Tesis Doctoral

Problema de empaquetamiento con conflictos generalizados

Pousa, Federico

2018-03-26

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Problema de empaquetamiento con conflictos generalizados

Tesis presentada para optar al título de
Doctor en el área de Ciencias de la Computación

Federico Pousa

Directoras: Isabel Méndez-Díaz
Paula Zabala

Consejero de estudios: Santiago Figueira

Lugar de trabajo: Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Buenos Aires, 2018

Fecha de defensa: 26/03/2018

Problema de empaquetamiento con conflictos generalizados

El problema de empaquetamiento con conflictos generalizados (PECG) es una generalización del problema de empaquetamiento en el cual los ítems a asignar presentan conflictos entre subconjuntos de ellos. Este problema surge naturalmente en situaciones donde se quiere resolver un problema de asignación minimizando la cantidad de contenedores utilizados, pero en donde los ítems no pueden ser asignados de forma irrestricta, sino que la asignación depende de los conflictos que se presentan entre ellos. En la literatura del área, no existe un tratamiento computacional para este problema, aunque sí se considera el caso particular en donde los ítems presentan conflictos de a pares.

En esta tesis se aborda el PECG mediante la formulación de modelos de Programación Lineal Entera. Para el modelo propuesto, se presenta un estudio teórico que consta de derivar el sistema minimal del poliedro asociado y familias de desigualdades válidas. Luego, en el caso de varias familias, se demuestra bajo que condiciones definen facetas del poliedro en cuestión. Simultaneamente, desde el aspecto práctico, se desarrolló un algoritmo branch-and-cut que incorpora diferentes componentes. En primer lugar se desarrollaron varias heurísticas y metaheurísticas para conseguir buenas soluciones primales. Luego, se incorporaron las familias de desigualdades más fuertes como planos de corte, junto con otras componentes particulares, para conseguir un algoritmo más robusto y eficiente. Por último, se experimentó con un extenso conjunto de instancias para analizar el desempeño, obteniendo muy buenos resultados que muestran la factibilidad de la utilización del enfoque en la práctica.

Palabras Clave: problema de empaquetamiento, optimización combinatoria, programación lineal entera, algoritmo branch-and-cut, estudio poliedral, grafo de conflictos, asignación en hipergrafos.

Bin Packing Problem with Generalized Conflicts

The Bin Packing Problem with Generalized Conflicts (BPGC) is a generalization of the Bin Packing Problem in which the items to be assigned present conflict among subsets of them. This problem arises naturally in situations when an assignment problem has to be solved minimizing the number of containers used, but with the addition that the items can not be assigned to a container without checking that the assignation is compliant with the conflicts between the items. In the literature of Bin Packing Problems, there is no computational treatment for this problem with an Integer Programming approach. However, there is a particular case of this problem, the one where the items only have conflicts among pairs of them, that has some papers devoted to it.

In this thesis the problem mentioned above is treated by proposing formulations based on Integer Linear Programming. For the formulations used, a theoretical study is presented consisting of the minimal system of the polyhedron along with several families of valid inequalities. Then, in the case of many families, proofs are given showing the special conditions that they need to fulfil in order to be facet defining for the polyhedron.

Then, from the practical point of view, a branch-and-cut algorithm was developed designing several components. Firstly, many heuristics and metaheuristics were developed in order to get primal solutions of good quality. Then, the strongest families of valid inequalities were added to the algorithm, besides many more specific components, willing to get a robust and efficient algorithm. As a final step, the algorithm was tested against a vast set of instances to analyze its efficiency. Very good quality results were obtained showing the feasibility of the usage of this approach in real life situations.

Keywords: bin packing problems, combinatorial optimization, integer programming, branch-and-cut algorithm, polyhedral study, conflicts graph, assignation in hypergraph.

Índice general

Capítulo 1

Introducción

1.1. Optimización combinatoria

La optimización es la rama científico-productiva que busca resolver problemas en donde hay que encontrar la mejor opción de entre un conjunto de posibles soluciones. En particular, el término *optimización combinatoria* se refiere al subconjunto de problemas en donde el conjunto de soluciones posibles es discreto. Los problemas de optimización combinatoria no son inherentemente difíciles. Todo problema en donde se quiera encontrar la mejor opción entre un conjunto discreto de posibilidades es un problema de este área. De tal manera que una cuestión tan simple como elegir el mejor precio entre dos productos iguales de diferentes marcas en un supermercado se puede catalogar como un problema de optimización combinatoria y su resolución es una trivial comparación.

En general, en muchos textos o charlas introductorios al tema, se asocia la dificultad de un problema de optimización combinatoria con la inmensidad del espacio de soluciones factibles. Este aspecto tiene una relación obvia, ya que si el espacio de soluciones fuese pequeño y conocido (como en el ejemplo del supermercado), podríamos resolver el problema correspondiente simplemente *mirando* todas las soluciones posibles y quedándonos con la mejor, sea cual fuese el concepto de mejor según el problema. Sin embargo, la correlación entre el tamaño del espacio de soluciones y la dificultad de un problema no es total. La dificultad de un problema no solo radica en el tamaño de su espacio de soluciones sino más bien en la forma en que se explora dicho espacio. Por ejemplo, el Problema del Viajante de Comercio consiste en encontrar la forma más rápida de realizar un tour entre n ciudades. En dicho problema, cualquier ordenamiento de las ciudades es una solución posible por lo que, a grandes rasgos, se tienen en el orden de $n!$ soluciones factibles. Por otro lado, el Problema del Árbol Generador Mínimo, consiste en encontrar un Árbol Generador en un grafo que sume el mínimo peso posible entre todas sus aristas. Gracias a la fórmula de Cayley, se sabe que la cantidad de árboles generadores de un grafo de n vértices puede ser hasta n^{n-2}. Si nos guiásemos solamente por el tamaño del espacio de soluciones, el segundo problema puede tener una cantidad de soluciones factibles mayor al primero para un n dado. Sin embargo, existen algoritmos polinomiales que indican como conseguir el Árbol Generador Mínimo de un grafo en tiempo polinomial, mientras que no se conocen algoritmos polinomiales para resolver el Problema del Viajante de Comercio, haciendo que en la actualidad sea un problema mucho más difícil.

Yendo al extremo de la comparación anterior, existen problemas de optimización de dominio continuo que son triviales por más que sus espacios de soluciones sean infinitos. Es por esto que si bien la cantidad de soluciones es un factor importante para que un problema sea difícil, no es el único factor a tener en cuenta. De hecho, hay problemas muy difíciles en donde la cantidad de soluciones no es tan grande, si no que lo difícil es justamente poder generar inclusive hasta una sola de las soluciones posibles.

Es por estas razones que en la resoluciones de este tipo de problemas es de vital importancia poder entender la estructura particular de cada problema para diseñar algoritmos que logren una

búsqueda guiada de la mejor solución en el inmenso espacio de soluciones posibles.

La optimización combinatoria se encarga de resolver problemas de muy variados dominios, a continuación mencionamos algunos ejemplos.

- Ruteo de vehículos: Implican la elección de las rutas óptimas para un vehículo o una flota de vehículos bajo un cierto conjunto de restricciones. En la práctica, el dominio puede ser variado por ejemplo ruteando camiones de una empresa de logística sujeto a restricciones de capacidad y de condiciones laborales. Sin embargo, los ejemplos se pueden aplicar a muchos otros sectores como ambulancias, aviones, barcos, etc., cada uno con sus restricciones operativas particulares y su objetivo distintivo.

- Asignaciones: Apareamientos de conjuntos de recursos bajo diferentes restricciones. Existen muchos ejemplos de este tipo de problemas como asignaciones de trabajadores a máquinas, puestos de trabajo u horarios. También son muy aplicados en problemas del campo de la salud donde los problemas de asignación resuelven cuestiones operativas como asignaciones de quirófanos, pero también pueden ser utilizados para situaciones más específicas como optimización de cadenas de transplantes.

- Empaquetamientos: Son problemas en los que se busca agrupar un conjunto de elementos de una forma específica, usualmente con el objetivo de crear la menor cantidad de grupos posibles. Nuevamente se presenta en dominios del mundo real muy diversos, pudiendo utilizarse para minimizar la cantidad de contenedores necesarios para el transporte de alimentos intercontinental, como así también para minimizar la cantidad de conexiones a un servidor web para traer todas las imágenes correspondientes a un videojuego online.

- Decisiones generales: Cualquier tipo de decisión determinante sujeta a diferentes restricciones operativas. Existen infinidad de ejemplos en esta dirección como las decisiones para crear una red ferroviaria, las decisiones para la utilización de una *smart grid* de electricidad, la colocación de productos o publicidades en espacios físicos o virtuales entre tantos otros.

Lo destacable del modelado de problemas de optimización combinatoria es que el estudio particular de uno tiene un impacto directo en la resolución de otros problemas, ya sea porque el modelo subyacente es el mismo o porque las técnicas utilizadas son transferibles entre las resoluciones. A modo de ejemplo, el Problema del Viajante de Comercio es posiblemente el problema de optimización combinatoria con mayor trato particular, aunque seguramente el día a día de los viajantes de comercio no sea el problema más importante o interesante del universo. Sin embargo, en las últimas décadas se han resuelto problemas de los más diversos al ser modelados como un Problema de Viajante de Comercio, tales como la optimización del camino de un taladro laser para la creación de placas de circuitos, o problemas de secuenciamiento de genoma humano. Además, muchas de las técnicas que se han desarrollado específicamente para este problema, han sido luego generalizadas para la resolución de problemas de optimización combinatoria en general, conformando hoy parte de los principales *solvers* computacionales de propósito general del área.

El impacto que tienen los problemas de optimización combinatoria sobre la vida cotidiana es inmenso. En cada instante, se resuelven problemas de este tipo para optimizar decisiones en muchas escalas diferentes. Un modelo parecido al que decide el movimiento de una flota de vehículos pequeña en alguna ciudad chica, puede también ser utilizado para resolver un problema de dimensiones mucho más grandes a nivel de un país.

En los últimos años la resolución de este tipo de problemas ha causado un gran impacto tanto económico como en otros aspectos. Centenares de aerolíneas resuelven día a día estos problemas para organizar sus operaciones y mejorar sus finanzas, siendo ésta la diferencia entre la bancarrota y la estabilidad financiera. La liga de fútbol americano de los Estados Unidos resuelve el problema de la calendarización de sus partidos para lograr satisfacer a todas las partes de un negocio que mueve millones de dólares [14], al igual que sucede en muchas otras ligas deportivas del mundo

[11, 51]. El servicio postal estadounidense UPS, ha desarrollado en el transcurso de varios años el ambicioso proyecto ORION que toma la mayoría de las decisiones asociadas a sus entregas por vía terrestre [30].

Al ver tantos ejemplos de arraigo de las soluciones de optimización en los problemas del mundo real, resulta válida la pregunta sobre si aún vale la pena el empuje académico sobre estos temas o si por el contrario las bases teóricas ya se encuentran bien definidas para poder atacar cualquier problema práctico. Esta pregunta resulta aún más válida si pensamos en la clase de subproblemas de optimización que se pueden modelar fácilmente mediante Programación Lineal, ya que son problemas para los cuales hace algunas décadas que se conocen algoritmos polinomiales para resolverlos [38]. Sin embargo, es sabido que aún contando con dichos algoritmos, existen problemas de tamaños tales que hacen que la resolución sea dificultosa y donde es importante explotar la estructura particular del problema para poder resolverlo. En el congreso Informs Annual Meeting 2017, Robert Bixby, fundador de CPLEX y Gurobi, dos de los *solvers* de propósito general más importantes del área, presentó una charla titulada *Optimization: past, present, future*. En dicha charla, Bixby contó su experiencia en sus vastos años de desarrollo de CPLEX y Gurobi, haciendo hincapié en la importancia de los avances matemáticos para lograr la escalabilidad que se tiene en la actualidad. Acotando solamente a un extenso conjunto de *benchmarks* de problemas de Programación Lineal, los avances puramente tecnológicos entre el año 1986 y 2004, han provisto una mejora de 1600 veces en los tiempos de cómputo. Mientras tanto, los avances matemáticos del área proveyeron una mejora de 3300 veces. Lo cual resulta aún más sorprendente considerando que el algoritmo con mejor clase de complejidad para Programación Lineal se desarrolló en 1984. Esto muestra que más allá del desarrollo de un algoritmo polinomial, fue muy importante la continua investigación académica del área para poder llegar a los niveles de resolución práctica actuales.

1.2. Problemas de empaquetamiento

Como mencionamos anteriormente, una categoría de problemas dentro del área de la optimización combinatoria son los problemas de empaquetamiento o más comúnmente conocidos como problemas de *Bin Packing*. Esta categoría de problemas ha tenido un particular trato en la literatura ya que engobla muchos problemas y situaciones reales en las que se busca agrupar un cierto conjunto de elementos, generalmente tratando de minimizar la cantidad de contenedores, pero cumpliendo con restricciones operativas particulares.

En su versión original, el Bin Packing Problem (BPP) consiste en ubicar una cierta cantidad de ítems, cada uno con un valor asociado que le llamaremos *tamaño*, en la menor cantidad de contenedores posibles. Los contenedores son indistinguibles y poseen un valor asociado llamado *capacidad*. El problema busca minimizar la cantidad de contenedores necesarios para ubicar los ítems de manera tal que la suma de los tamaños de los ítems asignados a un contenedor nunca exceda su capacidad.

La situación que modela el BPP resulta de utilidad en muchos problemas reales, pero otras veces se tienen variaciones del contexto o de las restricciones del problema que resultan en problemas de empaquetamiento que no son estrictamente el BPP. A continuación enumeramos algunas extensiones o variaciones al BPP:

- Empaquetamientos en donde los ítems no tienen un solo valor asociado, sino varios. Los ítems en este caso en vez de solamente tener un valor *tamaño*, pueden tener más de un valor que dan origen a restricciones independientes (peso, altura, etc.).
- Empaquetamientos en donde no todos los contenedores tienen la misma capacidad.
- Empaquetamientos donde existe una restricción de orden en la asignación de los ítems.
- Empaquetamientos en donde los ítems presenten conflictos entre sí. En estos casos existen restricciones para ubicar ciertos ítems juntos.

- Empaquetamientos donde no es obligatorio asignar todos los ítems sino que existe una penalización asociada a la no asignación.

- Empaquetamientos geométricos en 2 o 3 dimensiones.

Como mencionamos anteriormente, las variaciones presentadas suelen surgir de situaciones reales en donde las restricciones operativas no son modeladas completamente por el BPP original. Existen muchas aplicaciones asociadas a cada una de las variaciones presentadas:

- Generalizaciones de Bin Packing son vastamente utilizadas en la resolución de problemas asociados a la asignación de procesos a procesadores. En [19] se puede ver un ejemplo en donde los procesadores tienen recursos limitados que imponen restricciones conjuntamente a otras restricciones impuestas por precedencias entre los procesos a ejecutar.

- En la misma línea de asignar procesos a procesadores, puede suceder que dos procesos no puedan ser asignados al mismo procesador ya que compiten por un mismo único recurso. En [32] se presenta esta aplicación y un algoritmo aproximado para su resolución.

- En diferentes industrias de producción de materiales tales como caños o chapas, se cortan los productos a comerciar desde un trozo de materia prima inicial. Dada una demanda particular de, por ejemplo, chapas rectangulares, es importante minimizar la cantidad de chapas madre que se necesitan para recortar el pedido para minimizar costos. En este caso lo que se necesita resolver es un problema de Bin Packing geométrico en 2 dimensiones. En [9] se puede ver un trabajo fundacional sobre esta variación del problema.

- En un caso de aplicación parecido al anterior, a veces sucede que estos rectángulos no se obtienen de trozos separados de materia prima más grande, si no que se cortan de una gran banda de materia prima de ancho fijo y altura no acotada. Esta aplicación da lugar a una variación de problema de Bin Packing ligeramente diferente que se conoce como el Strip Packing Problem. En [42] se puede ver un enfoque de resolución exacta para este problema.

Estos casos son solo algunos de las varias aplicaciones que se modelan utilizando el Bin Packing Problem o alguna de sus variantes.

El Bin Packing Problem, y sus variaciones, han tenido un extenso trato en la literatura del área. Por un lado, como ya mencionamos antes, resulta de interés por la cantidad de situaciones reales que engloba un modelo de este tipo. Por otro lado, el problema ha despertado interés en otros aspectos. Por ejemplo, los problemas de empaquetamiento suelen ser problemas que se prestan muy bien para el desarrollo de algoritmos aproximados y esquemas de aproximación. Mencionando algunos ejemplos del tratamiento en la literatura, en [21] se demuestra que el Bin Packing Problem pertenece a la clase $\mathcal{NP}$-hard. En [34] se presentan varios resultados correspondientes a las cotas teóricas para diferentes algoritmos aproximados para el problema. En [10] y [20] se presentan recopilaciones y análisis de diferentes algoritmos para el Bin Packing Problem.

1.3. Objetivo de la tesis

En este trabajo nos proponemos investigar un problema particular dentro del conjunto de problemas de empaquetado. Estudiaremos el Problema de Empaquetamiento con Conflictos Generalizados. Para estudiar este problema lo haremos desde un enfoque de Programación Lineal Entera, desarrollando e investigando cada uno de los componentes clásicos en torno a este enfoque con el fin de no solo obtener un estudio teórico asociado al problema, si no también con el objetivo de poder volcar los conceptos teóricos desarrollados en el diseño de un algoritmo robusto y eficiente para dicho problema.

El trabajo se divide en 7 capítulos.

En el Capítulo 2 se realiza una revisión de los principales conceptos asociados a la Programación Lineal Entera. En dicho capítulo se describen los algoritmos más utilizados bajo este enfoque, dando una idea general de los principales componentes que los conforman, los cuales serán objeto de estudio a lo largo de este trabajo.

En el Capítulo 3 se introduce específicamente al Problema de Empaquetamiento con Conflictos Generalizados. Luego de dar una definición coloquial y formal del problema, se realiza una revisión de literatura del problema. Además se enuncian algunas aplicaciones reales en donde el problema puede ser utilizado, haciendo foco en un caso de estudio particular que hemos abordado como proyecto de transferencia tecnológica desde nuestro grupo de investigación. Por último, se menciona la relación del problema a estudiar con otros problemas conocidos en la literatura.

En el Capítulo 4 se estudian diferentes heurísticas para el problema con el fin de obtener soluciones iniciales de buena calidad, las cuales son condicionantes para el desempeño del algoritmo en su totalidad. Además de presentar diversas heurísticas constructivas, se experimentará con metaheurísticas de búsqueda local para generar soluciones factibles.

En el Capítulo 5 se redefine el modelo de Programación Lineal Entera que se utilizará para el problema y se realiza un estudio poliedral sobre el mismo. El objetivo es poder tener una descripción completa del sistema minimal, para luego también poder reconocer varias desigualdades válidas para el problema, demostrando la condición de facetitud de varias de ellas.

En el Capítulo 6 se combinan los resultados obtenidos hasta el momento junto con la investigación de diversas componentes algorítmicas para desarrollar un algoritmo de tipo branch-and-cut robusto y eficiente para el problema. Junto con la descripción de las diferentes partes del algoritmo, se presenta un estudio cuantitativo y cualitativo del desempeño de varias combinaciones posibles sobre diferentes tipos de instancias del problema.

Por último, en el Capítulo 7 se presentan las conclusiones generales del trabajo y se explayan algunas posibles direcciones para continuar las investigaciones.

Capítulo 2

Preliminares

En este trabajo se estudia un problema de optimización combinatoria. Los problemas de optimización combinatoria son aquellos en los que hay que encontrar una configuración óptima de entre un conjunto discreto, pero usualmente muy grande, de posibilidades.

Los problemas de optimización combinatoria muchas veces admiten un modelado mediante una formulación de Programación Lineal Entera Mixta (PLEM). En líneas generales, los modelos PLEM son simples de formular. Sin embargo, un mismo problema puede admitir muchas formulaciones PLEM distintas que pueden llevar a resoluciones del problema muy distantes. El objetivo de este capítulo es introducir los conceptos básicos y los algoritmos típicos utilizados para formulaciones PLEM. Para dichas explicaciones, se asumirá la familiaridad con los conceptos básicos de Programación Lineal (PL) y Teoría de Grafos.

2.1. Programación Lineal Entera Mixta

Dentro de la comunidad de optimización combinatoria, el uso de formulaciones PLEM para resolver este tipo de problemas ha sido y es muy aceptado y todavía es una de las técnicas más efectivas y utilizadas para resolver los problemas combinatorios más desafiantes.

Con este tipo de formulaciones se pueden modelar diversas situaciones en donde el objetivo es minimizar o maximizar una función lineal para un conjunto de soluciones factibles, las cuales surgen de satisfacer un conjunto de restricciones, también lineales. La diferencia con los problemas de Programación Lineal, es que en el caso de PLEM algunas de las variables del modelo solo admiten valores enteros. Esta diferencia no solo permite modelar cantidades enteras de alguna opción, como podría ser la producción de algún tipo de producto indivisible, sino que también admite el modelado de decisiones particulares. Se puede por ejemplo utilizar una variable binaria, solo admite valor 0 ó 1, para decidir si realizar o no cierta acción. Hay sobrados casos en la literatura de problemas muy conocidos donde se utilizan este tipo de variables, siendo tal vez el más conocido el Problema del Viajante de Comercio (TSP), aunque también hay otros problemas de disciplinas diversas como problemas de optimización sobre redes, asignación de recursos, teoría de grafos, y muchos otros.

En una definición formal, un formulación PLEM busca la solución óptima de una función lineal dentro de un espacio vectorial, restringiéndose a la región definida por el poliedro caracterizado por las igualdades y desigualdades del modelo, y la integralidad de las variables. En una forma genérica, una formulación PLEM se puede definir como:

$$\begin{aligned} \text{mín} \quad & \sum_{j\in I} c_j x_j + \sum_{j\in C} c_j x_j && (2.1)\\ \text{s.t.} \quad & \sum_{j\in I} a_{ij} x_j + \sum_{j\in C} a_{ij} x_j \geq b_i, \quad i = 1,\ldots,m \\ & x_j \in \mathbb{Z}_+ \quad \forall\ j \in I \\ & x_j \in \mathbb{R}_+ \quad \forall\ j \in C \end{aligned}$$

en donde I es el conjunto de variables que solo admiten valores enteros y C es el conjunto de variables continuas, con $I \cup C = \{1,\ldots,n\}$. El vector $c \in \mathbb{R}^n$ representa la función objetivo y los coeficientes a_{ij} se expresan como una matriz de restricciones $A \in \mathbb{R}^{m\times n}$. Si $C = \emptyset$, entonces a la formulación se la suele denominar como una de Programación Lineal Entera pura (PLE).

Con un propósito puramente notacional, el problema (2.1) también se expresa de la manera

$$\text{mín}\{cx \ :\ Ax \geq b,\ x \geq 0,\ x_j \in \mathbb{Z}_+\ \forall j \in I\}.$$

Este problema en su forma general pertenece a la clase de complejidad $\mathcal{NP}$-hard, lo que significa que hasta la actualidad no se conoce un algoritmo polinomial que lo resuelva.

El poliedro asociado a una formulación PLEM se define como $P = \{x \in \mathbb{R}^n_+ \ :\ Ax \geq b\}$ y el conjunto de las soluciones factibles se denota $S = P \cap \{x \in \mathbb{R}^n \ :\ x_j \in \mathbb{Z}\ \forall\ j \in I\}$. La *relajación lineal* (LP) del problema (2.1) es

$$\text{mín}\{cx \ :\ Ax \geq b,\ x \geq 0\}. \tag{2.2}$$

La principal diferencia entre el LP y el problema (2.1) es que para el primero se conocen algoritmos polinomiales para resolverlo (Karmarkar [37]). Es más, los algoritmos utilizados para resolver problemas LP funcionan bastante bien en la práctica, logrando resolver instancias con un gran número de variables y restricciones.

Denotamos $conv(S)$ a la cápsula convexa de S. Es decir al menor poliedro que contiene a todos los puntos de S. Existen instancias de PLEM que tienen la particularidad de que pueden ser resueltas en tiempo polinomial. Por ejemplo, si $P = conv(S)$ y el número de restricciones necesarias para describir dicha cápsula convexa es polinomial, entonces cualquier algoritmo que resuelva un LP general, podrá ser usado para resolver esta instancia hasta la optimalidad. Sin embargo, esto en la práctica sucede en un número muy acotado de problemas y todos los problemas que pertenecen a la clase $\mathcal{NP}$-hard no cumplen esta condición.

Cuando la caracterización de la cápsula convexa no es conocida, la relajación LP suele ser muy útil. Si dicha relajación resulta ser infactible, entonces el problema PLEM también lo es, ya que la relajación tiene en juego justamente a un espacio más grande de soluciones posibles en donde están incluídas las del problema original. Si el punto óptimo del LP satisface no solo las restricciones lineales, si no también las restricciones de integralidad, entonces también resulta ser el óptimo del problema PLEM. Por último, en el caso de que exista una solución factible pero que no sea entera, entonces el valor óptimo de la relajación LP es una cota dual del valor óptimo del problema PLEM. Luego, poder obtener relajaciones LP lo más ajustadas posibles le da aún más importancia a dicho valor. La diferencia entre el valor óptimo del problema PLEM y el valor óptimo de la relajación LP, es lo que se llama *gap* y se utiliza como una medida que sirve para evaluar la calidad de la relajación.

Luego, la diferencia entre el valor de la mejor solución conocida para un PLEM y el valor óptimo de la relajación LP es una cota por arriba del valor del gap real entre la solución conocida y la mejor solución posible. Este valor es útil para evaluar la calidad de una solución primal y, en el caso de que el gap sea 0, sirve para certificar la optimalidad de la solución.

Muchos de los algoritmos conocidos para resolver problemas PLEM se basan justamente en la relación entre el problema PLEM y su relajación LP. Algunos de los algoritmos más utilizados en la práctica se describen en la siguiente sección.

2.2. Algoritmos para problemas PLEM

La mayoría de los algoritmos para resolver un modelo PLEM se pueden clasificar dentro de uno de los siguientes enfoques:

- Algoritmos de *planos de corte*
- Algoritmos de *branch-and-bound*
- Algoritmos de *branch-and-cut*
- Algoritmos de *branch-and-price*

2.2.1. Algoritmos de Planos de corte

La principal idea detrás de los algoritmos de planos de cortes se basa en las sucesivas mejoras de la relajación LP mediante la adición de desigualdades lineales válidas para la cápsula convexa $conv(S)$ para cortar soluciones del LP que no pertenecen a $conv(S)$. El esquema general del algoritmo comienza omitiendo las restricciones de integralidad de las variables y resolviendo la relajación LP. Si el punto obtenido no es entero, entonces se intenta identificar una desigualdad lineal válida para $conv(S)$ que separe a la solución del LP hallada de todos los puntos enteros factibles de S. Al agregar esta nueva desigualdad a la formulación, se obtiene una nueva relajación LP que es al menos tan fuerte como la anterior y en muchos casos estrictamente más fuerte, lo cual permite reproducir este proceso iterativamente.

Claramente, el éxito de esta metodología depende fuertemente de la posibilidad y la eficiencia de encontrar desigualdades válidas que sean violadas por el punto óptimo del LP, que puedan ser agregadas al modelo para separar dicho punto. Esto significa que este enfoque requiere de un *algoritmo de separación* para encontrar dichos cortes.

El esquema básico de un algoritmo de planos de corte se puede expresar de la siguiente manera.

Entrada: Una formulación PLEM.

1. *Inicialización*: Considerar la relajación LP del problema.
2. *Resolviendo la relajación LP*: Sea x^* la solución óptima de la relajación LP. Si $x_j^* \in \mathbb{Z} \; \forall \; j \in I$, entonces parar. Si no, ir a 3.
3. *Separación*: Buscar una desigualdad válida violada por x^*. Si se encuentra, agregarla al problema actual e ir a 2. Si no, parar. El problema no pudo ser resuelto.

Algorithm 1: Algoritmo general de planos de corte

Los planos de cortes se pueden generar por dos tipos de enfoques diferentes:

- Planos de cortes de propósito general, aplicables a cualquier problema PLEM.

  A comienzo de la década del 60, Gomory [24] desarrolló un algoritmo general para generar desigualdades válidas para cortar la actual solución de toda relajación LP. En cada iteración, se genera una desigualdad basada en la expresión de las variables básicas en función de las variables no básicas, utilizando exclusivamente argumentos de integralidad. Se dice que es un algoritmo general porque no utiliza ninguna información particular del problema, sino que puede realizarse el procedimiento de la misma manera para cualquier instancia. Bajo ciertas circunstancias, este método converge a la solución óptima del problema. Este resultado es extremandamente importante desde un punto de vista teórico ya que asegura que se puede encontrar un plano de corte en cualquier situación. Sin embargo, en la práctica no suele ser eficiente como para poder resolver instancias PLEM en un tiempo razonable por sí solo.

En concordancia con este procedimiento, muchos algoritmos han sido desarrollados usando una variedad de cortes de propósito general como los cortes *disjuntivos*, los cortes *clique*, los cortes *cover*, etc. Muchos de estos cortes se pueden encontrar desarrollados en [46]. Al igual que los cortes Gomory, todos estos algoritmos constituyen resultados teóricos muy interesantes pero en general su aplicación en la práctica no tiene tanto éxito. Por un lado, tienen la ventaja de que siempre pueden ser aplicados, por lo cual ante la falta de otras herramientas posibles, estos algoritmos siempre se pueden aplicar. Por otro lado, casi nunca resultan en herramientas super adecuadas para el tratamiento particular de un problema. En general, un estudio más específico que dependa de la estructura del problema a resolver, logra proveer cortes de mayor calidad con un impacto en la práctica mucho más notorio.

- Planos de corte que explotan la estructura particular del problema

  Cada problema presenta propiedades particulares que pueden facilitar la identificación de mejores planos de corte. El primer trabajo en la literatura que considera este enfoque es el trabajo de Dantzig et al. [13] del año 1954 para el Problema del Viajante de Comercio. Con las técnicas propuestas, los autores fueron capaces de resolver una instancia con 49 ciudades, lo cual presentaba un serio desafío al conocimiento y el poder de cómputo de la época. De esta manera, sentaron la piedra fundamental para lo que hoy conforma una de las herramientas más poderosas para los PLEM, los resultados poliedrales.

  Una propiedad deseada para un plano de corte es la eliminación del mayor volumen posible de soluciones fraccionarias. Dado que los planos de corte son desigualdades válidas para la cápsula convexa $conv(S)$, es razonable asumir que las *facetas* de dicha cápsula resultarán particularamente útiles. El primer estudio poliedral en esta dirección fue realizado en la década del 70 para el Problema de Conjunto Independiente por Padberg en [48] y para el TSP por Gröetschel y Padberg en [28, 29]. Estos trabajos significaron un notorio avance en la resolución de este tipo de problemas.

  Para lograr un algoritmo particularizado para un problema dado, el estudio poliedral del mismo tiene que ser desarrollado conjuntamente con algoritmos de separación eficientes. En este sentido Grötschel et al. [27] establecieron uno de los resultados teóricos más importantes, el cual relaciona la complejidad computacional del problema de separación con el problema de optimización. Este resultado dice que la optimización del problema $\text{máx}\{cx \ : \ x \in conv(S)\}$ puede ser resuelta en tiempo polinomial si y solo si el problema de separación también. Esto también muestra entonces una medida de la dificultad de encontrar una descripción completa de la cápsula convexa del problema, y que tan difícil puede resultar para ciertas familias de desigualdades válidas desarrollar un algoritmo de separación.

  La desventaja de realizar un estudio poliedral particularizado a un problema es que las desigualdades encontradas solo pueden ser aplicadas al problema en cuestión. Sin embargo, algunas desigualdades obtenidas para problemas particulares pueden ser utilizadas o adaptadas a casos más generales. Por ejemplo, ciertas desigualdades válidas desarrolladas para el Problema de la Mochila [43], [46] se han generalizado y hoy son parte del núcleo básico de los *solvers* más importantes para problemas PLEM.

Finalmente, basándose en esta discusión, podemos concluir que un algoritmo de planos de corte puede no resolver hasta la optimalidad un problema, ya sea porque no encuentra un buen corte para la relajación actual, o porque excede un tiempo de cómputo razonable. De todas maneras, este enfoque puede ser utilizado para obtener buenas cotas duales para el valor óptimo del PLEM. Además, tomando el óptimo de la relajación LP, es posible también encontrar una buena solución primal aplicando algún tipo de heurística.

2.2.2. Algoritmos branch-and-bound

La idea principal detrás de los algoritmos branch-and-bound es la de realizar una enumeración inteligente del conjunto de soluciones factibles. Para esto se utiliza información del problema que

ayuda a reducir el tamaño del árbol de enumeración. Estos algoritmos pueden ser clasificados bajo el esquema *Divide and Conquer*, en el que la idea es separar el espacio de búsqueda en pedazos más pequeños para obtener subproblemas más fáciles de resolver, uniendo luego las resoluciones de estos subproblemas para obtener una solución al problema total.

Este esquema se representa con un árbol de enumeración en donde el nodo raíz corresponde al problema original y las ramas que se van abriendo resultan de ir partiendo el espacio de búsqueda de soluciones. Cada nodo del árbol tiene asociado un subproblema que involucra encontrar el óptimo de la función objetivo pero sujeto a una parte del espacio de solución. Además, argumentos de dominancia y factibilidad son usados para podar ramas completas del árbol durante el proceso de resolución.

Una posibilidad para podar ramas es la que se denomina *poda por cota*. En este caso se calcula en cada nodo del árbol una cota dual del valor óptimo del subproblema que resulta de restringir el espacio de búsqueda a la región asociada con dicho nodo. Si la cota dual obtenida es más grande que la mejor cota primal obtenida para el problema, entonces no es necesario seguir explorando esta rama porque esto indica que el valor óptimo entero de la rama seguro tiene que ser peor que la cota primal que ya se tiene. El cálculo de las cotas duales en cada nodo debe provenir de un *tradeoff* entre la calidad de la cota y el esfuerzo requerido para su cómputo.

El cómputo de cota duales y primales se puede realizar mediante muchos enfoques distintos. En el caso de las duales, cualquier relajación del problema puede producir una cota válida. En el caso más típico, la cota dual se puede obtener mediante la relajación LP del problema que proviene de descartar la condición de integralidad sobre las variables. Si bien la relajación LP es una de las relajaciones más utilizadas, existen otro tipo de relajaciones posibles para obtener cotas duales. Una posible alternativa es utilizar la relajación Lagrangeana [4], en la cual algunas de las desigualdades lineales son descartadas del conjunto de restricciones y agregadas mediante un procedimiento particular a la función objetivo. Otra posible forma de obtener una cota dual es mediante lo que se denomina una relajación combinatoria. En este tipo de relajaciones lo que se hace es descartar ciertas restricciones que hacen que el problema se transforme en otro problema conocido para el cual existe una forma más simple de resolverse, por ejemplo un algoritmo polinomial. También existen otras formas de relajación, tal como las relajaciones subrogadas que provienen de conglomerar muchas restricciones en una sola mediante una combinación de las mismas.

Si bien vemos que existen muchos tipos de relajaciones posibles, la relajación LP tiene un papel protagónico en los algoritmos branch-and-bound. Esto sobre todo sucede porque una vez que se tiene la solución de una relajación LP para un nodo, suele resultar simple en la práctica hallar la solución de la relajación LP de los nodos hijos en el árbol de búsqueda.

Por el lado de las cotas primales, el valor de cualquier solución factible al problema sirve. De esta manera, cualquier solución hallada mediante heurísticas o mediante la integralidad de un óptimo de la relajación LP de algún nodo se puede utilizar como una posible cota primal.

La partición del espacio de búsqueda, lo que se denomina *branching*, se hace mediante la división de la región factible analizada por el subproblema actual asociado a un nodo en dos o más regiones factibles más pequeñas. Cada nueva región genera un subproblema nuevo asociado a un nuevo nodo en el árbol, que se origina agregando algún tipo de restricción adicional al subproblema asociado al nodo padre. Una observación importante para el proceso de *branching* es que toda solución factible del problema padre debe estar presente en al menos uno de sus hijos. Estos nuevos subproblemas generados son insertados a la lista de nodos a explorar. Una de las reglas de *branching* más simple es considerar alguna variable entera d, que se encuentre con un valor fraccional, d^*, cuando se resuelve la relajación LP de un nodo. Con esta varible, el subproblema puede ser partido en dos nuevos subproblemas en donde se agregra la restricción $\lfloor d^* \rfloor$ como cota superior de la variable en uno de los subproblemas, y $\lceil d^* \rceil$ como cota inferior del otro subproblema. Este procedimiento se puede aplicar recursivamante a cada uno de los nodos del árbol.

El pseudocódigo para el esquema general de un algoritmo branch-and-bound se muestra en el algoritmo 2. Se debe tener en cuenta que en este esquema general hay dos componentes importantes que no se están especificando y tienen un alto impacto en el comportamiento del algoritmo. Estas componentes son la selección de cual va a ser el siguiente nodo a resolverse y el proceso por el cual

se generan los nuevos subproblemas.

En cuanto a la selección de nodo, entre las estrategias más utilizadas se puede mencionar la búsqueda en profundidad o DFS por sus siglas en inglés, la búsqueda en anchura o BFS y la búsqueda por mejor cota o *best bound* que se basa en tomar el nodo con la mejor cota dual posible, la cual dependiendo el caso puede ser fehacientemente obtenida, o bien puede ser estimada.

Por el lado del *branching*, si bien ya se mencionó una de las formas más típicas de hacerlo, queda pendiente la elección de sobre que variable se hará el procedimiento. Esta elección se puede hacer de varias maneras diferentes. Algunas de estas estrategias de selección se basan en los métodos de *mínima infactibilidad*, *máxima infactibilidad*, *pseudocostos* y *strong branching*. En [1] Achterberg et al. hacen un repaso completo de las diferentes estrategias mencionadas.

No existe una combinación de estas dos componentes que siempre produzca mejores resultados para cualquier problema que se presente. Es necesario tomar decisiones para cada problema, basándose en una mezcla entre la teoría, la estructura de cada problema y la experimentación que se lleva a cabo.

Entrada: Una formulación PLEM.

1. *Inicialización*: Se crea una lista L con el nodo raíz y la relajación LP del problema. Asignar $Z^{\text{ub}} = \infty$
2. *Selección de nodo*: Si L está vacía, entonces parar y reportar Z^{ub}. Si no, seleccionar un elemento de L y removerlo de la lista.
3. *Acotando*: Resolver la relajación lineal asociada al nodo seleccionado. Si es infactible, regresar a 2. Si no, sea x^* la solución óptima y Z^* el valor de la función objetivo.
   - Si x^* es factible para el problema original, asignar $Z^{\text{ub}} = \text{mín}\{Z^{\text{ub}}, Z^*\}$. Regresar a 2
   - Si $Z^* \geq Z^{\text{ub}}$, entonces en esta rama no puede haber una solución factible mejor que la cota primal actual. Regresar a 2.
4. *Branching*: Generar subproblemas para el nodo actual y agregarlos a la lista L. Regresar a 2.

Algorithm 2: Algoritmo Branch and Bound general

2.2.3. Algoritmos branch-and-cut

En el comienzo de la década de los 80, Crowder et al. en [12] fueron muy exitosos en aplicar una técnica mixta para resolver problemas con solo variables binarias. Consideraron un algoritmo branch-and-bound pero, antes de comenzar con el proceso de *branching*, aplicaron la técnica de planos de corte a la relajación lineal asociada al nodo raíz. De esta manera, pudieron mejorar notoriamente la cota dual obtenida por la relajación LP, lo cual se tradujo en una reducción del tamaño del árbol explorado.

Luego, a mediados de dicha década, comenzaron a aparecer los primeros trabajos en los que esta idea se extendió aplicando los algoritmos de planos de corte no solo al nodo raíz, sino también al resto de los nodos del árbol. Grötschel et al. en [26] utilizaron este enfoque para el Linear Ordering Problem (LOP) y Padberg et al. en [50] para el TSP. En este último, se introdujo el término branch-and-cut para referirse a este tipo de algoritmos. El esquema general para este tipo de algoritmos se muestra en el algoritmo 3.

La descripción del algoritmo no especifica algunas componentes importantes, además de las mencionadas cuando se explicó el algoritmo branch-and-bound. Por ejemplo, un primer punto a considerar es cuántas iteraciones de búsqueda de planos de corte se deben realizar por cada uno de los nodos. También se debe tener en cuenta cuántos cortes se quieren agregar en cada

iteración, qué se debe hacer con los cortes generados en diferentes subárboles, etc. Todas estos puntos representan decisiones de diseño que afectan en gran medida al desempeño de un algoritmo de estas características, afectando la efectividad del algoritmo y el tiempo de cómputo requerido por el mismo. En la práctica, es importante encontrar un balance entre las diferentes estrategias posibles para encontrar la combinación que mejor se adapte al problema en cuestión.

En términos generales, hay dos puntos principales a remarcar para explicar los buenos resultados que se obtuvieron con esta técnica:

- El uso de los planos de corte, que si bien se generan para un subproblema particular del árbol, son válidos para todo el árbol. Esto hace que el esfuerzo utilizado para un subproblema pueda ser reutilizado.
- El uso de desigualdades válidas específicas para el problema, que provienen de realizar un estudio poliedral particularizado.

En las últimas décadas, este tipo de algoritmos se utilizó de manera muy efectiva para resolver una gran variedad de problemas de optimización combinatoria. El uso de desigualdades válidas específicas para cada problema fue uno de los factores más importantes para la resolución de grandes instancias en problemas difíciles.

Entrada: una formulación PLEM.

1. *Inicialización*: Crear una lista L con el nodo raíz y la relajación LP del problema. Asignar $Z^{\text{ub}} = \infty$.
2. *Selección de nodo*: Si L está vacía, entonces parar y reportar Z^{ub}. Si no, seleccionar un nodo de L y removerlo de la lista.
3. *Acotando*: Resolver la relajación lineal asociada al nodo elegido. Si es infactible, regresar a 2. Si no, sea x^* la solución óptima y Z^* el valor de la función objetivo.
   - Si x^* es factible para el problema original, asignar $Z^{\text{ub}} = \text{mín}\{Z^{\text{ub}}, Z^*\}$. Regresar a 2
   - Si $Z^* \geq Z^{\text{ub}}$, entonces en esta rama no puede haber una solución factible con mejor cota primal que la actual. Regresar a 2.
4. *Branching vs. Cutting*: Decidir si buscar planos de corte o no. Si la respuesta es no, ir a 6.
5. *Separación*: Buscar una desigualdad válida violada por x^*. Si no se encuentra ninguna, ir a 6. Si no, agregarla a la formulación e ir a 3.
6. *Branching*: Generar los subproblemas para el nodo actual y agregarlos a L. Regresar a 2.

Algorithm 3: Algoritmo Branch and Cut general

2.2.4. Algoritmos branch-and-price

Los algoritmos del tipo branch-and-price son una generalización de los branch-and-bound que fueron introducidos por Savelsbergh en [54]. Están especialmente diseñados para formulaciones que tienen un gran número de variables, posiblemente exponenciales, y que por lo tanto no se pueden manejar explícitamente. Este tipo de modelos son muy frecuentes en la práctica y se consideran útiles por diferentes motivos. Por ejemplo, podría ser que una formulación con una cantidad exponencial de variables sea la forma más natural o hasta incluso la única manera para modelar un problema. También puede suceder que un modelo alternativo con una cantidad de variables menor tenga una relajación LP mucho más débil por lo que se prefiera entonces el modelo con mayor cantidad de variables.

Debido al tamaño del modelo, la resolución de la relajación LP asociada a cada nodo del árbol puede ser muy costosa en cuanto al tiempo de cómputo requerido. La estrategia en estos casos es resolver la relajación LP pero restringiéndose a un subconjunto de variables con una cantidad manejable. La solución obtenida es factible para la relajación original, pero no es necesariamente el valor óptimo. Para chequear la optimalidad lo que se debe hacer es buscar entre las variables que no fueron consideradas para ver si hay alguna candidata a entrar a la base. Eso es, sea y^* la solución óptima del dual del problema restringido, tenemos que determinar si existe una columna a con $c_a - y^*a < 0$, con c_a el costo de la columna a en la función objetivo. Si existe una columna que satisfaga esta condición, la columna a es incluída en el conjunto restringido de variables a ser consideradas y el problema es resuelto nuevamente. De lo contrario, la solución actual es la óptima y el algoritmo continúa con el esquema normal asociado a un algoritmo branch-and-bound. Esta técnica para resolver las relajaciones LP se conoce como *Generación de columnas*.

Como las columnas no se consideran explícitamente, el procedimiento utilizado para chequear la optimalidad de una solución en la relajación LP es un factor importante para el éxito de esta técnica. En varias situaciones, las columnas de la matriz de restricciones pueden ser descriptas implícitamente como vectores característicos de subconjuntos de un conjunto particular. Por ejemplo, dado un conjunto de ítems con diferentes pesos, los subconjuntos de ítems que tienen un peso total por debajo de un peso particular podrían ser un subconjunto a considerar. En estos casos, es posible formular el subproblema de generación de columna como un modelo PLEM que puede ser resuelto por un algoritmo exacto como por algún tipo de heurística. En el ejemplo de los ítems, obtenemos un problema de la mochila: de todos los posibles subconjuntos de ítems que están por debajo de cierto peso, se selecciona el que tiene el menor costo reducido. Uno de los primeros trabajos aplicando esta técnica es el trabajo de Gilmore y Gomory en [23] para el Cutting Stock Problem.

Dentro del contexto de un algoritmo branch-and-price, es importante desarrollar un algoritmo eficiente para la generación de columnas que permita resolver las relajaciones LP en una cantidad razonable de tiempo. Esto depende esencialmente de la estructura del subproblema de generación de columnas y del hecho de que la estrategia de *branching* no debería afectar la estructura de este subproblema considerablemente, para que no se vuelva cada vez más complicado de resolver.

2.2.5. Comentarios adicionales

Aún cuando no sea posible encontrar la solución óptima a un problema PLEM, todos los algoritmos descriptos son realmente útiles en la práctica. El valor óptimo de la relajación LP siempre constituye una cota dual para el valor óptimo entero. Luego, una de las principales características de todos estos enfoques es ir iterativamente mejorando esta cota dual y, en el caso de los algoritmos branch-and-bound y branch-and-cut, nuevas soluciones factibles pueden ir apareciendo durante el proceso de enumeración, las cuales sirven para mejorar la cota primal del problema. En la práctica, a veces no es necesario llegar a encontrar la solución óptima del problema, sino que ya puede ser bueno tener una solución entera de calidad, junto con una medida de cual es la máxima distancia posible que se tiene hasta el verdadero valor óptimo. Tener buenas cotas duales y primales para un problema es una parte esencial para estimar de la mejor manera posible la calidad de una solución factible dada.

Ninguno de los métodos presentados resultan en algoritmos de complejidad polinomial y, en general, resulta muy difícil estimar cuanto va a ser el tiempo requerido para encontrar la solución óptima de un PLEM, y luego poder probar que es la óptima. El tamaño de una instancia, teniendo en cuenta la cantidad de variables y restricciones, a veces puede resultar un estimador superficial de la dificultad de un problema, pero en general no es una métrica que baste para asociar un cierto nivel de dificultad. En la mayoría de los casos, la dificultad de resolver un problema PLEM no está asociado solamente al tamaño de la instancia particular, si no a la estructura que el problema pueda presentar.

En las últimas décadas, muchos investigadores han puesto esfuerzos considerables en desarrollar nuevas ideas que sustenten estos enfoques, generando mejores y más eficientes implementaciones

que logran reducir drásticamente los tiempos de cómputo necesarios. Estas ideas, en conjunción con los avances tecnológicos, hace que hoy en día sea posible resolver instancias de problemas PLEM que hasta hace algunos años eran completamente intratables.

2.3. Ejemplo: El Problema del Viajante de Comercio Asimétrico

En esta sección presentamos la formulación clásica del Problema del Viajante de Comercio Asimétrico (ATSP) para ilustrar algunos de los conceptos explicados en este capítulo.

Repasemos la definición del ATSP. Consideramos un digrafo completo $D = (V, A)$, donde $V = \{0, 1, \ldots, n\}$ es el conjunto de vértices y A es el conjunto de aristas que los une. Cada arista $(i, j) \in A$, $i \neq j$, tiene asociado un costo positivo c_{ij}. El objetivo del problema es encontrar un tour que visite a todos los vértices en V exactamente una vez con un costo total mínimo.

La figura 2.1 muestra un ejemplo de $D = (V, A)$ para $n = 3$. Es un digrafo completo, ya que para cada par de vértices i, j, $i \neq j$, hay un arco (i, j) uniéndolos. Además, el número asociado con cada $(i, j) \in A$ representa el costo c_{ij}.

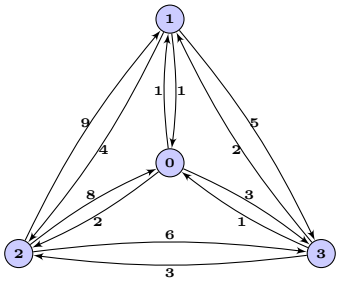


Figura 2.1: Digrafo completo ($n = 3$).

Para formular un modelo PLEM para el ATSP, comenzamos definiendo las variables de decisión. Consideramos la variables binarias x_{ij} para $(i, j) \in A$, en donde $x_{ij} = 1$ si y solo si el vértice j es visitado inmediatamente despues del vértice i en el tour. Luego, el siguiente paso es modelar la función objetivo que es minimizar el costo total del tour y el conjunto de restricciones que define qué es una solución factible para el problema utilizando expresiones lineales. El modelo clásico para el ATSP se define por la formulación PLEM (2.3) - (2.7).

$$\text{mín} \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2.3}$$

$$\text{s.t.} \quad \sum_{j\in V} x_{ij} = 1 \qquad i \in V \tag{2.4}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad j \in V \tag{2.5}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \qquad S \subset V,\ |S| \geq 2 \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad i \in V,\ j \in V \tag{2.7}$$

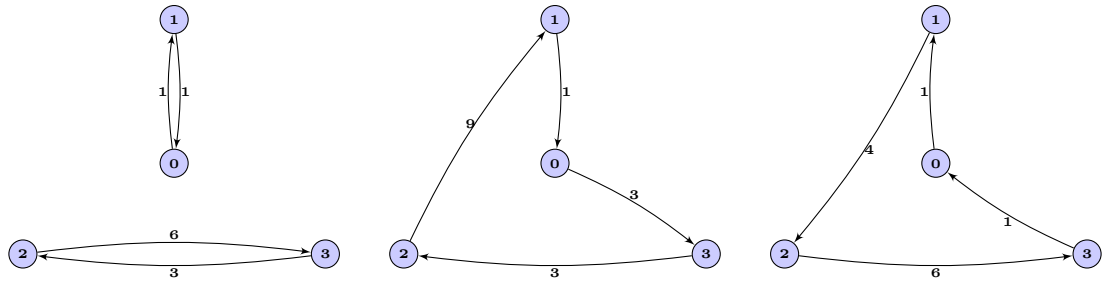
La función objetivo (2.3) minimiza la suma de los costos asociados con los arcos que se utilizan en el tour. Las ecuaciones (2.4) y (2.5) establecen que el viajante tiene que llegar a cada punto y salir de cada punto exactamente una vez, respectivamente. Estas restricciones se conocen como ecuaciones de grado de entrada y de grado de salida. Sin embargo, la imposición de estas dos condiciones

no es suficiente para obtener soluciones factibles para el ATSP, ya que una solución que realice dos subtours separados cumple las restricciones. Luego, se deben considerar la desigualdades de eliminación de subtour para evitar estas situaciones, las cuales se expresan mediante la restricción (2.6). Finalmente, en (2.7) se pueden ver las restricciones de integralidad de las variables de decisión.

Las desigualdades de eliminación de subtours imponen una cantidad exponencial de restricciones, lo cual claramente no puede ser incluído en el modelo explícitamente, ni siquiera para valores muy pequeños de n. Luego, estas restricciones usualmente se consideran como cortes dentro de un algoritmo branch-and-cut, como se explicó en la sección 2.2.3, y son agregadas a medida que sea necesario a la formulación dentro del árbol de búsqueda. Es importante remarcar que la separación de esta familia de desigualdades tiene una complejidad polinomial y en la práctica existen implementaciones eficientes de este algoritmo, como se ve en [49].

La figura 2.2 muestra tres situaciones diferentes relacionadas al ejemplo del ATSP mostrado en 2.1. En la figura 2.2a podemos observar una solución infactible para el ATSP por el mencionado problema de generación de subtours. En general, soluciones que sigan este patrón son resultado del problema de asignación que nace de la conjunción entre la función objetivo (2.3) y las restricciones (2.4), (2.5), junto con $x_{ij} \geq 0$ for $(i, j) \in A$. Esta solución puede prohibirse en el modelo incluyendo alguna de las desigualdades de eliminación de subtour asociadas, por ejemplo, por el conjunto $S = \{2, 3\}$.

Las figuras 2.2b y 2.2c muestran dos soluciones factibles diferentes para el ATSP. La primera corresponde al tour definido por la secuencia de vértices $(0, 1, 2, 3)$ y tiene un costo total de 16. La segunda representa la solución determinada por la secuencia $(0, 3, 2, 1)$ y tiene un costo total de 12. En particular, esta última corresponde a la solución óptima de la instancia descripta en la figura 2.1.



(a) Solución infactible (subtours). Costo: 11.

(b) Solución factible. Costo: 16.

(c) Solución óptima. Costo: 12.

Figura 2.2: Ejemplos relacionados al ATSP

Como se mencionó en la introducción, el TSP (y el ATSP) recibió mucha atención en las últimas décadas. El modelo presentado en este ejemplo es solo una de las varias alternativas que existen para modelar el ATSP. En [47] puede encontrarse una revisión detallada de los diversos modelos conocidos para este problema.

Capítulo 3

Problema de Empaquetamiento con Conflictos Generalizados

3.1. Introducción

En este capítulo introducimos el problema particular que vamos a tratar en este trabajo, el Problema de Empaquetamiento con Conflictos Generalizados (BPGC por sus siglas en inglés).

El problema es una variación del Bin Packing en donde se tienen n contenedores indistinguibles de capacidad W y un conjunto de n ítems. Cada ítem i tiene un valor asociado al que llamaremos *peso* y denotaremos como w_i para todo $i = 1, \ldots, n$. Además, en este problema se cuenta con una lista de conflictos C. Un conflicto (S,max_S) consiste de un subconjunto S de ítems y una cantidad max_S. La restricción de incompatibilidad asociada a cada uno de estos conflictos establece que como mucho max_S ítems del subconjunto S pueden ser asignados juntos al mismo contenedor.

El objetivo entonces es minimizar la cantidad de contenedores necesarios para asignar a todos los ítems. Por un lado se debe asegurar que la capacidad de cada contenedor no sea excedida por la suma de los pesos de los ítems asignados a él. Por otro lado, se deben cuidar las relaciones de conflicto entre ítems para que no haya más ítems asignados juntos de los que deberían.

Hasta donde llega nuestro conocimiento, el Problema de Empaquetamiento con Conflictos Generalizados no ha sido introducido en la literatura. Sin embargo, una variación en donde solo se consideran conflictos entre pares de ítems ha tenido cierto trato en la literatura en las últimas décadas. El Problema de Empaquetamiento con Conflictos (BPC) fue introducido en 1997 en [33] y es el nombre que se le dió en dicho trabajo al caso particular en donde los ítems solo pueden presentar conflictos de a pares, en lugar de cualquier tipo de conflictos entre subconjuntos de ítems. En ese problema, un grafo G es provisto para modelar los conflictos. Cada uno de los ítems representa un vértice en el grafo y se coloca una arista entre cada par de vértices asociados a ítems que no pueden ser asignados juntos.

Aunque, como veremos próximamente, el Problema de Empaquetamiento con Conflictos es útil para modelar diversas situaciones de la vida real, hay otras situaciones en las que no alcanzan los conflictos de a pares. En estos casos se dan situaciones en las que se necesitan conflictos generalizados para captar correctamente la restricción.

El Problema de Empaquetamiento con Conflictos Generalizados puede ser formalmente definido con el siguiente modelo de Programación Lineal Entera. Consideramos las variables binarias x_{ik} y y_k, en donde x_{ik} representa a la asignación del ítem i al contenedor k e y_k representa la utilización del contenedor k. Haciendo uso de estas definiciones para las variables, el siguiente modelo corresponde al Problema de Empaquetamiento con Conflictos Generalizados:

$$
\begin{aligned}
&Minimize && \sum_{k=1}^{n} y_k && && \\
&subject\ to && && && \\
& && \sum_{k=1}^{n} x_{ik} = 1 && \forall i = 1, \ldots, n && (3.1)\\
& && \sum_{i=1}^{n} w_i x_{ik} \leq W y_k && \forall k = 1, \ldots, n && (3.2)\\
& && \sum_{i \in S} x_{ik} \leq max_S && \forall (S, max_S) \in C && \\
& && && \forall k = 1, \ldots, n && (3.3)\\
& && y_k \in \{0, 1\} && \forall k = 1 \ldots n && (3.4)\\
& && x_{ik} \in \{0, 1\} && \forall i, k = 1 \ldots n && (3.5)
\end{aligned}
$$

A continuación explayamos coloquialmente la noción o restricción que captura cada componente del modelo:

- La función objetivo es la suma sobre las variables de utilización de los contenedores. El objetivo es minimizar la cantidad de contenedores utilizados.
- La primera familia de restricciones impone el hecho de que cada ítem debe ser asignado a un solo contenedor.
- La segunda familia se encuentra para restringir el peso total de un contenedor a su capacidad. La suma de los pesos de los ítems asignados a un contenedor dado no puede superar su capacidad W. Además del lado derecho de la restricción vemos a la variable de utilización de contenedor ya que el límite W solo aplica si el contenedor realmente está en uso. De lo contrario, esta familia indica que ningún ítem puede ser asignado a un contenedor no utilizado.
- La siguiente familia predica sobre los conflictos entre los ítems. El conflicto representado por la tupla (S, max_S) indica que de entre todos los ítems presentes en S, a lo sumo max_S pueden ser asignados juntos. De esta manera, la restricción asociada impone esto sumando las variables de asignación sobre todos los ítems de S para cada uno de los contenedores posibles y luego acota dicha suma por max_S.

Es simple ver que este problema pertenece a la clase de complejidad $\mathcal{NP}$-hard ya que es una generalización del Bin Packing Problem. Cualquier instancia del Bin Packing Problem puede ser pensada como una instancia de este problema en la cual la lista de conflictos C está vacía. Si pudiésemos resolver de manera simple el Problema de Empaquetamiento con Conflictos Generalizados para cualquier instancia, entonces podríamos resolver con el mismo nivel de dificultad cualquier instancia del Bin Packing Problem.

Aplicaciones

Existen varias aplicaciones en la literatura en donde se necesitan empaquetamientos con restricciones entre los elementos a asignar. Las aplicaciones a presentar solo consideran conflictos entre pares de ítems a asignar, aunque en la mayoría de ellas la generalización a la situación en donde se necesiten conflictos generalizados y no de a pares resulta natural.

En un primer ejemplo, ya en el año 1979, Christofides et al. en [8] presentan un problema de ruteo en donde buscan la mínima cantidad de vehículos para realizar entregas de ciertos ítems

con propiedades particulares que resultan en conflictos. Por ejemplo, en un mismo vehículo tienen la restricción de no poder asignar un ítem inflamable con uno explosivo. Esto claramente es una situación en donde los conflictos son de a pares, haciendo que sea una situación ideal para el Problema de Empaquetamiento con Conflictos. Sin embargo, la restricción también podría ser que entre todos los productos inflamables no se puede asignar más de cierta cantidad juntos por una cuestión de seguridad. En este caso el grafo de conflictos no puede capturar esta noción y el modelado quedaría incompleto. En ese caso se necesita introducir conflictos generalizados para satisfacer la restricción.

Más aplicaciones que involucran conflictos entre los elementos a asignar se sucedieron en los siguientes años. En [40], Laporte y Desroches presentan un problema de asignación de exámenes a períodos. En dicho problema el objetivo es minimizar la cantidad de períodos de tiempo necesario para realizar todos los exámenes. Sin embargo, se deben considerar dos tipos de restricciones. Por un lado, la cantidad de salones para tomar los exámenes está acotado, lo cual es claramente capturado por la noción de capacidad de un contenedor, que en este caso serían los períodos de tiempo. Por otro lado, los exámenes tienen alumnos en común, por lo cual hay varios exámenes que no puede ser evaluados simultáneamente, lo cual se modela mediante los conflictos entre los ítems a asignar.

Luego, en [32], se resuelve un problema de asignación en computación distribuída. El objetivo es asignar las diferentes tareas a los distintos procesadores, los cuales tienen una capacidad acotada. Además, hay tareas que no pueden ser asignadas juntas a un mismo procesador, ya sea por una cuestión de performance o porque compiten por un recurso particular. Nuevamente este es un caso donde la generalización de los conflictos resulta natural ya que puede pasar que no sean dos tareas las que compitan por un único recursos, si no varias tareas que compiten por una cantidad acotada de recursos.

Otras aplicaciones pueden ser encontradas en [2] y [5]. En estos trabajos también se presentan problemas de calendarización de tareas en procesadores bajo diferentes restricciones operativas. Por último, en [17] y [18] se encuentran aplicaciones parecidas a las anteriores, pero con la característica de que el grafo de conflictos presenta una estructura particular. Por ejemplo, es simple encontrar situaciones en donde los grafos de conflictos asociados a los problemas sean grafos intervalo. Esta situación se da por ejemplo cuando los ítems a asignar son un elemento asociado a un intervalo de tiempo.

3.1.1. Caso de estudio particular

Estudiando el Problema de Empaquetamiento con Conflictos Generalizados nos hemos encontrado directamente con una aplicación al desarrollar una solución para un problema logístico proveniente de una empresa de cosméticos en Argentina.

La empresa cuenta con una planta central en la ciudad de Buenos Aires, capital del país, la cual funciona como central administrativa y productiva. En esta planta se centraliza toda la distribución de cosméticos. Los mismos se empaquetan para ser enviados a diversos puntos de todo el país para su posterior reventa.

La compañía funciona mediante la utilización de revendedores que poseen un catálogo fijo por un período de tiempo. En este período el revendedor busca realizar las ventas pertinentes, las cuales luego son pedidas al depósito central que se las entregará al cierre del período de venta, lo cual usualmente toma 15 días. La compañía tiene seccionado el mapa del país en zonas y cada revendedor se encuentra asociado a una zona de influencia. Por otro lado, una zona puede tener muchos revendedores asociados. De esta manera, al finalizar el período de venta la compañía sabe perfectamente la demanda de cada una de las zonas, lo cual no es más que la sumatoria de las demandas de las ventas que lograron realizar los revendedores asociados a la zona. Sin embargo, la compañía no realiza los envíos directamente desde el depósito central a cada zona, sino que en realidad tiene varios depósitos intermedios por diferentes lugares del país. Cada uno de estos depósitos se utiliza para albergar los cosméticos de un cierto conjunto de zonas asociadas geográficamente al depósito. La tarea del depósito central es satisfacer la demanda de cada uno de los depósitos mediante la entrega de cosméticos con una flota de camiones prefijada. Una vez

entregados los cosméticos en cada depósito, cada uno de ellos satisfacerá la demanda de cada zona de manera particular utilizando su propia flota de vehículos pequeños.

Por restricciones operativas de la empresa, la demanda de cada zona no puede ser dividida y debe ser satisfecha en un solo camión. Como un primer paso de un problema de calendarización complejo, debemos encontrar la mínima cantidad de camiones necesaria para asignar las demandas de todas las zonas. Este subproblema sería la situación típica para el problema de Bin Packing, en donde cada zona es un ítem a ser asignado con un peso asociado que es la demanda en litros de cosméticos y los camiones son los contenedores que tienen una cierta capacidad límite. Sin embargo, los depósitos tienen restricciones adicionales relacionadas a aspectos de capacidad física, administrativa y operativa que hacen que sea imposible entregar todas las zonas correspondientes al mismo tiempo. Por ejemplo, hay depósitos que deben manejar los cosméticos correspondientes a diez zonas pero, dadas las restricciones mencionadas, solo puede recibir hasta tres zonas a la vez. Luego, el problema se convierte en una situación modelable mediante nuestro Problema de Empaquetamiento con Conflictos Generalizados en donde tenemos conflictos entre conjuntos de zonas. Para cada depósito, tenemos un conjunto de zonas S, para las cuales se sabe que a lo sumo una cierta cantidad, max_S, de ellas podrá ser asignada al mismo camión.

El Problema de Empaquetamiento con Conflictos Generalizados mostró ser una formulación correcta para este subproblema, la cual fue utilizada en una primera etapa de una posterior solución integral de logística que actualmente está siendo utilizada por la empresa.

3.2. Trabajos relacionados

Si bien el Problema de Empaquetamiento con Conflictos Generalizados no tiene un trato particular en la literatura, el Problema de Empaquetamiento con Conflictos sí tiene algunos trabajos dedicados a él durante las últimas décadas, cubriendo diferentes aspectos del problema. Esto resulta de interés ya que el Problema de Empaquetamiento con Conflictos es un caso particular de nuestro problema por lo que los resultados que se consigan para dicho problema tienen la posibilidad de ser generalizados.

En primer lugar, desde el punto de vista de algoritmos aproximados, el problema despertó cierto interés ya que, como mencionamos anteriormente, el Bin Packing Problem admite aproximaciones y heurísticas que se puede demostrar que acotan de manera constante al problema. En [32] y [33] se muestra que, cuando se consideran grafos de conflicto generales, no es posible desarrollar un esquema de aproximación de cota polinómica. En [16] se presentan cotas inferiores para el problema y esquemas de aproximación para ciertas clases de grafos de conflicto particulares. En particular, en dicho trabajo desarrollan un algoritmo aproximado con cota constante para grafos de conflicto perfectos. Luego, mejoran la cota obtenida particularizando el algoritmo de aproximación sobre algunas subclases de grafos perfectos, como los grafo intervalo.

Desde el punto de vista heurístico, tenemos el trabajo de Kalfakakou et al. [35] que introduce el problema de minimizar la cantidad de depósitos para albergar productos que conflictúan. En este trabajo los autores presentan una heurística de costo computacional elevado en la que intentan encontrar buenos subconjuntos candidatos para asignar a un mismo depósito. Esto lo hacen realizando una búsqueda de subconjuntos independientes del mayor tamaño posible en el grafo de incompatibilidades.

Luego, Gendreau et al. en [22] presentan un trabajo completo de estudio de heurísticas constructivas del Problema de Empaquetamiento con Conflictos, junto con dos cotas inferiores. Una de las cotas es una simple adaptación de una cota presentada en [43] para el Bin Packing Problem. Por otro lado, la segunda cota presentada está desarrollada particularmente para el Problema de Empaquetamiento con Conflictos y entrega un resultado más ajustado, utilizando fuertemente que los conflictos son de a pares. En cuanto a las heurísticas, en el trabajo se presentan 6 heurísticas constructivas diferentes, yendo desde simples adaptaciones de las heurísticas clásicas para el Bin Packing Problem para tener en cuenta el grafo de conflictos, hasta la última heurística que realiza búsquedas de clique en el grafo conflicto y en su complemento para generar soluciones primales de calidad. Para finalizar, se presentan los resultados computacionales obtenidos con las diferentes

heurísticas sobre un conjunto de instancias para el Problema de Empaquetamiento con Conflictos, generadas a partir de instancias clásicas de Bin Packing, y agregando luego el grafo conflicto.

Algunos años despues, Basnet et al. en [3] y Maiza et al. [41] presentan más aproximaciones heurísticas para el Problema de Empaquetamiento con Conflictos. La principal diferencia entre el trabajo de Gendreau y el resto de los mencionados es que el primero es el único que presenta procedimientos simples con una complejidad computacional polinomial baja. Los otros trabajos desarrollaron diferentes aproximaciones con un alto consumo de tiempo de cómputo, usualmente dados al resolver un problema $\mathcal{NP}$-hard hasta un cierto límite de tiempo o una cierta profundidad en el proceso de búsqueda. El aspecto más recalcable de las heurísticas desarrolladas en el trabajo de Gendreau es que dan buenos resultados en las instancias experimentadas con procedimientos de baja complejidad.

El Problema de Empaquetamiento con Conflictos también tiene algunos trabajos desde un enfoque exacto en la literatura. En el año 2010, Muritiba et al. [44] presentan el primer trabajo con un enfoque de Programación Lineal Entera. Luego, Elhedhli et al. [15] extienden dicho trabajo. Por último, Sadykov et al. [52] presentan un tercer trabajo con un enfoque de Programación Lineal Entera en el que hacen hincapié en ciertas componentes particulares del algoritmo para obtener mejores resultados. Además, desarrollan un caso particular del algoritmo para cuando el grafo de conflictos es un grafo intervalo. Todos estos trabajos no ahondan sobre el modelo presentado al introducir nuestro problema, particularizado con conflictos de a pares, si no que trabajan sobre la reformulación obtenida mediante la descomposición de Dantzig-Wolfe de dicho modelo. De esta manera, los modelos trabajados poseen una alta cantidad de columnas, lo cual lleva a desarrollar algoritmos del tipo branch-and-price para su resolución. En particular, la condición del grafo de conflictos de ser un grafo intervalo resulta en un problema esclavo simple de resolver para agregar columnas al modelo, por lo cual hacen hincapié en dicha situación.

Todos los trabajos mencionados tratan el Problema de Empaquetamiento con Conflictos, el cual solo tiene en cuenta los conflictos entre pares de ítems. Dado que durante este trabajo vamos a desarrollar heurísticas, resultados poliedrales y un algoritmo branch-and-cut para un problema más general, *a priori* podría resultar alentador la comparación de nuestros desarrollos en este problema particular, con el fin no solo de utilizar los trabajos anteriores sino también las instancias que fueron creadas oportunamente. Sin embargo, esta comparación no la consideramos adecuada ya que tanto los algoritmos como las instancias desarrolladas en dichos trabajos, hacen un muy fuerte uso de la existencia de conflictos entre pares de ítems que no es generalizable a nuestro problema de manera natural.

Por el lado de algoritmos heurísticos, tenemos 3 trabajos con resultados destacables. El primero, como ya mencionamos, corresponde a Gendreau et al. [22]. En la heurística que presenta mejores resultados, los autores buscan cliques en el grafo conflicto, ya que todos los elementos de una clique deben ser asignados a contenedores diferentes. Luego, por cada ítem de la clique, buscan una clique que contenga a dicho ítem en el grafo complemento. Todas estas son ideas que no generalizan cuando en vez de tener conflictos de a pares se tienen conflictos generalizados. En primer lugar, necesitamos definir qué es una clique cuando hablamos de conflictos generalizados. Esto se hará posteriormente en nuestro trabajo, pero no inducirá al fuerte hecho de que todos los ítems de la clique deban ser asignados a diferentes contenedores, por lo que la idea principal de la heurística no podrá ser utilizada. También se podrá definir el complemento de los conflictos generalizados, lo cual tampoco irá en concordancia con la idea que se utilizó para desarrollar esta heurística.

En segundo lugar, tenemos el trabajo de Muritiba et al. [44] que además de presentar un enfoque exacto, también presenta heurísticas constructivas y una metaheurística tabú. La metaheurística tabú es la que muestra mejores resultados. En particular, en dicho trabajo utilizan esta metaheurística para lograr reducir en un contenedor una asignación dada. Es decir, teniendo una asignación que utiliza $k+1$ contenedores, la metaheurística se esfuerza en buscar una asignación de k contenedores. Para lograr este fin, el algoritmo busca tener asignaciones factibles en los primeros k contenedores y una asignación posiblemente infactible en el contenedor $k+1$ la cual se irá arreglando iterativamente. En particular, para buscar ir haciendo estos arreglos, toma un ítem

i del contenedor $k+1$ y lo mueve a algún contenedor h. Al elegir este contenedor h, es posible que haya ítems que conflictúen con i, por lo cual los remueven. Al tener conflictos de a pares es simple y está unívocamente determinado quiénes son los ítems que conflictúan. Al tener conflictos generalizados, existen diferentes subconjuntos de ítems que se pueden remover para que no haya conflictos activos con el ítem i. El hecho de que existan muchos subconjuntos ya presenta una decisión más para el algoritmo que agranda abruptamente el espacio de exploración. Ni siquiera es tan simple tomar una única decisión para no considerar todas las opciones de subconjuntos. Si, por ejemplo, se tomase la decisión de sacar la mínima cantidad de ítems tal que ya no haya conflictos con i, entonces este problema se correspondería con el *Set Cover Problem*, el cual pertenece a la clase de complejidad $\mathcal{NP}$-hard. De esta manera, se ve que la heurística hace un fuerte uso del hecho de que los conflictos son de a pares, y su generalización no solo no sería trivial, sino que perdería la fortaleza que la situación particular otorga.

Por último, tenemos el trabajo de Sadykov et al. [52]. En este trabajo se presenta una heurística denominada *Pure Diving*, en la cual se realiza un recorrido en profundidad del árbol de branch-and-price para conseguir una solución primal de calidad. En esta heurística también se hace un fuerte uso de la particularidad de los conflictos de a pares, conjuntamente con la particularidad que presentan las instancias utilizadas en dicho trabajo. Este uso de los conflictos de a pares y de las instancias particulares sucede, a grandes rasgos, en dos aspectos diferentes. Por un lado, el *branching* lo realizan sobre las variables de su modelo, las cuales representan asignaciones posibles de ítems a contenedores. Al tener conflictos de a pares, y por la naturaleza de las instancias que utilizan, la cantidad de columnas es de orden polinomial por lo que al realizar una búsqueda en profundidad sobre el árbol de branch-and-price, no toma tanto tiempo encontrar una solución de calidad. En el caso de tener conflictos generalizados, la cantidad de columnas posibles sería mucho mayor lo cual seguramente iría en detrimento de la performance de un algoritmo basado en una búsqueda en profundidad. Por otro lado, para hacer la búsqueda en profundidad en el árbol de branch-and-price, lógicamente, deben ir resolviendo cada nodo del árbol. El hecho de que solo haya conflictos de a pares resulta en un problema esclavo de estructura particular (específicamente, el Problema de la Mochila con Conflictos) lo cual facilita la resolución. Esta estructura específica también se perdería al tener conflictos generalizados.

Por el lado de los algoritmos exactos, los tres trabajos hacen uso de la situación particular de los conflictos de a pares por los mismos motivos que explayamos con la heurística de [52]. Dado los conflictos de a pares, y la forma de generar las instancias particulares, la cantidad de columnas total en dichos trabajos es de orden polinomial, lo cual no sucederá al utilizar conflictos generalizados. Si aún así se quisiese aplicar un algoritmo branch-and-price, el problema esclavo no tendría la estructura específica de un Problema de la Mochila con Conflictos.

Como mencionamos anteriormente, tampoco consideramos adecuadas las instancias utilizadas en dichos trabajos. Si tomamos el trabajo de Sadykov et al. [52], el último desde un punto de vista exacto, se presentan 6 tipos de instancias u, t, d, ua, da, ta. Las instancias resultan bastante particulares aún teniendo en cuenta que son para la versión del problema con conflictos de a pares. En primer lugar, los grupos de instancias que no poseen la letra a como sufijo, es porque están asociadas a un grafo conflicto intervalo. Esto caracteriza fuertemente dichas instancias ya que el problema de coloreo en grafos intervalo es polinomial y se puede hacer uso de la estructura de dichos grafos para generar algoritmos con mejor performance que uno general. Por otro lado, los conjuntos de instancias que tienen como prefijo u o t, son instancias en las que en promedio entran 3 ítems por contenedor. Considerando que los trabajos exactos mencionados desarrollan un algoritmo branch-and-price, esto resulta bastante peculiar porque en dichas instancias la cantidad de columnas posibles es de orden polinomial. Por último se encuentran las instancias da. Estas instancias presentan un grafo conflicto aleatorio y poseen la propiedad de que en promedio deberían entrar unos 8 ítems por contenedor. Esto también resulta particular ya que se tienen por un lado los conflictos de a pares, y por el otro conflictos, en promedio, de conjuntos de 9 ítems. Esta disparidad entre el orden de las restricciones, será objeto de análisis en secciones posteriores de este trabajo. Es por todo lo dicho que desestimaremos la utilización de estas instancias particulares para la experimentación de nuestro problema más general.

3.3. Relación con el Problema de Coloreo en Hipergrafos

El Problema de Empaquetamiento con Conflictos Generalizados tiene una estrecha relación con el Problema de Coloreo en Hipergrafos. El Problema de Coloreo en Hipergrafos puede tener diferentes definiciones ya que por ejemplo se podría exigir que en una hiperarista todos los vértices tengan un color distinto, o también se podría exigir que no todos los vértices tengan el mismo color. Esta última es justamente la noción de coloreo en hipergrafos que tomaremos. Es decir, el Problema de Coloreo en Hipergrafos consiste en encontrar la mínima cantidad de colores necesarios para colorear todos los vértices de un hipergrafo, de manera tal que ninguna de sus hiperaristas quede monocromática. En [6] se puede encontrar esta definición de coloreo, junto con muchas de las primeras nociones fundamentales sobre hipergrafos. En dicho libro también se enuncian varios resultados teóricos sobre el problema de coloreo, relacionados a cotas y caracterización de coloreos en subclases particulares de hipergrafos.

En el Problema de Empaquetamiento con Conflictos tenemos restricciones sobre conjuntos de ítems. Estas restricciones constan de un subconjunto S y un valor máximo max_S de ítems que pueden ir juntos. Dicho conjunto S tiene un cierto cardinal y max_S podría ser un número bastante menor a ese cardinal. Por ejemplo, una restricción podría predicar sobre un conjunto de siete elementos diciendo que a lo sumo tres de ellos pueden ser asignados juntos. Si bien en la práctica esto es una sola restricción, en el plano teórico podríamos convertir esta restricción en muchas otras en donde el cardinal de S y max_S solo difieran en uno. En el ejemplo dado, decir que de esos siete elementos solo podemos asignar tres juntos, es análogo a decir que para todo conjunto de cuatro elementos dentro del original, solo podremos asignar a lo sumo tres juntos. De esta manera, podemos convertir la restricción original en varias restricciones que, en conjunto, caractericen el mismo espacio de soluciones factibles. La particularidad de las nuevas restricciones es que ahora sí estas restricciones entran en consonancia con el Problema de Coloreo en Hipergrafos ya que cada conjunto S correspondería exactamente a una hiperarista que no queremos colorear monocromáticamente.

Por otro lado, el Problema de Empaquetamiento con Conflictos Generalizados presenta justamente restricciones de empaquetamiento. Estas restricciones también se pueden pensar como restricciones de hiperarista en un hipergrafo. Por cada subconjunto minimal de ítems que excedan su peso total con respecto a la capacidad de los contenedores, se puede pensar que se tiene una hiperarista que no se quiere colorear monocromáticamente.

Para realizar una traducción directa, podemos pensar que los vértices del hipergrafo son los ítems a asignar en el Problema de Empaquetamiento con Conflictos Generalizados. Las hiperaristas del hipergrafo surgen de dos formas diferentes. Hay hiperaristas que representan los conflictos generalizados entre los ítems, y hay hiperaristas que representan la restricción de capacidad. La asignación de un color a un vértice tiene la misma connotación que la asignación de contenedor a un ítem.

De esta manera, se ve que existe una relación entre los dos problemas que será utilizada a lo largo del trabajo, por lo cual muchas veces nos referiremos a los conflictos como hiperaristas de un hipergrafo H asociado al problema. Si bien el Problema de Coloreo en Hipergrafos tiene literatura asociada, no hemos encontrado ningún trabajo relacionado a un enfoque exacto desde Programación Lineal Entera para su resolución. En general son descripciones de casos particulares y generación de resultados relacionados a cotas y posibles aproximaciones.

Capítulo 4

Heurísticas

En este capítulo introduciremos varias heurísticas para el Problema de Empaquetamiento con Conflictos Generalizados, con el objetivo de lograr cotas primales de calidad en tiempos acotados. En el marco de un algoritmo branch-and-cut las heurísticas juegan un papel muy importante por diversos motivos. En primer lugar, más allá de la búsqueda de una solución óptima conjuntamente con su certificado de optimalidad, en muchas aplicaciones reales puede ser tan o más importante tener una solución de calidad de forma temprana que lograr la solución óptima. Por otro lado, como fue explicado en el capítulo de conceptos preliminares, las cotas primales serán las responsables de lograr buenas podas en el árbol de búsqueda. Además, en el caso de nuestro problema, las heurísticas iniciales acotan la cantidad de contenedores a involucrar, lo cual también va a tener un impacto directo sobre el tamaño del problema a resolver.

Cabe recalcar que nuestro objetivo fue desarrollar algoritmos heurísticos no como una propuesta de solución algorítmica del problema en sí misma, sino como una herramienta que fundamentalmente fuese rápida y brindara soluciones de aceptable calidad dentro del contexto de un branch-and-cut.

En primer lugar, describiremos algunas heurísticas que son adaptaciones naturales de algoritmos que usualmente son utilizados para el problema de Bin Packing. Dichos algoritmos se encuentran bien detallados en [43] en donde, además de presentarlos, se demuestra que tienen un buen desempeño para el caso donde no hay conflictos. Por otro lado, algunos de estos algoritmos fueron adaptados por Gendreau et al. en [22] y por Muritiba et al. en [44] para el caso de conflictos de a pares. Esto hace que parezca una decisión razonable ver cómo se comportan en nuestra variación del problema.

Estos procedimientos construyen una solución de forma incremental. Comienzan con todos los contenedores vacíos y todos los ítems sin ubicar. En cada paso consideran un ítem aún no asignado y lo colocan en un contenedor factible. El procedimiento termina cuando se han asignado todos los ítems. El orden en que los ítems son considerados y la elección del contenedor donde ubicar cada uno definen las diferentes versiones del método.

Las adaptaciones de las propuestas que se encuentran en la literatura no son capaces de reflejar los distintos niveles de conflicto que se presentan en nuestro problema, perdiendo información que resulta relevante. Esto nos incentivó a definir una nueva noción de grado de conflictividad de un ítem que logra capturar mejor la esencia de la generalización que presentamos en esta tesis, dando así origen a nuevas versiones del procedimiento.

Luego, describiremos dos metaheurísticas basadas en Búsqueda Local. La primera de ellas consiste en un procedimiento de Hill Climbing, mientras que la segunda corresponde a la aplicación de la técnica de Simulated Annealing.

4.1. Heurísticas Constructivas

4.1.1. NextFitWC y FirstFitWC

Las dos primeras heurísticas a presentar son simples adaptaciones de las conocidas heurísticas Next Fit y First Fit para el Bin Packing Problem. En [43] se puede encontrar su definición y un estudio de la calidad de las aproximaciones que arrojan.

En la heurística Next Fit With Conflicts (NextFitWC), se consideran los ítems a asignar en algún orden dado. El algoritmo comienza con el primer contenedor vacío y los ítems en un orden arbitrario. Luego, por cada ítem que se va considerando, si se puede asignar al contenedor actual, se hace la asignación. En caso de no poder ingresar el ítem al contenedor actual, dicho contenedor se cierra y el siguiente contenedor se toma en consideración. Dado que todos los contenedores tienen las mismas condiciones y, por lo tanto, son indistinguibles, la elección de los contenedores durante el proceso de asignación es irrelevante.

En la heurística First Fit With Conflicts (FirstFitWC) la base es similar a la anterior, pero en vez de tratar de hacer la asignación al contenedor actual, debemos chequear todos los contenedores ya utilizados en orden y asignar el ítem al primer contenedor que se pueda. Al igual que en la heurística anterior, si ningún contenedor es capaz del albergar al ítem en cuestión, un nuevo contenedor debe abrirse.

La mayor diferencia entre estas versiones de las heurísticas y las versiones aplicadas al Bin Packing Problem es que en el caso de tener conflictos dos aspectos deben ser chequeados para decidir si un contenedor puede albergar un ítem particular. Al igual que en el Bin Packing Problem, debemos chequear si la capacidad residual del contenedor es mayor o igual al tamaño del ítem que estamos tratando de asignar. Además en este caso, también tenemos que chequear si el ítem tiene algún conflicto con algún subconjunto de ítems que ya se encuentre asignado al contenedor.

Es un hecho bien conocido que las heurísticas Next Fit y First Fit para el Bin Packing Problem casi siempre tienen un peor desempeño que las versiones de dichos algoritmos en donde se consideran los ítems en un orden particular y no en uno arbitrario. De hecho, en [43] se puede ver la demostración de que las cotas de aproximación son mucho más ajustadas. Sin embargo, la comparación entre las versiones mencionadas no necesariamente se traduce de manera análoga al Problema de Empaquetamiento con Conflictos Generalizados, por lo que estas dos heurísticas resultan algoritmos aceptables para experimentar, al menos como línea de base para comparar mejores alternativas.

De hecho, para dar una noción de cómo se pierde la traducción de resultados de un problema a otro, podemos ver que las heurísticas aquí presentadas no se pueden acotar en cuanto a la calidad de solución que arrojan. Supongamos que tenemos n ítems en donde cada uno de ellos tiene un tamaño igual a $\frac{2W}{n}$. Además, los conflictos generalizados de estos ítems son simplemente conflictos de a pares que forman un camino. Es decir, podemos pensar que los conflictos en esta instancia se pueden modelar con un grafo P_n. Un grafo P_n tiene n vértices y las únicas aristas que posee son las que une a cada vértice con su inmediato anterior e inmediato posterior, en caso de que existan. En este caso, si tomamos los ítems en orden del camino, cada vez que queramos insertar a un ítem con la heurística Next Fit With Conflicts, nos encontraremos con un contenedor que alberga al ítem anterior del camino, por lo que existirá un conflicto que resultará en la apertura de un nuevo contenedor. Luego, la solución arrojada por la heurística utilizará n contenedores, cuando en realidad existe una solución muy simple con dos contenedores que surge de poner a los ítems impares del camino en un contenedor y a los ítems pares en el otro. Además, esta instancia de características negativas para el NextFitWC se puede armar para cualquier valor de n y el valor óptimo siempre seguirá siendo dos. Demostrando así que no puede haber ninguna cota constante para el desempeño de la heurística.

El mismo procedimiento se puede realizar para demostrar que la heurística FirstFitWC no tiene una cota constante. En este caso, no se puede utilizar el grafo de conflictos P_n ya que la heurística entregaría satisfactoriamente la respuesta óptima. Sin embargo, si utilizamos un grafo bipartito completo al que se le remueven las aristas de un matching máximo, entonces lograremos un caso mal condicionado. Supongamos entonces nuevamente n ítems, con n par para simplificar,

en donde cada uno tiene un tamaño $\frac{2W}{n}$. El grafo conflicto de la instancia será el mencionado, un bipartito completo en donde cada bipartición tendrá $\frac{n}{2}$ ítems y en donde se remueven las aristas de un matching perfecto. Luego, tomaremos el orden de los ítems según las parejas del matching perfecto. Es decir, se irán considerando secuencialmente cada una de las parejas del matching.

En esta instancia, es simple ver que la solución óptima es dos, asignando cada bipartición a un contenedor. Sin embargo la heurística utilizará un contenedor nuevo para cada pareja del matching, utilizando así un total de $\frac{n}{2}$ contenedores. Nuevamente esto se puede hacer para cualquier valor de n, demostrando que no existe una cota constante para este algoritmo heurístico.

De acuerdo a nuestra experimentación preliminar, FirstFitWC siempre mostró comportamiento superior a NextFitWC. Dado que estas heurísticas básicas son tomadas como referentes para comparar otras alternativas, en los reportes solo consideraremos la versión FirstFitWC.

4.1.2. FirstFitDecreasingWC

El objetivo de esta heurística es analizar si las mejoras del ordenamiento que se observan en el Bin Packing Problem se trasladan a nuestro Problema de Empaquetamiento con Conflictos Generalizados. El algoritmo en sí es igual a la heurística FirstFitWC, pero en este caso se consideran los ítems en un orden específico. El orden se determina mediante los tamaños de los ítems, ordenándolos de forma decreciente. Esta versión también fue utilizada en [22] y en [44] como una de sus alternativas para el caso de conflictos de a pares.

El principio de esta heurística radica en el hecho de que usualmente es más fácil asignar ítems más pequeños a contenedores que ya se encuentren en uso, a hacer dicha asignación pero con los ítems grandes. Esto pasa porque es más simple asignar los ítems pequeños en el poco espacio residual que dejan los ítems grandes en cada uno de los contenedores utilizados. En cambio, si muchos ítems pequeños fueran asignados al primer contenedor en los primero pasos, entonces un ítem grande no podría ser asignado a dicho contenedor y tendrían que necesitar un contenedor nuevo, usualmente llevando a una peor solución.

De todas maneras, en esta versión del problema también hay que chequear los conflictos entre los subconjuntos de ítems, por lo que no queda claro que el mejor desempeño de la versión ordenada de la heurística se preserve en esta situación.

4.1.3. FirstFitDecreasingDegreeWC

En esta heurística presentamos otra opción para ordenar los ítems a asignar. En el Problema de Empaquetamiento con Conflictos Generalizados no solo hay que prestar atención a las capacidades, sino también a los conflictos. Es por esto que la misma idea del ordenamiento se puede aplicar a la cantidad de conflictos de un ítem. Así como argumentamos que usualmente es más fácil asignar ítems más pequeños más tarde, una sospecha similar se puede aplicar para pensar que es posible que sea más simple asignar los ítems menos conflictivos más tarde que los más conflictivos. Una vez que los ítems con mayor cantidad de conflictos son asignados, debería ser más simple encontrarles un lugar a los ítems que no presentan tantos conflictos. Luego, esta heurística es la misma que First Fit With Conflicts, pero tomando en cuenta el orden de los ítems a asignar según la cantidad de conflictos en los que están involucrados.

4.1.4. FirstFitDecreasingWC + FirstFitDecreasingDegreeWC

En [44], Muritiba et al. presentan una combinación basada en las dos propuestas anteriores. Para cada ítem i, se calcula un peso subrogado $w_i^s = \alpha\frac{w_i}{\bar{w}} + (1-\alpha)\frac{d(i)}{\bar{d}}$ donde $\bar{w}$ es el peso promedio, $d(i)$ el grado del ítem i en el grafo conflicto (cantidad de conflictos en los que el ítem i está involucrado) y $\bar{d}$ el grado promedio. Variando en el intervalo $[0, 1]$ el parámetro α es posible ponderar la importancia que se le quiere dar a cada criterio. Los ítems son tomados en orden decreciente de su peso subrogado. Los casos en que α toma valor 1 ó 0 corresponden a las dos heurísticas mencionadas anteriormente.

4.1.5. FirstFitDecreasingConflictividadWC

En el caso de conflictos de a pares, la conflictividad de un ítem está fuertemente asociada al grado del ítem en el grafo conflicto. Sin embargo, en nuestro caso de conflictos generalizados, no solo es relevante la cantidad de conflictos en los que está involucrado dicho ítem, sino también el tamaño de los mismos. Por ejemplo, no tiene el mismo impacto la restricción impuesta por un conflicto entre 9 ítems que entre 3. Esto nos sugiere la siguiente nueva definición de conflictividad. Comenzamos definiendo el costo de un conflicto de tamaño k. Esto lo haremos en función de la cantidad de posibilidades de satisfacer el mismo con 2 contenedores (mínima cantidad necesaria de contenedores para ubicar los ítems del conflicto), es decir $\frac{\sum_{j=1}^{k-1}\binom{k}{j}}{2}$. Entonces, el costo de un conflicto de tamaño k, $CostoConf_k$, se define como el inverso de dicho valor, reflejando que a mayor cantidad de posibilidades menor es la restricción que impone el conflicto.

Usando esta noción, definimos $dc(i)$, la conflictividad de un ítem i, como la suma de los costos de los conflictos en los que está envuelto. Es decir:

$$dc(i) = \sum_{S \in C, i \in S} CostoConf_{|S|}$$

Notar que, en el caso de conflictos de a pares, la noción de conflictividad coincide con la noción de grado.

En el proceso de asignación de contenedores, los ítems son tomados en orden decreciente respecto a su conflictividad.

4.1.6. FirstFitDecreasingWC + FirstFitDecreasingConflictividadWC

Siguiendo el mismo modelo expuesto anteriormente, combinamos la conflictividad y el peso de un ítem. Para cada ítem i, se calcula un peso subrogado $w_i^s = \alpha \frac{w_i}{\bar{w}} + (1-\alpha)\frac{dc(i)}{\bar{dc}}$ donde $\bar{w}$ es el peso promedio, $dc(i)$ el grado de conflictividad y $\bar{dc}$ el grado de conflictividad promedio. Nuevamente los ítems son tomados en orden decreciente según este peso subrogado.

4.1.7. FirstFitOrdenDinamicoWC

En general, las heurísticas secuenciales suelen tener un buen comportamiento cuando el próximo elemento a considerar se elige en forma dinámica (ver [25]), en lugar de seguir un orden preestablecido desde el inicio. En este sentido, nuestra propuesta es elegir en cada iteración aquel ítem aún no ubicado que tenga la mayor cantidad de contenedores prohibidos en la solución parcial obtenida hasta el momento.

4.2. Hill Climbing

Dada una instancia de nuestro problema, una vez que se tiene una asignación factible de los ítems a los contenedores, es relativamente simple generar diferentes asignaciones que no disten demasiado de la asignación previa. Esta es una propiedad muy importante para tener la posibilidad de generar cualquier tipo de esquema de búsqueda local para el problema. El propósito de las metaheurísticas de búsqueda local es poder definir una buena vecindad para ser explorada, buscando una mejor solución. Una buena vecindad debe ser una lo suficientemente grande como para contener soluciones de buena calidad, pero a su vez, debe ser lo suficientemente simple para poder ser explorada en límites de tiempo pequeños, dado que usualmente toma varios pasos moverse de una solución inicial a un óptimo local.

Nuestra primera aproximación para utilizar un procedimiento de búsqueda local es definir un algoritmo del tipo Hill Climbing, en el cual comenzaremos de una solución factible y trataremos de mejorarla explorando todas las soluciones presentes en la vecindad definida. En cada paso, se actualizará la solución actual con la mejor solución explorada en la vecindad, y se repetirá el procedimiento desde esta nueva solución hasta que se encuentre un óptimo local, es decir una

solución que no puede ser mejorada por ninguno de sus vecinos. Para poder desarrollar el algoritmo Hill Climbing, debemos definir sus componentes principales.

Solución inicial Como mencionamos anteriormente, la intención de un algoritmo Hill Climbing es ir iterativamente mejorando la solución actual. Luego, antes de poder definir como se van a generar mejores soluciones, esta heurística necesita ser inicializada con una solución factible. Diferentes soluciones iniciales pueden resultar en grandes variaciones de los resultados finales, ya que pueden converger a óptimos locales muy diferentes. En general, es una buena práctica comenzar desde una solución de buena calidad, para luego tratar de mejorarla. En este caso, comenzaremos con la solución inicial arrojada por la heurística FirstFitDecreasingConflictividadWC, ya que es computacionalmente barata para ejecutar y en los experimentos computacionales mostró un buen comportamiento.

Vecindad La segunda componente principal de una procedimiento de búsqueda local es la definición de la vecindad de una solución, en donde buscaremos las mejoras. Para nuestro problema, generaremos los vecinos de una solución a partir de tomar un ítem y reasignarlo a todo contenedor posible. Los contenedores posibles para el ítem serán aquellos tales que al poner el nuevo ítem no se exceden de la capacidad total, y donde no se viola ningún conflicto por realizar la asignación. Por la naturaleza de las restricciones, no necesitamos realizar ningún tipo de chequeo sobre el contenedor del que estamos removiendo el ítem, ya que si partimos de una solución factible, la remoción de un ítem nunca puede generar que se viole una restricción.

Función objetivo Por último, debemos definir una función objetivo para comparar la solución actual con cada uno de sus vecinos. La primera idea lógica que se podría barajar es la de utilizar la misma función objetivo que la planteada en el problema, ya que es esta función la que en el fondo queremos optimizar. Sin embargo, esto no siempre resulta en la mejor opción para un procedimiento de búsqueda local. En el Problema de Empaquetamiento con Conflictos Generalizados, la función objetivo solo admite valores enteros ya que es exactamente la cantidad de contenedores utilizados en la asignación. Si usásemos la misma función objetivo para el algoritmo de Hill Climbing, sería muy dificultoso conseguir mejoras. Si tomamos una solución cualquiera y exploramos la vecindad que definimos anteriormente, es muy posible que nunca difieran en la cantidad de contenedores utilizados. Solamente encontraremos una mejor función objetivo en el caso de que tengamos un contenedor con un único ítem que puede ser reasignado a otro contenedor que ya se encuentra utilizado. Por lo tanto, esta no es una muy buena función objetivo ya que si la solución inicial tuviera más de un ítem en cada contenedor, entonces el algoritmo terminaría en la primera iteración sin conseguir ninguna mejora.

Para evitar el problema que acabamos de mencionar, nos resultará útil definir una nueva función objetivo que permita al algoritmo moverse entre diferentes soluciones hasta llegar a un óptimo local. Para nuestro problema, maximizaremos la siguiente función objetivo

$$\frac{\sum_{i=1}^{n} l_i^2}{b} \tag{4.1}$$

en donde b es el número de contenedores utilizados y l_i es la suma de los tamaños de los ítems asignados al i-ésimo contenedor.

En primer lugar, notamos que esta función es consistente con la original en el sentido de que si tenemos una menor cantidad de contenedores, entonces esta nueva función tendrá mayor valor. La principal diferencia entre las dos funciones objetivos, es que esta nueva considera la carga actual de los contenedores. Dado que la función suma sobre los cuadrados de las cargas de los contenedores, si dos soluciones tienen la misma cantidad de contenedores, la asignación más desbalanceada es elegida. Es preferible tener asignaciones más desbalanceadas dado que esto llevará a tener contenedores casi completamente llenos y otros casi vacíos. Después de algunas

iteraciones, sería esperable encontrarse con el caso en donde un contenedor es completamente vaciado resultando en una asignación que utiliza menos contenedores.

4.3. Simulated Annealing

La última heurística que presentaremos es un algoritmo basado en la metaheurística de Simulated Annealing (SA). La técnica de Simulated Annealing es una reconocida metaheurística dentro de la familia de los procedimientos de búsqueda local. Funciona de manera similar a la técnica de Hill Climbing recientemente mencionada, pero con la posibilidad de evitar quedar estancado en ciertos mínimos locales que no representan tan buenas soluciones. La mayor diferencia entre Simulated Annealing y Hill Climbing radica en la posibilidad de ir hacia una solución en la vecindad que tenga peor función objetivo que la actual.

Tener la opción de moverse a una solución peor que la actual, usualmente funciona muy bien para evitar caer en mínimos locales, con el objetivo de poder ir en futuras iteraciones a soluciones que son aún mejores.

El método comienza con una solución inicial *xactual* y un parámetro T denominado *temperatura*. En primer lugar, se selecciona de manera aleatoria una solución vecina *xvecina* y se evalúa la función objetivo. Si la nueva función objetivo es más alta, $\Delta f = f(xactual) - f(xvecina) < 0$, entonces se acepta la solución vecina como nueva solución actual. Sin embargo, si la solución vecina tiene un decrecimiento en la función objetivo ($\Delta f >= 0$), entonces solo se aceptará como solución actual con cierta probabilidad que depende de una función de aceptación que va decayendo con cada iteración. El decaimiento de esta función está regulada por el parámetro T de temperatura, el cual va decreciendo con el correr del algoritmo.

El sustento detrás de este procedimiento es que un ocasional decrecimiento del valor de la función objetivo puede ayudar a no quedarse estancados en un mínimo local. Como mencionamos, la temperatura es el factor que determina el nivel de aceptación. A altas temperaturas, la probabilidad debe ser cercana a 1, haciendo que se acepten la mayoría de las soluciones, mientras que las temperaturas bajas, resultan en probabilidades cercanas a 0, haciendo que se rechacen la mayoría de las soluciones que no mejoran la función objetivo.

Nuestra implementación considera una probabilidad de aceptación que sigue el criterio Metropolis [39], el cual se determina por $e^{-\Delta f/T}$. Luego, el procedimiento de Simulated Annealing debe comenzar con una alta temperatura para poder considerar muchos saltos entre las soluciones para explorar el espacio de búsqueda, y luego comienza a reducirse la temperatura para darle más énfasis al hecho de mejorar las soluciones actuales.

La temperatura inicial y el factor de enfriamiento influyen notoriamente en el desempeño de la heurística de Simulated Annealing propuesta. De esta manera, realizamos experimentos preliminares para determinar una buena fijación de parámetros. En cuanto al factor de enfriamiento, se observó que un decrecimiento lento genera mejores resultados. Después de analizar los experimentos preliminares, se fijó la siguiente función para realizar la actualización de la temperatura en cada iteración, $T = \alpha T$, con $\alpha < 1$. En cuanto a la temperatura inicial T, se calculó utilizando el método presentado en [55].

Además del criterio de aceptación, otra diferencia entre el procedimiento de Hill Climbing y el de Simulated Anneling desarrollados para este problema, es que el Simulated Annealing no explora toda la vecindad de una solución, solamente compara la solución actual con algún vecino elegido de manera aleatoria. El procedimiento de Simulated Annealing propuesto, contiene las mismas componentes principales que el procedimiento de Hill Climbing mencionado: inicialización, vecindad y función objetivo. El siguiente pseudocódigo engloba todo el funcionamiento del procedimiento de Simulated Annealing.

Data: Tamaños de los ítems, Capacidad de los contenedores, Conjunto de conflictos, T, α
Result: xincumbent: asignación factible de ítems a contenedores
xactual=asignacion inicial mediante procedmiento heuristico;
xincumbent=xactual ;
while *no se alcance límite de tiempo* **do**
  xvecina = generarVecinoAleatorio(xactual);
  Δf= funcionObjetivo(xactual) - funcionObjetivo(xvecina);
  if $\Delta f < 0$ **then**
    xactual = xvecina;
  else
    if *random()* $< e^{-\Delta f/T}$ **then**
      xactual = xvecina;
    end
  end
  $T = \alpha T$;
  if *funciónObjetivoOriginal(xactual)* > *funciónObjetivoOriginal(xincumbent)* **then**
    xincumbent=xactual ;
  end
end

Algorithm 4: Simulated Annealing

4.4. Resultados experimentales

Como paso inicial para evaluar el comportamiento de las heurísticas, diseñamos experimentos sobre instancias que presentan conflictos generalizados entre los ítems. Naturalmente, como no tenemos conocimiento de trabajos previos sobre este problema, no tenemos punto de comparación para nuestros desarrollos.

Las corridas fueron realizadas en una máquina con procesador Intel Core i7-2600 3.40GHz con 16 GB de RAM, con Ubuntu como sistema operativo.

4.4.1. Análisis de Heurísticas Constructivas

Nuestro primer objetivo es comparar la calidad de las soluciones brindadas por las diferentes versiones de heurísticas constructivas.

Se ejecutaron experimentos en instancias con 100 ítems, variando el número de conflictos (NumConf) y el tamaño máximo (MaxItemConf) del conjunto de conflictos. Para cada conflicto S, el cardinal de S se generó aleatoriamente con una distribución uniforme en el intervalo $[2, MaxItemConf]$. De igual manera, los miembros de S son seleccionados de forma aleatoria.

Concentramos nuestros experimentos computacionales en el caso particular cuando $max_S = |S| - 1$ dado que las instancias con valores más pequeños de max_S están, en cierto sentido que fue explayado anteriormente, representadas mediante las restricciones que se generan por este caso particular.

La capacidad de los contenedores se fijó en 10000 y los tamaños de los ítems fueron generados aleatoriamente por una distribución uniforme en el intervalo $[500, 2500]$. La idea detrás de esta configuración es prevenir que la capacidad de los contenedores y los tamaños de los ítems resulten en restricciones muy duras que hagan superfluas las limitaciones impuestas por el conjunto de conflictos.

Para cada instancia, computamos el porcentaje de gap que hay entre una cota inferior LB y la solución encontrada por la heurística. La cota inferior LB la obtuvimos como el valor de la relajación LP de la reformulación de Dantzig-Wolfe del modelo presentado en este trabajo.

Para cada combinación de parámetros se generaron 10 instancias para obtener resultados más representativos, por lo que luego se calculó el gap porcentual promedio entre todas las instancias

del mismo tipo.

En la tabla 4.1 mostramos los resultados.

Max Item Conf	Num Conf	FFWC	FFDWC	FFDWC+FFDDWC				FFDWC+FFDCWC				FFODWC
						α				α		
				0,25	0,5	0,75	1 (FFDDWC)	0,25	0,5	0,75	1 (FFDCWC)	
3	2500	24,04	33,83	33,83	29,42	24,08	15,63	33,83	29,42	24,08	15,63	**13,67**
	5000	49,24	47,72	47,69	43,36	42,28	38,57	47,69	43,36	42,28	38,57	**30,00**
	10000	34,39	29,74	29,76	31,31	27,64	27,01	29,76	31,31	27,64	27,01	**18,72**
	15000	18,32	17,92	17,72	17,05	15,27	14,46	17,72	17,05	15,27	14,46	**7,38**
	20000	6,48	6,51	6,82	5,99	6,65	5,64	6,82	5,99	6,65	5,64	**0,96**
	25000	1,54	1,16	1,16	1,16	1,03	0,64	1,16	1,16	1,03	0,64	**0,00**
	30000	0,00	0,11	0,11	0,11	0,11	0,00	0,11	0,11	0,11	0,00	**0,00**
5	2500	5,21	12,38	11,13	11,08	7,83	5,21	9,79	8,50	4,63	3,25	**2,58**
	5000	14,96	24,08	21,54	24,00	18,88	11,08	22,13	20,17	13,08	**6,50**	9,08
	10000	43,38	49,17	52,33	47,17	47,21	40,17	49,21	46,54	45,21	35,04	**27,96**
	15000	47,68	48,68	49,76	45,58	45,55	43,42	46,61	47,13	44,03	38,79	**30,39**
	20000	41,29	41,38	41,30	40,14	38,86	35,85	40,94	41,39	36,30	32,50	**25,34**
	25000	36,61	33,83	34,50	35,22	37,70	34,16	33,78	34,89	34,90	29,24	**22,19**
	30000	31,94	30,12	31,94	30,11	31,93	30,14	31,02	31,02	31,01	26,77	**19,15**
10	2500	3,29	3,92	6,50	5,83	5,88	**0,00**	3,88	3,25	1,29	0,67	1,33
	5000	4,54	9,17	10,46	8,46	7,83	2,67	6,50	4,54	4,54	1,96	**1,29**
	10000	9,13	17,58	18,21	14,33	17,58	7,83	13,71	13,04	9,75	**5,17**	7,17
	15000	14,33	24,75	26,79	24,75	22,83	16,21	24,75	22,75	16,96	**7,79**	10,46
	20000	25,42	34,63	37,21	37,75	34,54	22,79	35,79	33,25	28,71	18,29	**13,71**
	25000	36,46	48,83	44,21	45,63	45,58	37,25	41,71	44,25	40,42	31,25	**24,75**
	30000	42,50	52,50	53,13	49,38	51,25	42,50	51,25	46,88	47,50	36,25	**25,00**
15	2500	2,63	5,21	4,54	3,92	3,25	1,96	2,63	0,67	0,67	0,67	**0,00**
	5000	2,67	5,88	4,58	5,83	5,88	4,54	5,88	3,92	2,67	**0,67**	2,58
	10000	4,54	12,33	11,67	11,08	10,42	5,21	9,75	5,83	5,88	**1,96**	3,88
	15000	7,79	16,88	16,29	13,71	17,00	9,71	14,33	13,00	9,13	**3,96**	5,25
	20000	12,38	21,46	21,50	24,08	21,50	11,04	19,54	18,25	12,42	8,46	**7,79**
	25000	16,96	26,75	26,08	26,04	24,71	14,96	24,71	24,13	18,25	11,75	**8,50**
	30000	22,13	31,71	34,42	34,38	31,08	19,46	33,00	29,21	24,00	16,21	**11,75**
20	2500	1,33	3,29	2,00	3,33	**0,00**	1,33	0,67	1,33	**0,00**	**0,00**	**0,00**
	5000	2,67	4,58	4,58	5,25	4,54	2,58	2,58	1,33	1,33	**0,67**	2,63
	10000	4,58	8,50	9,79	8,50	9,08	2,63	8,50	5,21	3,96	**0,67**	4,54
	15000	5,88	11,13	10,46	11,13	13,00	5,88	10,42	8,46	7,21	**2,63**	3,83
	20000	7,83	16,29	15,63	16,33	14,38	7,17	15,67	12,33	8,54	**2,54**	4,54
	25000	9,79	20,83	18,92	19,54	18,25	8,50	18,25	16,96	11,75	6,46	**4,54**
	30000	13,00	23,42	24,63	24,71	23,38	11,71	20,83	18,83	16,92	7,79	**6,54**

Cuadro 4.1: Heuristicas constructivas

Cuadro 4.2: Instancias PECG - 100 ítems - 5 segundos

NumConf	MaxItemConf			
	3	5	10	15
2500	2.46	0.00	0.00	0.00
5000	9.62	0.00	0.00	0.00
10000	2.37	9.81	0.00	0.00
15000	0.42	9.83	0.63	0.00
20000	0.00	11.83	5.99	5.48
25000	0.00	5.00	15.18	14.22
30000	0.00	4.34	24.12	19.62

La primera conclusión que podemos sacar es que en todas las instancias, los mejores resultados son obtenidos por FFDCWC y FFODWC. Cuando la cantidad máxima de ítems involucrados en un conflicto aumenta, mejor es el comportamiento de FFDCWC.

Cuando $MaxItemConf = 3$, los resultados de FFDDWC y FFDCWC son similares. En este caso, la conflictividad y el grado son comparables. Sin embargo cuando $MaxItemConf$ crece, la noción de conflictividad representa en forma mas fidedigna la incompatibilidad de un ítem. Estas mismas conclusiones pueden derivarse si comparamos para un mismo α, los resultados de FFDDWC+FFDWC y FFDCWC+FFDWC.

Esto evidencia la importancia de haber considerado la nueva noción de conflictividad. Es decir, que es tan relevante la cantidad de conflictos en los cuales está involucrado un ítem como también el tamaño de los mismos.

Otro punto a destacar es el hecho de que no resultó beneficioso la utilización del orden de los ítems por su peso. Se puede ver en las tablas presentadas que la heurística FirstFitWC, que considera un orden arbitrario, obtiene mejores resultados. Al analizar las corridas en detalle, se puede ver que en general los pesos no imponen una restricción fuerte. Observamos que la cantidad de ítems por contenedor en las soluciones óptimas suele ser bajo, motivo por el cual las restricciones de empaquetamiento no imponen grandes limitaciones.

4.4.2. Análisis de Simulated Annealing

Ahora analizaremos los resultados obtenidos con el procedimiento de Simulated Annealing. Respecto a Hill Climbing, en todos los casos fueron de inferior calidad, por lo que omitimos su presentacion para concentrarnos en Simulated Annealing.

En la tabla 4.2 reportamos los resultados obtenidos con un límite de ejecución de 5 segundos. La primera conclusión que se extrae, es que a medida que tenemos una mayor cantidad de conflictos, se tiene un mayor gap. Esto es un comportamiento esperado para la heurística. Más restricciones sobre las asignaciones de ítems factibles reducen enormemente la vecindad. Luego, es más dificultoso para el procedimiento encontrar distintas soluciones y alejarse de un mínimo local no tan bueno. De todas maneras, cuando el número de conflictos es grande relativamente al número total de ítems y al número de ítems en cada conjunto de conflicto, el gap tiende a reducirse. En este caso, el número de contenedores utilizados por las asignaciones factibles difieren fuertemente uno de otro.

La misma tendencia se puede observar cuando el número de ítems que pertenecen a un conjunto de conflicto se incrementa. La vecindad de una solución es menos restrictiva, mostrando que la heurística encuentra posibilidades para mejorar la solución actual.

También analizamos la calidad de la solución cuando el límite de tiempo está fijado en 10 y 30 segundos. Estos resultados se encuentran presentados en las tablas 4.3 y 4.4. En primer lugar, el principal mensaje de estas tablas es que la tendencia sobre el gap en cuanto a la cantidad de conjuntos de conflicto y a la cantidad de ítems en un conflicto es similar a la de los experimentos anteriores. Eso es, a mayor cantidad de conflicto o menor cantidad de ítems que pertenecen a un conjunto de conflicto, mayor es el gap.

Al tener más tiempo, como es de esperar, la heurística obtiene mejores soluciones. Se puede

Cuadro 4.3: Instancias PECG - 100 ítems - 10 segundos

NumConf	MaxItemConf			
	3	5	10	15
2500	2.46	0.00	0.00	0.00
5000	9.62	0.00	0.00	0.00
10000	2.37	9.81	0.00	0.00
15000	0.42	9.83	0.63	0.00
20000	0.00	8.06	4.23	0.00
25000	0.00	4.69	9.35	1.21
30000	0.00	3.51	13.92	6.47

Cuadro 4.4: Instancias PECG - 100 ítems - 30 segundos

NumConf	MaxItemConf			
	3	5	10	15
2500	2.46	0.00	0.00	0.00
5000	9.62	0.00	0.00	0.00
10000	2.37	9.81	0.00	0.00
15000	0.42	9.83	0.63	0.00
20000	0.00	6.62	4.23	0.00
25000	0.00	3.07	9.35	0.63
30000	0.00	2.08	13.45	0.63

notar que la mejora es más pronunciada en el caso en el que la cantidad de ítems en un conjunto de conflicto se incrementa. Por ejemplo, en el caso de 30000 conjuntos, es posible reducir el gap por un 54 % en el caso de 5 ítems, 45 % en el caso de 10 ítems y 97 % en el caso de 15 ítems. Esto se da porque la heurística, al tener más tiempo, puede inspeccionar más soluciones y esto es más simple cuando la cantidad de ítems en cada conflicto es mayor.

Finalmente, experimentamos con instancias más grandes. Aunque no podemos hacer comparaciones con la cota dual (dado que la relajación del modelo a resolver es computacionalmente muy cara) y que no tenemos resultados previos de la literatura, es útil analizar el desempeño de la heurística con diferentes límites de tiempo. Experimentamos con instancias de 2500, 5000 y 7500 ítems, variando el número de conflictos (NumConf) y el tamaño máximo (MaxItemConf) de cada conjunto de conflicto, siguiente el mismo proceso aleatorio uniforme mencionado anterioremente.

En la tabla 4.5 reportamos la solución promedio alcanzada por el algoritmo en hasta 5, 10 y 30 segundos de tiempo límite total. En la tabla 4.6, mostramos la reducción promedio de porcentaje del valor de la solución en 10 y 30 segundos, con respecto a la solución obtenida con 5 segundos, respectivamente.

Cuadro 4.5: Valor promedio de la solución

MaxItemConf	NumItem	NumConf														
		2500			5000			10000			20000			40000		
		5s	10s	30s	5s	10s	30s	5s	10s	30s	5s	10s	30s	5s	10s	30s
5	2500	379	379	379	379	379	379	379	379	379	379	379	379	379	379	379
	5000	1262	757	757	1281	758	758	1303	758	758	1342	757	757	1391	761	758
	7500	3281	2066	1134	3333	2114	1135	3398	2136	1135	3508	2232	1135	3644	2288	1135
10	2500	379	379	379	379	379	379	379	379	379	379	379	379	379	379	379
	5000	1259	758	758	1274	757	757	1320	757	757	1370	758	758	1431	826	757
	7500	3266	2058	1134	3313	2090	1135	3387	2146	1134	3507	2224	1134	3661	2318	1134
15	2500	379	379	379	379	379	379	379	379	379	379	379	379	446	379	379
	5000	1262	757	757	1293	758	758	1348	758	758	1389	767	757	3289	2045	1129
	7500	3288	2083	1137	3320	2074	1135	3572	2317	1131	3544	2263	1134	3689	2367	1135
20	2500	379	379	379	379	379	379	379	379	379	380	379	379	486	379	379
	5000	1261	757	757	1323	757	757	1332	758	758	1411	826	758	1539	1057	758
	7500	3281	2056	1135	3342	2077	1134	3500	2202	1135	3714	2439	1127	3774	2417	1135

Cuadro 4.6: Porcentaje promedio de reducción

MaxItemConf	NumItem	NumConf									
		2500		5000		10000		20000		40000	
		10s	30s	10s	30s	10s	30s	10s	30s	10s	30s
5	2500	0	0	0	0	0	0	0	0	0	0
	5000	40	40	41	41	42	42	44	44	45	46
	7500	37	65	36	66	37	67	36	68	37	69
10	2500	0	0	0	0	0	0	0	0	0	0
	5000	40	40	41	41	43	43	45	45	42	47
	7500	37	65	37	66	37	67	37	68	37	69
15	2500	0	0	0	0	0	0	0	0	15	15
	5000	40	40	41	41	44	44	45	45	33	66
	7500	37	65	38	66	36	68	36	68	36	69
20	2500	0	0	0	0	0	0	1	1	22	22
	5000	40	40	43	43	43	43	41	46	31	51
	7500	37	65	38	66	37	68	34	70	36	70

Los resultados computacionales indican que, excepto por el caso de 2500 ítems, la solución en 5 segundos no es la de mejor calidad y puede ser substancialmente mejorada en la mayoría de los casos por encima de un 40 % en un tiempo máximo de 10 segundos.

Hasta 5000 ítems, las soluciones obtenidas en hasta 30 segundos parecen ser marginalmente mejores que las obtenidas a los 10 segundos y el algoritmo tiende a ser más exitoso cuando el número de conjuntos de conflictos crece. En el caso de 7500 ítems, una mejora significativa es evidente cuando el tiempo límite es de 30 segundos. En todos los casos, extendiendo el límite hasta 60 segundos, no hemos encontrado mejoras significativas, aun experimentando con variaciones en los parámetros del algoritmo (α y temperatura inicial).

Capítulo 5

Estudio poliedral

En este capítulo se introducen un conjunto de resultados poliedrales que sirven para identificar componentes importantes asociadas a nuestro modelo, tales como igualdades válidas, desigualdades y facetas. Además de la caracterización teórica, cabe destacar que en muchos casos los resultados teóricos obtenidos tienden a tener una traducción interesante en el aspecto práctico, ayudando en el desarrollo de un algoritmo robusto y eficiente.

Al igual que en los experimentos realizados en el capítulo de heurísticas, nuestro trabajo se concentró en el caso que $max_S = |S| - 1$. Esto es así dado que se puede mostrar que si existe una restricción con un max_S diferente, entonces se la puede cambiar por un conjunto de restricciones donde sí se cumple la condición pedida y el conjunto de soluciones factibles es el mismo.

De la misma manera, las restricciones de capacidad de los contenedores también se pueden reemplazar por restricciones del mismo tipo. Es decir, por cada conjunto de ítems que sobrepase la capacidad de un contenedor, la restricción se puede pensar como una hiperarista que indica que todos esos ítems no pueden ir juntos. En la práctica esto puede tener un impacto importante ya que cada restricción de capacidad sobre los contenedores, debe ser reemplazada por un número mayor de restricciones, posiblemente exponencial ya que cada subconjunto de ítems que se exceda en tamaño será asociado a una nueva hiperarista. De todas maneras, desde el estudio teórico el reemplazo es válido por lo que no hay pérdida de generalidad estudiando las restricciones que provienen de haber realizado el cambio de las restricciones.

El modelo original presentado en la introducción es correcto y completo para nuestro problema. Sin embargo, ese modelo presenta muchas soluciones simétricas. Es decir, soluciones que esencialmente representan a la misma situación pero que para el modelo son diferentes, por ejemplo, por un renombre de variables. Esto usualmente tiende a generar problemas en el contexto de un algoritmo exacto, ya que se debe explorar un espacio de soluciones inherentemente redundante. Dada esta característica no deseada, primero redefinimos nuestro modelo base para el cual realizamos el estudio poliedral. Siguiendo las ideas de [45], este nuevo modelo cortará muchas soluciones enteras factibles para las cuales sabemos que existe una solución alternativa con mismo valor de función objetivo, resultando en un poliedro con menor cantidad de soluciones factibles, pero correcto. Nuestro modelo consiste de las mismas variables, función objetivo y restricciones que el modelo presentado anteriormente, pero ahora se le agregan dos nuevas restricciones. Una de estas restricciones cortará soluciones simétricas mediante la imposición de un orden en el uso de los contenedores, dado por $y_k \geq y_{k+1} \quad \forall k = 1 \dots n-1$. Con esta nueva restricción, un contenedor solo puede ser utilizado si el contenedor previo está siendo utilizado. De esta manera, se dejan afuera varias soluciones simétricas ya que los contenedores dejan de ser indistinguibles. Esta restricción no estaba en el modelo presentado originalmente, aunque las heurísticas constructivas tienden a satisfacerla dado que muchas de ellas intentan llenar los contenedores en orden.

Luego, se añade otra restricción para indicar que la variable de uso de contenedor (y_k) solo puede encontrarse prendida en el caso que fehacientemente haya algún ítem asignado al contenedor, modelado como $y_k \leq \sum_{i=1}^{n} x_{ik} \quad \forall k = 1 \dots n$. Si bien esta restricción parece obvia, el modelo original

no la necesita para ser correcto dado que en un punto óptimo la función objetivo se encargaría de apagar todas las variables de uso de contenedor asociadas a contenedores vacíos.

Esto resulta en el siguiente nuevo modelo para el PECG:

$$
\begin{aligned}
\textit{Minimizar} \quad & \sum_{k=1}^{n} y_k & & \\
\textit{sujeto a} & & & \\
& \sum_{k=1}^{n} x_{ik} = 1 & \forall i = 1, \ldots, n & (5.1)\\
& \sum_{i=1}^{n} w_i x_{ik} \leq W y_k & \forall k = 1, \ldots, n & (5.2)\\
& \sum_{i \in S} x_{ik} \leq max_S & \forall (S, max_S) \in C & \\
& & \forall k = 1, \ldots, n & (5.3)\\
& y_k \geq y_{k+1} & \forall k = 1 \ldots n-1 & (5.4)\\
& y_k \leq \sum_{i=1}^{n} x_{ik} & \forall k = 1 \ldots n & (5.5)\\
& y_k \in \{0, 1\} & \forall k = 1 \ldots n & (5.6)\\
& x_{ik} \in \{0, 1\} & \forall i, k = 1 \ldots n & (5.7)
\end{aligned}
$$

Este nuevo modelo es el que de aquí en más llamaremos *modelo base* para los resultados y experimentación posterior. Presentamos entonces un conjunto de resultados para el poliedro P_{base}asociado con este modelo (cápsula convexa de la soluciones factibles).

En primer lugar, comenzaremos enunciando la dimensión del poliedro asociado. Este resultado es crítico tanto desde el punto de vista teórico como desde el práctico para el estudio del problema ya que, de esta manera, podremos saber en que dimensión realmente se encuentra el conjunto de soluciones factibles.

Una vez obtenido el sistema minimal, se presentarán varias desigualdades válidas para el modelo. Algunas de las desigualdades presentadas resultan ser faceta y en esos casos se presentará su correspondiente demostración.

Antes de comenzar con el estudio poliedral propiamente dicho, se listan una serie de consideraciones a tener en cuenta a lo largo de todos los resultados presentados.

Suposición 1. *Todo ítem puede compartir un contenedor con algún otro ítem. Es decir, no es universal en el sentido de que conflictúa directamente con todos los demás.*

Esta condición se logra fácilmente preprocesando cualquier instancia del problema. Si un ítem es universalmente conflictivo entonces es seguro que en cualquier asignación tiene que ocupar un contenedor por sí solo, por lo que esta asignación se puede hacer de manera previa a la resolución de la instancia. De esta manera, no hay pérdida de generalidad en solo considerar instancias que satisfagan la suposición.

Suposición 2. *Hay al menos cinco ítems en la instancia, y por lo tanto al menos cinco contenedores disponibles.*

Esta segunda suposición no es estrictamente necesaria para la mayoría de los resultados que se van a presentar, pero constituye una simplificación en pos de evitar tener que dividir las demostraciones en casos que no resultan de interés teórico ni práctico.

Suposición 3. *No existen hiperaristas superfluas. Es decir, no existen hiperaristas contenidas en otras.*

Nuevamente esta condición se logra preprocesando las instancias que no cumplan con esta restricción. El objetivo es poder quedarse con hiperaristas que representen información por sí mismas y no que resulten de una derivación de otra restricción.

Todo este conjunto de suposiciones no resultan en una pérdida de generalidad importante de las instancias a tratar y, sin embargo, resultarán de ayuda para normalizar y simplificar las posibles situaciones que pueden surgir en cada uno de los análisis a realizar.

5.1. Dimensión del poliedro

El poliedro definido por el modelo claramente no es de dimensión completa dado que ya posee un conjunto de igualdades en su definición básica. Luego, es importante saber si alguna otra ecuación es válida para el modelo con el objetivo de obtener una descripción completa del sistema minimal de ecuaciones. Obtener todas las ecuaciones de un modelo es importante desde un punto de vista teórico ya que luego es utilizado vastamente para lograr demostrar si una desigualdad válida dada es faceta. Además, las ecuaciones usualmente implican restricciones muy fuertes que tienden a resultar en mejores tiempos computacionales en el contexto de un algoritmo branch-and-cut.

Para nuestro modelo, se puede demostrar que el sistema minimal está definido por las igualdades de asignación que ya están presentes en la definición original, más algunas ecuaciones adicionales que están asociadas al hecho de que el uso de los contenedores ahora tiene un orden impuesto.

Proposición 1. *El sistema minimal de P_{base} está dado por las siguientes ecuaciones:*

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad \forall i = 1, \ldots, n \tag{5.8}$$

$$y_k = 1 \qquad \forall k = 1 \ldots \chi^b \tag{5.9}$$

$$y_n = \sum_{i=1}^{n} x_{in} \tag{5.10}$$

El primer grupo de ecuaciones se corresponden con las restricciones de asignación que ya estaban presentes en el modelo original. El segundo grupo de ecuaciones establecen que, cuando los contenedores se usan de manera ordenada, todos los contenedores hasta el número χ^b seguro tienen que ser utilizados en todos los puntos factibles del poliedro, siendo χ^b la cantidad óptima de contenedores necesitada para la instancia particular. Finalmente, la última ecuación está diciendo que, en caso de que se use el último contenedor, entonces debe ser utilizado solamente con un ítem. Este sería el caso en donde cada ítem es asignado a un contenedor diferente, en cualquier otro caso, el contenedor debe permanecer vacío.

Demostración. Para demostrar que las ecuaciones dadas conforman el sistema minimal del poliedro debemos probar que:

- Las ecuaciones son linealmente independientes. Es decir que ninguna de ellas puede escribirse como combinación lineal de las demás.
- Toda otra ecuación satisfecha por todos los puntos del poliedro puede ser escrita como una combinación lineal de las ecuaciones dadas.

El primer punto resulta fácil de ver ya que las ecuaciones dadas imponen restricciones sobre conjuntos disjuntos de variables. Luego, resta probar la segunda condición.

En una combinación lineal de las ecuaciones dadas denotamos A_i como el coeficiente que multiplica a la ecuación (5.8) correspondiente al ítem i. B_k al coeficiente de la ecuación que impone que

el contenedor k es utilizado si es menor o igual a χ^b. Por último, denotamos como C al coeficiente de la última ecuación.

Con las anteriores definiciones, cualquier ecuación satisfecha por los puntos del poliedro se puede escribir de forma genérica como

$$\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_{ik}x_{ik} + \sum_{k=1}^{n}\beta_k y_k = \gamma \tag{5.11}$$

Si esta ecuación genérica se puede escribir como una combinación lineal del presunto sistema minimal del poliedro, entonces las siguientes igualdades deben ser ciertas.

$$\alpha_{ik} = A_i \quad \forall k \neq n \ \ \forall i \in 1 \dots n \tag{5.12}$$
$$\alpha_{in} = A_i + C \quad \forall i \in 1 \dots n \tag{5.13}$$
$$\beta_k = B_k \quad 1 \leq k \leq \chi^b \tag{5.14}$$
$$\beta_k = 0 \quad \chi^b < k < n \tag{5.15}$$
$$\beta_n = -C \tag{5.16}$$

Luego, para poder asegurar que es posible generar la ecuación genérica a partir de una combinación de las ecuaciones dadas, debemos demostrar los siguientes resultados que implicarían que las ecuaciones (5.12) a (5.16) son compatibles:

$$\alpha_{ik_1} = \alpha_{ik_2} \quad \forall i \quad \forall k_1, k_2 \neq n \tag{5.17}$$
$$\alpha_{in} = \alpha_{ik_1} - \beta_n \quad \forall i \quad \forall k_1 \neq n \tag{5.18}$$
$$\beta_k = 0 \quad \chi^b < k < n \tag{5.19}$$

Por lo explicado en las consideraciones previas, no existen ítems que no puedan compartir un contenedor con ningún otro. Es decir, por cada ítem i_1, existe al menos algún otro ítem i_2, con el cual pueden ser asignados juntos a un mismo contenedor. Tomamos entonces un punto entero factible del poliedro en donde dichos dos ítems son asignados a un mismo contenedor, digamos k_1, y cada uno del resto de los ítems es asignado solo, a un contenedor distinto. Al armar este punto, tenemos exactamente un contenedor vacío. Lo que es más, este contenedor debe ser el último porque ahora se está en el caso donde los contenedores tienen impuesto un orden de uso. Desde este punto, podemos generar otro punto factible que proviene de mover i_1 al contenedor vacío. Una vez obtenidos estos puntos, podemos evaluarlos en la ecuación genérica que queremos demostrar que es redundante. Luego de evaluar los puntos en la ecuación, procedemos a restar los resultados entre sí. De este procedimiento resulta la siguiente ecuación válida.

$$\alpha_{i_1 k_1} - \alpha_{i_1 n} - \beta_n = 0 \tag{5.20}$$

Esto es exactamente lo que necesitamos demostrar en (5.18) para el ítem i_1 y para el contenedor k_1. Además, este argumento puede ser utilizado para todo ítem y para todo contenedor que no sea el último, demostrando así (5.18) para todo par de ítems y contenedores.

Con este resultado, para un ítem dado i, podemos reordenar (5.18) para cualquier contenedor de la siguiente manera

$$\alpha_{ik_1} = \alpha_{in} + \beta_n \tag{5.21}$$

Como esta ecuación es válida para todo contenedor, es claro que (5.17) vale para todo par de contenedores.

Una vez probados (5.17) y (5.18), solo resta probar la validez de (5.19) para terminar la demostración.

Como χ^b es el mínimo número de contenedores necesarios para una asignación óptima de los ítems, debe entonces existir un punto factible del poliedro en donde todos los ítems puedan ser asignados a los primeros χ^b contenedores. Además, debe existir algún punto que utilice solo χ^b contenedores en el que un ítem dado i_1 no esté asignado solo, de otra manera $\chi^b = n$. Si tomamos un punto donde i_1 está asignado solo, debe existir otro ítem i_2 que puede ser movido de su contenedor actual al contenedor donde está i_1. Este ítem i_2 existe dado que no hay ningún ítem universalmente conflictivo. Es importante notar que el movimiento de i_2 solo podría resultar en un punto infactible si al irse de su contenedor este quedase vacío. Sin embargo, esto no puede suceder, ya que si no esto significaría que se puede realizar una asignación de los ítems utilizando $\chi^b - 1$ contenedores, lo cual resulta en un absurdo.

Una vez que tenemos este punto óptimo donde i_1 no se encuentra solo, podemos generar un nuevo punto factible del poliedro en donde movemos i_1 al contenedor $\chi^b + 1$. Podemos entonces evaluar estos dos puntos en la ecuación genérica. Una vez hecha las evaluaciones, restamos los resultados obtenidos, quedando:

$$\alpha_{i_1 k_1} - \alpha_{i_1 \chi^b+1} - \beta_{\chi^b+1} = 0 \tag{5.22}$$

En el caso de que no haya ningún contenedor entre χ^b y n, entonces no hay ningún resultado más para demostrar. Además, en este caso todo nodo sería universalmente conflictivo, por lo cual no podría suceder dicho caso.

En el caso de que sí haya contenedores intermedios, dado que ya demostramos que $\alpha_{i_1 k_1} = \alpha_{i_1 \chi^b+1}$, entonces $\beta_{\chi^b+1} = 0$.

Luego, al haber movido i_1 a $\chi^b + 1$, obtuvimos un nuevo punto factible en donde con seguridad no hay un solo ítem en cada contenedor dado que estamos en el caso en donde n es más grande a $\chi^b + 1$. Entonces, nuevamente tomamos algún ítem que este compartiendo contenedor y se lo mueve al siguiente contenedor vacío, en este caso $\chi^b + 2$. Nuevamente, si $\chi^b + 2$ es n entonces no hay nada que demostrar, concluyendo. Si, por el contrario, $\chi^b + 2$ no es n, entonces se puede realizar el mismo procedimiento que para $\chi^b + 1$, demostrando que el coeficiente β_{χ^b+2} es nulo.

Este procedimiento puede repetirse para todos los contenedores intermedios hasta llegar al contenedor n, quedando entonces demostrado (5.19)

De esta manera quedan demostradas todas las identidades necesarias para probar que el sistema minimal del poliedro P_{base} es el propuesto. □

Corolario 1. *Dada una instancia del Problema de Empaquetamiento con Conflictos Generalizados con n ítems, todos ellos no universalmente conflictivos, la dimensión del poliedro asociado a nuestro modelo es* $n^2 - \chi^b - 1$.

Demostración. Este resultado es una simple derivación de la anterior proposición. Nuestro modelo tiene $n^2 + n$ variables y ya hemos demostrado que el sistema minimal consta de $n + \chi^b + 1$ ecuaciones. Luego, la dimensión del poliedro es $n^2 + n - (n + \chi^b + 1) = n^2 - \chi^b - 1$. □

5.2. Desigualdades Válidas

En esta sección presentamos varias familias de desigualdades válidas para nuestro poliedro. Algunas provienen de las restricciones originales necesarias para definir correctamente el problema y otras son derivaciones de restricciones que nacen en general de la combinación de las restricciones del problema con las características particulares del hipergrafo que define la instancia a resolver. Encontrar estas combinaciones resulta de vital interés práctico para poder explotar los atributos específicos de cada instancia con el objetivo de lograr mejores tiempos computacionales en las resoluciones.

Para varias de estas familias, se observa que definen facetas del poliedro bajo ciertas circunstancias y se presenta una demostración en esos casos junto a la definición de la familia.

Antes de continuar con el tratamiento de cada familia en particular, remarcamos que en todos los casos que se realice una demostración de *facetitud* sobre la familia de desigualdades, se tomará como vigente la siguiente hipótesis:

Hipótesis 1. *Un punto factible donde exactamente χ^b contenedores son usados debe pertenecer a la cara definida por la desigualdad.*

Esta hipótesis es necesaria para lograr que cualquier desigualdad sea faceta del poliedro sin importar su naturaleza. Esto sucede porque el uso de los contenedores tiene un orden impuesto. Si la hipótesis no se cumpliese, significaría que todos los puntos que pertenecen a la cara de la desigualdad en cuestión están usando necesariamente al menos $\chi^b + 1$ contenedores. Esto traería aparejada la igualdad $y_{\chi^b+1} = 1$, la cual es una igualdad que no se deriva del sistema minimal, por lo que en este caso la desigualdad nunca podría definir una faceta del poliedro.

5.2.1. Desigualdad de conjunto independiente

En nuestro problema, la restricción clave para conseguir asignaciones válidas viene del hecho de que ciertos subconjuntos de ítems conflictúan como un todo, por lo que no pueden ser asignados todos juntos al mismo contenedor. En las restricciones del modelo base se predica que en ciertos conjuntos de k elementos como mucho $k-1$ ítems pueden ser asignados al mismo contenedor. Sin embargo, las combinaciones de estas restricciones entre diferentes hiperaristas en general tienden a generar subconjuntos de ítems en donde la máxima cantidad que puede ser asignada juntos es aún menor.

Dado un conjunto de ítems S, denotamos como $\alpha(S) = r$ a la máxima cantidad de ítems que pueden estar juntos, es decir, la cardinalidad del máximo conjunto independiente (en este contexto, la máxima cantidad de ítems de S que pueden ser asignados juntos a un mismo contenedor). Luego, para cada contenedor $\bar{k}$ se deriva que la desigualdad

$$\sum_{i \in S} x_{i\bar{k}} \leq r y_{\bar{k}} \tag{5.23}$$

es válida para P_{base}.

Si bien es claro que la desigualdad propuesta es una restricción válida, es simple mostrar que en general esta desigualdad no define una faceta para el poliedro. En primer lugar, es necesario tener un punto que pertenezca a la cara donde se cumpla $y_{\bar{k}} = 1$. De lo contrario, todos los puntos satisfacerían la igualdad $y_{\bar{k}} = 0$, bajando en al menos uno la dimensión de la cara asociada. Entonces, en los puntos que se usa el contenedor $\bar{k}$, la única forma de cumplir la restricción por igualdad es si r ítems de S son asignados al contenedor $\bar{k}$. Sin embargo, como los contenedores son usados en orden, esto lleva a la restricción de que cada contenedor anterior a $\bar{k}$ tiene que ser utilizado con al menos un ítem. Luego, esto solo puede ser satisfecho cuando $\bar{k} \leq n - r + 1$.

Además, utilizando el mismo argumento del orden de los contenedores, se puede mostrar que en la cara los últimos $r-1$ contenedores nunca pueden ser utilizados. Si el contenedor $\bar{k}$ no se utiliza, entonces ningún contenedor posterior se utiliza. Si el contenedor $\bar{k}$ se utiliza, entonces debe tener al menos r ítems asignados. Luego, la cantidad de ítems restantes nunca pueden alcanzar para llenar los últimos $r-1$ contenedores.

Lo expuesto indica que las variables asociadas al uso de los últimos contenedores se encontrarían siempre apagadas en la cara, haciendo que la desigualdad sea más débil que lo deseado.

Dado que varias ecuaciones son válidas en la cara definida por la desigualdad inicial, es necesario reforzar la restricción agregando más información. Usando argumentos similares a los expuestos recientemente, se pueden agregar nuevas variables a la desigualdad presentada. Si se quiere poner r ítems en el contenedor $\bar{k}$, entonces el contenedor $n-r+1$ solo podrá tener un ítem asignado. Además,

si se quiere asignar un ítem a cualquier contenedor con número más grande a $n-r+1$, entonces la cantidad de ítems en $\bar{k}$ debe disminuir. Por cada ítem que se usa en los últimos contenedores, un ítem menos puede ser asignado al contenedor $\bar{k}$. Estas observaciones se pueden volcar reforzando la desigualdad de la siguiente manera:

$$\sum_{i \in S} x_{i\bar{k}} + \sum_{i=1}^{n} \sum_{k=n-r+1}^{n} x_{ik} \leq r y_{\bar{k}} + y_{n-r+1} \tag{5.24}$$

Esta desigualdad es más fuerte que la presentada originalmente y se puede demostrar que en muchas situaciones define una faceta del poliedro asociado.

Proposición 2. *Dado un contenedor $\bar{k} < n-r+1$ y un conjunto de vértices S,*

$$\sum_{i \in S} x_{i\bar{k}} + \sum_{i=1}^{n} \sum_{k=n-r+1}^{n} x_{ik} \leq r y_{\bar{k}} + y_{n-r+1} \tag{5.25}$$

es válida para P_{base}. Además, bajo algunas hipótesis la desigualdad define una faceta para el poliedro asociado.

Demostración. La validez de la desigualdad fue explayada parcialmente durante la derivación de la misma. Sin embargo, para realizar las argumentaciones de la derivación, resultó necesario pensar que el contenedor $\bar{k}$ no se encontraba entre los últimos r contenedores. Esta situación quedará más clara al enunciar las hipótesis necesarias para que la desigualdad sea válida y defina una faceta.

Luego, para demostrar que la restricción define una faceta del poliedro asociado en algunas circunstancias, demostramos que la cara definida por la desigualdad tiene exactamente una dimensión menos que todo el poliedro. Esto se hace mostrando que el sistema minimal de la cara no es más que el sistema minimal del poliedro más la restricción presentada pero considerada como una igualdad.

Para completar esta demostración haremos uso de las siguientes necesarias y suficientes hipótesis para que la desigualdad sea válida y defina una faceta.

Hipótesis 2. $\bar{k} < n-r+1$

Esta hipótesis no solo es necesaria para que la desigualdad pueda ser faceta, si no que es necesaria para que la desigualdad sea válida. Si se cumpliese que $\bar{k}$ es mayor o igual a $n-r+1$ eso significaría que el contenedor $\bar{k}$ está entre los últimos r contenedores. Al suceder esto, poner algo de S en $\bar{k}$ sumaría doble en el lado izquierdo de la restricción. De esta manera el lado izquierdo podría llegar a sumar hasta $2r$, violando la restricción. En el único caso que no se podría violar la desigualdad por este tipo de puntos, es si $\bar{k}$ fuese igual a n. Pero en ese caso se viola simplemente armando un punto en donde haya un ítem por contenedor, el último vacío y 2 ítems en el contenedor $n-r+1$.

Hipótesis 3. $\chi^b < n-r+1$

Esta hipótesis indica que la cantidad mínima de contenedores necesarios está por debajo de $n-r+1$. Si no fuera así, entonces el lado derecho de la desigualdad siempre tendría su valor máximo $r+1$. De esta manera, para cumplir la restricción por igualdad, nunca se podría enviar más de un ítem a los contenedores que no suman en el lado izquierdo. Es decir a los contenedores anteriores a $n-r+1$ que no son $\bar{k}$.

Hipótesis 4. $\nexists i \notin S \mid \alpha(S+i) = r$

Esta condición dice que el subconjunto S es maximal en el sentido de tener un conjunto independiente de tamaño r. Cualquier otro ítem que quiera ser agregado a S, debe incrementar el tamaño del máximo conjunto independiente. Esta hipótesis es necesaria porque, de lo contrario, se podría agregar la variable $x_{\bar{i}\bar{k}}$ a la primera sumatoria, en donde $\bar{i}$ sería el ítem que no es condescendiente con la hipótesis. De esta manera se obtendría una desigualdad que domina a la presentada.

Hipótesis 5. *El conjunto S tiene al menos dos elementos*

Esto es necesario porque si S tiene un solo elemento, entonces si se usa el contenedor $\bar{k}$, debe ser con el ítem de S. Si no, al usarse el contenedor $\bar{k}$ alguien tendría que sumar del lado izquierdo para poder cumplir con la igualdad, sin embargo lo único que puede sumar es asignar un ítem al último contenedor. Por el orden del uso de los contenedores esto nunca puede sumar más que uno, que es lo mismo que sumaría la variable y_{n-r+1} del lado derecho, por lo que la igualdad nunca puede ser satisfecha. Es decir, la hipótesis es necesaria porque de lo contrario los puntos de la cara satisfacerían la igualdad $x_{s\bar{k}} = y_{\bar{k}}$, donde s sería el único ítem en S.

Hipótesis 6. *Debe existir algún conjunto independiente $R \subset S$ de cardinal r, tal que $V - R$ no es una clique considerando las aristas de tamaño 2.*

Esta condición se debe cumplir, ya que de lo contrario todos los puntos de la cara cumplen la igualdad $\sum_{i=1}^{n} x_{i1} = 1$. Por un lado, esta igualdad es válida para todos los puntos de la cara que usan más de $n - r$ contenedores independientemente de esta hipótesis. Esto es porque si se usan más de $n - r$ contenedores, entonces el lado derecho de la desigualdad suma $r + 1$. Esto implica que entre el contenedor $\bar{k}$ y los contenedores a partir del $n - r + 1$ tiene que haber al menos $r + 1$. Esto deja $n - (r + 1)$ ítems para los primeros $n - r$ contenedores, pero sin contar el $\bar{k}$. Es decir que necesariamente tiene que ir un ítem asignado a cada contenedor.

Por otro lado, en el caso particular de que la hipótesis no valga, entonces se tendría una clique de tamaño $n - r$, por lo que $\chi^b = n - r$, ya que por la anterior hipótesis el valor de χ^b no puede ser más alto. Pero en este caso, todos los puntos que usen $n - r$ contenedores, necesariamente tienen que tener la misma forma. Esta forma es un conjunto de r ítems de S en el contenedor $\bar{k}$ y el resto de los ítems se tiene que esparcir uno en cada contenedor, en particular el primero.

Luego, por ambos casos se ve que si la condición no se cumpliese entonces los puntos de la cara cumplirían una nueva igualdad. De esta manera, la hipótesis resulta necesaria.

Para demostrar que (5.25) define una faceta del poliedro bajo las circunstancias explayadas, vamos a mostrar que el sistema minimal de la cara está dado por el sistema minimal del poliedro más la nueva restricción considerada como igualdad.

Si bien no hace falta para la demostración, se asume que $\bar{k}$ es mayor a χ^b para realizar la prueba. En caso de que esto no se cumpla, la demostración es más simple, habiendo menos condiciones a probar. Este punto quedará más claro en el transcurso de la demostración.

Para mostrar que toda ecuación es linealmente dependiente con las ecuaciones del presunto sistema minimal, necesitamos demostrar los siguientes resultados:

$$\alpha_{ik} = \alpha_{i1} \quad \forall i \in S \quad \forall k < n - r + 1, k \neq \bar{k} \tag{5.26}$$

$$\alpha_{i\bar{k}} = \alpha_{i1} - \beta_{n-r+1} \quad \forall i \in S \tag{5.27}$$

$$\alpha_{ik} = \alpha_{i1} - \beta_{n-r+1} \quad \forall i \in S \quad \forall k \geq n - r + 1, k \leq n - 1 \tag{5.28}$$

$$\alpha_{in} = \alpha_{i1} - \beta_{n-r+1} - \beta_n \quad \forall i \in S \tag{5.29}$$

$$\alpha_{ik} = \alpha_{i1} \quad \forall i \notin S \quad \forall k < n - r + 1, k \neq \bar{k} \tag{5.30}$$

$$\alpha_{i\bar{k}} = \alpha_{i1} \quad \forall i \notin S \tag{5.31}$$

$$\alpha_{ik} = \alpha_{i1} - \beta_{n-r+1} \quad \forall i \notin S \quad \forall k \geq n - r + 1, k \leq n - 1 \tag{5.32}$$

$$\alpha_{in} = \alpha_{i1} - \beta_{n-r+1} - \beta_n \quad \forall i \notin S \tag{5.33}$$

$$\beta_k = 0 \quad \chi^b < k < n \quad k \neq \bar{k}, n - r + 1 \tag{5.34}$$

$$\beta_{\bar{k}} = r\beta_{n-r+1} \tag{5.35}$$

En primer lugar comenzamos tomando un punto donde se utilicen $n - r$ contenedores que pertenezca a la cara. Este punto tiene que existir porque se puede crear a partir de un punto que use χ^b contenedores y luego moviendo, de ser necesario, ítems hasta llegar a utilizar el contenedor $n - r$ con el detalle de que tiene que haber r ítems del conjunto S en el contenedor $\bar{k}$. Además,

por las hipótesis que se están utilizando, este punto debe poder tener algun contenedor distinto de $\bar{k}$ con más de un ítem, ya que existe algún conjunto independiente de S de tamaño r que no deja una clique completa en los ítems restantes.

Una vez que se tiene este punto, se puede pensar que el contenedor que tiene más de un ítem es el primero (y distinto de $\bar{k}$) sin pérdida de generalidad. Vamos a generar un nuevo punto factible moviendo a uno de los ítems, i, del primer contenedor al contenedor $n-r+1$, produciendo otro punto válido para la cara. De la interacción entre estos dos puntos evaluados en la ecuación genérica obtenemos la siguiente relación:

$$\alpha_{i1} = \alpha_{in-r+1} + \beta_{n-r+1} \tag{5.36}$$

Dependiendo de si el ítem que podemos pasar pertenece o no a S, esta ecuación corresponde a (5.28) o (5.32), pero solamente para el contenedor $n-r+1$. Además, si bien para ejemplificar se tomó el primer contenedor, se podría haber utilizado cualquiera de los contenedores anteriores a $n-r+1$ sin contar $\bar{k}$ que debe ser distinto para cumplir las restricciones de la cara. Luego, para los ítems que pueden compartir contenedor en puntos de esta familia se demuestra (5.26) o (5.30), dependiendo si el ítem i pertenece o no a S.

Así, con estos al menos dos ítems que comparten contenedor por fuera de $\bar{k}$ se puede generar una asignación de $n-1$ contenedores, en donde se manden estos 2 ítems al contenedor $n-r+1$ y todos los demás contenedores tengan un ítem asignado. Para que este punto pertenezca a la cara, lo único que debemos tener cuidado es que en $\bar{k}$ debe haber un ítem de S. Entonces, si tomamos este punto y pasamos a uno de los dos ítems al último contenedor obtenemos la relación:

$$\alpha_{in-r+1} = \alpha_{in} + \beta_n \tag{5.37}$$

de donde se deriva:

$$\alpha_{in} = \alpha_{i1} - \beta_{n-r+1} - \beta_n \tag{5.38}$$

demostrando o bien (5.29) o (5.33) para estos ítems. Además, si bien este pasaje se hizo desde el contenedor $n-r+1$, se podría haber hecho desde cualquier contenedor de los últimos r. Esto demuestra que los α asociados son los mismos. Es decir demuestra (5.28) o (5.32) para estos ítems en particular. Es importante destacar que por las hipótesis utilizadas, existe al menos uno de estos ítems (y por ende al menos dos).

Luego, una vez realizadas estas demostraciones para estos ítems en particular, es simple mostrar los mismos resultados para el resto de los ítems. Si tomamos los puntos que utilizan n contenedores y pertenecen a la cara entonces suceden dos cosas. En primer lugar, todos los contenedores tiene un ítem asignado. En segundo lugar, para pertenecer a la cara, el contenedor $\bar{k}$ tiene un ítem de S. Además, como el cardinal de S es mayor a 1, hay varios candidatos a ir a $\bar{k}$. Luego, se pueden crear nuevos puntos a partir de intercambiar 2 ítems que no estén en $\bar{k}$. Supongamos que lo hacemos con los contenedores 1 y 2, entonces de la interacción entre esos dos puntos obtendríamos la relación:

$$\alpha_{i_1 1} + \alpha_{i_2 2} = \alpha_{i_1 2} + \alpha_{i_2 1} \tag{5.39}$$

Como i_1 e i_2 pueden ser cualquier par de ítems, podemos tomar i_2 como alguno de los ítems para los cuales ya demostramos todos los resultados pertinentes (exceptuando el resultado que predica sobre el contenedor $\bar{k}$). Como ya sabemos que $\alpha_{i_2 1} = \alpha_{i_2 2}$, entonces ahora queda demostrado $\alpha_{i_1 1} = \alpha_{i_1 2}$. Este procedimiento se puede hacer para todo ítem y lo que es más, se puede hacer para todo contenedor que no sea $\bar{k}$. De esta manera se puede demostrar que todos los ítems tienen la misma naturaleza para sus coeficientes α como se puede ver en los resultados a demostrar si no consideramos el contenedor $\bar{k}$ que es donde difieren los ítems que pertenecen a S con los que no.

Con estos intercambios en asignaciones de n contenedores que pertenecen a la cara, se demuestra entonces (5.26), (5.28), (5.29), (5.30), (5.32), (5.33).

Como siguiente paso, vamos a querer demostrar (5.31) para cada ítem i que no está en S. Cada uno de estos ítems i, tiene que poder ser asignado a $\bar{k}$ en una asignación de $n-r$ contenedores. De no suceder esto, no se cumpliría la hipótesis que indica que no se puede agregar un ítem a S y

seguir teniendo un máximo conjunto independiente de tamaño r. Tomamos entonces un punto que utiliza $n-r$ contenedores y donde i está en $\bar{k}$. Luego, obtenemos un nuevo punto factible para la cara moviendo a i desde $\bar{k}$ al contenedor $n-r+1$. De estos dos puntos obtenemos la relación:

$$\alpha_{i\bar{k}} = \alpha_{in-r+1} + \beta_{n-r+1} \tag{5.40}$$

Luego, utilizando (5.32), se deriva:

$$\alpha_{i\bar{k}} = \alpha_{i1} \tag{5.41}$$

demostrando (5.31) para todo ítem fuera de S.

Por último en cuanto a los coeficientes α, debemos demostrar (5.27). Para esto vamos asumir que $r \geq 2$. El caso $r = 1$ es un caso particular que solo se da en el caso que S sea una clique de hiperaristas de tamaño 2. En este caso, varias ecuaciones se pueden escribir de forma simplificada ya que, por ejemplo, cada vez que nos referimos al contenedor $n-r+1$, nos estamos refiriendo al último. Este caso particular será considerado en una sección posterior.

Luego, si $r \geq 2$, entonces podemos tomar un punto que utilice $n-1$ contenedores donde existe un conjunto de 2 elementos de S que podemos asignar a $\bar{k}$, asignando un ítem a cada uno del resto de los contenedores sin contar el último. A partir de este punto, generamos un nuevo punto que surge moviendo uno de estos dos ítems al contenedor n. De estos dos puntos sale la relación:

$$\alpha_{i\bar{k}} = \alpha_{in} + \beta_n \tag{5.42}$$

Lo cual, en conjunción con los resultados anteriores, demuestra (5.27), pero solo para los ítems de S que pueden ser asignados con otro de S. No tienen porque ser todos, pero al menos dos deben existir ya que estamos en el caso de que el conjunto independiente máximo es al menos 2.

Luego, para el resto de los ítems de S que no puedan compartir contenedor con ningún otro ítem de S, podemos realizar el mismo tipo de intercambio que hicimos antes tomando puntos que utilicen n contenedores. Esto no lo podíamos realizar antes porque para seguir perteneciendo a la cara, necesitamos intercambiar a un ítem de S en $\bar{k}$ por otro de S. Ahora que ya hemos demostrado (5.27) para al menos dos ítems de S, tenemos ítems para realizar los intercambios, demostrando la condición para todos los ítems.

Por último, debemos demostrar las condiciones de los coeficientes β, lo cual ahora resulta más simple al saber exactamente la relación entre los coeficientes α al mover cualquier ítem de contenedor.

En primer lugar, comenzamos tomando un punto en la cara donde se utilicen χ^b contenedores. A partir de este punto podemos pasar un ítem que comparta contenedor con otros al contenedor χ^b+1 sin perder la pertenencia a la cara. De esta manera se demuestra que $\beta_{\chi^b+1} = 0$. Este mismo procedimiento lo podemos hacer para todos los contenedores intermedios entre χ^b y $\bar{k}$ (si no hay ninguno, no hay nada para probar). De esta manera hasta $\bar{k}$ exclusive queda demostrado que los β asociados son 0.

Luego, para llenar $\bar{k}$ y seguir consiguiendo puntos factibles de la cara, debemos mover r ítems de S a $\bar{k}$. Esto siempre se puede hacer porque tiene que haber r ítems que sean un conjunto independiente. Además, si el movimiento de estos ítems generase contenedores vacíos, esto se podría arreglar reacomodando otros ítems que por argumentos de cantidad tienen que alcanzar y como ya probamos que sus coeficientes α son todos iguales para los contenedores antes de $\bar{k}$ entonces podemos excluirlos del razonamiento ya que moverlos o no es inocuo. Al pasar r ítems a $\bar{k}$ conseguimos un nuevo punto en la cara. De la interacción de los dos puntos resulta la relación:

$$\sum_{i=1}^{r} \alpha_{i1} = \sum_{i=1}^{r} \alpha_{i\bar{k}} + \beta_{\bar{k}} \tag{5.43}$$

en donde denominamos, sin pérdida de generalidad, de 1 a r a los ítems que movemos al contenedor $\bar{k}$. Además, como para estos ítems todos los α de los contenedores antes de $\bar{k}$ son

intercambiables, se simplificó la relación utilizando α_{i1} sin importar de que contenedor realmente provenía cada ítem. Luego, utilizando el resultado (5.27), se deriva:

$$\sum_{i=1}^{r} \alpha_{i1} = \sum_{i=1}^{r} (\alpha_{i1} - \beta_{n-r+1}) + \beta_{\bar{k}} \tag{5.44}$$

Por último, simplificando y despejando se obtiene:

$$r\beta_{n-r+1} = \beta_{\bar{k}} \tag{5.45}$$

demostrando (5.35). Todo esto se hizo en el caso de que $\chi^b < \bar{k}$. En caso contrario, este procedimiento no se podría hacer, pero no hace falta porque el coeficiente $\beta_{\bar{k}}$ quedaría libre haciendo que este resultado no sea necesario de demostrar.

Luego, a partir del punto que utiliza exactamente $\bar{k}$ contenedores, se puede seguir con el procedimiento de desprender de a un ítem y se continúa con la demostración de $\beta_k = 0$ hasta el contenedor $n - r + 1$ exclusive. A partir de ese contenedor el mismo procedimiento no se puede continuar porque los coeficientes α comienzan a ser distintos por lo que no resulta inocuo mover un ítem. Sin embargo, si tomamos un punto que utilice $n-r+1$ contenedores, lo que seguro podemos hacer es ir pasando de un ítem de $\bar{k}$ a cada uno de los contenedores restantes hasta el $n-1$. Luego, como por (5.27) y (5.28) vale que $\alpha_{i\bar{k}} = \alpha_{ik}$ para todo i en S y para todo k mayor o igual a $n-r+1$ y menor a n, nuevamente se demuestra que $\beta_k = 0$ para esos mismos contenedores, demostrando entonces todo (5.34).

De esta manera, queda finalizada la demostración de *facetitud* de esta familia de desigualdades bajo las condiciones dadas.

□

En la familia presentada recientemente, se toma un conjunto S de ítems y se escribe una nueva restricción en función de su conjunto independiente máximo. Si bien esta desigualdad engloba a cualquier subconjunto de ítems con su r correspondiente, resulta de interés identificar estructuras recurrentes en el hipergrafo de conflictos. En el caso de un subconjunto genérico, el cálculo del conjunto independiente máximo resulta un problema difícil, perteneciente a la clase $\mathcal{NP}$-hard. Sin embargo, para ciertas estructuras del hipergrafo, se puede identificar el cardinal de su conjunto independiente máximo de manera simple. A continuación se presentan algunos ejemplos de dichas estructuras, para las cuales se muestra su conjunto independiente máximo asociado.

Hiperclique maximal

Una estructura clásica a considerar en problemas que poseen un grafo conflicto es la de clique. En problemas con un grafo de conflicto simple, una arista indica que dos vértices conflictúan y por lo tanto no pueden ser asignados juntos a un mismo contenedor. En el caso de una clique, la información es más restrictiva aún ya que indica que no importa cuales dos elementos tomemos dentro de esa estructura, nunca van a poder ser asignados juntos. Visto de otra manera, no queda otra opción que asignar cada vértice a un contenedor diferente.

En nuestro caso no poseemos un grafo conflicto, sino un hipergrafo. En los hipergrafos para definir una clique no solo importa qué vértices están involucrados, sino cuál es el cardinal de las hiperaristas que conforman la clique. Es decir, una clique $K_{n_1}^{n_2}$ es un subgrafo de n_1 vértices en donde todas las hiperaristas de tamaño n_2 se encuentran presentes. Por ejemplo, una clique K_5^3, es un conjunto de 5 vértices en donde todas las hiperaristas de 3 vértices se encuentran presentes. De esta manera, no importa de qué manera hagamos una posible asignación, nunca podrán ir más de 2 vértices de este subconjunto juntos en un mismo contenedor.

En el caso general, si se tiene una clique $K_{n_1}^{n_2}$, entonces el cardinal de su conjunto independiente máximo es $n_2 - 1$. Es importante notar que estamos bajo la hipótesis que no existen hiperaristas superfluas. De lo contrario, podrían existir hiperaristas más pequeñas incluidas en la clique que restrinjan más fuertemente el número máximo de vértices que pueden ser asignados juntos.

Luego, para esta estructura, podemos particularizar la desigualdad presentada en la anterior sección de la siguiente manera

$$\sum_{i \in K_{n_1}^{n_2}} x_{i\bar{k}} + \sum_{i=1}^{n} \sum_{k=n-n_2-1+1}^{n} x_{ik} \leq (n_2 - 1) y_{\bar{k}} + y_{n-n_2-1+1} \tag{5.46}$$

Para que esta desigualdad pueda ser faceta, se deben cumplir las hipótesis presentadas para la desigualdad de conjunto independiente. En particular, las cliques de un elemento no son tenidas en cuenta, lo cual es razonable ya que no presentan ninguna restricción en nuestro problema. Además, se debe cumplir que no se puede agregar un nuevo elemento al conjunto mientras se mantiene el cardinal del conjunto independiente máximo. Esto significa que las cliques que pueden definir facetas son las cliques maximales, es decir las que no se les puede agregar otro vértice del hipergrafo para el cual también existan toda las hiperaristas de tamaño n_2 con el resto de los nodos de la clique.

Un punto importante a tener en cuenta es que el caso donde el tamaño del conjunto independiente máximo es 1 fue dejado de lado en la demostracción presentada para la desigualdad de conjunto independiente. Esto hace entonces que la desigualdad asociada a las cliques $K_{n_1}^2$ no se encuentren contempladas dentro de la demostración. Lo que es más, esta es la única estructura no contemplada en dicha demostración. Si el cardinal del conjunto independiente máximo de un conjunto es 1, eso significa que ningún par de ítems puede ser asignado junto. La restricción de que dos ítems no puedan ir juntos, solamente se puede dar si explícitamente existe la arista entre los dos ítems como vemos en la siguiente proposición

Proposición 3. *Ningún conjunto de hiperaristas de tamaño mayor a 2 puede implicar una hiperarista de 2 ítems.*

Demostración. Podemos tomar un hipergrafo que contenga todas las hiperaristas de tamaño 3 (por lo cual se encuentran implicadas absolutamente todas las de tamaño mayor) y todas las hiperaristas de tamaño 2, menos la que relaciona a dos ítems particulares i_1 e i_2. Este es el caso de mayor restrictividad posible sin tener explícitamente la relación entre i_1 e i_2. Aún en dicho caso, la asignación en donde i_1 e i_2 van a un mismo contenedor y todo el resto de los ítems se asigna de manera individual es factible. Por lo cual, podemos asumir que ninguna combinación de hiperaristas de tamaño 2 o de mayor orden puede implicar una hiperarista de tamaño 2 que no se encuentre explícitamente en el hipergrafo. □

Este resultado es de particular interés ya que, en un problema de asignación como el tratado en este trabajo, la aristas de tamaño 2 contienen información muy particular. Estas aristas dicen que en ninguna asignación factible los dos ítems involucrados pueden estar en el mismo contenedor. De hecho, si se tiene una clique de aristas de tamaño 2, se sabe que nunca pueden ir juntos y hasta se podrían preasignar cada uno a un contenedor diferente como punto de partida. Con las hiperaristas de tamaño 3 o más, la proposición anterior nos dice que esto ya no es cierto. No importa que estructura presente el hipergrafo, si dos ítems no tienen arista explícitamente involucrando a ellos dos, van a existir asignaciones factibles en donde se encuentren en el mismo contenedor.

Dado este resultado, queda claro entonces que la única estructura que tiene conjunto independiente máximo de tamaño 1, es una clique con aristas de tamaño 2.

Luego, para demostrar la condición de faceta para la desigualdad de conjunto independiente para las cliques de aristas de tamaño 2, en primer lugar vemos que la desigualdad en este caso particular queda de la siguiente manera:

$$\sum_{i \in S} x_{i\bar{k}} \leq y_{\bar{k}} \tag{5.47}$$

Para realizar entonces la demostración en este caso, los resultados a mostrar (en el caso de que $\chi^b < \bar{k}$), son los siguientes:

$$\alpha_{ik} = \alpha_{i1} \quad \forall i \in S \quad \forall k \neq \bar{k}, n \tag{5.48}$$
$$\alpha_{i\bar{k}} = \alpha_{i1} - \beta_{\bar{k}} \quad \forall i \in S \tag{5.49}$$
$$\alpha_{in} = \alpha_{i1} - \beta_n \quad \forall i \in S \tag{5.50}$$
$$\alpha_{ik} = \alpha_{i1} \quad \forall i \notin S \quad \forall k \neq \bar{k}, n \tag{5.51}$$
$$\alpha_{i\bar{k}} = \alpha_{i1} \quad \forall i \notin S \tag{5.52}$$
$$\alpha_{in} = \alpha_{i1} - \beta_n \quad \forall i \notin S \tag{5.53}$$
$$\beta_k = 0 \quad \chi^b < k < n \quad k \neq \bar{k} \tag{5.54}$$
$$\tag{5.55}$$

Los resultados a demostrar son una particularización de las ecuaciones demostradas en la demostración general de conjunto independiente, en donde ahora $n - r + 1$ no es otra cosa que n por lo que algunos casos y variables se colapsaron en uno solo. La demostración de todos los resultados excepto (5.49) salen de manera análoga a lo realizado para la demostración general de conjunto independiente. En el caso de (5.49), fue el único momento en la demostración en el que se utilizó que $r \geq 2$ para poder generar puntos que tuvieran dos representantes de S en $\bar{k}$ para luego mover uno solo de ellos y así poder probar el resultado correspondiente al coeficiente $\alpha_{i\bar{k}}$. En este caso, para mostrar que esa identidad se satisface, se puede generar una asignación que utilice $\bar{k} - 1$ contenedores (la cual existe en el caso $\chi^b < \bar{k}$) y luego se mueve cualquiera de los ítems de S al contenedor $\bar{k}$, logrando demostrar así el resultado faltante para el caso particular de estas cliques. En el caso de que $\chi^b \geq \bar{k}$, la ecuación (5.49) no es tal, si no que lo que hay que demostrar en ese caso es que $\alpha_{i_1\bar{k}} - \alpha_{i_1 1} = \alpha_{i_2\bar{k}} - \alpha_{i_2 1}$ para todos los pares i_1,i_2 de ítems en S. Lo cual se demuestra facilmente tomando asignaciones de n contenedores e intercambiando los ítems i_1 e i_2 entre los contenedores 1 y $\bar{k}$.

De esta manera, queda demostrado el caso particular de la desigualdad de conjunto independiente para todos los tipos de cliques posibles en el hipergrafo.

Hiperaristas

Otra estructura muy simple en la que se conoce su conjunto independiente máximo es la de hiperarista. Al tener una hiperarista de tamaño r, sabemos que $r-1$ ítems de la hiperarista pueden ser asignados juntos. Para eso es importante estar en el caso de que no existen hiperaristas incluídas en otras, ya que esto podría hacer que el cardinal del conjunto independiente máximo sea menor que $r - 1$.

Entonces, para que la desigualdad de conjunto independiente instanciada en una hiperarista genere una faceta del poliedro, se deben cumplir nuevamente todas las hipótesis enunciadas en el caso general. En particular, la hipótesis que indica que el conjunto elegido (en este caso la hiperarista), es maximal en el sentido de poseer un conjunto independiente máximo de determinado tamaño, nos está indicando que las hiperaristas definirán facetas si no se encuentran englobados en una estructura más restrictiva, como por ejemplo una clique.

Lo que esto quiere decir es que, si bien el modelo presentado es correcto ya que las restricciones que predican sobre las hiperaristas capturan correctamente la noción de que un cierto conjunto de ítems no puede ser asignado juntos, dichas restricciones pueden ser reforzadas. El refuerzo se realiza simplemente reemplazando la restricción original por la que surge de tomar la desigualdad de conjunto independiente particularizada en una hiperarista dada. Dependiendo de la combinación entre la hiperarista elegida, el $\bar{k}$ y el χ^b de la instancia, este reemplazo puede resultar en una restricción de igual fuerza en algunos pocos casos, de mayor fuerza en la mayoría de los casos, o bien puede resultar en una faceta cuando se satisfagan todas las hipótesis necesarias.

Ciclo

Al referirse a estructuras con conjunto independiente máximo conocido, otra estructura que aparece recurrentemente es la de ciclo. Por ejemplo, si un ciclo no presenta cuerdas, es sabido que el conjunto independiente máximo se alcanza con la parte entera de dividir la cantidad de vértices por dos. Cuando en vez de un grafo se tiene un hipergrafo de conflictos, la noción de ciclo tiene que ser redefinida dado que existen diferentes maneras de generalizar el concepto utilizado en grafos simples para hipergrafos.

Una de las primeras definiciones de ciclos para hipergrafos se puede encontrar en [6], conjuntamente con varios de los primeros resultados generalizados en hipergrafos. En dicho libro, se puede encontrar la siguiente definición:

Sea H un hipergrafo conformado por el conjunto de vértices X y el conjunto de hiperaristas E. Sea k un entero mayor o igual a 2. Un *ciclo* de longitud k es una secuencia $(x_1, E_1, x_2, E_2, x_3, E_3, \ldots, x_k, E_k, x_1)$ que cumple:

1. $E_1, E_2, \ldots, E_k$ son hiperaristas distintas de H.
2. $x_1, x_2, \ldots, x_k$ son vértices distintos de H.
3. $x_i, x_{i+1} \in E_i \quad 1 \leq i \leq k-1$
4. $x_k, x_1 \in E_k$.

Es decir, que en esta definición se considera un ciclo a una secuencia de vértices distintos que se encuentran unidos por al menos una hiperarista distinta. Si bien la definición guarda una relación obvia con la análoga en grafos, es necesariamente distinta en el hecho de que para hipergrafos los vértices involucrados en formar un ciclo no son solamente los que van a estar en el ciclo, sino que los vértices que están por fuera juegan un rol importante dado que las aristas entre vértices ahora tienen a más de dos involucrados.

Esta diferencia hace que sea mucho más dificil obtener resultados generales para esta definición de ciclos en cuanto al cardinal del conjunto independiente máximo y, luego, a la cantidad de contenedores necesarios para asignar los ítems en nuestro problema. El caso más extremo se consigue pensando en un hipergrafo conformado de la siguiente manera. Tomamos un grafo que sea un ciclo simple sin cuerdas de m vértices. Luego, en vez de unir a cada par consecutivo de vértices con una clásica arista, los unimos con una hiperarista de tamaño 3 que involucra a estos dos vértices y a un tercero por fuera del ciclo. En este caso es claro que el cardinal del conjunto independiente máximo es m, ya que podemos asignar todos los ítems a un mismo contenedor y no violar ninguna de las restricciones. Este es un caso extremo para la caracterización del conjunto independiente máximo ya que nos indica que esta estructura puede no presentar ningún tipo de restricción.

De esta manera, no podemos definir un resultado general para la estructura de ciclos ya que bajo la misma definición caen estructuras con cardinal de conjunto independiente muy dispar. De todas formas, sí se puede buscar la caracterización no de todos los ciclos en general, si no de algunos ciclos particulares. Dos particularizaciones de ciclos en hipergrafos que tienen cierto estudio son los *loose cycle* y *tight cycle*, los cuales pueden encontrarse por ejemplo en [53]. Por cuestiones de simplicidad, la mayoría de las referencias a estos tipos de ciclos se hace asumiendo la regularidad en el tamaño de las hiperaristas involucradas. Es decir, que todas poseen el mismo cardinal k.

- Los *loose cycles* o *ciclos sueltos*, son ciclos conformados por una sucesión de hiperaristas diferentes que tiene intersección en tan solo un vértice. Si tomamos un ciclo de este tipo que no contenga *cuerdas*, es decir otras hiperaristas que involucren vértices del ciclo, es simple ver cual es el cardinal del conjunto independiente máximo. Se puede tomar todos los vértices que no están en las intersecciones para cada una de las hiperaristas, acumulando $k-2$ por cada una de las hiperaristas del ciclo. Luego, para los vértices en la intersección, sucede algo análogo a lo que ocurre en el caso de ciclos en grafos simples. Nunca podremos tomar dos consecutivos de ellos, porque al ya haber tomado todos los que están en la intersección, si

tomamos dos consecutivos entonces estaremos tomando la hiperarista en su totalidad. De esta manera, lo máximo que se le puede agregar al conjunto independiente es la parte entera de dividir por 2 los vértices que están en la intersección.

- Los *tight cycles* o *ciclos ajustados*, son ciclos que se forman por una sucesión de hiperaristas que tiene una intersección casi completa de sus vértices. Cada hiperarista se interseca en todos sus vértices menos 1 con la siguiente. Formalmente, un *tight cycle* es una secuencia de n vértices $(v_0, \ldots, v_{n-1})$ tal que $(v_i, \ldots, v_{i+k-1})$, con $k < n$, es una arista para cada i, en donde todos los subíndices se toman módulo n. En este caso, y nuevamente asumiendo la no existencia de otras hiperaristas que involucren vértices del ciclo, se puede llegar a caracterizar el cardinal del conjunto independiente máximo mediante una asignación golosa. Si se toman los primeros $k-1$ vértices, es claro que no se puede tomar el siguiente porque se completaría la primera hiperarista. Luego, a partir del vértice tomado como intervalo, nuevamente se pueden tomar los siguientes $k-1$ vértices y no más alla de eso ya que, por la definición del ciclo, todos los conjuntos de k vértices consecutivos conforman una hiperarista. De esta manera, en un ciclo que involucre n vértices, el conjunto independiente máximo involucrará a $n - \lfloor \frac{n}{k} \rfloor$.

Si bien no se puede caracterizar el conjunto independiente máximo para la definición general y abarcativa de ciclo, se mostró cómo se puede caracterizar en algunos casos particulares, existiendo seguramente muchas otras particularizaciones de ciclos en donde es simple determinar el cardinal del conjunto independiente máximo.

5.2.2. Suma reforzada de hiperarista a partir de un contenedor

En la sección anterior se mostró una familia de desigualdades que estaban determinadas por el conjunto independiente máximo de un conjunto de ítems dado. En particular, cuando una hiperarista no está incluída en estructuras particulares, también se puede ver como un conjunto de vértices para los cuales se puede escribir la desigualdad presentada. Sin embargo, a partir de la desigualdad de hiperarista básica que se encuentra en el modelo, existen otras maneras de realizar un refuerzo de variables para conseguir una familia de desigualdades fuerte. Recordando la desigualdad de hiperarista, la misma dice que a lo sumo $r-1$ ítems pueden ser asignados a un contenedor dado, donde r es el tamaño de la hiperarista. Luego, dado que el uso de los contenedores se encuentra ordenado, se puede reforzar la desigualdad de hiperarista de la siguiente manera:

$$\sum_{k=\bar{k}}^{n} \sum_{i \in H} x_{ik} \leq (r-1) y_{\bar{k}} + y_{\bar{k}+1} \tag{5.56}$$

Esta desigualdad es válida ya que la sumatoria del lado izquierdo suma sobre los contenedores mayores o iguales a $\bar{k}$. Luego, cualquier ítem de la hiperarista que se asigne a alguno de estos últimos colores estará obligatoriamente encendiendo la variable $y_{\bar{k}}$. Si se quisieran asignar los r ítems a toda la cola de contenedores, necesariamente no pueden ir todos juntos en $\bar{k}$ justamente porque conforman una hiperarista, por lo que se tendrá que encender la variable $y_{\bar{k}+1}$, por lo que la desigualdad es válida para P_{base}.

Al obtener la desigualdad, se experimentó con diferentes poliedros para analizar la dimensión de la cara asociada a la desigualdad, teniendo en cuenta las particularidades del hipergrafo asociado a cada instancia y la elección del índice de contenedor desde que se empieza a sumar. Se observó que en un gran conjunto de instancias la dimensión de la cara se redujo solamente en 1 con respecto a la dimensión del poliedro, mostrando así que en varios casos la cara definida por la desigualdad resulta ser faceta.

Proposición 4. *Dado un contenedor $\bar{k}$ y una hiperarista H de tamaño r*

$$\sum_{k=\bar{k}}^{n} \sum_{i \in H} x_{ik} \leq (r-1) y_{\bar{k}} + y_{\bar{k}+1} \tag{5.57}$$

es válida para P_{base}. Además, bajo ciertas hipótesis, la desigualdad define una faceta de P_{base}.

Demostración. La validez de la desigualdad fue explicada anteriormente para derivar la formula en sí misma y combina la naturaleza de las hiperaristas con el rasgo distintivo del modelo estudiado, que son los contenedores usados de manera incremental.

Para demostrar que esta desigualdad define una faceta para nuestro poliedro, nuevamente vamos a demostrar que el sistema minimal de la cara posee exactamente una restricción más que el sistema minimal del poliedro.

Como se esgrimió anteriormente, las desigualdades generadas por esta familia no siempre resultan ser facetas. Sin embargo, a continuación se listan las hipótesis necesarias y suficientes para que la desigualdad defina una faceta del poliedro.

Hipótesis 7. $\bar{k} < n - r + 1$

Esta hipótesis resulta necesaria ya que de lo contrario se pueden derivar las siguientes igualdades para la cara:

- Si $\bar{k} > n - r + 1$ implicaría $y_{\bar{k}+1} = 0$ porque de lo contrario, si se usase el contenedor $y_{\bar{k}+1}$, se tendrían que usar r ítems en la *cola* de los contenedores para cumplir la condición de la cara. Sin embargo, quedarían al menos $n - r + 1$ contenedores antes de $\bar{k}$ y solo $n - r$ ítems para rellenarlos, lo cual sería imposible.

- Si $\bar{k} = n - r + 1$ entonces o bien no se usa el contenedor $\bar{k} + 1$, y por ende nadie puede ser asignado a él, o bien lo utilizo, pero entonces necesito r ítems de la hiperarista asignados a contenedores mayores o iguales a $\bar{k}$ para satisfacer las condiciones de la cara. Una vez usados r en la *cola* de los contenedores, quedarían $n - r$ ítems para los exactamente $n - r$ primeros contenedores que estarían antes de $\bar{k}$. Luego, en ambos casos es verdad que los ítems que no se encuentran en la hiperarista están restringidos a no ser asignados a $\bar{k} + 1$ (ni a ningún contenedor posterior). Esto hace que valgan las igualdades del tipo $x_{i\bar{k}+1} = 0$ para todos los ítems que no están en la hiperarista, resultando entonces en una cara de mucha menor dimensión a la deseada.

Hipótesis 8. $\chi^b \leq \bar{k}$

Esta hipótesis es necesaria para que no se generen las siguientes situaciones en la cara:

- Si $\bar{k} < \chi^b$ y $\bar{k} > 1$, lo que sucede es que las variables del lado derecho de la desigualdad están obligatoriamente encendidas por ser menores o iguales a χ^b. Luego, para cumplir la restricción por igualdad, los r ítems de la hiperarista están obligados a ir a contenedores mayores o iguales a $\bar{k}$ generando por ejemplo la familia de igualdades $x_{i1} = 0$ para todo ítem en la hiperarista.

- Si $\bar{k} < \chi^b$ y $\bar{k} = 1$, entonces la igualdad puede ser derivada del resto de las igualdades del sistema minimal, por lo que la desigualdad resulta superflua.

Hipótesis 9. *$V - H$ no es una clique considerando las hiperaristas de tamaño 2.*

Es decir, es necesario que al menos un par de ítems por fuera de la hiperarista H puedan ser eventualmente asignados juntos. Si esto no se cumpliese, entonces $\chi^b \geq n - r$. Por las hipotesis presentadas anteriormente, la única opción que todavía no resultaría cubierta es cuando $\chi^b = n - r = \bar{k}$. Luego, en este caso, se pueden dar dos situaciones para los puntos que pertenecen a la cara:

- El punto utiliza χ^b contenedores. Como fuera de H hay $n - r$ ítems y en este caso hay $n - r$ contenedores en uso, tiene que cumplirse que los ítems que no están en la hiperarista se separan de a uno en cada uno de los contenedores, dado que si son una clique de hiperaristas de tamaño 2 no pueden ir juntos.

- El punto utiliza más de χ^b contenedores. En ese caso, los r ítems de H van a la *cola* de los contenedores, pero alguien debe ir al contenedor 1 para que no quede vacío, y por la condición que se está planteado, solo podría ir un ítem de $V - H$.

Luego, en ambos casos se cumple $\sum_{i \notin H} x_{i1} = 1$, por lo que esta hipótesis es necesaria para que la desigualdad pueda resultar faceta.

Para realizar la demostración vamos a probar que toda igualdad satisfecha por los puntos de la cara se puede escribir como una combinación lineal de las igualdades del sistema minimal del poliedro, más la restricción actual considerada como una igualdad. Para poder asegurar esto, hace falta probar las siguientes igualdades.

$$\alpha_{ik} = \alpha_{i1} \quad \forall i \in H \quad \forall k < \bar{k} \tag{5.58}$$
$$\alpha_{ik} = \alpha_{i1} - \beta_{\bar{k}+1} \quad \forall i \in H \quad \forall k \geq \bar{k}, k \leq n-1 \tag{5.59}$$
$$\alpha_{in} = \alpha_{i1} - \beta_{\bar{k}+1} - \beta_n \quad \forall i \in H \tag{5.60}$$
$$\alpha_{ik} = \alpha_{i1} \quad \forall i \notin H \quad \forall k < \bar{k} \tag{5.61}$$
$$\alpha_{ik} = \alpha_{i1} \quad \forall i \notin H \quad \forall k \geq \bar{k}, k \leq n-1 \tag{5.62}$$
$$\alpha_{in} = \alpha_{i1} - \beta_n \quad \forall i \notin H \tag{5.63}$$
$$\beta_k = 0 \quad \chi^b < k < n, k \neq \bar{k}, \bar{k}+1 \tag{5.64}$$
$$\beta_{\bar{k}} = (r-1)\beta_{\bar{k}+1} \tag{5.65}$$

Cabe destacar que para derivar las igualdades anteriores se está utilizando que $\bar{k} > \chi^b$. En el caso de la igualdad entre $\bar{k}$ y χ^b, la condición (5.65) desaparece y el resto de la demostración es análoga por lo que no se hará dicha distinción.

En primer lugar, comenzaremos utilizando a los ítems que no pertenecen a la hiperarista H y que pueden ser asignados con otro ítem que no pertenezca a la hiperarista. Al menos un par de ellos debe existir dado que, por hipótesis, estamos en el caso en donde los ítems que están por fuera de H no forman una clique con hiperaristas de tamaño 2.

Una vez identificados al menos dos de estos ítems, se puede armar un punto que cumpla la restricción de la cara en las que dos de estos ítems estén juntos asignados al contenedor número 1. Luego, en el resto de los contenedores se coloca un solo ítem, exceptuando por el último contenedor que se deja vacío. Para que sea posible armar un punto de este tipo que pertenezca a la cara es importante notar que deben estar los r ítems de H en contenedores a partir de $\bar{k}$. Por las hipótesis de los rangos posibles para $\bar{k}$, siempre hay una cantidad suficiente de contenedores para realizar dicha asignación. Una vez que se obtiene un punto de este tipo, se puede pasar alguno de los dos ítems del contenedor 1, al contenedor n. De la interacción entre estos dos puntos se obtiene la relación:

$$\alpha_{i1} = \alpha_{in} + \beta_n \tag{5.66}$$

Es decir, vale (5.63) para todos los ítems que puedan ser asignados conjuntamente con otro ítem no perteneciente a H.

En el procedimiento realizado, el contenedor número 1 resulta indistinguible de todos los contenedores anteriores al $\bar{k}$ para generar este tipo de puntos. Cualquiera de ellos puede albergar los dos ítems en cuestión dejando un ítem en el resto de los contenedores, exceptuando el último. Es por esto que la relación:

$$\alpha_{ik} = \alpha_{in} + \beta_n \tag{5.67}$$

es válida para todos los k anteriores a $\bar{k}$, probando (5.61) nuevamente solo para los ítems que tienen permitido compartir contenedor.

Una vez que se encuentra demostrado (5.61), y (5.63) para los ítems *no conflictivos*, es simple ver que estas propiedades también valen para los *conflictivos*. Esto se logra realizando intercambios de ítems en asignaciones que utilicen todos los contenedores.

Por ejemplo, para demostrar (5.63) para los ítems que no pueden compartir contenedor se puede hacer de la siguiente manera. Se asigna el ítem en cuestión al contenedor 1. Se deja en el contenedor n algún ítem *no conflictivo* para el cual ya demostramos (5.63). A todo el resto de los contenedores se asigna un ítem. Esto se puede cumplir en la cara porque los contenedores desde $\bar{k}$ hasta $n-1$ alcanzan para asignar a los r ítems de H. Una vez que se tiene este punto, se intercambian el ítem en cuestión del contenedor 1 con el del contenedor n, consiguiendo la relación:

$$\alpha_{i_1 1} + \alpha_{i_2 n} = \alpha_{i_1 n} + \alpha_{i_2 1} \tag{5.68}$$

En donde i_1 es el ítem conflictivo que se envió al contenedor 1, i_2 es uno de los ítems no conflictivos de $V - H$. Dado los resultados demostrados anteriormente se puede reemplazar $\alpha_{i_2 n}$ por $\alpha_{i_2 1} - \beta_n$. Al realizar el reemplazo se puede ver que se cumple lo pedido en (5.63).

De la misma manera, se puede demostrar (5.61) para todos los demás ítems *conflictivos*.

A continuación, probaremos (5.62). Para esto dividiremos la demostración en dos casos. Un primer caso cuando $r = 2$, y otro caso cuando $r \geq 3$.

Cuando $r = 2$, tiene que existir un ítem, i_1, por fuera de H que pueda ser asignado junto a algún ítem de H, i_h, ya que por hipótesis no tenemos nodos universalmente conflictivos. Tomamos entonces un punto que utiliza $n-1$ contenedores en donde i_1 se encuentra asignado al contenedor 1 e i_h se encuentra en el contenedor $\bar{k}$, y algún contenedor entre 1 y $\bar{k}-1$ tiene dos ítems. Luego, creamos un segundo punto factible para la cara que proviene de mover el ítem i_1 al contenedor $\bar{k}$. Si el ítem i_1 se encontraba acompañado en su contenedor, el pasaje se puede hacer directamente. Si se encontraba solo, entonces existía algún otro contenedor entre el 2 y el $\bar{k}-1$ con dos ítems por fuera de H. Se puede tomar uno de estos ítems y pasarlo al contenedor 1 para que el punto sea factible y, dados los resultados probados, este movimiento no tendrá injerencia ya que los coeficientes asociados a ese par de contenedores para ese ítem son iguales. Si $\bar{k}$ fuese 2, entonces χ^b también es 2. Por lo tanto en el punto armado, el ítem i_1 puede ir acompañado de otro ítem que no está en H.

De los dos puntos mencionados sale la relación

$$\alpha_{i_1 1} = \alpha_{i_1 \bar{k}} \tag{5.69}$$

Este procedimiento en realidad se puede realizar para cualquier contenedor entre $\bar{k}$ y $n-1$, por lo que se demuestra (5.62) para el ítem i_1 en particular.

En el caso $r \geq 3$, también probaremos (5.62) para un ítem en particular, valiéndonos primero de un resultado intermedio. Como $r \geq 3$, podemos armar una asignación que utilice $n-1$ contenedores en donde en el contenedor $\bar{k}$ se coloquen dos ítems de H y en todo el resto de los contenedores haya un único ítem, cumpliendo la restricción de la cara. A partir de este punto, generamos otro punto factible para la cara que proviene de pasar uno de los dos ítems que esta en $\bar{k}$, i_h, al último contenedor. De estos dos puntos obtenemos el resultado

$$\alpha_{i_h \bar{k}} = \alpha_{i_h n} + \beta_n \tag{5.70}$$

Si bien hicimos esto para el contenedor $\bar{k}$, lo mismo se puede hacer con cualquier contenedor entre $\bar{k}$ y $n-1$.

Una vez que tenemos este resultado para i_h, tomamos una asignación que utilice $n-1$ contenedores en donde haya 2 ítems por fuera de H en el contenedor 1 y en donde i_h esté asignado a $\bar{k}$. A partir de este punto, generamos uno nuevo que nace de realizar dos movimientos. Por un lado pasamos un ítem del contenedor 1, i_1, al contenedor $\bar{k}$. Además, pasamos i_h del contenedor $\bar{k}$ al último contenedor. Así, con estos dos puntos y el resultado anterior sobre i_h, obtenemos

$$\alpha_{i_1 1} = \alpha_{i_1 \bar{k}} \tag{5.71}$$

Nuevamente esto se podría realizar con cualquier contenedor entre $\bar{k}$ y $n-1$ por lo que se demuestra todo (5.62) para i_1, que es algún ítem que posee un ítem compatible por fuera de H.

Como acabamos de argumentar, en ambos casos podemos mostrar que existe al menos un ítem para el que podemos probar (5.62). Recordando que aparte ya se encuentra demostrado (5.61) y (5.63) para todos los ítems pertinentes, podremos demostrar (5.62) para todos los ítems. Esto se puede realizar utilizando asignaciones en donde se usen todos los contenedores. Para cumplir la restricción de la cara, todos los ítems de la hiperarista deben estar en contenedores a partir del $\bar{k}$. Dada la hipótesis $\bar{k} < n - r + 1$, el ítem i_1 para el cual ya probamos el resultado, tiene permitido ir a cualquier contenedor entre $\bar{k}$ y $n - 1$. A partir de esta situación, para cada ítem i_2 que debamos probar (5.62), podemos armar un punto que utilice n contenedores en donde i_2 se asigne al contenedor 1 e i_1 se asigne al $\bar{k}$. Luego intercambiamos i_1 e i_2 y obtendremos la relación

$$\alpha_{i_2 1} = \alpha_{i_2 \bar{k}} \tag{5.72}$$

Esto lo podemos realizar con todo contenedor entre $\bar{k}$ y $n - 1$. A su vez, i_2 puede ser cualquier ítem no perteneciente a la arista. De esta manera, queda demostrado (5.62) para todos los ítems pertinentes.

Para demostrar los resultados pertinentes a los ítems pertenecientes a la hiperarista comenzamos por crear una asignación en donde se usen exactamente $\bar{k}$ contenedores. Esta asignación es posible dado que $\bar{k}$ no es menor a χ^b. Para que esta asignación corresponda a un punto que pertenece a la cara, el contenedor $\bar{k}$ debe tener $r - 1$ ítems de H asignados. Al ítem de H que no se encuentra en $\bar{k}$, lo asignamos al contenedor 1. Luego, se puede mover dicho ítem del contenedor 1 al contenedor $\bar{k}+1$ y se siguen cumpliendo las restricciones de la cara. Es posible que al realizar este movimiento el contenedor 1 quede vacío pero, análogamente a lo que se hizo anteriormente, se puede solucionar este problema enviando algun ítem no perteneciente a H y sin dejar otro contenedor vacío dado que por las hipótesis, siempre alcanzan los ítems para asignar a dos no pertenecientes a H juntos. Al realizar estos movimientos se obtiene la relación:

$$\alpha_{i1} = \alpha_{i\bar{k}+1} + \beta_{\bar{k}+1} \tag{5.73}$$

Siendo i un ítem de H. Es importante destacar que esto se puede realizar para todo ítem de H, ya que al no haber hiperaristas superfluas entonces todo subconjunto de $r - 1$ ítems de H puede ser asignado junto. Además, este procedimiento es indistinto para todos los contenedores que están antes de $\bar{k}$. De esta manera, queda probado (5.58).

Para continuar se utilizan asignaciones de n contenedores. Para que estos puntos pertenezcan a la cara, es necesario que los r ítems de H estén en los contenedores desde $\bar{k}$. Dadas las hipótesis explicitadas, hay al menos un contenedor más disponible desde $\bar{k}$ para algún ítem que no sea de H. En primer lugar, podemos poner a este ítem (i_1) en el contenedor n y luego se puede generar un nuevo punto intercambiando este ítem con alguno (i_2) de H que está en algún contenedor a partir de $\bar{k}$. Haciendo este intercambio nos queda la relación:

$$\alpha_{i_1 n} + \alpha_{i_2 k} = \alpha_{i_2 n} + \alpha_{i_1 k} \tag{5.74}$$

Sin embargo, al ser i_1 un ítem en $V - H$ podemos utilizar los resultados ya demostrados, quedando:

$$\alpha_{i_2 k} = \alpha_{i_2 n} + \beta_n \tag{5.75}$$

Dado que este intercambio se puede hacer con todo ítem en H y con todo contenedor desde $\bar{k}$ hasta $n - 1$, queda demostrado (5.60). Análogamente, si i_2 se posicionase originalmente en algún contenedor entre $\bar{k}$ y $n-1$ y se hiciese el mismo intercambio, se obtendría la relación que demuestra (5.59).

Una vez que se encuentran demostradas todas las propiedades para los α es simple ver las relaciones faltantes, dado que se conoce las igualdades que se cumplen al mover cualquier ítem entre cualquier par de contenedores.

En primer lugar, demostraremos todos los β_k que deben tener un valor nulo. Los contenedores que tienen un valor nulo son los que están entre χ^b y $\bar{k}$ (que podría no ser ninguno y en ese caso

no hay resultado a demostrar para ellos) y los que están entre $\bar{k}+1$ y n (que también podrían no existir). Para el primer grupo, podemos comenzar de una asignación de χ^b contenedores, que por hipótesis existe. A partir de esta asignación, podemos pasar a algún ítem que no se encuentre asignado solo al contenedor χ^b+1. Como para todo ítem se encuentra probado que sus coeficientes α son indistinguibles para los contenedores anteriores a $\bar{k}$, se desprende entonces que $\beta_{\chi^b+1}=0$. Análogamente se puede ir *desprendiendo* de a un ítem de la asignación previa y enviándolo al siguiente contenedor vacío. De esta manera se demuestra la nulidad de los coeficientes hasta el contenedor $\bar{k}$ exclusive. Para el contenedor $\bar{k}$ no se puede seguir haciendo el mismo procedimiento porque el punto entonces no cumpliría las restricciones de la cara.

Luego, para los contenedores entre $\bar{k}+1$ y n exclusive, lo que se puede hacer es comenzar de una asignación de $\bar{k}+1$ contenedores en donde $r-1$ ítems de H estén en $\bar{k}$ y 1 de ellos esté en $\bar{k}+1$. A partir de este punto se puede ir moviendo de a un ítem de $\bar{k}$ al primer contenedor vacío. Esto genera puntos factibles para la cara y, por los resultados demostrados anteriormentes, va derivando que los sucesivos β_k involucrados deben ser nulos. En el caso de que ya no alcancen los ítems de H para ir desplazando a los contenedores vacíos, y que todavía queden contenedores vacíos que no sean el último, entonces deben existir ítems por fuera de H que estén asignados a los contenedores anteriores a $\bar{k}$ en compañía. Estos ítems excedentes son los que utilizamos para desplazar a los contenedores vacíos para continuar con la demostración. De esta manera queda demostrado (5.64).

Por último, falta demostar (5.65). Para esto tomamos una asignación de $\bar{k}-1$ contenedores. Esto solo es posible si $\bar{k}>\chi^b$. Como se explicó anteriormente, este va a ser el único punto en donde se supone esto, para el resto de los resultados no fue necesario. En el caso que se cumpla la igualdad, el resto de la demostración sigue vigente y no es necesario demostrar (5.65).

Una vez que se tiene el punto donde se utilizan $\bar{k}-1$ contenedores, se pueden pasar $r-1$ ítems a $\bar{k}$ para generar un punto factible para la cara. En el caso de que este pasaje genere contenedores intermedios vacíos, esto siempre puede ser subsanado moviendo ítems que no estén en H para los cuales ya probamos que su coeficiente α_k es el mismo para todo k distinto de n, por lo que las relaciones derivadas serán las mismas. Al hacer este pasaje, obtenemos la relación:

$$\sum_{i=i_{H_1}}^{i_{H_{r-1}}} \alpha_{i1} = \sum_{i=i_{H_1}}^{i_{H_{r-1}}} \alpha_{i\bar{k}} + \beta_{\bar{k}} \tag{5.76}$$

Siendo $\{i_{H_1} \dots i_{H_{r-1}}\}$ los $r-1$ ítems pertenecientes a H que decidimos mover al contenedor $\bar{k}$. Cabe destacar que por los resultados demostrados anteriormente, no importa realmente de que contenedor provengan los ítems, sus coeficientes se pueden intercambiar por el coeficiente del contenedor 1. Luego, utilizando (5.59) y reemplazando queda:

$$\sum_{i=i_{H_1}}^{i_{H_{r-1}}} \alpha_{i1} = \sum_{i=i_{H_1}}^{i_{H_{r-1}}} \alpha_{i1} - (r-1)\beta_{\bar{k}+1} + \beta_{\bar{k}} \tag{5.77}$$

Simplificando:

$$(r-1)\beta_{\bar{k}+1} = \beta_{\bar{k}} \tag{5.78}$$

Quedando demostrado (5.65)

De esta manera, quedan demostrados todos los resultados necesarios para mostrar que la desigualdad define una faceta del poliedro bajo las condiciones mencionadas.

□

5.2.3. Desigualdad de vecindad

En esta sección presentamos una familia de desigualdades que se genera a partir de tomar un vértice y su vecindad en el hipergrafo de conflictos, entendiendo por vecindad a todos aquellos ítems con los cuales comparte alguna hiperarista. Como ya mencionamos anteriormente, los problemas que utilizan un grafo de conflictos en vez de un hipergrafo guardan una información particularmente

fuerte en cada una de sus aristas, que no se traduce al tener hiperaristas de tamaño mayor o igual a 3. Una arista en un grafo de conflictos indica que si sucede algo con uno de los vértices involucrados (como ser asignado a un contenedor), entonces no sucede con el otro vértice. En un hipergrafo esto no es cierto, ya que las hiperaristas contienen información restrictiva para un conjunto en su totalidad, y no para pares de vértices.

En el problema de coloreo de grafos, el hecho de asignarle un color a un vértice implica inmediatamente que todos sus vecinos no pueden ser coloreados con dicho color. Luego, si r es el cardinal del conjunto independiente máximo de la vecindad de un vértice $\bar{i}$. Entonces la siguiente desigualdad es válida para cualquier color [45]:

$$\sum_{i \in N(\bar{i})} x_{i\bar{k}} + r x_{\bar{i}\bar{k}} \leq r y_{\bar{k}} \tag{5.79}$$

Esta desigualdad es válida porque se está en uno de tres casos posibles:

- El contenedor $\bar{k}$ no se usa, entonces ningún ítem puede ser asignado a $\bar{k}$.
- El contenedor $\bar{k}$ se usa con $\bar{i}$, entonces nadie de su vecindad puede ser asignado a $\bar{k}$.
- El contenedor $\bar{k}$ se usa pero no con $\bar{i}$, entonces seguro no pueden ir más que r de su vecindad ya que es el cardinal del conjunto independiente máximo.

Sin embargo, en nuestro problema que posee un hipergrafo de conflictos, esta desigualdad deja de ser válida. Pensemos en un ejemplo muy simple de un hipergrafo con tan solo una hiperarista de tamaño 3 que involucra a los primeros 3 ítems. Una posible desigualdad particularizada para este caso quedaría:

$$x_{21} + x_{31} + 2x_{11} \leq 2y_1 \tag{5.80}$$

Pero es claro que como la hiperarista es de tamaño 3, entonces se puede asignar a los ítems 1 y 2 al contenedor 1, violando la desigualdad. La diferencia importante con el caso original, es que en ese caso era claro que la presencia del ítem 1 en el contenedor 1, hacía imposible la presencia de cualquier otro ítem de su vecindad, lo cual ya no sucede.

Para poder generalizar esta desigualdad al caso donde tenemos un hipergrafo, debemos generalizar la noción de la restricción que impone sobre su vecindad el hecho de asignar un ítem particular a un contenedor dado. Nuevamente tomamos al ítem $\bar{i}$ sobre el que vamos a escribir la desigualdad y su vecindad $N(\bar{i})$. Llamamos r al cardinal del conjunto independiente máximo en la vecindad de $\bar{i}$, es decir $r = \alpha(N(\bar{i}))$. Ahora, para poder escribir la generalización, denominamos $\bar{r}$ a la cantidad máxima de ítems en $N(\bar{i})$ que pueden ser asignados juntos con $\bar{i}$ a un contenedor en particular. De esta manera, podemos seguir el razonamiento utilizado en la derivación de la desigualdad anterior para presentar la siguiente desigualdad:

$$\sum_{i \in N(\bar{i})} x_{i\bar{k}} + (r - \bar{r}) x_{\bar{i}\bar{k}} \leq r y_{\bar{k}} \tag{5.81}$$

Esta desigualdad recupera la validez para nuestro problema dado que nuevamente se presentan 3 posibles casos:

- El contenedor $\bar{k}$ no se utiliza y la desigualdad se cumple trivialmente.
- El contenedor $\bar{k}$ se utiliza pero no por $\bar{i}$, entonces al igual que antes podemos asegurar que no más de r ítems de su vecindad serán asignados a $\bar{k}$.
- El contenedor $\bar{k}$ se utiliza con el ítem $\bar{i}$, además por la definición de $\bar{r}$, sabemos que esa es la máxima cantidad de vecinos de $\bar{i}$ que lo pueden acompañar en el contenedor $\bar{k}$, haciendo que nuevamente el lado izquierdo sume r como máximo.

Se destaca que esta desigualdad se presenta como una generalización de la primera, ya que en el caso de un grafo el $\bar{r}$ es 0.

Si bien esta desigualdad es válida, no es lo suficientemente restrictiva para generar una faceta del poliedro asociado. Para reforzar la desigualdad se pueden utilizar dos ideas. En primer lugar, si el ítem $\bar{i}$ es asignado a algún contenedor que esté después del $n - r + 1$, entonces no va a ser posible asignar r ítems a $\bar{k}$ simplemente porque la cantidad de ítems no alcanzaría para rellenar todos los contenedores intermedios. Por cada contenedor más hacia el final que se asigna $\bar{i}$, un ítem menos que se puede asignar al $\bar{k}$. También, del mismo modo que se utilizó al momento de reforzar la desigualdad de conjunto independiente, el argumento de orden de los contenedores nuevamente hace que varias variables asociadas a la cola de los contenedores puedan ser sumadas del lado izquierdo. El uso de estas variables hace que las variables sobre el contenedor $\bar{k}$ no puedan sumar demasiado como para violar la desigualdad ya que necesitamos sacar ítems de $\bar{k}$ para rellenar los contenedores intermedios si queremos asignar ítems a los últimos contenedores. Estas ideas derivan en la siguiente desigualdad:

$$\sum_{i \in N(\bar{i})} x_{i\bar{k}} + (r-\bar{r}) x_{\bar{i}\bar{k}} + \sum_{k=n-r+2}^{n-\bar{r}} (k+r-1-n) x_{\bar{i}k} + \sum_{k=n-\bar{r}+1}^{n} (r-\bar{r}-1) x_{\bar{i}k} + \sum_{i \in V} \sum_{k=n-\bar{r}}^{n} x_{ik} \leq r y_{\bar{k}} + y_{n-\bar{r}} \tag{5.82}$$

Esta desigualdad condensa todas las ideas propuestas y nuevamente resulta en una generalización para hipergrafos reforzada de la desigualdad presentada en primer lugar.

La nueva desigualdad sí resulta faceta del poliedro asociado a nuestro modelo en varios casos, por lo que se procede con la pertinente demostración.

Proposición 5. *Dado un contenedor $\bar{k} < n - \bar{r}$ y un ítem $\bar{i}$ y asumiendo $\bar{r} < r$*

$$\sum_{i \in N(\bar{i})} x_{i\bar{k}} + (r-\bar{r}) x_{\bar{i}\bar{k}} + \sum_{k=n-r+2}^{n-\bar{r}} (k+r-1-n) x_{\bar{i}k} + \sum_{k=n-\bar{r}+1}^{n} (r-\bar{r}-1) x_{\bar{i}k} + \sum_{i \in V} \sum_{k=n-\bar{r}}^{n} x_{ik} \leq r y_{\bar{k}} + y_{n-\bar{r}} \tag{5.83}$$

es válida para P_{base}. Además, bajo ciertas hipótesis, la desigualdad define una faceta de P_{base}.

Demostración. En primer lugar haremos las observaciones pertinentes sobre la validez. La derivación realizada contiene ciertos casos bordes que son excluídos en la proposición para lograr la validez de la desigualdad.

Hipótesis 10. $\bar{k} < n - \bar{r}$

Para pensar la derivación de la desigualdad, se asumía que $\bar{k}$ era un contenedor que no está entre los últimos. Es decir, que $\bar{k} < n - \bar{r}$. En caso contrario, la desigualdad no resulta válida. Esto es porque se puede armar un punto en donde se utilice el contenedor $n - \bar{r}$ y no el $\bar{k}$. En este caso el lado derecho de la desigualdad sería 1. Luego, se asigna $\bar{i}$ a $n - \bar{r}$ junto con $\bar{r}$ de su vecindad y entonces el lado izquierdo ya sumaría 2, violando la desigualdad.

Hipótesis 11. $\bar{r} < r$

Al momento de realizar los refuerzos pertinentes, se pensó que si se quiere utilizar el contenedor $\bar{k}$, entonces la sumatoria de los ítems utilizados desde $n - \bar{r}$ en adelante se puede balancear con la variable de utilización del contenedor $n - \bar{r}$, ya que si bien la primera sumatoria puede sumar más que uno, solo lo puede hacer quitando ítems de $\bar{k}$, por lo que el lado izquierdo de la desigualdad nunca puede sumar lo suficiente como para violar la misma. Esto no sucede cuando $\bar{r} = r$, ya que se puede armar el siguiente punto infactible. Se asigna r ítems de la vecindad de $\bar{i}$ a $\bar{k}$, $\bar{i}$ al contenedor $n - r$ y algún otro ítem al contenedor $n - r + 1$. Este punto se puede conformar ya que las cantidades de ítems alcanzan y, sin embargo, viola la desigualdad.

Luego de enunciar las hipótesis necesarias para que la desigualdad sea válida, enunciamos las hipótesis para que la desigualdad pueda generar una faceta del poliedro asociado.

Hipótesis 12. $\bar{k} \leq n - r + 1$

Esta condición es necesaria porque de lo contrario se tendría que $\bar{k} > n - r + 1$. En ese caso, nunca se puede asignar r ítems al contenedor $\bar{k}$, simplemente porque la cantidad de ítems no alcanzaría para rellenar todos los contenedores anteriores. Luego, como nunca se puede asignar r ítems a $\bar{k}$, entonces para cumplir la restricción por igualdad se debe caer sí o sí en el caso de que $\bar{i}$ sea asignado o bien a $\bar{k}$, o bien a algunos de los contenedores desde $n-r+2$ para que pueda sumar en el lado izquierdo de la ecuación. Esto hace que, por ejemplo, se cumpla la igualdad $x_{\bar{i}1} = 0$, haciendo que la desigualdad no pueda generar una faceta.

Hipótesis 13. $\chi^b < n - r$

Si esta hipótesis no se cumpliese, entonces los puntos que pertenecen a la cara cumpliendo la restricción por igualdad nunca pueden tener más de un ítem asignado a varios contenedores. Por ejemplo, el contenedor 1 siempre tendría un solo ítem asignado generando una nueva igualdad cumplida por todos los puntos de la cara, que no se deriva del resto de las igualdades.

Hipótesis 14. *Todo ítem fuera de la vecindad de $\bar{i}$ tiene que poder ser asignado con algún conjunto de r ítems en $N(\bar{i})$, o bien tiene que poder ser asignado con $\bar{i}$ y algún conjunto de $\bar{r}$ ítems en $N(\bar{i})$.*

Recordemos que para los puntos de la cara se cumple alguna de las siguientes tres situaciones. El contenedor $\bar{k}$ no se usa. $\bar{i}$ y $\bar{r}$ de sus vecinos están asignados a $\bar{k}$. r vecinos de $\bar{i}$ están asignados a $\bar{k}$. De lo recientemente observado, esta hipótesis resulta necesaria porque de no cumplirse para algún ítem fuera de la vecindad de $\bar{i}$, entonces este ítem estaría obligado a nunca ser asignado al contenedor $\bar{k}$ generando así una nueva igualdad cumplida por todos los puntos de la cara, privando a la desigualdad de las condiciones necesarias para definir una faceta del poliedro.

Por último, presentamos una hipótesis que no resulta necesaria para que la desigualdad sea faceta pero simplifica el análisis a realizar

Hipótesis 15. $\bar{r} \geq 1$

Es importante destacar que con esta hipótesis no se está perdiendo ningún caso de interés, ya que cuando $\bar{r}$ es 0, se está en el caso de un grafo de conflictos con aristas de a pares de ítem, en donde la desigualdad ya se encuentra demostrada que es faceta. [45].

Para demostrar que la desigualdad presentada define una faceta del poliedro bajo las hipótesis consideradas, vamos a probar que toda igualdad satisfecha por los puntos de la cara definida se puede pensar como una combinación lineal del sistema minimal del poliedro más la restricción actual impuesta como una igualdad.

Para poder hacer dicha demostración, se debe mostrar la validez de los siguientes resultados:

$$\alpha_{ik} = \alpha_{in} + \beta_{n-\bar{r}} + \beta_n \quad \forall i \notin N(\bar{i}), i \neq \bar{i} \quad \forall k \leq n - \bar{r} - 1 \tag{5.84}$$

$$\alpha_{ik} = \alpha_{in} + \beta_n \quad \forall i \notin N(\bar{i}), i \neq \bar{i} \quad n - \bar{r} \leq k \leq n - 1 \tag{5.85}$$

$$\alpha_{ik} = \alpha_{in} + \beta_{n-\bar{r}} + \beta_n \quad \forall i \in N(\bar{i}) \quad \forall k < n - \bar{r} + 1, k \neq \bar{k} \tag{5.86}$$

$$\alpha_{i\bar{k}} = \alpha_{in} + \beta_n \quad \forall i \in N(\bar{i}) \tag{5.87}$$

$$\alpha_{ik} = \alpha_{in} + \beta_n \quad \forall i \in N(\bar{i}) \quad n - \bar{r} \leq k \leq n - 1 \tag{5.88}$$

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}n} + \beta_n + (r - \bar{r})\beta_{n-\bar{r}} \quad k \neq \bar{k}, k \leq n - r + 1 \tag{5.89}$$

$$\alpha_{\bar{i}\bar{k}} = \alpha_{\bar{i}n} + \beta_n \tag{5.90}$$

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}n} + \beta_n \quad n - \bar{r} \leq k \leq n - 1 \tag{5.91}$$

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}n} + \beta_n + (n - \bar{r} - k + 1)\beta_{n-\bar{r}} \quad n - r + 2 \leq k \leq n - \bar{r} - 1 \tag{5.92}$$

$$\beta_{\bar{k}} = r\beta_{n-\bar{r}} \tag{5.93}$$

$$\beta_k = 0 \quad \chi^b + 1 \leq k \leq n - 1, k \neq \bar{k}, n - \bar{r} \tag{5.94}$$

$$\tag{5.95}$$

En primer lugar, se observa que las igualdades presentadas son tales en el caso de que $\bar{k} > \chi^b$. En caso contrario, la restricción (5.93) desaparece y el resto de los resultados no sufren modificaciones.

Vamos a comenzar demostrando (5.85) y (5.88). Para esto se toman dos vértices no adyacentes y distintos de $\bar{i}$, los cuales existen gracias a las hipótesis que indican que todo ítem no es universalmente conflictivo y que $\bar{r} < r$. Construimos un punto factible de la cara en el cual estos dos ítems sean asignados al contenedor $n - \bar{r}$, $\bar{i}$ es asignado a $\bar{k}$ y el resto de los contenedores tiene un ítem cada uno, exceptuando el último. Luego, se genera un nuevo punto válido que surge de mover uno de los dos ítems que comparten contenedor, al contenedor n. De estos dos puntos surge el resultado:

$$\alpha_{in-\bar{r}} = \alpha_{in} + \beta_n \tag{5.96}$$

Si bien esto se hizo desde el contenedor $n-\bar{r}$, se podría haber hecho desde cualquier contenedor entre $n - \bar{r}$ y $n - 1$, ya que en todos esos casos los puntos pertenecen a la cara. Luego, con este resultado, queda demostrado (5.85) o (5.88) para los dos ítems en cuestión, dependiendo de si estos pertenecen o no a la vecindad de $\bar{i}$.

Una vez demostrados los resultados para estos dos ítems, se pueden generar puntos factibles que utilicen los n contenedores. Una forma de generar estos puntos es asignando el ítem $\bar{i}$ al contenedor $\bar{k}$ y uno de los ítems (i_1) para los que ya se demostró el resultado al contenedor n y un ítem para el que se quiere demostrar este resultado (i_2) al contenedor $n-\bar{r}$. Luego, se genera un nuevo punto factible que surge de intercambiar i_1 con i_2. De esta interacción se obtiene:

$$\alpha_{i_1 n} + \alpha_{i_2 n-\bar{r}} = \alpha_{i_2 n} + \alpha_{i_1 n-\bar{r}} \tag{5.97}$$

Dado que para i_1 ya se demostró la equivalencia necesaria, de este resultado se deriva la misma equivalencia para i_2 y el contenedor $n - \bar{r}$. Sin embargo, este procedimiento se puede realizar con todo ítem distinto de $\bar{i}$ y con todo contenedor entre $n - \bar{r}$ y $n - 1$, demostrando así (5.85) y (5.88) en su totalidad.

Ahora procedemos a demostrar (5.87). Para esto tomamos dos ítems que están en la vecindad de $\bar{i}$ y que pueden ser asignados juntos, los cuales existen porque, gracias a las hipótesis, se está en el caso donde se cumple $1 \leq \bar{r} \leq r$, por lo que $r \geq 2$ y entonces $|N(i)| \geq 2$. Formamos entonces un punto que utilice $n - 1$ contenedores en donde los dos ítems en cuestión se encuentren asignados juntos en $\bar{k}$ e $\bar{i}$ se encuentra asignado al contenedor $n - 1$, cumpliendo así la restricción para pertenecer a la cara. Desde este punto, se genera uno nuevo que nace de mover a uno de los dos ítems desde $\bar{k}$ al contenedor n. Al utilizar estos dos puntos se obtiene:

$$\alpha_{i\bar{k}} = \alpha_{in} + \beta_n \tag{5.98}$$

demostrando así (5.87) para los puntos de la vecindad que pueden compartir contenedor con otro ítem de la vecindad de $\bar{i}$. Luego, para el resto de los ítems, los que no pueden compartir, se genera un punto de la siguiente manera. Se utilizan los n contenedores, se asigna en $\bar{k}$ algún ítem de la vecindad de $\bar{i}$ para el cual ya valga (5.87), $\bar{i}$ al contenedor n y un ítem para el que queremos demostrar la relación al contenedor $n-\bar{r}$. Ahora se intercambian los ítems de $\bar{k}$ y $n-\bar{r}$ generando un nuevo punto factible. De este intercambio obtenemos:

$$\alpha_{i_1\bar{k}} + \alpha_{i_2 n-\bar{r}} = \alpha_{i_2\bar{k}} + \alpha_{i_1 n-\bar{r}} \tag{5.99}$$

Luego, utilizando (5.87) para i_1 y (5.88) para i_2, se deduce que vale (5.87) para i_2. Este procedimiento se puede realizar para todos los ítems de la vecindad de $\bar{i}$, demostrando así (5.87) en su totalidad.

Con estos resultados, ya estamos en condiciones de demostrar (5.90). Para esto generamos un punto factible en el que se utilicen los n contenedores. El ítem $\bar{i}$ es asignado al último contenedor y en el contenedor $\bar{k}$ se asigna algún ítem de la vecindad de $\bar{i}$ ya que de otra manera no se cumpliría la restricción de la cara. Al intercambiar estos dos ítems se sigue sumando lo mismo del lado izquierdo de la restricción por lo que se siguen estando en la cara. De estos dos puntos se deriva:

$$\alpha_{i\bar{k}} + \alpha_{\bar{i}n} = \alpha_{\bar{i}\bar{k}} + \alpha_{in} \tag{5.100}$$

Usando (5.87) queda:

$$\alpha_{in} + \beta n + \alpha_{\bar{i}n} = \alpha_{\bar{i}\bar{k}} + \alpha_{in} \tag{5.101}$$

Lo cual al simplificar los α_{in} demuestra (5.90)

De manera análoga ahora podemos demostrar (5.91). Tomamos un punto que utilice n contenedores en donde $\bar{i}$ está asignado al contenedor $n - \bar{r}$ y en $\bar{k}$ hay algún ítem de la vecindad de $\bar{i}$ para pertenecer a la cara. Se genera un segundo punto al intercambiar estos dos ítems quedando:

$$\alpha_{i\bar{k}} + \alpha_{\bar{i}n-\bar{r}} = \alpha_{\bar{i}\bar{k}} + \alpha_{in-\bar{r}} \tag{5.102}$$

Utilizando (5.87) y (5.90), se puede ver entonces que se cumple (5.91) para el contenedor $n - \bar{r}$. Sin embargo este procedimiento se puede hacer asignando a $\bar{i}$ a cualquier contenedor entre $n - \bar{r}$ y $n - 1$ (ya que los coeficientes en la desigualdad son iguales para este ítem). De esta manera, queda demostrado (5.91).

La demostración continúa mostrando la validez de (5.84) y (5.86). Para poder concluir estas igualdades, debemos primero hacer una observación. Si tomamos los puntos de la cara que utilizan $n - \bar{r} - 1$ contenedores, entonces debe existir alguno de estos puntos que tenga al menos 2 ítems asignados por fuera del contenedor $\bar{k}$. Para ver que esta observación es cierta dividiremos el razonamiento en dos casos:

- Si $r \geq \bar{r} + 2$, entonces siempre se puede armar una asignación en donde haya $\bar{r} + 1$ ítems en $\bar{k}$, dos ítems en algún otro contenedor y un ítem en el resto. Los $\bar{r} + 1$ ítems de $\bar{k}$ se obtienen utilizando el $\bar{i}$ y sus $\bar{r}$ vecinos con los que puede compartir contenedor por definición. Los dos ítems pueden provenir de la vecindad de $\bar{i}$ ya que al diferir r y $\bar{r}$ en al menos dos, tiene que haber dos ítems en $N(\bar{i})$ que pueden ir juntos y que no están siendo utilizados en $\bar{k}$.

- Si $r = \bar{r} + 1$, entonces hay dos tipos posibles de puntos que pertenecen a la cara:

  - Los puntos en donde se asigna $\bar{i}$ y $\bar{r}$ ítems de su vecindad al contenedor $\bar{k}$. En estos puntos se pone un ítem en el resto de los contenedores y todavía sobra un ítem i a asignar. Si i puede asignarse a algún contenedor distinto de $\bar{k}$, entonces ya tenemos el tipo de punto que buscamos, en donde hay 2 ítems por fuera de $\bar{k}$. Si i debe ser asignado a $\bar{k}$, entonces significa que $i \notin N(\bar{i})$. Pero entonces podemos armar un punto en donde $\bar{i}$ e i esten asignados al contenedor 1, r ítems de $N(\bar{i})$ estén asignados a $\bar{k}$ y todos los demás contenedores tengan un ítem asignado.
  - Los puntos en donde se asignan r ítems de $N(\bar{i})$ en $\bar{k}$. En estos puntos se asigna un ítem a cada contenedor restante y luego resta un ítem más por ubicar. Este ítem o bien puede ser ubicado en algún contenedor que no sea $\bar{k}$, generando el punto buscado, o bien tiene que ser asignado a $\bar{k}$. Luego, si fue asignado a $\bar{k}$ es porque no pertenece a la vecidad de $\bar{i}$, porque si no tendríamos $r + 1$ independientes en la vecindad de $\bar{i}$. Como no pertenece a la vecindad de $\bar{i}$, entonces lo podemos mover al contenedor donde esta $\bar{i}$ sin problemas.

De esta manera, demostramos que siempre tenemos un punto que usa $n - \bar{r} + 1$ contenedores en el que hay al menos dos ítems por fuera del contenedor $\bar{k}$.

Una vez que tenemos este punto armado, podemos obtener un nuevo punto que surge de mover a uno de los ítems que comparte contenedor, y no es $\bar{i}$, al contenedor $n - \bar{r}$. De estos dos puntos obtenemos la siguiente igualdad:

$$\alpha_{i_1 k} = \alpha_{i_1 n-\bar{r}} + \beta_{n-\bar{r}} \tag{5.103}$$

Usando (5.85) o (5.88), según i_1 esté o no en la vecindad de $\bar{i}$, se demuestra (5.84) o (5.86) para el ítem i_1 y el contenedor k. Este intercambio se puede hacer desde cualquier contenedor anterior a $n - \bar{r}$ que no sea $\bar{k}$. Luego, análogamente a lo hecho en los resultados anteriores, se puede demostrar (5.84) y (5.86) para todos los demás ítems a través del procedimiento que se basa en ubicar a i_1 en el contenedor n y al ítem que se quiera realizar la demostración en uno de los contenedores deseados. Al realizar el intercambio entre estos dos ítems, se estará probando para el ítem y contenedor buscado. Esto se puede realizar con todo ítem distinto de $\bar{i}$ y con todo contenedor hasta el $n - \bar{r}$ exclusive, excluyendo también a $\bar{k}$. De esta forma se demuestra (5.84) y (5.86), exceptuando por el caso donde el contenedor en la relación es $\bar{k}$, para $i \notin N(\bar{i})$.

Con todos los resultados demostrados, vamos a continuar por mostrar que las igualdades (5.94) son válidas. Estas igualdades dicen que los coeficientes asociados a las variables de uso de contenedor son 0 para todos los contenedores desde $\chi^b + 1$ hasta $n - 1$, exceptuando por $n - \bar{r}$ y $\bar{k}$. Por los tipos de puntos utilizados para la demostración, diviremos la demostración en dos intervalos.

En primer lugar, tomamos un punto en donde se encuentran utilizados $n - \bar{r}$ contenedores, en donde $\bar{i}$ y $\bar{r}$ de sus vecinos están asignados al contenedor $\bar{k}$. Luego, movemos uno de los $\bar{r}$ ítems al contenedor $n - \bar{r} + 1$ y obtenemos la siguiente relación:

$$\alpha_{i\bar{k}} = \alpha_{in-\bar{r}+1} + \beta_{n-\bar{r}+1} \tag{5.104}$$

Luego, utilizando (5.87) y (5.88), se deriva:

$$\alpha_{in} + \beta_n = \alpha_{in} + \beta_n + \beta_{n-\bar{r}+1} \tag{5.105}$$

de donde se deduce que $\beta_{n-\bar{r}+1}$ es 0. Una vez probado esto, se puede utilizar el último punto, y pasar otro ítem más de los que acompaña a $\bar{i}$ en $\bar{k}$ al siguiente contenedor libre. Así, con la misma cuenta utilizada para $n - \bar{r} + 1$, se puede mostrar que todos los β de ahí en adelante son nulos hasta el anteúltimo contenedor.

Una vez realizada esta última demostración, resta por demostrar el mismo resultado para los contenedores que van desde $\chi^b + 1$ hasta el $n - \bar{r} - 1$, exceptuando por el $\bar{k}$. Para esto nuevamente abriremos la demostración en 2 casos, según si χ^b está por debajo o no de $\bar{k}$.

Comenzamos entonces con el caso $\bar{k} \leq \chi^b$. En este caso siempre se puede armar una asignación que utilice χ^b contenedores en la que $\bar{i}$ sea asignado a $\bar{k}$ junto con $\bar{r}$ vecinos. Este punto se puede armar con seguridad porque, por hipótesis, existe un punto que utiliza χ^b contenedores en la cara y las únicas configuraciones con χ^b contenedores son del tipo pedido. Luego, iremos demostrando que los coeficientes β son nulos de manera secuencial como se hizo para el otro intervalo. Si en el resto de los contenedores de este punto hay alguno que tenga más de un ítem, entonces se puede utilizar estos ítems que sobran para ir pasándolos a los contenedores libres y así demostrando la nulidad de la misma manera que lo hecho antes, ya que con este procedimiento quedaría:

$$\alpha_{i1} = \alpha_{ik} + \beta_k \tag{5.106}$$

En donde se utiliza α_{i1} ya que está demostrado que los coeficientes α son todos iguales para los contenedores que están al principio. En este caso el contenedor k es cada uno de los contenedores que van quedando libres sucesivamente y a los que se les mueve un ítem. De esta manera, utilizando (5.84), se demuestra que β_k es 0. Esto lo podemos hacer hasta que agotemos todos los ítems sobrantes en los contenedores que no son $\bar{k}$. Si al realizar esto todavía quedan contenedores en el intervalo para los cuales hay que realizar la demostración, entonces no queda otra opción que tener a todos los ítems faltantes en $\bar{k}$. Además, mientras que todavía queden contenedores libres en el intervalo a demostrar, la cantidad de ítems en $\bar{k}$ por fuera de $\bar{i}$ y los $\bar{r}$ vecinos tiene que ser mayor a 1 por un argumento de cantidad total de ítems. Luego, con estos ítems podemos realizar sucesivamente el siguiente procedimiento con cada uno de los contenedores que faltan. Tomamos dos ítems i_1 e i_2 en $\bar{k}$ por fuera de $\bar{i}$ y sus $\bar{r}$ vecinos. Generamos 3 nuevos puntos factibles que nacen de pasar a i_1, a i_2 y a i_1 e i_2 al primer contenedor libre. De la interacción entre estos 3 puntos y el original, obtenemos las siguientes relaciones:

$$\alpha_{i_1\bar{k}} = \alpha_{i_1 k} + \beta_k \tag{5.107}$$
$$\alpha_{i_2\bar{k}} = \alpha_{i_2 k} + \beta_k \tag{5.108}$$
$$\alpha_{i_1\bar{k}} + \alpha_{i_2\bar{k}} = \alpha_{i_1 k} + \alpha_{i_2 k} + \beta_k \tag{5.109}$$
$$\tag{5.110}$$

Luego, sumando las dos primeras ecuaciones y restando la tercera obtenemos:

$$0 = 2\beta_k - \beta_k \tag{5.111}$$

demostrando el resultado deseado. Esto lo podemos hacer sucesivamente con cada contenedor libre que siga en la lista, hasta que en $\bar{k}$ solo quede un ítem por fuera de $\bar{i}$ y sus $\bar{r}$. En este momento estaremos utilizando $\bar{r}+2$ ítems en $\bar{k}$ y uno en todo el resto, por lo que el siguiente contenedor libre es $n-\bar{r}$ habiendo terminado la demostración para este intervalo.

Para cerrar todos los casos, debemos realizar la demostración cuando $\bar{k} > \chi^b$. En este caso tenemos nuevamente dos intervalos, los contenedores que están antes de $\bar{k}$ y los que se encuentran después. Para los que están antes simplemente podemos empezar de un punto que utilice χ^b contenedores e ir desprendiendo de a un ítem hacia los contenedores nuevos hasta llegar a $\bar{k}$. Utilizando los resultados (5.84) o (5.86), según la naturaleza del ítem que estemos moviendo, se podrá realizar la demostración pertinente de la misma manera que se realizó anteriormente. Luego, solo resta realizar la demostración para los contenedores entre $\bar{k}+1$ y $n-\bar{r}-1$. Para esto comenzamos tomando un punto que utilice $\bar{k}$ contenedores y donde el contenedor $\bar{k}$ tenga asignado solamente a $\bar{i}$ y a $\bar{r}$ de sus vecinos. El resto de los ítems tiene que poder ser ubicado en los anteriores contenedores ya que, por el caso en que estamos, $\bar{k}-1 \geqq \chi^b$. Una vez que se tiene este punto, todos los ítems que sobran en los contenedores anteriores a $\bar{k}$ se pueden ir moviendo sucesivamente al primer contenedor vacío, generando así puntos que pertenecen a la cara y que demuestran la nulidad de los coeficientes buscados al combinar las ecuaciones resultantes con los resultados (5.84) y (5.86). Nuevamente, en el momento que no sobren más ítems es porque entonces hemos llegado a llenar hasta el contenedor $n-\bar{r}-1$ terminando así con todo el intervalo necesario.

De esta manera, queda demostrado (5.94).

Continuamos realizando la demostración de (5.89). Para esto tomamos un punto en donde se utilizan $n-r+1$ contenedores. Asignamos r ítems de la vecindad a $\bar{k}$ e $\bar{i}$ queda en algún contenedor diferente. Luego, queremos armar un segundo punto factible para la cara, también con $n-r+1$ contenedores, en el que $\bar{i}$ esté asignado al contenedor $\bar{k}$. Para que el punto pertenezca en la cara necesitamos hacer que $\bar{i}$ esté acompañado por $\bar{r}$ ítems de su vecindad. Este punto podría no ser simple de armar debido a que los $\bar{r}$ vecinos que pueden ser asignados con $\bar{i}$ pueden no tener nada que ver con los r que ya están asignados a $\bar{k}$. De la misma manera, puede ser que algunos de los r sean parte de los $\bar{r}$ indistintamente. No importa cual sea el caso, siempre se puede tomar $\bar{r}$ ítems y asignarlos con $\bar{i}$ en $\bar{k}$ y reordenar el resto de los ítems para que no quede ningún contenedor vacío. Dado que ya demostramos (5.86) y (5.87), de estos dos puntos se puede derivar la siguiente relación:

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}\bar{k}} + (r-\bar{r})\beta_{n-\bar{r}} \tag{5.112}$$

Esta ecuación nace del hecho de que no importa cómo fueron los movimientos de los ítems, todos los movimientos necesarios, exceptuando el de $\bar{i}$, fueron realizados por ítems que están en la vecindad de $\bar{i}$. Para estos ítems ya tenemos demostrado (5.86) y (5.87). No importa cómo hayan sido los movimientos, si no que lo que importa es que actualmente tenemos $r-\bar{r}$ ítems menos en $\bar{k}$ que ahora están en algún otro contenedor hasta el $n-\bar{r}-1$. Por cada uno de esos movimientos entonces se suma una vez la variable $\beta_{n-\bar{r}}$, que es la diferencia entre (5.86) y (5.87). Ahora, utilizando (5.90), nos queda:

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}n} + \beta_n + (r-\bar{r})\beta_{n-\bar{r}} \tag{5.113}$$

demostrando (5.89) para el contenedor k. Sin embargo, este procedimiento fue indistinto para todos los contenedores k desde 1 hasta $n-r+1$, sin contar a $\bar{k}$, por lo que queda demostrado (5.89).

Ahora demostramos el resultado (5.92). En primer lugar lo hacemos para el contenedor $n-r+2$ y luego iremos haciendo sucesivamente el mismo procedemiento para hacer los siguientes contenedores hasta llegar al $n-\bar{r}-1$. Comenzamos tomando un punto que utilice $n-r+1$ contenedores. En este punto asignamos a $\bar{i}$ al contenedor $n-r+1$, r ítems a $\bar{k}$, y un ítem al resto. A partir de este punto factible de la cara, generamos otro que también cumple las restricciones de la cara pasando a $\bar{i}$ al contenedor $n-r+2$, y pasando un ítem de $\bar{k}$ al contenedor $n-r+1$. Con estos dos puntos obtenemos la relación:

$$\alpha_{\bar{i}n-r+1} + \alpha_{i\bar{k}} = \alpha_{\bar{i}n-r+2} + \alpha_{in-r+1} \tag{5.114}$$

reemplazamos usando (5.86), (5.87) y (5.89), quedando:

$$\alpha_{\bar{i}n} + \beta_n + (r-\bar{r})\beta_{n-\bar{r}} + \alpha_{in} + \beta_n = \alpha_{\bar{i}n-r+2} + \alpha_{in} + \beta_{n-r} + \beta_n \tag{5.115}$$

simplificando y reagrupando, se deriva:

$$\alpha_{\bar{i}n-r+2} = \alpha_{\bar{i}n} + \beta_n + (r-\bar{r}-1)\beta_{n-\bar{r}} \tag{5.116}$$

que es lo que queríamos demostrar para el contenedor $n-r+2$. Luego, cada contendor sucesivo tiene casi el mismo resultado pero sustrayendo un $\beta_{n-\bar{r}}$ más con respecto al anterior. Esto sale de realizar el mismo procedimiento que acabamos de hacer pero ahora con este nuevo punto. Es decir, con el punto que se generó, el que utiliza $n-r+2$ contenedores, se mueve el ítem $\bar{i}$ al contenedor $n-r+3$, y se mueve un ítem de $\bar{k}$ a $n-r+2$. Este último movimiento, al igual que antes, es el responsable de que aparezca una vez el coeficiente $\beta_{n-\bar{r}+1}$ para ser restado. De esta manera, el mismo procedimiento se puede hacer hasta que en $\bar{k}$ queden $\bar{r}$ ítems (para seguir perteneciendo a la cara) y por lo tanto este procedimiento se puede hacer exactamente hasta el contenedor $n-\bar{r}-1$, demostrando (5.92) en su totalidad.

Por último, nos restó demostrar (5.84) para el caso $\bar{k}$. Para demostrar la relación para $\bar{k}$, usaremos la hipótesis que indica que todo ítem que no está en la vecindad de $\bar{i}$ debe poder ser asignado conjuntamente o bien con r ítems de la vecindad, o bien con $\bar{i}$ y $\bar{r}$ ítems de su vecindad. Tomamos entonces al ítem a demostrar junto con el conjunto de ítems (r o $\bar{r}$ más $\bar{i}$) que puede ser asignado, y los asignamos al contenedor $\bar{k}$. Luego, generamos un segundo punto factible que surge de mover al ítem en cuestión desde $\bar{k}$ al primer contenedor vacío. Este contenedor depende el caso puede ser distinto, pero lo importante es que por la relación entre r y $\bar{r}$, siempre es menor o igual a $n-\bar{r}-1$. De estos dos puntos se desprende:

$$\alpha_{i\bar{k}} = \alpha_{ik} + \beta_k \tag{5.117}$$

Utilizando (5.84) y (5.94), nos queda:

$$\alpha_{i\bar{k}} = \alpha_{in} + \beta_{n-\bar{r}} + \beta_n \tag{5.118}$$

que es lo que queríamos demostrar. Por hipótesis, este procedimiento se puede realizar con todo ítem que no pertenezca a la vecindad de $\bar{i}$, por lo que queda demostrado (5.84) en su totalidad.

De esta manera, quedan demostrados todos los resultados necesarios para definir la condición de faceta de la cara generada por la desigualdad propuesta bajo las hipótesis planteadas.

□

5.2.4. Inhabilita contenedores

La utilización de los contenedores en orden es un concepto que ya estuvo presente en todas las derivaciones de las desigualdades presentadas. En algunos casos se ha utilizado como idea primordial para derivar una desigualdad en conjunto con alguna estructura particular del hipergrafo. En otros casos se ha utilizado solamente como un refuerzo de la desigualdad derivada de otras ideas.

En este caso, presentamos una desigualdad válida que nace puramente de esta idea sin utilizar ninguna estructura del hipergrafo, sino simplemente un vértice cualquiera. Si un contenedor no es utilizado, es claro que un vértice particular no puede ser asignado a dicho contenedor. Además, por la restricción de orden de uso de los contenedores, tampoco puede ser asignado a ningún contenedor posterior. En particular, podemos sumar las variables de pertenencia a los contenedores y generar una desigualdad más fuerte. La misma queda definida para un vértice $\bar{i}$ y un contenedor $\bar{k}$ de la siguiente manera:

$$\sum_{k=\bar{k}}^{n} x_{\bar{i}k} \leq y_{\bar{k}} \tag{5.119}$$

Esta desigualdad resulta válida para el modelo y es un refuerzo de la idea básica de que la variable de contenedor limita a la variable de asignación de un ítem a un contenedor. Además, como mostramos en la siguiente proposición, esta desigualdad resulta en una restricción fuerte en varios casos, definiendo una faceta del poliedro asociado al modelo.

Proposición 6. *Dado un contenedor $\bar{k}$ y un ítem $\bar{i}$.*

$$\sum_{k=\bar{k}}^{n} x_{\bar{i}k} \leq y_{\bar{k}} \tag{5.120}$$

es válida para P_{base}. Además, si se cumple $\chi^b < \bar{k} < n$, entonces la desigualdad genera una faceta del poliedro.

Demostración. La validez de la desigualdad se puede ver analizando dos casos posibles.

- $y_{\bar{k}} = 0$. En este caso las desigualdades del modelo implican que $x_{\bar{i}\bar{k}}$ es 0. Además también deben ser nulas todas las variables asociadas a los contenedores posteriores y, a su vez, estas fuerzan a que las variables de asignación del ítem $\bar{i}$ a dichos contenedores sean nulas.

- $y_{\bar{k}} = 1$. En este caso la igualdad de asignación sobre el ítem $\bar{i}$ dice que la suma sobre todos los contenedores es igual a 1, por lo cual al truncar la sumatoria desde cierto contenedor tiene que ser menor o igual a lo anterior por ser todas variables no negativas.

Luego, una vez probada la validez, mostramos entonces que la desigualdad genera una faceta del poliedro. Esto sucede solo bajo las hipótesis enunciadas en la proposición.

Hipótesis 16. $\bar{k} > \chi^b$

Esta hipótesis es necesaria porque de no cumplirse, significaría que el contenedor particular es uno de los que seguro se debe utilizar en todo punto de la cara y del poliedro. Luego, para que se cumpla la restricción de la cara, esto significa que entonces en todos los puntos factibles el ítem $\bar{i}$ tiene que estar asignado a algún contenedor desde $\bar{k}$ en adelante. Esto genera la igualdad $x_{\bar{i}1} = 0$, entre otras posibles. Haciendo entonces que la dimensión de la cara asociada sea menor a la necesaria para ser una faceta. En el caso particular en que $\bar{k} = 1$, no se cumpliría la igualdad propuesta. Sin embargo, en este caso la desigualdad tampoco define una faceta del poliedro ya que la desigualdad generada es superflua y se deriva del resto de las restricciones.

Hipótesis 17. $\bar{k} \neq n$

Esta hipótesis dice que el contenedor particular para el cual estamos definiendo la desigualdad no puede ser el último. Si $\bar{k}$ fuese el último, esto significaría que en los puntos de la cara el único ítem que tiene la posibilidad de ir al último contenedor es el ítem $\bar{i}$. Así, tenemos muchas igualdades cumplidas por los puntos de la cara que son de la forma $x_{in} = 0$ para todos los ítems que no son $\bar{i}$.

Para el resto de las posibilidades para $\bar{k}$, debemos demostrar las siguientes identidades para ver que la desigualdad define una faceta.

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}n} + \beta_n + \beta_{\bar{k}} \quad \forall k < \bar{k} \tag{5.121}$$

$$\alpha_{\bar{i}k} = \alpha_{\bar{i}n} + \beta_n \quad \forall \bar{k} \leq k < n \tag{5.122}$$

$$\alpha_{ik} = \alpha_{in} + \beta_n \quad \forall i \neq \bar{i} \ \forall k < n \tag{5.123}$$

$$\beta_k = 0 \quad \chi^b < k < n \quad k \neq \bar{k} \tag{5.124}$$

La única restricción para los puntos que pertenecen a la cara, es que si se usa el contenedor $\bar{k}$, entonces el ítem $\bar{i}$ se encuentra asignado a algún contenedor desde ese hasta el último. Esto deja libertad a la generación de muchos puntos simples para poder conseguir los resultados a demostrar. En primer lugar, comenzamos con un punto en donde los ítems se asignen a $n-1$ contenedores. Ponemos al ítem $\bar{i}$ con algún ítem posible en el contenedor $n-1$ y todo el resto de los contenedores tendrá un solo ítem. Luego, generamos un segundo punto factible de la cara mediante el traspaso del ítem $\bar{i}$ al contenedor n. De estos dos puntos obtenemos la siguiente relación:

$$\alpha_{\bar{i}n-1} = \alpha_{\bar{i}n} + \beta_n \tag{5.125}$$

Este mismo procedimiento se puede hacer no solo desde el contenedor $n-1$, si no desde cualquier contenedor mayor o igual a $\bar{k}$, demostrando (5.122).

Luego, nuevamente comenzamos con un punto que utilice $n-1$ contenedores. En este caso enviaremos a un ítem i al contenedor 1 con algún posible vecino. Todos los demás contenedores tendrán un solo ítem y, para que el punto pertenezca a la cara, el ítem $\bar{i}$ se asignará a alguno de los últimos contenedores, por ejemplo el $n-1$. Al igual que en el caso anterior, el segundo punto a tener en cuenta nacerá de mover este ítem que comparte contenedor al último. Así obtenemos:

$$\alpha_{i1} = \alpha_{in} + \beta_n \tag{5.126}$$

Este razonamiento se puede hacer con al menos un ítem, ya que ninguno es universalmente conflictivo, y por lo tanto tiene algún otro ítem con el que pueden ser asignados. Aparte, no puede ser que todos tengan como único posible compañero al ítem $\bar{i}$, ya que esto significaría que χ^b es $n-1$, lo cual no es un caso a tratar en esta demostración ya que estamos bajo la hipótesis de que $\chi^b < \bar{k} < n$.

Luego, para los ítems que pueden compartir contenedor con alguien distinto a $\bar{i}$, queda demostrado (5.123) para todos los contenedores excepto para el $n-1$, ya que podría suceder que este sea el único contenedor posible para que vaya $\bar{i}$ en una asignación de $n-1$ contenedores. Para demostrar el resultado para este anteúltimo contenedor, tomamos un punto donde se utilicen los n contenedores, asignando $\bar{i}$ a n y el ítem i que queremos demostrar al contenedor $n-1$. Luego armamos un segundo punto intercambiando estos dos ítems, con lo cual obtenemos:

$$\alpha_{in-1} + \alpha_{\bar{i}n} = \alpha_{\bar{i}n-1} + \alpha_{in} \tag{5.127}$$

A partir de este resultado, utilizamos (5.122) y nos queda:

$$\alpha_{in-1} = \alpha_{\bar{i}n-1} + \beta n \tag{5.128}$$

Demostrando así todo (5.123), pero solo para los ítems que pueden compartir contenedor con alguien distinto a $\bar{i}$. Lo importante, como se notó, es que algún ítem de este tipo existe. Llamamos a este ítem i_1. Luego, para todo ítem i_2 que no cumpla esta propiedad podemos demostrar su

resultado (5.123) correspondiente mediante un punto que utilice n contenedores y luego intercambiándolo con el ítem i_1 para el cual ya se demostró el resultado. Esto demuestra por completo (5.123).

Probaremos ahora la validez de (5.121). Para esto tomamos un punto en donde se utilicen $\bar{k}-1$ contenedores y donde $\bar{i}$ esté asignado con alguien más al contenedor 1. Este punto existe porque estamos bajo la hipótesis de que siempre existe un punto en la cara donde se utilizan exactamente χ^b contenedores. Luego, como $\chi^b < \bar{k}$, se puede armar un punto como el explicado. A partir de este primer punto, podemos pasar el ítem $\bar{i}$ al contenedor $\bar{k}$ y seguir cumpliendo las restricciones de la cara. De la interacción entre estos dos puntos nos queda:

$$\alpha_{\bar{i}1} = \alpha_{\bar{i}\bar{k}} + \beta_{\bar{k}} \tag{5.129}$$

Además como este pasaje se puede hacer indistintamente desde cualquier contenedor previo a $\bar{k}$, esto demuestra (5.121).

Por último, para demostrar (5.124), utilizamos el mismo procedimiento que se uso en varias de las demostraciones previas. Comenzamos de un punto que represente una asignación donde se utilizan exactamente χ^b contenedores. A partir de este punto, se va moviendo secuencialmente un ítem cualquiera que no esté solo en un contenedor al primer contenedor vacío. Gracias a que ya hemos demostrado (5.123), este procedimiento va demostrando que el coeficiente β asociado a cada contenedor es nulo. Este procedimiento encuentra un problema cuando llegamos al contenedor $\bar{k}$. Al comenzar a utilizar este contenedor, el único ítem que se puede pasar es $\bar{i}$, es por esto que el resultado no es válido para dicho contenedor. Luego, a partir de ese punto se puede seguir sin problemas por lo que queda demostrado el resultado en su totalidad.

Habiendo demostrado todos los resultados necesarios, queda probado que la desigualdad define una faceta en los casos propuestos.

□

5.2.5. Fuerza contenedor a cero

Al comenzar con el estudio poliedral, hemos agregado ciertas restricciones al modelo. Si bien estas restricciones no eran estrictamente necesarias para obtener un modelo correcto, el agregado de las mismas genera un poliedro con muchos menos puntos simétricos, lo cual es un resultado deseable para achicar el espacio de búsqueda de soluciones óptimas. Una de las desigualdades agregadas es la que genera una relación inversa entre las variables de uso de contenedor y las variables de asignación de ítem a contenedor. En el modelo original ya se encontraba la restricción que dice que si un ítem es asignado a un contenedor, entonces la variable de uso del contenedor debe estar encendida. Sin embargo, esta restricción deja que las variables de uso de contenedor se enciendan sin tal vez tener ningún ítem asignado. De esta manera, se agregó la restricción que induce la lógica inversa. Si ningún ítem está asignado a un contenedor dado, entonces la variable de uso del contenedor debe ser nula.

Esta restricción no era válida en el modelo original ya que no solo corta puntos fraccionarios, si no también puntos enteros. Sin embargo, los puntos cortados no son de interés por ser subóptimos. Luego de ser agregada al modelo como restricción básica, no solamente, obviamente, pasa a ser válida, sino que además en varios casos define una faceta del nuevo poliedro como se ve en la próxima proposición.

Proposición 7. *Dado un contenedor $\bar{k}$, la desigualdad*

$$y_{\bar{k}} \leq \sum_{i \in V} x_{i\bar{k}} \tag{5.130}$$

es válida para P_{base}. Además, bajo ciertas hipótesis la desigualdad define una faceta del poliedro asociado.

Demostración. Como ya se mencionó, la desigualdad es válida para P_{base} dado que es una de la restricciones básicas que define el problema.

Para demostrar que define una faceta del poliedro bajo ciertas hipótesis, mostraremos que el sistema minimal de la cara del poliedro generada al tomar esta desigualdad como una igualdad tiene exactamente una restricción más que el sistema minimal de todo el poliedro.

Para que la desigualdad genere una faceta, enunciamos la siguiente hipótesis de caracter necesario y suficiente:

Hipótesis 18. $\bar{k} \neq n$

Esta condición resulta necesaria ya que, de lo contrario, tendríamos $\bar{k} = n$. La restricción particularizada en este caso se encuentra dominada por una igualdad ya presente en el sistema minimal de P_{base}, por lo que no podría definir una faceta.

Luego, para demostrar que (5.130) define una faceta bajo las hipótesis necesarias, debemos probar que se cumplen las siguientes identidades:

$$\alpha_{ik_1} = \alpha_{ik_2} \quad \forall i \quad \forall k_1, k_2 \neq n, \bar{k} \tag{5.131}$$

$$\alpha_{in} = \alpha_{ik_1} - \beta_n \quad \forall i \quad \forall k_1 \neq n, \bar{k} \tag{5.132}$$

$$\beta_k = 0 \quad \chi^b < k < n \tag{5.133}$$

$$\alpha_{i_1\bar{k}} + \alpha_{i_2k_1} = \alpha_{i_1k_1} + \alpha_{i_2\bar{k}} \quad \forall i_1, i_2 \forall k_1 \neq n \tag{5.134}$$

(5.131), (5.132), (5.133) pueden ser demostradas de la misma manera que se hizo para el sistema minimal de P_{base}. Los puntos que fueron usados para realizar las demostraciones pueden ser representados por puntos que satisfacen las restricciones para pertenecer a la cara.

Para probar (5.134) usaremos otra familia de puntos pertenecientes a la cara. Los puntos en donde todo ítem es asignado a un contenedor diferente satisfacen no solo las restricciones de P_{base}, si no también la restricción actual por igualdad dado que exactamente un ítem es asignado al contenedor $\bar{k}$. Utilizamos entonces dos puntos de este tipo. Uno en donde el ítem i_1 está en el contenedor $\bar{k}$ y el ítem i_2 es asignado al contenedor k_1. El segundo punto será casi igual al primero, con la diferencia de que los ítems mencionados se encuentran intercambiados en la asignación.

Ahora evaluamos ambos puntos en el ecuación genérica y restamos los resultados, obteniendo:

$$\alpha_{i_1\bar{k}} + \alpha_{i_2k_1} - \alpha_{i_1k_1} - \alpha_{i_2\bar{k}} = 0 \tag{5.135}$$

Esto es simplemente un despeje de (5.134). Dado que no impusimos ninguna restricción adicional sobre i_1, i_2 y k_1, este resultado puede ser utilizado para todo par de ítems y todo contenedor, concluyendo así la demostración.

□

5.2.6. Variables positivas

La no negatividad de las variables utilizadas es una restricción del modelo que no está dada por una desigualdad si no directamente por la naturaleza binaria de las variables utilizadas. En esta sección mostraremos que la condición de no poder ser negativas para las variables x, definen facetas del poliedro asociado en varios casos. Este mismo aspecto no es analizado para las variables de uso de contenedor ya que la imposición del uso de los contenedores en orden hace que, una vez fijada una variable de uso en 1 o en 0, muchas otras variables se encuentren fijadas haciendo que estas restricciones no puedan generar facetas.

Veremos entonces en que casos las desigualdades $x_{ik} \geq 0$ definen facetas del poliedro. Para realizar esta demostración, se debe hacer por cada ítem $\bar{i}$ y cada contenedor $\bar{k}$ en particular. En primer lugar, notamos que si al sacar el ítem $\bar{i}$ del conjunto de vértices queda una clique con hiperaristas de tamaño 2, entonces se genera una condición para que la cara no sea una

faceta. Básicamente, esta condición se traduce en que ningún par de ítems por fuera de $\bar{i}$ pueden ser asignados juntos. Luego, para los puntos que pertenecen a la cara, se tiene que cumplir que $\sum_{i \neq \bar{i}} x_{i\bar{k}} = y_{\bar{k}}$. Esto, a menos que $\bar{k} = n$, es una nueva igualdad que no se desprende del sistema minimal del poliedro por lo que hace que la cara asociada no sea una faceta. El caso $\bar{k} = n$ será analizado posteriormente.

Luego, el anterior argumento nos indica que la siguiente hipótesis resulta necesaria:

Hipótesis 19. $\exists i_1, i_2 \in V - \bar{i}$ *que pueden ser asignados juntos.*

Para demostrar que la desigualdad genera una faceta bajo la condición dada, se debe poder concluir que las siguientes identidades son válidas

$$\alpha_{ik_1} = \alpha_{ik_2} \quad \forall i \neq \bar{i} \quad \forall k_1, k_2 \neq n \tag{5.136}$$

$$\alpha_{\bar{i}k_1} = \alpha_{\bar{i}k_2} \quad \forall k_1, k_2 \neq \bar{k}, n \tag{5.137}$$

$$\alpha_{in} = \alpha_{ik_1} - \beta_n \quad \forall i \quad \forall k_1 \neq \bar{k}, n \tag{5.138}$$

$$\beta_k = 0 \quad \chi^b < k < n \tag{5.139}$$

Estos resultados son simples de demostrar utilizando unicamente puntos en donde se utilicen n o $n-1$ contenedores. Por la hipótesis manejada, se pueden armar puntos que cumplan las restricciones de la cara en donde se utilicen $n-1$ contenedores y donde dos ítems, que no son $\bar{i}$, se encuentren asignados juntos. Desde este punto, se puede generar otro punto que también pertenezca a la cara en donde se mueva uno de estos ítems al último contenedor. Mezclando la información de estos dos puntos se obtiene la relación:

$$\alpha_{ik} = \alpha_{in} + \beta_n \tag{5.140}$$

Este movimiento se puede hacer desde cualquier contenedor distinto del último por lo que, para los ítems que pueden compartir con otro ítem que no sea $\bar{i}$, queda demostrado (5.136) y (5.138).

Luego, al tomar puntos en la cara que utilizan n contenedores, todos los contenedores tiene un único ítem. Estos puntos lo único que tienen que cumplir es que el ítem $\bar{i}$ no se encuentre asignado al contenedor $\bar{k}$. Tomando estos puntos e intercambiando cada uno de los ítems por aquel al cual ya se le demostró (5.136) y (5.138), es simple ver que entonces estas propiedades valen para todos los ítems. De la misma manera, intercambiando los ítems ya demostrados con el ítem $\bar{i}$ se puede demostrar (5.137). Justamente el resultado (5.137) tiene de distinto a (5.136) que no se debe demostrar para el contenedor $\bar{k}$, lo cual nunca se podría hacer porque el único intercambio que se tiene prohibido, para seguir perteneciendo a la cara, es aquel que asigna al $\bar{i}$ a $\bar{k}$.

Una vez demostrados estos resultados, se demuestra (5.139) de manera análoga a lo hecho en demostraciones anteriores. Se comienza con una asignación que utilice χ^b contenedores, la cual existe por hipótesis. A partir de esta asignación se van moviendo sucesivamente de a un ítem al primer contenedor vacío. El hecho de haber mostrado que los coeficientes α son iguales para todos los contenedores excepto el último, deriva en que entonces todos los coeficientes β asociados desde el contenedor $\chi^b + 1$ hasta el anteúltimo son nulos.

De esta manera queda demostrada la condición de faceta en el caso de que $\bar{k} < n$. El caso donde se cumple la igualdad es más simple porque ni siquiera se necesita de la hipótesis. No se realiza la demostración correspondiente ya que es análoga a la anterior, con la diferencia de que ahora no está la distinción para un contenedor intermedio ($\bar{k}$) para $\bar{i}$ dado que ahora es el último contenedor. Entonces, al saber que los ítems no son universalmente conflictivos, se puede comenzar de un punto con $n-1$ contenedores utilizados donde algún par de ítems estén asignados juntos y a partir de ahí continuar con el mismo procedimiento explayado.

5.2.7. Uso secuencial

Una de las familias de desigualdades agregadas al comenzar con el estudio poliedral es la que impone una restricción en el orden que se utilizan los contenedores. Si bien el modelo original sin

estas restricciones es correcto y completo, resulta innecesario pensar a todos los contenedores como indistinguibles, ya que esto lleva a que existan muchísimas soluciones equivalentes. El objetivo último de la resolución de este problema es minimizar la cantidad de contenedores utilizados, teniendo un conjunto de contenedores que son todos iguales. Es por esto que se quieren dejar afuera soluciones que solamente varíen en los identificadores de los contenedores utilizados. Una forma de resolver esta simetría de solución es imponiendo un orden de uso secuencial de los contenedores. Nunca puede utilizarse un contenedor si no están siendo utilizados todos los contenedores anteriores. De esta manera, se agregó una nueva familia de desigualdades que indican que la variable de uso de un contenedor siempre es mayor o igual a la variable de uso del contenedor siguiente.

Como vemos en la proposición siguiente, la familia agregada para cortar la simetría explayada, define facetas del poliedro en la mayoría de los casos.

Proposición 8. *Sea $\bar{k}$ un contenedor entre χ^b y $n-2$, la desigualdad*

$$y_{\bar{k}} \geq y_{\bar{k}+1} \tag{5.141}$$

define una faceta para P_{base}.

Demostración. En la proposición se puede ver que no se incluye a toda la familia, sino que se excluye a todos los contenedores previos a χ^b y al contenedor $n-1$. En los casos que $\bar{k}$ sea menor a χ^b, la desigualdad no define una faceta ya que en realidad las dos variables involucradas se encuentran encendidas en todos los puntos de la cara, resultando en la redundante desigualdad $1 \geq 1$. En el caso de que $\bar{k}$ sea $n-1$, la desigualdad es válida y no es superflua como en el caso anterior, pero no define una faceta del poliedro asociado. En este caso, los puntos de la cara cumplen la igualdad $\sum_i x_{in-1} = y_{n-1}$, la cual no se deriva del resto de las igualdades, por lo que la cara tiene menor dimensión a la necesaria para ser faceta del poliedro.

Para realizar la demostración correspondiente al resto de los casos, debemos probar las siguientas identidades:

$$\alpha_{ik} = \alpha_{in} + \beta_n \quad \forall i \quad \forall k \neq n \tag{5.142}$$

$$\beta_k = 0 \quad \chi^b < k < n, k \neq \bar{k}, \bar{k}+1 \tag{5.143}$$

$$\beta_{\bar{k}} + \beta_{\bar{k}+1} = 0 \tag{5.144}$$

Estos resultados deben ser probados en el caso de que $\bar{k}$ sea mayor a χ^b. En el caso de la igualdad, solo se deben demostrar las primeras dos igualdades, por lo que no se hará la distinción de dicho caso.

Comenzamos la demostración entonces probando (5.142). Para esto tomaremos puntos de la cara en donde se utilicen $n-1$ contenedores. Dado que $\bar{k}$ es menor a $n-1$, estos puntos cumplen la restricción por igualdad para pertenecer a la cara. Para probar la relación para un ítem i en particular, tomamos un punto en donde el ítem i se encuentre asignado al contenedor 1 con algún vecino (existe dado que no hay nodos universalmente conflictivos). El resto de los contenedores tendrá un solo ítem asignado. Luego, generamos un segundo punto factible moviendo a i desde el contenedor 1 al último. De estos dos puntos se obtiene:

$$\alpha_{i1} = \alpha_{in} + \beta_n \tag{5.145}$$

Si bien esta relación se hizo con un ítem en particular y el primer contenedor para ejemplificar, lo mismo puede hacerse con todo ítem y con cualquier contenedor distinto al último. Esto demuestra (5.142) en su totalidad.

Al igual que en varias demostraciones anteriores, una vez que se tiene demostrada la relación entre los coeficientes α de un ítem, es más simple demostrar el resto de los resultados ya que el movimiento del ítem entre contenedores produce varios términos simplificables. Para demostrar (5.143) lo haremos nuevamente secuencialmente para cada uno de los contenedores. Comenzamos

tomando una asignación de los ítems que utilice χ^b contenedores. Este punto existe por hipótesis y aparte notamos que descartamos el caso $\chi^b = \bar{k}$. Luego de generar este punto, iremos generando puntos moviendo de a un ítem desde algún contenedor que no quede vacío al siguiente contenedor vacío hasta llegar al contenedor $\bar{k}$ exclusive. Del primer movimiento se genera la relación:

$$\alpha_{ik} = \alpha_{i\chi^b+1} + \beta_{\chi^b+1} \tag{5.146}$$

En donde k es algún contenedor entre el 1 y χ^b. Por (5.142), sabemos que $\alpha_{ik} = \alpha_{i\chi^b+1}$. Luego, simplificando, obtenemos β_{χ^b+1} igual a 0. Este procedimiento de ir moviendo a un ítem lo podemos seguir realizando hasta llegar al contenedor $\bar{k}$ exclusivo, por lo que iremos sucesivamente obteniendo los mismos resultados para los coeficientes β de cada contenedor intermedio. Una vez que llegamos a esta situación, no podemos generar un nuevo punto factible moviendo un solo ítem al contenedor $\bar{k}$, ya que esto no cumpliría la restricción de la cara. En vez de esto, generamos un nuevo punto en donde se mueva un ítem i_1 al contenedor $\bar{k}$ y un ítem i_2 al contenedor $\bar{k}+1$. De este nuevo punto con el anterior obtenemos la relación:

$$\alpha_{i_1k_1} + \alpha_{i_2k_2} = \alpha_{i_1\bar{k}} + \alpha_{i_2\bar{k}+1} + \beta_{\bar{k}} + \beta_{\bar{k}+1} \tag{5.147}$$

Nuevamente, utilizando (5.142), podemos simplificar y obtenemos:

$$\beta_{\bar{k}} + \beta_{\bar{k}+1} = 0 \tag{5.148}$$

Demostrando así (5.144). A partir de este punto podremos continuar con el procedimiento de mover de a un ítem al siguiente contenedor vacío, demostrando así $\beta_k = 0$ para todos los contenedores a partir de $\bar{k}+1$ y hasta el $n-1$. De esta manera queda también demostrado (5.143).

Habiendo demostrado todos los resultados necesarios, se puede afirmar que la desigualdad genera una faceta del poliedro asociado para todos los casos propuestos.

□

5.2.8. Intersección de hiperaristas

Como vimos en familias anteriores, la interacción entre varias hiperaristas del hipergrafo da lugar a restricciones más complejas, como en el caso de las cliques. En este caso, presentamos una familia de desigualdades que surge de la interacción entre solo dos hiperaristas que tienen una gran intersección entre sí.

Dadas dos hiperaristas $H_1 = \{v_1, v_2, \ldots, v_r\}$ y $H_2 = \{v_2, v_3, \ldots, v_{r+1}\}$, podemos ver que se intersecan en todos sus ítems excepto en dos. Si quisieramos asignar una gran cantidad de ítems de estas hiperaristas a un contenedor particular $\bar{k}$, es claro que no podríamos asignarlos todos ya que entonces ambas hiperaristas estarían siendo violadas. Sin embargo, sí podríamos asignar r de los $r+1$ ítems que se encuentran en consideración. De esta manera, es claro que vale la siguiente desigualdad

$$\sum_{i \in H_1 \cup H_2} x_{i\bar{k}} \leq r y_{\bar{k}} \tag{5.149}$$

Si bien la anterior es una desigualdad válida, podemos sumarle más información. En el caso de que la relación se cumpla por igualdad, tiene que ser porque tanto v_1 como v_{r+1} son asignados al contenedor $\bar{k}$. No hay otra manera de asignar r ítems de los considerados sin violar alguna de las dos hiperaristas. Entonces, si luego utilizamos el contenedor siguiente, $\bar{k}+1$, tiene que ser con algún ítem que no sea ni v_1 ni v_{r+1}. Esto se traduce en la siguiente desigualdad

$$\sum_{i \in H_1 \cup H_2} x_{i\bar{k}} + y_{\bar{k}+1} \leq r y_{\bar{k}} + \sum_{i \in V \setminus \{v_1, v_{\bar{k}+1}\}} x_{i\bar{k}+1} \tag{5.150}$$

La cual también es una relación válida para nuestro modelo. Por último, podemos utilizar el mismo proceso de refuerzo que hemos utilizado en familias anteriores, que proviene del orden de uso

que se le da a los contenedores. Si queremos cumplir la anterior relación por igualdad, utilizando el contenedor $\bar{k}$, entonces tendremos que haber asignado r ítems al contenedor $\bar{k}$, haciendo que varios contenedores del final no puedan ser llenados con ningún ítem. Al igual que se hizo anteriormente, esta situación resulta en el siguiente refuerzo

$$\sum_{i \in H_1 \cup H_2} x_{i\bar{k}} + y_{\bar{k}+1} + \sum_{i \in V} \sum_{k=n-r+2}^{n} x_{ik} \leq r y_{\bar{k}} + \sum_{i \in V \setminus \{v_1, v_{\bar{k}+1}\}} x_{i\bar{k}+1} + y_{n-r+2} \tag{5.151}$$

La desigualdad presentada resulta en una restricción válida para nuestro modelo y de mayor fuerza que la presentada originalmente. No se tiene una demostración fehaciente de que esta familia de desigualdades genere facetas del poliedro asociado a nuestro modelo. Sin embargo, se corroboró empíricamente dicho resultado en variadas instancias del problema.

Capítulo 6

Algoritmo branch-and-cut

En este capítulo se describe el algoritmo diseñado a partir del estudio realizado. Conjuntamente, se van presentando resultados experimentales que ilustran sobre la performance de los diferentes componentes del algoritmo. Con dichos resultados se realiza un análisis y discusión que sustenta las diferentes elecciones que se fueron haciendo para tratar de obtener un algoritmo robusto y eficiente. El problema tratado presenta una gran generalidad y no se espera que exista una combinación de componentes que sea la óptima para todas las diferentes situaciones particulares que pueden surgir a partir del problema planteado. Sin embargo, se trata de elegir componentes que tengan un buen comportamiento general y, en algunos casos, se detallarán componentes que se adapten a algunos casos específicos.

En el trascurso de este trabajo se mostró un estudio detallado de diferentes heurísticas iniciales, las cuales son utilizadas en nuestro algoritmo branch-and-cut para generar buenas cotas primales. También se presentó un estudio poliedral para definir familias de restricciones válidas para nuestro modelo. Las mismas serán utilizadas durante el diseño del algoritmo con el objetivo de encontrar cortes que ayuden a recortar secciones del poliedro de la relajación lineal que no sean necesarias explorar, para así poder podar nodos del árbol de búsqueda resultando, generalmente, en un menor tiempo de cómputo necesario.

Además de los componentes detallados durante el trabajo, se explotarán también otros aspectos que resultan fundamentales para un algoritmo de esta característica, como por ejemplo el preprocesamiento de las instancias o la elección de la regla de *branching*, entre otros.

Este tipo de algoritmo tiene muchos parámetros a tener en cuenta si pensamos en cada una de las decisiones de diseño tales como el recorrido del árbol, la elección del siguiente nodo, las reglas de *branching*, etc. A eso se le suma la utilización o no de cada una de los componentes agregados tales como familias de desigualdades o diferentes tipos de heurísticas. La explosión combinatoria de todos estos parámetros hace que sea imposible un estudio experimental en donde todas estas variables estén en juego. Se busca entonces tratar de aislar la injerencia de cada uno de los componentes en el análisis del algoritmo en su totalidad. Esto en general también resulta imposible dado que la performance de cada decisión está altamente influenciada por el resto de las decisiones, llegando al extremo en donde una componente puede tener un impacto negativo sobre el desempeño del algoritmo al ser agregada de forma individual, pero que luego puede llegar a ser altamente beneficiosa cuando se agrega en conjunto con otra componente. Dicho esto, se irá experimentando con cada una de las decisiones tenidas en cuenta tratando de entender en lo posible la relación con las decisiones previamente analizadas.

6.1. Preprocesamiento

Una primera etapa de preprocesamiento tiene como idea principal la reducción del espacio de búsqueda del problema por características de la instancia en particular a resolver, que son

observables antes de comenzar con la resolución del algoritmo branch-and-cut. Tener la posibilidad de eliminar una cantidad no despreciable de variables del modelo suele ser un resultado muy beneficioso ya que puede hasta mover el orden de magnitud de las instancias resolubles por el algoritmo.

6.1.1. Eliminación de variables

Si se quisiera realizar un estudio comparativo entre un algoritmo branch-and-cut particularizado para nuestro problema, y una resolución genérica provista por un software de propósito general para problemas de programación lineal entera, se podría tomar como punto de partida la resolución del modelo presentado originalmente por un *solver* de programación lineal entera. En este caso, el modelo contiene $n^2 + n$ variables. Al considerar n ítems y n contenedores posibles, se tienen n^2 variables de asignación en el modelo, sumadas a las n variables de uso de contenedor. Este es un número de variables bastante grande que puede ser fácilmente reducido.

Como primer preprocesamiento de las variables del modelo, podemos utilizar la mejor cota primal obtenida para una instancia mediante las diferentes heurísticas y metaheurísticas explicadas en capítulos anteriores. Una vez que se obtiene esta cota, es claro que no son necesarios los contenedores posteriores ya que ya se tiene una solución factible que utiliza menos contenedores. De esta manera se pueden anular todas las variables de uso de contenedor asociadas a los contenedores posteriores a la cota primal obtenida. De la misma manera, se anulan también las variables de asignación de todos los ítems a dichos contenedores. Lo que es más, ni siquiera hace falta anular estas variables, directamente pueden no existir en la definición del modelo. Si bien el impacto de este preprocesamiento es dependiente de la instancia en particular a resolver, en general la mejor cota primal obtenida tiene un valor bastante más bajo que n, resultando en una gran reducción del número de variables.

Este preprocesamiento resulta muy importante ya que, si se trabajase con el modelo original, para muchas instancias sería prohibitivo ya solamente el tiempo de la resolución de la relajación lineal en el nodo raíz. Si bien se podría pensar que la resolución del modelo original por un *solver* de propósito general es el primer punto de comparación en la que el problema todavía no recibió un tratamiento particular, la diferencia en tiempos de resolución es tan grande que obviaremos este punto de comparación. Como primer punto de resolución general del problema, tomaremos el modelo presentado en el estudio poliedral, el cual posee más restricciones que el original para evitar un tipo de simetría en las soluciones. A este modelo le sumaremos el preprocesamiento recientemente indicado para reducir la cantidad de variables. Esto es lo que en nuestras comparaciones tomaremos como opción *Base*.

Si bien se agregaron restricciones al modelo original con el objetivo de eliminar simetrías, todavía quedan varios puntos factibles que son equivalentes en una solución real, pero que son tomados como alternativas diferentes por el modelo. Existen diferentes maneras de cortar estas simetrías, nosotros utilizaremos una que resulta muy simple de definir y que ayuda a reducir el número de variables, siguiendo la idea de [45]. Con las restricciones que se tienen hasta el momento, el orden de los contenedores solo se utiliza para distinguir hasta que punto están siendo utilizados. Sin embargo, para una cantidad fija de contenedores utilizados, los mismos son indistinguibles y cualquier permutación de ellos resulta en asignaciones equivalentes desde la aplicación práctica. Para mitigar en parte esta situación, podemos utilizar un argumento similar al esbozado al momento de definir el problema. Dado que tenemos n ítems, no tiene sentido considerar más de n contenedores. De la misma manera, al tener un ítem, no tiene sentido considerar más de un contenedor. Al tener dos ítems, no tiene sentido considerar más de dos contenedores, y así sucesivamente. Esto nos lleva a la siguiente observación. No es necesario permitir que un ítem i sea asignado a un contenedor mayor al $i - esimo$. Es decir, podemos preprocesar las variables x_{ik} con $k > i$ sabiendo que pueden ser anuladas. De igual manera que en el caso anterior, no solo se pueden anular estas variables, si no que directamente pueden no definirse en el modelo. Si bien este corte de simetría no es total, porque todavía quedan puntos equivalentes en el modelo, llamaremos a esta opción *Base antisimétrico (BaseAS)*

Como primera experimentación, veremos qué tanto impacto en los tiempos de cómputo tiene el último corte de simetría ($BaseAS$) respecto del modelo que tomamos como punto de partida para los resultados empíricos ($Base$). Para esto, procederemos a definir las instancias que se utilizaron para este experimento, como para varios de los posteriores.

6.1.2. Instancias

Como ya mencionamos, hasta donde llega nuestro conocimiento, el Problema de Empaquetamiento con Conflictos Generalizados no tiene un tratamiento particular en la literatura desde un punto de vista algorítmico, por lo que no tenemos conocimiento de instancias que se hayan utilizado previamente. De esta manera, nos disponemos a crear instancias para ir realizando los diferentes experimentos que decidirán qué componentes son adecuadas para agregar al algoritmo.

Nuestro problema contiene restricciones de, básicamente, dos orígenes diferentes. Por un lado están las restricciones del hipergrafo conflicto. Estas restricciones provienen del problema de coloreo en hipergrafos, en donde tenemos que minimizar la cantidad de contenedores usados mirando solamente los conflictos entre conjuntos de ítems. Por otro lado, tenemos las restricciones de capacidad de los contenedores. Estas restricciones provienen del problema de la mochila, en donde se tiene que rellenar un contenedor con muchos ítems sin exceder su capacidad.

Como ya hemos notado anteriormente, las restricciones de capacidad pueden ser pensadas como restricciones del hipergrafo conflicto. En el problema de la mochila se utiliza continuamente el concepto de *cover* (ver [36]). Un *cover* es un conjunto de ítems que no pueden ser asignados juntos porque exceden la capacidad del contenedor. En nuestro problema lo que se puede hacer es evitar utilizar las restricciones de capacidad *per se*, y en vez de ellas, agregar una hiperarista al hipergrafo por cada uno de los covers minimales. Sin embargo esto no significa que en la práctica las situaciones sean análogas, dado que las restricciones de capacidad son una por cada contenedor, mientras que la cantidad de *covers* minimales puede ser de orden exponencial.

Para crear instancias para este problema, es importante crear situaciones en las que la relevancia de cada tipo de restricción sea comprendida. Como se demostró en el capítulo de estudio poliedral, las aristas de tamaño 2 son mucho más restrictivas que las de tamaño 3, al punto que aún teniendo todas las aristas de tamaño 3 en el hipergrafo, esto no nos induce ninguna arista de 2. De la misma manera, se puede ver que las aristas de tamaño 3 son más restrictivas que las de tamaño 4, y así sucesivamente. Luego, si el hipergrafo conflicto tiene muchas aristas de 2, es posible que las restricciones de capacidad no sumen mucha información a menos que sean muy ajustadas. Es decir, si las capacidades de los contenedores hacen que fácilmente entren 5 o 6 ítems por contenedor, entonces los *covers* que generan aristas en el hipergrafo tendrán un tamaño muy poco restrictivo en comparación a las aristas ya presentes.

Dado que los resultados mostrados en el estudio poliedral se apoyan, en general, sobre el hipergrafo conflicto, vamos a comenzar experimentando con instancias en las que las capacidades de los contenedores no impongan restricciones. Además, para mantener aristas comparables en cuanto a restrictividad, vamos a considerar en principio hipergrafos *regulares*, es decir hipergrafos en los que todas las aristas tienen el mismo tamaño. Evitaremos en primer lugar utilizar hipergrafos con aristas de tamaño 2, ya que caeríamos en el caso del problema de coloreo de grafos, para el cual existen extensos resultados particulares en la literatura.

De esta manera, definiremos nuestro primer conjunto de instancias que se basan en hipergrafos 3-regulares. En primer lugar, se realizaron experimentos preliminares para detectar el número de ítems para el cual es esperable poder realizar mejoras tangibles. Es decir, una cantidad de ítems tal que no sea trivial la resolución de la instancia, pero donde tampoco sea completamente intratable el problema. Para este primer conjunto de instancias, se utilizaron 22, 24, 26, 28, 30 y 32 ítems.

Una vez fijado el número de vértices, se crea al hipergrafo conflicto de manera aleatoria hasta alcanzar cierto porcentaje de conflictividad. Los porcentajes utilizados son 10 %, 25 %, 50 %, 75 % y 90 %. Por cada una de las combinaciones posibles entre cantidad de ítems y porcentaje de conflictividad, se generaron 10 instancias para obtener una mayor representatividad de las muestras.

Estas instancias serán utilizadas no solamente para experimentar con el preprocesamiento propuesto, sino también luego para diferentes componentes del algoritmo.

A continuación presentamos los resultados experimentales de las variantes de preprocesamiento bajo las instancias mencionadas.

6.1.3. Resultados experimentales

Para mostrar el impacto de aplicar un mayor preprocesamiento al problema, realizamos corridas de ambas alternativas (*Base* y *BaseAS*) sobre las instancias propuestas. Si bien existen diferentes métricas para medir el desempeño de algoritmos de este tipo, en este caso comenzaremos por reportar el tiempo de cómputo utilizado por cada alternativa. La diferencia en esta ocasión será tan notoria que se reporta únicamente sobre dos instancias de cada grupo tamaño-porcentaje propuesto.

Todas las experimentaciones presentadas en este capítulo se realizaron con algoritmos implementados en C++, utilizando CPLEX 12.7 [31]. Las corridas fueron realizadas en una máquina con procesador Intel Core i7-2600 3.40GHz con 16GB de RAM, con Ubuntu como sistema operativo. Cada instancia en particular tiene asignada una hora como tiempo límite de ejecución y su tiempo efectivamente utilizado se reporta en segundos.

Cuadro 6.1: Instancias 3 regulares - 22 ítems

Algoritmo	10 %-0	10 %-1	25 %-0	25 %-1	50 %-0	50 %-1	75 %-0	75 %-1	90 %-0	90 %-1
Base	0.1	0.05	0.27	0.31	10	7	469	317	3600	3600
BaseAS	**0.02**	**0.02**	**0.23**	**0.14**	**0.39**	**0.44**	**5**	**6**	3600	**85**

Cuadro 6.2: Instancias 3 regulares - 24 ítems

Algoritmo	10 %-0	10 %-1	25 %-0	25 %-1	50 %-0	50 %-1	75 %-0	75 %-1	90 %-0	90 %-1
Base	0.37	0.12	0.33	0.34	1107	1848	3600	3600	3600	3600
BaseAS	**0.09**	**0.03**	**0.22**	**0.24**	**18**	**35**	**760**	**375**	3600	3600

Cuadro 6.3: Instancias 3 regulares - 26 ítems

Algoritmo	10 %-0	10 %-1	25 %-0	25 %-1	50 %-0	50 %-1	75 %-0	75 %-1	90 %-0	90 %-1
Base	1.54	0.18	42	44	2502	3292	3600	3600	3600	3600
BaseAS	**0.12**	**0.02**	**2**	**3**	**45**	**39**	**1148**	**781**	3600	3600

Cuadro 6.4: Instancias 3 regulares - 28 ítems

Algoritmo	10 %-0	10 %-1	25 %-0	25 %-1	50 %-0	50 %-1	75 %-0	75 %-1	90 %-0	90 %-1
Base	0.49	0.13	39	69	3600	3600	3600	3600	3600	3600
BaseAS	**0.41**	**0.05**	**2**	**3**	**76**	**53**	3600	3600	3600	3600

Cuadro 6.5: Instancias 3 regulares - 30 ítems

Algoritmo	10 %-0	10 %-1	25 %-0	25 %-1	50 %-0	50 %-1	75 %-0	75 %-1	90 %-0	90 %-1
Base	0.65	0.53	39	78	3600	3600	3600	3600	3600	3600
BaseAS	**0.48**	**0.42**	**4**	**3**	3600	3600	3600	3600	3600	3600

Cuadro 6.6: Instancias 3 regulares - 32 ítems

Algoritmo	10 %-0	10 %-1	25 %-0	25 %-1	50 %-0	50 %-1	75 %-0	75 %-1	90 %-0	90 %-1
Base	0.75	0.71	81	77	3600	3600	3600	3600	3600	3600
BaseAS	**0.46**	**0.39**	**6**	**5**	3600	3600	3600	3600	3600	3600

Con estos resultados ya se pueden extraer conclusiones generales de las instancias, así como también conclusiones sobre la comparación de las alternativas presentadas. En primer lugar, es bastante notorio como la dificultad de las instancias es creciente en dos direcciones bien marcadas. Por un lado, a medida que las instancias crecen en cuanto a cantidad de ítems a considerar, las instancias se dificultan, lo cual era un resultado más que esperado. Además, al todavía no tener los algoritmos ninguna componente que explote particularmente los conflictos entre los ítems, se nota que al crecer la conflictividad del hipergrafo las instancias también se van haciendo cada vez más difíciles de tratar.

Por otro lado, es muy claro el dominio en el desempeño de la variante *BaseAS* por sobre *Base*. En primer lugar, tenemos instancias en donde ambas variantes logran resolver satisfactoriamente en el tiempo provisto. En todas estas instancias el tiempo utilizado por *BaseAS* es menor al tiempo utilizado por *Base*, incluso en las instancias que se resuelven en fracciones de segundo. Luego, se puede ver que hay un subconjunto de instancias que *BaseAS* es capaz de resolver mientras que *Base* no. De las 60 instancias presentadas, *BaseAS* resuelve 39 instancias, mientras que *Base* resuelve 32. Además, las instancias que no logra resolver *Base* y sí *BaseAS*, son resueltas con holgura respecto del tiempo límite.

Por último, hay 21 instancias que no son resueltas por ninguna de las dos variantes. Sin embargo, en estas instancias también se puede ver una diferencia en el desempeño. Ambas variantes alcanzan la misma cota primal para todas las instancias en el tiempo límite, pero no sucede lo mismo con las cotas duales. A continuación se muestran algunas cotas duales obtenidas en las instancias no resueltas.

Cuadro 6.7: Instancias 3 regulares - Cotas duales

Algoritmo	22-90 %-0	24-90 %-0	26-90 %-0	28-90 %-0	30-90 %-0	32-90 %-0
Base	5.25	4.57	5	4.75	4	5
BaseAS	**6**	**7**	**6.18**	**6**	**6**	**6**

En la tabla presentada se puede ver la tendencia, que se repite en el resto de las instancias, que muestra una mejor cota dual para *BaseAS*.

Con todos los resultados mostrados, parece ineludible la conclusión de que el rompimiento de simetría que no permite a un ítem ir a un contenedor más alto que su propio número es muy beneficioso. De hecho, una posible pregunta disparada por estos resultados es por qué en la mayoría de las instancias hay tanta diferencia entre las variantes. En teoría, la inclusión del nuevo rompimiento de simetría reduce aproximadamente a la mitad la cantidad de variables de asignación de ítem a contenedores. Sin embargo, en la práctica no son tantas las variables eliminadas porque hay que recordar que ambas variantes tienen el agregado de la eliminación de contenedores gracias a la mejor cota primal conseguida por las heurísticas iniciales. Es posible que la amplia diferencia de performance no solo provenga por el número real de variables eliminadas, sino porque también esta eliminación de variables viene aparejada con una reducción importante de las posibilidades para algunos ítems. Por ejemplo, al agregar el rompimiento de simetría, el ítem 1 pierde completamente sus opciones y debe ser asignado al contenedor 1.

Fijación de ítems

Otro preprocesamiento posible para el problema surge a partir de la idea de fijar cierta información cuando la instancia así lo permita. Por ejemplo, en el preprocesamiento de rompimiento de

simetría ya presentado, finalmente también se consiguió fijar el ítem 1 al contenedor 1 como efecto secundario. En esta dirección, existen otras fijaciones posibles a realizar cuando consideramos la naturaleza del hipergrafo de conflictos. Es importante notar que estas fijaciones surgen cuando la información de los conflictos es tan restrictiva como para poder eliminar todas las alternativas posibles. Por ejemplo, si existe una arista que solamente involucre al ítem 1 con el ítem 2, no quedará otra opción para el ítem 2 que ser asignado al contenedor 2. Esta fijación nació de un conflicto de a pares de ítems, ya que este tipo de conflictos indica automáticamente que si algo sucede con uno de los vértices involucrados, no puede suceder con el otro. Luego, este tipo de resultados fuertes solo puede ser derivado si contamos con la presencia de aristas de tamaño 2 en el hipergrafo ya que, como se mostró anteriormente, las aristas de mayor tamaño no implican una arista de tamaño 2. Además si no hay una arista de tamaño 2 entre dos ítems, los mismos tiene la posibilidad de ser asignados juntos o separados por igual.

Luego, asumiendo la existencia de aristas de tamaño 2 en el hipergrafo, contamos con las siguientes formas de preprocesamiento.

Nodos universalmente conflictivos Son aquellos nodos que tienen aristas de tamaño 2 con todos los demás ítems del hipergrafo. Estos nodos directamente pueden sacarse de la instancia ya que deberán ser asignados a un contenedor propio. Luego de retirarlos se resuelve la instancia resultante con el algoritmo branch-and-cut. Al finalizar, simplemente se anexan los nodos extraídos a la solución final, agregándolos cada uno en contenedor extra.

Prefijación de clique Si tenemos una clique del hipergrafo con aristas de tamaño 2, entonces podemos estar convencidos que cada uno de esos ítems será asignado a un contenedor diferente. Luego, podemos renumerar los ítems para que sean los primeros y así poder asignarlos cada uno al contenedor con su mismo número. Cuanto más grande sea la clique obtenida, mayor la cantidad de variables que quedarán fijadas por este preprocesamiento. Luego, lo que se buscará es una clique maximal de mayor tamaño posible para eliminar la mayor cantidad de opciones posibles.

Este tipo de preprocesamiento no tiene influencia sobre las instancias 3-regulares presentadas por lo que no se realiza experimentación sobre las mismas. Oportunamente serán utilizados cuando se experimente con instancias de diferente naturaleza.

6.2. Desigualdades válidas

En esta sección estudiaremos el impacto que tienen algunos de los resultados poliedrales propuestos en el desempeño de nuestro algoritmo. Como mencionamos anteriormente, las desigualdades válidas que se encuentran para un modelo generan planos de corte para el poliedro asociado que, en general, son muy deseables ya que reducen el espacio de búsqueda. Para cada familia de desigualdades presentada se ha anexado su demostración de validez o hasta se ha mostrado su condición de facetitud. Sin embargo, esto no necesariamente quiere decir que la inclusión de estos componentes en el algoritmo vayan a tener un impacto beneficioso en el objetivo práctico principal, que es poder resolver una mayor cantidad de instancias en un menor tiempo de cómputo.

Hay varias razones por las que los planos de corte propuestos podrían no representar una mejora en el desempeño. En algunos casos podría ser que el tiempo utilizado para hallar dichos planos sea demasiado en comparación con el beneficio que trae aparejado. Otro efecto negativo posible es que la inclusión de los planos de corte haga crecer de manera abrupta el tamaño del modelo a resolver, resultando tal vez en tiempos de cómputo o uso de memoria prohibitivos. Si solo importase el hecho de que los planos corten puntos fraccionarios sin tener en cuenta el resto de los problemas, entonces siempre se deberían poner directamente como restricciones del modelo original, pero es claro que en la práctica esta decisión está lejos de ser la óptima. De hecho, en algunas ocasiones sucederá que la mejor opción es ni tener en cuenta los resultados poliedrales. Esto por ejemplo sucede cuando las instancias son tan pequeñas que la resolución es muy rápida solamente gracias al

preprocesamiento y la exploración del árbol de *branching*. En estos casos es posible que el tiempo incurrido en buscar planos de corte puede resultar meramente en una sobrecarga.

Es por todo lo dicho que la inclusión de las familias de desigualdades no se puede hacer a la ligera esperando siempre un resultado beneficioso, sino que se debe analizar si vale la pena la inclusión o no. Lo que es más, se puede analizar si podría servir una respuesta intermedia en la cual solo se agreguen algunos pocos representantes de cada familia.

6.2.1. Desigualdades Clique

La desigualdades clique fueron presentadas como un caso particular de las desigualdades de conjunto independiente. Estas desigualdades resultan atractivas para utilizar en la práctica por dos motivos. En primer lugar, como mencionamos anteriormente, son un caso de desigualdad de conjunto independiente en donde es sabido de antemano cual es el valor necesario para instanciar la desigualdad sin caer en costosos cómputos. Por otro lado, de la experimentación anterior pudimos ver que una de las direcciones que dificultaba las instancias era el creciente porcentaje de conflictos en el hipergrafo. A medida que crece el porcentaje de hiperaristas presentes en el hipergrafo, también crece abruptamente la posibilidad de encontrarse con cliques de mayor tamaño.

Por ejemplo, observando las instancias 3 regulares, en una instancia de 30 ítems con porcentaje 25 % deberían encontrarse aproximadamente 107 cliques K_4^3. En cambio, con porcentaje 50 % ya pasamos a un promedio de aproximadamente 1712 cliques K_4^3.

En la siguiente tabla, podemos ver aproximadamente cuántas cliques de cada tamaño diferente deberían hallarse en cada tipo de instancia.

Cuadro 6.8: Instancias 3 regulares - Cantidad de cliques K_k^3 en instancias de 28 ítems

Tamaño	10 %	25 %	50 %	75 %	90 %
4 ítems	2	80	1280	6478	13434
5 ítems	0	0	96	5534	34268
6 ítems	0	0	0	1195	45803
7 ítems	0	0	0	50	29638

En esta tabla se pueden observar dos cosas. En primer lugar, vemos que la cantidad de cliques crece mucho conjuntamente con el porcentaje de conflictos, generando confianza entonces en que la búsqueda de estos planos de cortes generará buenos resultados, ya que estas eran las instancias más difíciles. Por otro lado, se puede ver que la cantidad de cliques es muy dispar según el porcentaje de conflictos. Esto posiblemente hará que haya casos donde la cantidad de cliques es tan poca que la mejor opción sería ni buscarlas. En otros casos la cantidad de cliques podría justificar la inclusión de entrada de todas ellas. Mientras que en los casos más populosos hay que buscar heurísticamente que cliques utilizar ya que no se pueden utilizar todas. El objetivo es poder buscar un punto de equilibrio lo más general posible.

Algoritmo de separación

Existen diferentes maneras de introducir las restricciones clique al modelo. Como se mencionó anteriormente, una posibilidad es poner todas las desigualdades de entrada en el modelo. Esto suele resultar muy costoso si la familia es abundante. En nuestro caso, se realizaron pruebas preliminares introduciendo todas las desigualdades clique correspondientes a las cliques de tamaño 5 y 6. Lo que se pudo observar en dichos experimentos es que la calidad de la cota dual aumenta considerablemente con respecto al modelo *BaseAS*, pero a costa de un alto tiempo de cómputo de la relajación del nodo raíz y del resto del árbol de búsqueda. De hecho, en las instancias donde el

porcentaje y la cantidad de ítems es alta, el tiempo de cómputo y la memoria necesaria resultó completamente prohibitivo.

Si bien era esperable que este experimento preliminar no tuviese buenos resultados generales en cuanto a los tiempos de cómputo necesarios, sí sirvió para mostrar el potencial de esta familia de cortes en el impacto de la cota dual. De esta manera, debemos ver de qué manera incluiremos los cortes clique en nuestra formulación.

En la experimentación preliminar, se agregaron todos los cortes clique a la formulación. Para esta tarea, se necesitó computar todas las cliques de los tamaños requeridos para el hipergrafo de conflictos de cada instancia. Debido a los tamaños de las instancias utilizadas, este es un procedimiento que se puede realizar de forma exhaustiva requiriendo muy poco tiempo de cómputo. Entonces, lo que se hizo fue generar una tabla de cliques, para luego generar todas las desigualdades necesarias.

Un primer algoritmo de separación posible para esta familia consiste en utilizar dicha tabla de cliques para buscar en cada relajación lineal, qué desigualdades clique están siendo violadas. De esta manera, nuevamente se está buscando sobre todas las desigualdades clique posibles de cierto tamaño, pero solamente se agregarán al modelo las que se encuentren violadas en algún nodo del árbol de búsqueda. Esto genera un lógico *tradeoff* con respecto a la opción anterior. Las relajaciones en este caso no son tan pesadas dado que la cantidad de cortes cliques que se agrega es mucho menor al total y se hace gradualmente, pero se debe utilizar tiempo en cada nodo del árbol para buscar si el óptimo de la relajación lineal está violando alguna desigualdad que no se tuvo en cuenta hasta el momento.

Nuevamente se realizaron experimentaciones preliminares con esta primera opción de separación. Los resultados se encontraron en una dirección parecida a los experimentos anteriores, aunque mitigando un poco la disparidad en los extremos. Nuevamente la inclusión de los cortes mostró una mejora en la calidad de la cota dual. Por el lado negativo, en varias instancias se necesitó más tiempo para alcanzar la misma cota dual que con el enfoque anterior se habían alcanzado en el nodo raíz. Por el lado positivo, los tiempos de ejecución fueron mucho más equilibrados, haciendo que instancias con mayor porcentaje de conflictividad pudieran obtener mejores resultados. De todas maneras, el tiempo requerido por el algoritmo de separación exhaustivo sigue resultando prohibitivo si el objetivo es diseñar un algoritmo que se comporte razonablemente en el caso general.

Dados los resultados obtenidos en las experimentaciones preliminares, se optó por un esquema de separación no tan agresivo. En vez de recorrer exhaustivamente todas las cliques de cierto tamaño en el algoritmo de separación, se buscará heurísticamente desigualdades clique que se encuentren violadas. Para esto, se utilizan los siguientes pasos en cada relajación lineal del árbol de búsqueda.

1. Para cada contenedor posible, se aislan los valores de la relajación lineal correspondientes a las variables de asignación de todos los ítems a dicho contenedor y se ordenan en forma no creciente.

2. Por cada ítem en la lista ordenada, se recorren el resto de los ítems en orden y se va armando la clique más grande posible. Al hacerlo de forma ordenada, es esperable que las cliques resultantes tengan un valor elevado de las variables en la relajación.

3. Por cada clique que se genera, se chequea si la desigualdad clique correspondiente con la actual clique y el actual contenedor se encuentra violada. En caso afirmativo, se agrega como plano de corte al modelo.

Con el procedimiento explicado, lo que se busca es generar cliques en donde el lado izquierdo de la desigualdad correspondiente tenga un valor elevado, teniendo así mayor chance de violar a la desigualdad.

Al realizar el algoritmo de separación exhaustivo, se estaba chequeando cada una de las cliques por cada uno de los contenedores. Con este nuevo algoritmo de separación, se chequea cada uno de

los contenedores y por cada uno de ellos se realiza un procedimiento cuadrático sobre la cantidad de ítems. Cuando el porcentaje de conflictos es alto, este enfoque resulta muchísimo menos costoso que el anterior.

Si bien es claro que este último enfoque tiene la desventaja de que puede obviar muchas desigualdades válidas que están siendo violadas, es el que se utilizará en las experimentaciones posteriores ya que resulta en un punto de compromiso aceptable.

6.2.2. Inhabilita contenedores

Otra de las desigualdades presentadas en el estudio poliedral del modelo era la denominada *Inhabilita contenedores*, la cual se expresó de la siguiente manera

$$\sum_{k=\bar{k}}^{n} x_{\bar{i}k} \leq y_{\bar{k}} \tag{6.1}$$

Esta desigualdad resulta atractiva en la práctica por dos motivos. En primer lugar, no utiliza ninguna estructura particular del hipergrafo, por lo que la búsqueda de desigualdades violadas sobre esta familia no debería requerir un costo computacional alto. Por otro lado, a medida que la cota dual y la primal se acerquen en la resolución de una instancia, son pocos los contenedores que estarán en discordia sobre si deben ser utilizados o no. Esta familia de desigualdades justamente impone restricciones sobre asignaciones fraccionarias a dichos contenedores, por lo cual sería esperable que puedan generar un impacto positivo.

Dada la cantidad de desigualdades que existe para esta familia, el algoritmo de separación que se utilizará para la experimentación será simplemente exhaustivo sobre cada uno de los contenedores y cada uno de los ítems.

6.2.3. Desigualdad Sin Agujeros

Como última restricción para la próxima experimentación, presentamos la desigualdad *Sin Agujeros*. Esta restricción no es una desigualdad válida para el modelo, ya que no solamente corta puntos fraccionarios del poliedro sino que también corta puntos enteros. Sin embargo, con el mismo argumento esbozado anteriormente, los puntos enteros que va a cortar esta desigualdad siempre resultan en puntos subóptimos, o bien en puntos para los cuales hay un óptimo alternativo.

La desigualdad *Sin Agujeros* para un contenedor $\bar{k}$ y un ítem $\bar{i}$ dados se puede escribir de la siguiente manera:

$$x_{\bar{i}\bar{k}} \leq \sum_{i=1}^{\bar{i}-1} x_{i\bar{k}-1} \tag{6.2}$$

Si la variable $x_{\bar{i}\bar{k}}$ tiene valor 0, entonces esta desigualdad no impone ninguna restricción. Por el contrario, si la variable $x_{\bar{i}\bar{k}}$ tiene valor 1, entonces lo que está diciendo es que al mirar todos los ítems anteriores a $\bar{i}$, alguno tiene que estar asignado al contenedor $\bar{k}-1$. Dicho de otra manera, si ningún ítem anterior a $\bar{i}$ fue asignado a $\bar{k}-1$, no hay ninguna razón para dejar un agujero y asignarlo al contenedor $\bar{k}$. Es simple ver que todos los puntos que corta esta desigualdad tienen puntos alternativos que usan la misma cantidad de contenedores y que son conseguibles mediante simples permutaciones. De esta manera, la desigualdad corta puntos simétricos de la formulación preservando algún punto óptimo. Claramente, además de funcionar en el caso de puntos enteros, la desigualdad también corta puntos fraccionarios del poliedro.

Al igual que en el caso anterior, la cantidad de desigualdades presentes en esta familia hace que sea factible la implementación de un algoritmo de separación exhaustivo que revise todos los pares de contenedores e ítems. Este algoritmo de separación es el que se utilizará para la experimentación pertinente.

6.2.4. Resultados experimentales

En esta sección presentamos los resultados obtenidos al experimentar sobre las instancias 3-regulares ya mencionadas. Como comparación, tomaremos por un lado la versión *BaseAS* que mostró un claro mejor comportamiento que el modelo básico. Por otro lado, tomaremos *BaseAS*, juntos con los diferentes cortes mencionados recientemente.

Comenzamos entonces reportando los tiempos de ejecución utilizados en cada instancia por los enfoques *BaseAS* y *BaseASC*, este último correspondiente a la versión donde se utilizan los cortes. Nuevamente se aplicó un límite de 3600 segundos para la resolución de cada instancia en particular. La decisión del llamado a las rutinas de cortes fue dejado a cargo del *solver*. A priori, los algoritmos de separación deberían ser llamados en todos los nodos del árbol de búsqueda, la cantidad de rondas que el *solver* crea suficiente.

Cuadro 6.9: Instancias 3 regulares - 22 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.12	0.02	0.02	0.01	0.01	**0.03**	0.01	0.01	0.01	0.03
	BaseASC	**0.02**	0.02	0.02	0.01	0.01	0.04	0.01	0.01	0.01	0.03
25	BaseAS	0.17	0.2	0.19	**0.18**	0.17	**0.15**	0.17	0.16	0.15	**0.18**
	BaseASC	**0.16**	**0.18**	**0.15**	0.22	0.17	0.18	0.17	**0.15**	**0.13**	0.22
50	BaseAS	**0.38**	**0.45**	**0.52**	**0.53**	**0.64**	**0.59**	**0.43**	**0.48**	**0.48**	0.99
	BaseASC	0.49	0.75	0.68	0.67	0.83	0.72	0.83	0.57	1.15	**0.76**
75	BaseAS	**5.08**	**6.03**	**5.89**	**10.17**	370	**5.8**	**6.37**	**10.61**	**8.76**	**10.35**
	BaseASC	11.2	13.44	8.16	11.25	**322.13**	15.43	14.22	12.23	14.41	11.28
90	BaseAS	3600	87.26	78.17	3600	99.06	3600	**2081**	3600	87.08	3600
	BaseASC	**1884**	**46.21**	**22.85**	3600	**60.88**	**1830**	2402	3600	**46.54**	3600

Cuadro 6.10: Instancias 3 regulares - 24 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.04	0.02	0.01	0.01	0.03	0.01	**0.05**	0.04	**0.03**	0.01
	BaseASC	0.04	0.02	0.01	0.01	**0.02**	0.01	0.06	0.04	0.04	0.01
25	BaseAS	0.19	**0.22**	0.21	0.23	0.24	**0.18**	0.22	1.69	**0.28**	0.21
	BaseASC	0.19	0.24	**0.2**	0.23	0.24	0.21	0.22	**1.68**	0.31	0.21
50	BaseAS	17.81	36.13	27.13	63.51	46.18	31.58	**0.7**	33.36	36.75	57
	BaseASC	**13.9**	**21.54**	**20.92**	**15.94**	**18.92**	**25.04**	0.88	**31.35**	**29.82**	**29.92**
75	BaseAS	**769**	**380**	**666**	**314**	**623**	**908**	**390**	**426**	**684**	**845**
	BaseASC	853.88	1464	871	961	920	937	504	1173	690	1147
90	BaseAS	3600	3600	3600	3600	3600	3600	3045	3600	3600	3600
	BaseASC	**2415**	**1361**	**1734**	**937**	**2726**	**1866**	**1967**	**2319**	3600	**2149**

Los tiempos de ejecución mostrados son dispares y no se ve una clara tendencia en favor de alguna de las opciones. Esencialmente, se pueden observar tres grupos de instancias diferentes. El grupo de las instancias que son resueltas en tiempos muy bajos por ambos enfoques. Las instancias que se resuelven pero tomando un mayor tiempo de cómputo, y las instancias que no llegan a ser resueltas en el tiempo límite.

En el primer grupo de instancias caen todas las que presentan un porcentaje menor de conflictividad. Si bien los tiempos de ambos enfoques tienen algunas disparidades, son bastante comparables.

Luego se encuentran las instancias que también son resueltas hasta la optimalidad, pero en un tiempo no despreciable. En este grupo se observan las instancias con porcentajes intermedios. Aquí

Cuadro 6.11: Instancias 3 regulares - 26 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.09	0.02	**0.04**	**0.23**	**0.18**	**0.23**	**0.18**	**0.28**	0.01	**0.23**
	BaseASC	**0.03**	**0.01**	0.05	0.31	0.32	0.33	0.26	0.34	0.01	0.29
25	BaseAS	2.33	2.5	2.87	**1.68**	2.88	3.15	**1.74**	2.31	2.91	3.23
	BaseASC	**1.78**	**1.85**	**1.95**	2.68	2.88	**2.47**	2.16	**1.91**	**2.48**	**2.75**
50	BaseAS	45.5	**38.07**	80.11	**49.39**	**74.84**	70.58	62.39	**50.64**	**53.6**	**77.52**
	BaseASC	**25.94**	56.55	**35.36**	106	142	**66.99**	**54.17**	60.95	65.96	98.15
75	BaseAS	**1175**	**803**	**1287**	**832**	**592**	3600	**485**	**661**	**710**	**1047**
	BaseASC	2206.63	2090	1455	1624	3313	**2494**	3331	3600	1840	2552
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

Cuadro 6.12: Instancias 3 regulares - 28 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.46	**0.04**	**0.22**	0.42	**0.24**	0.33	**0.26**	**0.24**	**0.23**	0.33
	BaseASC	**0.3**	0.05	0.31	**0.35**	0.27	0.29	0.34	0.32	0.29	0.33
25	BaseAS	2.38	2.74	**2.06**	3.79	4.42	4.33	2.68	4.04	2.52	6.29
	BaseASC	**2.08**	**2.39**	3.54	**3.08**	**3.54**	**2.4**	**2.33**	**2.44**	**2.28**	**4**
50	BaseAS	75.83	**53.09**	**84.34**	140	143	**136**	**67.88**	**88.33**	**68.23**	**145**
	BaseASC	**30.37**	76.9	138	**62.45**	**113**	358	121	109	192	330
75	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

Cuadro 6.13: Instancias 3 regulares - 30 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.45	0.42	**0.3**	**0.3**	**0.31**	0.34	**0.33**	**0.41**	**0.34**	**0.31**
	BaseASC	**0.35**	**0.38**	0.36	0.32	0.35	**0.33**	0.4	0.44	0.36	0.33
25	BaseAS	3.74	**3.2**	3.33	**3.38**	5.09	4.99	3.93	4.81	3.9	**5.97**
	BaseASC	**3.01**	3.37	**2.32**	3.42	**4.07**	**4.26**	**3.45**	**2.88**	**3.81**	6.15
50	BaseAS	3600	3600	3600	3600	3600	3600	158.22	3600	3600	3600
	BaseASC	3600	3600	**2844.98**	3600	3600	3600	**109.85**	3600	3600	3600
75	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

se puede observar que en general el tiempo de la versión sin cortes, suele ser un poco mejor que la versión con cortes. Al parecer, el preprocesamiento incluído en *BaseAS* es suficiente para luego poder cerrar la instancia en un tiempo razonable y el tiempo necesario para separar los cortes incluídos no termina resultando en un decremento del tiempo de cómputo total. Algo interesante a notar en este conjunto de instancias, es que el comportamiento de cada enfoque es distinto. En el caso de la versión con cortes, la cota dual sube, en casi todas las instancias del grupo, mucho más rápidamente que las cotas duales que se van obteniendo en la versión sin cortes. Luego, al llegar al punto en donde el valor óptimo está entre dos opciones, ya que se tiene la cota primal en un valor y la cota dual en una unidad menos, la versión con cortes necesita más tiempo para lograr cerrar la instancia y termina equiparando su tiempo total con la versión que no utiliza los cortes.

Cuadro 6.14: Instancias 3 regulares - 32 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.56	**0.42**	0.41	**0.39**	0.4	**0.39**	**0.36**	**0.34**	**0.38**	**0.36**
	BaseASC	**0.48**	0.43	0.41	0.43	0.4	0.41	0.4	0.42	0.43	0.38
25	BaseAS	5.8	5.36	5.55	**128.21**	6.01	3600	4.69	2.49	**6.16**	**376.87**
	BaseASC	**3.9**	**2.95**	**3.52**	548.7	**3.95**	**261.4**	**3.56**	**2.37**	6.67	422.67
50	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
75	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

Por último, están las instancias que no son resueltas por ambos enfoques en el tiempo límite. En este grupo sí se ve un mejor desempeño de la versión con cortes. Si miramos solamente los tiempos, podemos ver que hay varias instancias no resueltas por *BaseAS*, que sí son resueltas por la versión con cortes. Por ejemplo, si miramos las instancias de 90 % de conflictividad en la tabla de 24 ítems, podemos ver que este comportamiento sucede con todas las instancias menos una. Además, en las instancias que no son resueltas por ninguna de las dos versiones en la hora de corrida, hay una diferencia muy clara a favor de la versión con cortes cuando se mira la calidad de la cota dual final obtenida.

A continuación, se muestran las cotas duales obtenidas por cada uno de los enfoques en varias instancias pertenecientes al grupo de las que no fueron resueltas en el tiempo límite.

Cuadro 6.15: Instancias 3 regulares - 26 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
90	BaseAS	6.18	6	6	6	6	6	6	6	6	6
	BaseASC	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**

Cuadro 6.16: Instancias 3 regulares - 28 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
75	BaseAS	6	6	6	5.5	5.8	6	6	6	5.5	5.5
	BaseASC	6	6	6	**6**	**6**	6	6	6	**6**	**6**
90	BaseAS	6	6	6	6	6	5.3	6	6	6	6
	BaseASC	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**

Cuadro 6.17: Instancias 3 regulares - 30 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
50	BaseAS	5	5	5	5	5	5.3	5	5	5	5
	BaseASC	**5.6**	5	**7**	**6**	**6**	**5.6**	**6**	**6**	**6**	5
75	BaseAS	6	5	6	6	5.6	5.2	5.3	5.1	5.8	5
	BaseASC	6	**6**	6	6	**6**	**6**	**6**	**6**	**6**	**6**
90	BaseAS	6	5.5	5.3	5.2	5	6	5	5.7	5.2	5.5
	BaseASC	**7**	**6.5**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**

Cuadro 6.18: Instancias 3 regulares - 32 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
50	BaseAS	5	5	5	5	5	5	5	5.1	5	5
	BaseASC	**5.3**	**5.3**	5	**5.3**	**6**	5	**5.2**	**6**	5	**6**
75	BaseAS	5.3	5	5	5	5	5	6	6	5	5
	BaseASC	**5.7**	**6**	**6**	**6**	**6**	**6**	6	6	**6**	**6**
90	BaseAS	6	5	5.3	6	5.2	5	5	5.3	5.3	5
	BaseASC	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**

En estos resultados se puede observar cómo en las instancias más complejas, la versión con cortes obtiene una cota dual igual o superior a *BaseAS* en la totalidad de las instancias.

En resumen, concluímos en que la inclusión de los planos de corte no parecen resultar en un beneficio claro en cuanto a los tiempos de ejecución de las instancias propuestas, pero sí tienen un impacto muy positivo en la calidad de las cotas duales de las instancias. En conjunto con esto, la versión con cortes logra resolver 11 instancias más que *BaseAS* hasta la optimalidad. Por otro lado, el agregado de los cortes resulta en un decremento muy abrupto en la cantidad de memoria necesaria para la resolución de las instancias. Este aspecto no es un punto fundamental en las instancias que son resueltas en poco tiempo, pero sí constituyen un problema importante para las instancias que son ejecutadas por un intervalo mayor de tiempo.

Además, si bien no se logra un beneficio claro en las instancias que sí son resueltas en cuanto a su tiempo de cómputo al agregar los cortes por sí solos, se podrá ver en secciones posteriores que los cortes elegidos tienen un buen comportamiento con otras componentes validando la importancia de su inclusión en el algoritmo.

6.2.5. Otras desigualdades

Como ya se mencionó, por diversos motivos no siempre una desigualdad válida fuerte como una faceta se traduce en una mejora notable en la práctica. Puede suceder que sea realmente muy costoso encontrar la desigualdad que es violada por el punto óptimo de la relajación lineal. También podría suceder que la familia de desigualdades corte puntos del poliedro que se encuentren lejos del óptimo que marca la función objetivo entonces nunca serán elegidas por la resolución de la relajación lineal, por lo que tampoco sería utilizada esta desigualdad. Varias familias de desigualdades que fueron derivadas durante el estudio poliedral no mostraron un buen comportamiento en la práctica por diferentes motivos. En esta sección mostramos una experimentación acotada sobre una de estas familias.

Suma reforzada de hiperarista

Una de las familias de desigualdades derivadas a partir de reforzar la desigualdad original de hiperarista es la siguiente:

$$\sum_{k=\bar{k}}^{n} \sum_{i \in H} x_{ik} \leq (r-1) y_{\bar{k}} + y_{\bar{k}+1} \tag{6.3}$$

En el capítulo de estudio poliedral se pudo ver que esta familia define facetas del poliedro bajo ciertas hipótesis. Si bien esto es un indicador de que la familia podría comportarse bien en la práctica, se notan algunas particularidades de la familia que podrían generar dudas sobre su posible funcionamiento. En primer lugar, la cantidad de variables del lado izquierdo no es alta con respecto a las del lado derecho. En particular en las instancias que estamos experimentando, el lado izquierdo como mucho podrá sumar 3 mientras que del lado derecho hay dos variables pero una con coeficiente 2. Además las variables del lado derecho se encontrarán prendidas para todos los

primeros contenedores. Por lo que en el caso de lograr violar esta desigualdad, solo podrá suceder para los contenedores más altos. Esto nos lleva a la segunda observación. Esta familia se probó que define facetas del poliedro si se cumple la hipótesis $\bar{k} < n - r + 1$. Sin embargo, en la mayoría de las instancias que estamos experimentando, la cota dual sube rápidamente haciendo que los contenedores bajos se encuentren utilizados fácilmente. De esta manera, los contenedores donde se podría violar la desigualdad son justamente los que no caen en esta hipótesis, donde la desigualdad es más débil desde un punto de vista teórico.

Al realizar una experimentación preliminar con esta familia, se pudo ver que no resultó beneficiosa su inclusión al enfoque *BaseAS*. A continuación, se muestran los resultados obtenidos para las instancias de 24 ítems y 50 % de conflictividad.

Cuadro 6.19: Instancias 3 regulares - 24 ítems - 50 % conflictividad

Algoritmo	0	1	2	3	4	5	6	7	8	9
BaseAS	17.81	36.13	27.13	63.51	46.18	31.58	**0.7**	33.36	36.75	57
BaseASC	**13.9**	**21.54**	**20.92**	**15.94**	**18.92**	**25.04**	0.88	**31.35**	**29.82**	**29.92**
BaseAS+SR	161	122	91.66	145	190	176	1.41	113	109	222

Se puede observar que el desempeño del enfoque con la nueva familia está bastante lejos tanto del obtenido por *BaseAS*, como por la versión con cortes. Observando el detalle de los experimentos corridos, se puede ver que este incremento del tiempo sucedió por dos motivos. Por un lado, el árbol de búsqueda resultó más extenso al utilizar la nueva familia de cortes, para lo cual no hay una explicación clara. Por otro lado, al ir agregando los nuevos cortes, las relajaciones resultaron más pesadas que en el caso *BaseAS*, pero sin obtener un beneficio tangible en la cota dual como sucede en la versión con cortes. En este caso se presentó el resultado sobre un subgrupo de instancias pero este comportamiento es análogo en la mayoría de las instancias 3-regulares. Por otro lado, al realizar los experimentos tampoco se observó un beneficio desde la métrica de la cota dual obtenida. En resumen, se considera que la inclusión de esta familia de desigualdades no resultó beneficioso por lo que se desestima su utilización en experimentos futuros.

6.3. Branching

Luego de experimentar con diferentes planos de corte posibles, en esta sección nos proponemos experimentar con otra componente fundamental de un algoritmo branch-and-cut, el *branching*. Al resolver la relajación lineal de un nodo del árbol de búsqueda, necesitamos apartar el punto óptimo en el caso de que éste sea fraccionario. En primera instancia se hace una búsqueda de todos los planos de corte agregados al algoritmo para ver si alguno de ellos logra separar a este punto del poliedro. En el caso de que ningún plano logre deshacerse de este óptimo, la siguiente acción es realizar un *branching*. La forma más clásica de *branching* consiste en tomar alguna variable que tenga valor fraccionario en el óptimo del nodo y luego crear dos subproblemas. Un subproblema en el cual solo se consideren valores para esta variable que están por debajo de la parte entera por debajo del valor fraccionario que mostraba en la relajación lineal, y otro subproblema en el que se consideren los valores que estén por encima de la parte entera por arriba. En el caso de contar solo con variables binarias, como en nuestro problema, esto significa que un subproblema será el que considere la variable igualada a 0, y el otro subproblema considerará la variable igualada a 1.

En caso de que no sea especificado, la decisión de sobre qué variable realizar el *branching* queda a cargo del framework utilizado, en este caso CPLEX. Si bien CPLEX tiene diferentes heurísticas para tratar de optimizar este tipo de decisiones basándose en la estructura del problema, muchas veces resulta beneficioso introducir una regla de *branching* que haga un uso del entendimiento particular del problema.

En nuestro problema, se detectaron algunas situaciones particulares que motivan la inclusión de una regla de *branching* particular.

En primer lugar, un resultado que ya se vió que resulta beneficioso es la fijación de variables. Durante la etapa de preprocesamiento se pudo ver que la eliminación de varias variables tuvo un impacto muy positivo en los tiempos de cómputo al reducir considerablemente el espacio de búsqueda. El *branching* no es más que una fijación de una variable, creando dos subproblemas donde la variable esta fijada con diferentes valores. Si bien el *branching* es un procedimiento que fija una sola variable, en realidad puede ser que esto derive en más variables fijadas. El caso más simple se da cuando se fija una variable de asignación en 1, lo cual automáticamente fija la variable de uso de contenedor en 1.

En nuestro problema, una vez aplicado los preprocesamientos explicados, no todos los ítems tienen las mismas posibilidades. Los ítems de numeración más baja tiene menos posibilidades que los ítems más altos, los cuales pueden ser asignados a cualquier contenedor. De esta manera, realizar el *branching* sobre los ítems más bajos tiene un impacto más grande en cuanto a la reducción de posibilidades que se deriva. Por ejemplo, en el caso extremo, si realizamos el *branching* sobre la variable de asignación del ítem 2 al contenedor 1, automáticamente también estaremos fijando la variable de asignación del ítem 2 al contenedor 2, ya que son las únicas dos opciones que tiene este ítem.

En consonancia con lo explicado, se realizaron experimentos previos para visualizar el comportamiento de la estrategia de branching utilizada por CPLEX. En estos experimentos se pudo ver que muchas veces el *solver* decidía realizar el *branching* por variables de asignación de ítems altos, habiendo tenido la posibilidad de realizar el procedimiento con un ítem mucho más bajo. Luego de cambiar la regla de *branching* del *solver* por una en donde se le diese prioridad a los ítems más bajos, los tiempos de cómputo de las instancias utilizadas mejoraron notablemente.

Además, la noción de *branching* mencionada, no solo parece una idea interesante por sí misma, sino que además intuitivamente parece tener una buena cooperación con las desigualdades válidas utilizadas. Al ir elevando las cotas duales, los primeros contenedores siempre se encuentran utilizados por la restricción de uso en orden de los mismos. Luego, el juego de las variables se empieza a dar con las variables correspondientes a los últimos contenedores, ya que son las que dictaminarán si hacen falta tantos contenedores como indica la cota primal actual, o si ésta puede ser mejorada. De esta manera, se utilizaron algunos cortes en la sección anterior, tales como la desigualdad *Inhabilita contenedores* o la desigualdad *Sin agujero*, que se benefician cuando las variables sobre los contenedores altos tienen variabilidad, haciendo que entonces se violen o no los planos de corte asociados. Si se realiza el branching por estas variables y se las fija, estos cortes casi con seguridad no tendrán ninguna injerencia. Nuevamente, esto indica que la elección de realizar el *branching* sobre los primeros ítems y los primeros contenedores parecería ser, al menos en una idea intuitiva, más adecuada.

Por lo explicado en los párrafos anteriores, se propone entonces una prioridad de *branching lexicográfica*. Tendrán prioridad para caer en el proceso de *branching* los ítems con numeración más baja y, luego, las variables de asignación asociadas a los contenedores más bajos.

A continuación mostraremos los tiempos de cómputo obtenidos en las instancias 3-regulares ya presentadas. En este punto realizaremos una comparación de las alternativas propuestas, por lo que reportaremos las siguientes versiones. *BaseAS*, *BaseASC*, *BaseASCB* que representa a la inclusión del *branching* junto con los cortes, y *BaseASB* que es la versión con *branching* pero sin incluir los cortes.

En los resultados presentados se puede observar que la variante *BaseASCB* resulta claramente ganadora. Obviando algunas instancias particulares donde el tiempo de resolución está en la fracción de segundo, el enfoque que junta los planos de corte y la estrategia de *branching* obtiene el mejor tiempo de cómputo en todas las instancias. Además, en varias instancias la diferencia del

Cuadro 6.20: Instancias 3 regulares - 22 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	**0.12**	0.02	0.02	0.01	0.01	**0.03**	0.01	0.01	0.01	0.03
	BaseASC	0.02	0.02	0.02	0.01	0.01	0.04	0.01	0.01	0.01	0.03
	BaseASB	0.02	0.02	0.02	0.01	0.01	**0.03**	0.01	0.01	0.01	0.03
	BaseASCB	0.02	0.02	0.02	0.01	0.01	**0.03**	0.01	0.01	0.01	0.03
25	BaseAS	0.17	0.2	0.19	0.18	0.17	0.15	0.17	0.16	0.15	0.18
	BaseASC	0.16	0.18	0.15	0.22	0.17	0.18	0.17	0.15	0.13	0.22
	BaseASB	0.15	**0.13**	**0.13**	**0.17**	**0.16**	**0.14**	**0.13**	**0.13**	0.14	**0.17**
	BaseASCB	**0.13**	0.16	**0.13**	**0.17**	0.17	0.18	0.15	**0.13**	**0.12**	**0.17**
50	BaseAS	0.38	**0.45**	0.52	**0.53**	0.64	0.59	0.43	0.48	0.48	0.99
	BaseASC	0.49	0.75	0.68	0.67	0.83	0.72	0.83	0.57	1.15	0.76
	BaseASB	0.34	0.48	0.68	0.81	**0.62**	**0.58**	**0.37**	**0.39**	**0.47**	0.65
	BaseASCB	**0.32**	0.61	**0.46**	0.59	0.69	0.78	0.91	0.44	0.56	**0.51**
75	BaseAS	5.08	6.03	5.89	10.17	370.99	5.8	6.37	10.61	8.76	10.35
	BaseASC	11.2	13.44	8.16	11.25	322	15.43	14.22	12.23	14.41	11.28
	BaseASB	3.55	4.92	5.01	8.48	222	**4.41**	**3.24**	3.44	4.53	8.02
	BaseASCB	**2.25**	**2.47**	**2.55**	**3.01**	**68.98**	7.59	5.62	**2.61**	**2.82**	**2.99**
90	BaseAS	3600	87.26	78.17	3600	99.06	3600	2081	3600	87.08	3600
	BaseASC	1884	46.21	22.85	3600	60.88	1830	2402	3600	46.54	3600
	BaseASB	2919	74.13	88.7	3600	114	3600	1822	2619	65.07	3250
	BaseASCB	**690**	**11.23**	**9.35**	**1101**	**8.52**	**493**	**899**	**711**	**8.97**	**621**

Cuadro 6.21: Instancias 3 regulares - 24 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.04	**0.02**	0.01	0.01	0.03	0.01	**0.05**	0.04	**0.03**	0.01
	BaseASC	0.04	**0.02**	0.01	0.01	**0.02**	0.01	0.06	0.04	0.04	0.01
	BaseASB	0.04	0.03	0.01	0.01	**0.02**	0.01	**0.05**	0.04	0.04	0.01
	BaseASCB	0.04	**0.02**	0.01	0.01	**0.02**	0.01	0.07	0.04	**0.03**	0.01
25	BaseAS	0.19	0.22	0.21	0.23	0.24	**0.18**	0.22	1.69	0.28	0.21
	BaseASC	0.19	0.24	0.2	0.23	0.24	0.21	0.22	1.68	0.31	0.21
	BaseASB	0.24	0.26	**0.17**	**0.21**	**0.22**	0.26	**0.16**	1.38	**0.23**	0.21
	BaseASCB	**0.17**	**0.2**	0.19	0.23	**0.22**	0.19	0.19	**1.31**	0.24	0.21
50	BaseAS	17.81	36.13	27.13	63.51	46.18	31.58	**0.7**	33.36	36.75	57
	BaseASC	13.9	21.54	20.92	15.94	18.92	25.04	0.88	31.35	29.82	29.92
	BaseASB	16.89	27.28	13.99	25.86	22.19	31.89	0.71	21.37	21.15	46.53
	BaseASCB	**10.15**	**13.03**	**7.74**	**13.85**	**11.5**	**17.36**	1.1	**12.1**	**14.72**	**19.12**
75	BaseAS	769	380	666	314	623	908	390	426	684	845
	BaseASC	853	1464	871	961	920	937	504	1173	690	1147
	BaseASB	328	308	340	317	269	570	172	496	430	672
	BaseASCB	**77.49**	**67.87**	**103**	**97.69**	**87.6**	**131**	**77.71**	**130**	**113**	**122**
90	BaseAS	3600	3600	3600	3600	3600	3600	3045	3600	3600	3600
	BaseASC	2415	1361	1734	937	2726	1866	1967	2319	3600	2149
	BaseASB	3600	3600	3600	3600	3600	3600	2796	3600	3600	3600
	BaseASCB	**435**	**269**	**520**	**431**	**280**	**463**	**489**	**386**	**463**	**553**

tiempo de resolución es realmente alta, llegando a ser hasta menos del 20 % del tiempo utilizado por el enfoque que consiguió el segundo mejor tiempo.

A priori, se podría pensar que la notable mejora de desempeño se debe a la introducción de una regla específica de *branching*, ya que la inclusión de cortes no había mostrado tal diferencial en los

Cuadro 6.22: Instancias 3 regulares - 26 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.09	0.02	**0.04**	0.23	0.18	0.23	0.18	**0.28**	**0.01**	0.23
	BaseASC	**0.03**	**0.01**	0.05	0.31	0.32	0.33	0.26	0.34	**0.01**	0.29
	BaseASB	0.11	0.02	**0.04**	**0.19**	**0.19**	**0.2**	**0.16**	0.3	**0.01**	**0.19**
	BaseASCB	**0.03**	0.02	**0.04**	0.38	0.25	0.36	0.22	0.32	0.02	0.23
25	BaseAS	2.33	2.5	2.87	1.68	2.88	3.15	1.74	2.31	2.91	3.23
	BaseASC	1.78	1.85	1.95	2.68	2.88	**2.47**	2.16	1.91	2.48	2.75
	BaseASB	1.19	2.47	**1.48**	1.96	2.7	2.61	1.82	1.59	2.08	2.64
	BaseASCB	**0.89**	**1.52**	1.52	**1.55**	**1.72**	2.58	**1.61**	**1.09**	**1.77**	**2.41**
50	BaseAS	45.5	38.07	80.11	49.39	74.84	70.58	62.39	50.64	53.6	77.52
	BaseASC	25.94	56.55	35.36	106	142	66.99	54.17	60.95	65.96	98.15
	BaseASB	17.83	23.86	26.95	44.42	26.46	33.48	23.97	28.04	54.14	45.16
	BaseASCB	**13.84**	**14.85**	**18.58**	**28.71**	**13.6**	**19.36**	**14.97**	**17.06**	**26.91**	**26.98**
75	BaseAS	1175	803	1287	832	592	3600	485	661	710	1047
	BaseASC	2206	2090	1455	1624	3313	2494	3331	3600	1840	2552
	BaseASB	398	555	543	514	423	716	528	330	838	779
	BaseASCB	**83.95**	**110**	**102**	**99.41**	**98.4**	**141**	**91.73**	**108**	**106**	**104**
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	**246**	**1059**	3600	3600	3600	3600	3600	3600	3600	3600

Cuadro 6.23: Instancias 3 regulares - 28 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.46	**0.04**	**0.22**	0.42	**0.24**	0.33	0.26	**0.24**	0.23	0.33
	BaseASC	0.3	0.05	0.31	0.35	0.27	0.29	0.34	0.32	0.29	0.33
	BaseASB	0.31	0.05	**0.22**	**0.29**	0.31	**0.22**	**0.2**	0.3	**0.21**	0.33
	BaseASCB	**0.25**	0.05	0.26	**0.29**	0.27	0.3	0.33	0.36	0.32	**0.27**
25	BaseAS	2.38	2.74	2.06	3.79	4.42	4.33	2.68	4.04	2.52	6.29
	BaseASC	2.08	2.39	3.54	3.08	3.54	2.4	2.33	2.44	2.28	4
	BaseASB	1.66	2.56	1.81	2.3	2.53	**2.13**	2.09	2.26	1.98	3.93
	BaseASCB	**1.42**	**1.88**	**1.74**	**1.81**	**2.2**	2.43	**1.81**	**1.63**	**1.92**	**2.65**
50	BaseAS	75.83	53.09	84.34	140	143	136	67.88	88.33	68.23	145
	BaseASC	30.37	76.9	138	62.45	113	358	121	109	192	330
	BaseASB	86.64	59.11	48	53.93	36.58	49.64	52.21	43.42	42.82	91.49
	BaseASCB	**20.22**	**39.83**	**34.12**	**34.04**	**19.63**	**28.79**	**23.97**	**28.87**	**30.92**	**44.77**
75	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

tiempos de cómputo. Es por esto que se incluyeron en la tablas los resultados correspondientes a la versión que utiliza el *branching* sin los cortes. De dicha versión se notan dos cosas interesantes. En primer lugar, es claro que la regla de *branching* tiene un buen desempeño, mejorando en muchos casos la performance del modelo *BaseAS*. Sin embargo, sacando las instancias de resolución trivial,

Cuadro 6.24: Instancias 3 regulares - 30 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.45	0.42	0.3	**0.3**	0.31	0.34	0.33	0.41	0.34	0.31
	BaseASC	**0.35**	0.38	0.36	0.32	0.35	0.33	0.4	0.44	0.36	0.33
	BaseASB	0.43	**0.33**	**0.25**	0.31	**0.3**	**0.25**	**0.29**	0.39	**0.3**	**0.3**
	BaseASCB	0.39	**0.33**	0.37	0.33	0.31	0.28	0.34	**0.37**	0.31	0.32
25	BaseAS	3.74	3.2	3.33	3.38	5.09	4.99	3.93	4.81	3.9	5.97
	BaseASC	3.01	3.37	2.32	3.42	4.07	4.26	3.45	2.88	3.81	6.15
	BaseASB	2.33	**2.02**	1.7	3.07	3.1	3	2.54	2.69	2.94	4.89
	BaseASCB	**2.23**	2.16	**1.39**	**2.93**	**2.02**	**2.78**	**1.83**	**2.02**	**2.28**	**4.02**
50	BaseAS	3600	3600	3600	3600	3600	3600	158	3600	3600	3600
	BaseASC	3600	3600	2844	3600	3600	3600	109	3600	3600	3600
	BaseASB	3600	489	3600	3600	3600	3600	57.86	3600	3600	3600
	BaseASCB	**2352**	**797**	**1887**	**2219**	**1768**	**2048**	**32.41**	**1885**	**2023**	**2909**
75	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

Cuadro 6.25: Instancias 3 regulares - 32 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
10	BaseAS	0.56	0.42	0.41	0.39	0.4	0.39	0.36	**0.34**	0.38	0.36
	BaseASC	0.48	0.43	0.41	0.43	0.4	0.41	0.4	0.42	0.43	0.38
	BaseASB	0.46	**0.33**	0.38	**0.3**	**0.34**	**0.32**	**0.35**	0.36	**0.3**	**0.35**
	BaseASCB	**0.41**	0.36	**0.36**	0.37	0.35	0.33	0.37	0.39	0.34	0.38
25	BaseAS	5.8	5.36	5.55	128	6.01	3600	4.69	2.49	6.16	376
	BaseASC	3.9	2.95	3.52	548	3.95	261	3.56	2.37	6.67	422
	BaseASB	2.83	25.71	3.41	**25.38**	3.39	281	**2.4**	2.74	**3.79**	386
	BaseASCB	**2.7**	**2.61**	**3.18**	129	**2.88**	**115**	3.32	**2.3**	3.86	**99.18**
50	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	**2212**	**2723**	3600	**3306**	**2434**	3600	**2449**	**2544**	**3448**	**3507**
75	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600	3600	3600	3600	3600	3600

este enfoque nunca es ganador con respecto a la conjunción de *branching* y cortes, quedando bastante lejos en muchas de las instancias. Lo que es más, el *branching* solo en muchas instancias tampoco resulta ganador contra la versión que sí utiliza cortes pero no *branching*. De esta manera, la evidencia parece sostener que la mejora del último enfoque proviene de una buena interacción

entre las diferentes componentes.

Más allá de los tiempos de cómputo en sí, el enfoque *BaseASCB* logró resolver varias instancias que no se encontraban resueltas hasta la optimalidad en el límite de 3600 segundos. En particular, con la nueva versión, todas las instancias de 22 y 24 ítems se logran resolver. También hay otros grupos de instancias donde la mejora es notoria, como las instancias de 50 % de conflictividad en 30 y 32 ítems.

Por otro lado, veremos que en las instancias que todavía no pueden ser resueltas en el tiempo límite, las mejoras también son considerables. A continuación mostramos las cotas duales obtenidas en los conjuntos de instancias que todavía no se encuentran resultas. En las próximas tablas, no se muestran los resultados correspondientes a *BaseAS*, ya que ya se mostró que en este aspecto es siempre inferior a la versión que utiliza cortes.

Cuadro 6.26: Instancias 3 regulares - 26 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
	BaseASC	7	7	7	7	7	7	7	7	7	7
90	BaseASB	6.75	6	6	6	6	6	6	6.75	6	6
	BaseASCB	**8**	**8**	**7.3**	**7.3**	**7.2**	**7.2**	**7.1**	**7.3**	**7.2**	**7.2**

Cuadro 6.27: Instancias 3 regulares - 28 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
	BaseASC	6	6	6	6	6	6	6	6	6	6
75	BaseASB	6	6	6	6	6	6	6	6	6	6
	BaseASCB	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
	BaseASC	7	7	7	7	7	7	7	7	7	7
90	BaseASB	6	6	6	6	6	6	6	6	6	6
	BaseASCB	**7.3**	**7.4**	**7.3**	**7.2**	**7.3**	**7.1**	**7.2**	**7.2**	**7.4**	**7.4**

Cuadro 6.28: Instancias 3 regulares - 30 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
	BaseASC	6	6	6	6	6	6	6	6	6	6
75	BaseASB	6	6	6	6	6	6	6	6	6	6
	BaseASCB	**6.7**	**7**	**6.8**	**6.6**	**7**	**7**	**7**	**7**	**6.6**	**6.5**
	BaseASC	7	6.5	7	7	7	7	7	7	7	7
90	BaseASB	6	6	6	6	6	6	6	6	6	6
	BaseASCB	**7.2**	**7.4**	**7.2**	**7.3**	**7.2**	**7.4**	**7.3**	**7.3**	**7.2**	**7.1**

Cuadro 6.29: Instancias 3 regulares - 32 ítems

%	Algoritmo	0	1	2	3	4	5	6	7	8	9
	BaseASC	5.7	6	6	6	6	6	6	6	6	6
75	BaseASB	5	5	5.5	6	5.8	5	6	5.5	5	5.8
	BaseASCB	**6.3**	**6.4**	**6.4**	**6.4**	**6.3**	**6.3**	**6.4**	**6.4**	**6.4**	**6.4**
	BaseASC	7	7	7	7	7	7	7	7	7	7
90	BaseASB	6	5	6	6	6	5	5.6	5.5	6	6
	BaseASCB	**7.2**	**7.1**	**7.3**	**7.3**	**7.2**	**7.1**	**7.2**	**7.3**	**7.3**	**7.3**

En este aspecto podemos obtener conclusiones muy similares a cuando se reportan los tiempos. La versión *BaseASCB* tiene un desempeño muy superior al resto de los enfoques. En casi todas las instancias consigue cotas duales más altas. Hay dos puntos interesantes a remarcar que no se pueden observar en las tablas pero que valen la pena tener en cuenta para entender la diferencia de desempeño. Por un lado, destacamos que si bien las tablas presentan valores fraccionarios, en realidad es sabido que la función objetivo solo admite valores enteros, por lo que se podría pensar que la cota dual real es la parte entera por arriba de la informada. Esto implica que si en una instancia la cota dual del enfoque con cortes es 7, y la cota dual del enfoque con cortes y *branching* es 7.08, entonces la diferencia real no es de 0.08 si no de una unidad entera. Lo que esto significa es que al enfoque con *branching* le alcanzó la hora de cómputo para poder descartar todo el hiperplano de soluciones posibles donde se utilizan solamente 7 contenedores, mientras que el enfoque que solo usa cortes todavía debe explorar una porción de ese hiperplano.

Por otro lado, por la naturaleza convexa del poliedro, a medida que la cantidad de contenedores aumenta, el tamaño de cada uno de los hiperplanos va aumentando abruptamente. Esto quiere decir que la eliminación de cada uno de los hiperplanos, cantidad de contenedores igual a 5, cantidad de contenedores igual a 6, etc, resulta cada vez más complicada. Es decir, una diferencia de una unidad en la cota dual, no significa siempre lo mismo. Si comparásemos el enfoque *BaseASBC* contra *BaseAS* podríamos ver que hay instancias en donde la cota dual en una hora de cómputo subió, por ejemplo de 5 a 8. Esto significa que el enfoque *BaseASBC* logró sacarse de encima el hiperplano de utilización de 5 contenedores, pero luego también pudo sacar de encima el 6 y el 7 que son mucho más dificiles de explorar debido a sus tamaños.

Por todo lo remarcado, consideramos que la versión *BaseASCB* es la de mejor rendimiento, logrando un gran diferencial por sobre la versión básica de nuestro algoritmo.

Con estos últimos experimentos finalizamos el estudio de las diferentes componentes sobre las instancias 3-regulares. En la secciones siguientes buscaremos analizar si los resultados obtenidos se generalizan para instancias de otra naturaleza, observando así la generalidad y robustez del enfoque propuesto.

6.4. Experimentación 2-3-4

En las secciones anteriores, se utilizaron instancias 3-regulares para realizar toda la experimentación pertinente y así poder extraer conclusiones sobre qué componentes son deseables para el algoritmo general. La idea detrás de dicha elección de instancias fue tener un ambiente en donde la relación entre las diferentes hiperaristas fuese bien comprendida al tener todas ellas el mismo nivel de restrictividad. Como ya mencionamos varias veces durante este trabajo, el tamaño de las hiperaristas parece imponer un nivel de restrictividad que varía en órdenes de magnitud, haciendo que sea difícil *a priori* comparar el nivel de restricción que impone un arista de tamaño 2, contra varias hiperaristas de tamaño 5 por ejemplo.

Si bien las instancias presentadas resultaron útiles para la experimentación propuesta, el objetivo es diseñar un algoritmo lo más general posible, por lo que en esta sección nos abocaremos a experimentar con instancias un poco más generales que las anteriores. Consideramos entonces instancias con diferentes tamaños de hiperaristas en el hipergrafo conflicto. Esta elección de tamaños tampoco puede ser aleatoria ya que la disparidad en la restrictitividad de cada tamaño hace que sea difícil mezclar hiperaristas de tamaño 2 con otras de tamaño 9, por decir un ejemplo, y poder entender si realmente las hiperaristas de mayor tamaño tienen algún rol en la instancia. Además, al insertar una arista de tamaño 2 al hipergrafo, implícitamente se están insertando muchísimas hiperaristas de tamaño 9 al mismo. De esta manera, aún un bajo porcentaje de aristas de tamaño 2, ya implicaría casi todas las hiperaristas de tamaño 9.

Para estas instancias, consideraremos conflictos de tamaño 2, 3 y 4. Como mencionamos recientemente, el porcentaje de conflictos de tamaño 2, no puede ser muy elevado ya que casi todos

los conflictos de tamaño mayor se encontrarían implicados. Por un lado, no queremos que suceda esto porque entonces no estaríamos experimentando sobre instancias más generales, si no que las mismas tenderían fuertemente a parecerse a las instancias 2-regulares. Por otro lado, este tipo de instancias corresponden al Problema de Coloreo en Grafos, para el cual existe una vasta cantidad de resultados particulares.

En relación a los porcentajes de cada tipo de conflicto, veamos el siguiente análisis. Si tomamos un hipergrafo de una cierta cantidad de vértices y le agregamos aristas de tamaño 2 aleatoriamente hasta llegar al 50 % de conflictividad, entonces lógicamente entre dos vértices cualesquiera hay una probabilidad de 0.5 de que exista un conflicto entre ellos. Si pensamos en conjuntos de tres vértices, entonces la existencia de cualquiera de las aristas que une a dos de ellos hace que el conjunto de 3 vértices conflictúe para lo que es la definición de nuestro problema. La probabilidad de que el conjunto conflictúe por una restricción de arista es de 1 menos la probabilidad que no conflictúe. Luego, como un conjunto de tres vértices posee 3 aristas de tamaño 2 que los relaciona, entonces la probabilidad de que el conjunto no conflictúe por restricciones de arista es de $\frac{1}{2}^3$=0.125. Esto quiere decir que si al hipergrafo se le insertan 50 % de aristas de tamaño 2, automáticamente tendremos un 87.5 % de las hiperaristas de tamaño 3 implicadas.

Con el mismo argumento, un 50 % de aristas de tamaño 2, implican un 97 % de las hiperaristas de tamaño 4 y prácticamente el 100 % de las hiperaristas de tamaño 9. Es por esto que si queremos crear instancias donde la interrelación entre los conflictos de diferentes tamaños tenga un rol que no sea subsumido por los conflictos de tamaño 2, debemos utilizar porcentajes de conflictos específicos.

En la siguiente tabla podemos observar más valores como los analizados recientemente. Es decir, dado un porcentaje de aristas de tamaño 2, que porcentaje de hiperaristas de tamaño 3 y 4 son implicadas.

Cuadro 6.30: Porcentajes de hiperaristas implicadas

Tamaño 2	Tamaño 3	Tamaño 4
5 %	15 %	27 %
10 %	27 %	47 %
15 %	39 %	62 %
20 %	48 %	74 %
25 %	58 %	83 %
30 %	66 %	89 %

Como se puede observar en la tabla anterior, ni bien el porcentaje de aristas de tamaño 2 empieza a crecer, las hiperaristas implicadas de tamaño 3 y 4 también crecen abruptamente. La idea es poder tener instancias en donde se pueda variar los porcentajes de los tres tamaños y no caer en casos donde solo importen las aristas de un tipo porque las de otro tipo están todas implicadas. De esta manera, se eligieron 4 combinaciones de porcentajes. De aristas de tamaño 2 solamente se utilizarán 5 % o 10 %. Se tomó esta decisión porque por sobre las hiperaristas de tamaño 3 y 4 implicadas, se quieren agregar un cierto porcentaje de hiperaristas propias. Por lo cual ya tomar valores más grandes de porcentajes de aristas de tamaño 2, no dejaría casi lugar a las hiperaristas de tamaño 4. Luego, cuando se utilice 5 % de aristas de tamaño 2, se utilizarán 10 % tanto de tamaño 3 como de tamaño 4 en un caso, y 20 % en otro. Cuando el porcentaje de aristas sea 10 % se combinará con 10 % y 15 % de hiperaristas de tamaño 3 y 4. Cabe observar que los porcentajes mencionados de hiperaristas de tamaño 3 y 4 son por encima de las hiperaristas implicadas. Las hiperaristas a insertar tienen como requisito no estar siendo implicadas por restricciones de menor tamaño.

A continuación se muestran los resultados obtenidos para la experimentación con este tipo de instancias. Por cada combinación de porcentajes elegida, se crearon 5 instancias. La idea de esta sección es poder visualizar la concordancia o no del comportamiento observado en las instancias 3-regulares en instancias más generales. De esta manera, lo primero que se reportará es el tiempo

en segundos utilizados para resolver cada instancia, nuevamente con un tiempo límite de una hora. Al igual que al finalizar la sección anterior, se contemplarán 4 versiones del algoritmo. *BaseAS* que involucra al modelo base junto con el preprocesamiento para rompimiento de simetrías en las variables, junto con los preprocesamientos pertinentes al tener aristas de tamaño 2. *BaseASC*, que se refiere al modelo *BaseAS* más los diferentes cortes ya mencionados. *BaseASB*, que se refiere al modelo *BaseAS* más la utilización del *branching* lexicográfico. *BaseASCB*, que se refiere al modelo *BaseASB* más los cortes ya mencionados y el *branching* lexicográfico. En el caso de las versiones con cortes clique, no se buscarán cliques con aristas de tamaño 2, ya que las densidades utilizadas generan una probabilidad muy baja de cliques de tamaños interesantes. Se procederá entonces, al igual que en las instancias 3-regulares, a buscar cliques con hiperaristas de tamaño 3, teniendo en cuenta tanto las hiperaristas de tamaño 3 explícitas del hipergrafo, como las implicadas por las de tamaño 2.

Cuadro 6.31: Instancias 2-3-4 - 22 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	0.47	**0.45**	0.29	0.8	0.38
	BaseASC	0.62	1.13	0.29	1.2	0.38
	BaseASB	**0.46**	0.54	**0.27**	**0.23**	**0.33**
	BaseASCB	**0.46**	0.74	**0.27**	**0.23**	0.47
5-20-20	BaseAS	123.93	113.49	197.61	131.75	197.59
	BaseASC	38.2	47.55	**39.27**	47.49	90.55
	BaseASB	65.77	95.45	123.72	88.51	171.24
	BaseASCB	**29.68**	**40.62**	41.83	**40.56**	**63.32**
10-10-10	BaseAS	0.4	0.54	0.89	0.51	0.85
	BaseASC	0.4	**0.51**	0.76	0.46	0.79
	BaseASB	0.38	0.72	0.71	0.49	0.74
	BaseASCB	**0.35**	0.7	**0.67**	**0.41**	**0.67**
10-15-15	BaseAS	31.8	33.54	60.27	43.44	43.26
	BaseASC	20.88	18.43	16.81	15.44	20.77
	BaseASB	15.17	30.37	42.97	24.04	43.72
	BaseASCB	**8.42**	**16.07**	**15.29**	**10.81**	**19.26**

Cuadro 6.32: Instancias 2-3-4 - 24 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	3.24	4.28	3.73	6.17	8.08
	BaseASC	2.22	3.64	3.54	3.09	5.53
	BaseASB	3.17	4.58	4.18	3.23	6.28
	BaseASCB	**2.06**	**3.63**	**3.39**	**1.72**	**4.11**
5-20-20	BaseAS	231.36	225.87	356.39	318.05	314.08
	BaseASC	130.55	106.3	101.11	95.59	173.95
	BaseASB	114.86	195.93	185.99	160.58	338.53
	BaseASCB	**57.93**	**86.07**	**85.89**	**79.49**	**163.49**
10-10-10	BaseAS	1	1.38	0.92	0.87	2.34
	BaseASC	0.85	1.44	0.88	1.2	1.84
	BaseASB	0.73	1.22	0.8	**0.86**	1.57
	BaseASCB	**0.6**	**1.09**	**0.69**	0.89	**1.52**
10-15-15	BaseAS	94.28	49.85	77.19	71.2	130.43
	BaseASC	32.22	34.12	34.34	17.44	33.3
	BaseASB	50.1	66.18	66.66	36.16	66.53
	BaseASCB	**21.84**	**30.35**	**31.99**	**16.7**	**31.94**

Cuadro 6.33: Instancias 2-3-4 - 26 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	3.87	3.73	6.21	3.68	11.48
	BaseASC	3.09	**3.67**	4.98	4.93	9.38
	BaseASB	3.78	4.6	5.34	3.75	8.57
	BaseASCB	**2.73**	3.72	**4.47**	**3.39**	**5.95**
5-20-20	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
10-10-10	BaseAS	34.75	**2.45**	5.85	559.69	9.59
	BaseASC	26.4	3.77	15.27	178.15	7.06
	BaseASB	**20.37**	45.16	**3.92**	255.96	6.96
	BaseASCB	27.11	60.84	6.34	**110.06**	**6.41**
10-15-15	BaseAS	3600	118.72	177.43	3600	3600
	BaseASC	3600	111.18	155.08	3600	3600
	BaseASB	3600	118.27	146.98	3600	3600
	BaseASCB	**3227.57**	**102.75**	**96.04**	3600	3600

Cuadro 6.34: Instancias 2-3-4 - 28 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	19.65	18.67	12.51	12.03	**8.35**
	BaseASC	42.35	**12.89**	34.2	10.37	35.7
	BaseASB	**11.97**	16.02	**20.87**	8.95	8.64
	BaseASCB	24.72	18.48	28.29	**8.77**	12.47
5-20-20	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
10-10-10	BaseAS	798.49	220.42	644.86	473.54	366.58
	BaseASC	118.27	**103.06**	93.74	190.9	129.09
	BaseASB	219.68	464.19	322.97	236.96	187.05
	BaseASCB	**75.26**	185.33	**87.85**	**88.76**	**113.94**
10-15-15	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600

En las tablas presentadas se observan varias conclusiones que concuerdan con las obtenidas anteriormente. Dentro de los puntos más simples, vemos cómo influye la cantidad de ítems considerados. En el extremo más bajo tenemos el grupo de instancias con 22 ítems en donde todas se encuentran resueltas en muy poco tiempo. En el otro extremo, tenemos las instancias de 32 ítems en donde casi ninguna instancia se encuentra resuelta hasta la optimalidad.

Nuevamente podemos ver un comportamiento parecido a las instancias 3-regulares cuando observamos cada uno de los posibles algoritmos. En la mayoría de las instancias, la versión *BaseAS* es la que tiene un peor desempeño, y la versión *BaseASCB* es la que presenta los mejores resultados. Nuevamente podemos observar que la ganancia de performance parece provenir de la interacción entre el *branching* y los cortes, ya que las versiones que consideran cada aspecto por separado nuevamente muestran comportamientos que suelen estar mejor posicionados que los de *BaseAS*, pero peor que los de *BaseASCB*, aunque en estas instancias también podemos encontrar casos en

Cuadro 6.35: Instancias 2-3-4 - 30 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	3600	**100.9**	3600	196.75	439.6
	BaseASC	**86.15**	492.16	1543.96	**44.24**	1881.52
	BaseASB	2562.53	1539.96	1802.79	155.66	906.21
	BaseASCB	1440.02	1196.5	**1038.13**	109.5	**867.39**
5-20-20	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
10-10-10	BaseAS	727.36	130.58	356.07	520.99	516.49
	BaseASC	242.86	**107.98**	**106.65**	147.72	**213**
	BaseASB	320.54	440.97	473.15	260.86	232.54
	BaseASCB	**175.27**	215.9	213.5	**130.76**	225.6
10-15-15	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600

Cuadro 6.36: Instancias 2-3-4 - 32 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	**2816.63**	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	2821.49	3600
5-20-20	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
10-10-10	BaseAS	**3168.26**	3600	1292.28	1123.39	3600
	BaseASC	3600	3600	**1199.18**	**513.35**	3600
	BaseASB	3600	3600	3600	1017.43	3600
	BaseASCB	3600	3600	3600	2055.74	3600
10-15-15	BaseAS	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600

donde *BaseASC* tiene un muy buen desempeño. Cabe destacar que, a diferencia de las instancias 3-regulares, la dominancia de *BaseASCB* por sobre *BaseAS* no es absoluta. De hecho, en las instancias de 32 ítems, hay una instancia resuelta por *BaseAS* que no fue resuelta por ningún otro enfoque. Este resultado resulta contradictorio con las conclusiones anteriores ya que, siendo las instancias más dificultosas, se podría esperar una mayor diferencia entre los dos enfoques y no solo no sucede eso sino que en algunas instancias se revierte el desempeño.

Analizando las pocas instancias en donde gana *BaseAS* de forma particular, se puede observar siempre el mismo comportamiento. Los casos en donde *BaseAS* tiene un mejor desempeño que el resto de los enfoques, son casos en donde esta versión logró encontrar una mejor cota primal lo cual le ayudó a podar una gran cantidad de ramas del árbol de búsqueda, cerrando así instancias que no fueron resueltas por los otros enfoques. No tenemos una explicación certera de por qué este enfoque fue capaz de encontrar estas cotas primales pero muy posiblemente esté ligado a un

hecho puramente implementativo, en el cual el *solver* puede valerse de mayor información cuando el modelo se encuentra sin ningún aditivo no estandar, tales como los cortes o el *branching* particular, para generar mejores soluciones primales.

Este es un comportamiento que no se visualizaba en las instancias 3-regulares ya que las heurísticas utilizadas en ese caso conseguían cotas primales muy buenas, siendo muchas veces el valor óptimo. Las restricciones de a pares de ítems, como ya hemos mencionado, son mucho más restrictivas y hacen más complicada la generación de soluciones factibles de calidad.

Dejando de lado las pocas instancias en donde un enfoque puede dar un salto de calidad al encontrar una solución primal que reduzca fuertemente el árbol de búsqueda, podemos seguir viendo que en el resto de las instancias el comportamiento es similar al observado en las instancias 3-regulares. A continuación se pueden ver las cotas duales obtenidas por los diferentes enfoques en todas las instancias de 30 y 32 ítems, ya que son los grupos que presentan una mayor cantidad de instancias no resueltas.

Cuadro 6.37: Cotas Duales - Instancias 2-3-4 - 30 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	5	5	5	5	5
	BaseASC	5	5	5	5	5
	BaseASB	5	5	5	5	5
	BaseASCB	5	5	5	5	5
5-20-20	BaseAS	4.3	4.73	4.33	5	4.38
	BaseASC	4.75	**5**	**5**	5	**5**
	BaseASB	**5**	**5**	**5**	5	**5**
	BaseASCB	**5**	**5**	**5**	5	**5**
10-10-10	BaseAS	6	6	6	6	6
	BaseASC	6	6	6	6	6
	BaseASB	6	6	6	6	6
	BaseASCB	6	6	6	6	6
10-15-15	BaseAS	5	5.26	5	5	5
	BaseASC	5.16	**6**	**5.25**	5.12	5
	BaseASB	5.18	5	5	5	5
	BaseASCB	**5.57**	**6**	**5.25**	**5.33**	**5.46**

En las tablas de cotas duales se pueden ver las mismas tendencias que observamos para las instancias 3-regulares y que ayudaron a definir las mejores componentes para lograr un algoritmo general. Exceptuando unas pocas instancias en donde las tendencias se ven revertidas gracias a que un enfoque encuentra una mejor cota primal por sobre el resto, en todas las demás instancias se puede ver que el algoritmo *BaseASCB* es el que siempre tiene un mejor desempeño. En general, el algoritmo *BaseAS* es el que tiene una menor calidad de cotas duales, exceptuando las instancias excluídas. Entre *BaseASB* y *BaseASC* nuevamente no hay un claro ganador, pero sí es claro que ambos están por debajo de *BaseASCB*.

Todo lo experimentado y analizado con las instancias 2-3-4 refuerzan las conclusiones observadas en las instancias 3-regulares. Es alentador ver cómo las conclusiones que se fueron observando en las anteriores instancias parecen presentar un nivel de generalización suficiente como para al menos mostrar un comportamiento similar en estas instancias nuevas.

6.5. Experimentación Bin Packing

Como hemos mencionado en ocasiones anteriores, las restricciones impuestas por el tamaño de los contenedores pueden ser emuladas por restricciones del hipergrafo conflicto, aunque esta

Cuadro 6.38: Cotas Duales - Instancias 2-3-4 - 32 ítems

%	Algoritmo	0	1	2	3	4
5-10-10	BaseAS	4.29	4.5	5	5	4.51
	BaseASC	**5**	**5**	5	**6**	**5**
	BaseASB	**5**	**5**	5	5	**5**
	BaseASCB	**5**	**5**	5	**6**	**5**
5-20-20	BaseAS	4.27	4.25	4.6	4.19	4.36
	BaseASC	4.28	**5**	4.38	**5**	4.53
	BaseASB	**5**	**5**	**5**	**5**	**5**
	BaseASCB	**5**	**5**	**5**	**5**	**5**
10-10-10	BaseAS	**6**	5	**6**	6	5
	BaseASC	5	5	**6**	6	5
	BaseASB	5	5	5	6	5
	BaseASCB	5	5	5	6	**5.14**
10-15-15	BaseAS	5	4.53	5	5	5
	BaseASC	5	**5**	5	5	5
	BaseASB	5	**5**	5	5	5
	BaseASCB	**5.25**	**5**	**5.12**	**5.17**	**5.5**

traducción no tiene porque resultar inocua en la práctica. Es por esto que en esta sección nos abocaremos a experimentar con instancias en donde haya restricciones activas desde los dos aspectos, tanto desde el tamaño de los ítems y los contenedores, como desde los conflictos.

En el estudio poliedral realizado en el capítulo anterior, la mayoría de las desigualdades propuestas tienen relación con el hipergrafo de conflictos, ya sea utilizando las restricciones de conflictos originales o las de capacidad de contenedor traducidas. Sin embargo, en la literatura se pueden encontrar familias de desigualdades válidas que están asociadas directamente al Problema de la Mochila que provienen de intentar asignar la mayor cantidad de elementos a un contenedor de capacidad fija. Si bien existen diferentes tipos de desigualdades, las que usualmente están asociadas a un mayor éxito en dicho problema son las desigualdades que involucran *covers*. Los *covers* son conjuntos de elementos para los que se sabe que no pueden ser asignados juntos ya que la suma de sus tamaños excede la capacidad de los contenedores a utilizar. En [36], Kaparis y Letchford realizan una revisión de diferentes cortes sobre el Problema de la Mochila, mostrando el desempeño en la práctica.

Además de presentar desigualdades relacionadas con *covers*, en [36] se presentan las desigualdades *weight inequalities* y *lifted pack inequalities*. Dichas desigualdades se muestran débiles tanto desde el punto de vista teórico como desde los resultados prácticos por lo que no haremos utilización de ellas para nuestro problema. Sin embargo, las desigualdades relacionadas con *covers* sí presentan un buen comportamiento para el Problema de la Mochila, por lo que haremos inclusión de ellas para nuestro análisis.

En primer lugar, se puede considerar las desigualdades *cover* en su estado inicial. Como indicamos antes, estas desigualdades simplemente indican que un conjunto de ítems que supera en la suma de sus tamaños a la capacidad de un contenedor, no pueden ser asignados todos juntos. De esta manera, si tenemos un conjunto de ítems C con la propiedad *cover*, la siguiente desigualdad es válida para cualquier contenedor particular $\bar{k}$.

$$\sum_{i \in C} x_{i\bar{k}} \leq (|C| - 1) y_{\bar{k}} \tag{6.4}$$

Es decir que si no se utiliza el contenedor, entonces ningún elemento del *cover* puede ser asignado a él, y si se utiliza el contenedor, entonces no pueden estar todos los elementos del *cover* asignados juntos en él. Si bien esta desigualdad es válida para cualquier conjunto que cumpla la

condición de *cover*, se busca encontrar las desigualdades más fuertes que se consiguen cuando el *cover* es minimal. Es decir, cuando al quitar cualquier elemento del conjunto sucede que sí se podría asignar todo el resto de ítems juntos.

Dado un *cover* minimal C, existen diversas formas de reforzar la desigualdad presentada utilizando los ítems que están por fuera del *cover*. Obteniendo una desigualdad genérica de la siguiente forma:

$$\sum_{i \in C} x_{i\bar{k}} + \sum_{i \notin C} \alpha_{i\bar{k}} x_{i\bar{k}} \leq (|C| - 1) y_{\bar{k}} \tag{6.5}$$

Hay diversas maneras de conseguir los coeficientes de refuerzo $\alpha_{i\bar{k}}$ lo cual lleva a diferentes desigualdades. Un procedimiento fácil de realizar y, que según lo experimentado en [36], muestra muy buenos resultados, es que el deriva en las desigualdades denominadas *extended cover*. Para generar el conjunto denominado *extended cover*, sumaremos al *cover* C a todo ítem que tenga un tamaño mayor o igual al ítem más grande en C. Formalmente, siendo $w^* = max_{i \in C} w_i$, definimos el *extended cover* $E(C)$ como $C \cup \{i | w_i \geq w^*\}$. Dado este nuevo conjunto de ítems, la siguiente es la desigualdad *extended cover* asociada:

$$\sum_{i \in E(C)} x_{i\bar{k}} \leq (|C| - 1) y_{\bar{k}} \tag{6.6}$$

Si bien existen otras maneras de reforzar la desigualdad original, la presentada mostró buenos resultados en [36], con el agregado de que en el mismo trabajo presentan un algoritmo de separación heurístico para la familia que tiene casi tan buenos resultados como los algoritmos de separación exactos, con un costo computacional mucho más reducido.

De esta manera, a todos los componentes desarrollados y analizados en las secciones anteriores, le sumaremos la familia de desigualdades propuesta para realizar la experimentación pertinente a la instancias que presenten restricciones tanto de empaquetamiento como de conflictos.

Una vez determinado el componente particular que se le sumará a la experimentación de este tipo de instancias debemos, justamente, generar instancias pertinentes. Al igual que nos sucedió en la sección de experimentación 2-3-4, debemos ser cuidadosos en la generación de instancias para que la interacción entre los dos tipos de restricciones tenga sentido y que no suceda que un tipo subsuma completamente al otro. En el capítulo 3, hemos hecho alusión a las instancias del tipo *da* presentadas en [52]. Las mismas presentaban conflictos solo de a pares de ítems, mientras que en promedio entraban 8 ítems por contenedor. De esta manera, lo que sucede es que por un lado tenemos un conjunto de restricciones muy fuertes, que son los conflictos de a pares, y por el otro tenemos *covers* de en promedio nueve elementos que imponen restricciones mucho más débiles. A tal punto es tan distinto el nivel de restricción impuesto, que al observar los óptimos conseguidos en dichas instancias, la cantidad de ítems asignados a cada contenedor está por debajo del tamaño promedio de los *covers*. Esto indica que posiblemente sean un número muy bajo de *covers* los que están imponiendo una restricción activa.

Por lo dicho recientemente, generaremos instancias donde el orden de las restricciones sean comparables. Por el lado de los conflictos, tomaremos nuevamente un hipergrafo 3-regular. Es decir, los conflictos presentes en el hipergrafo serán todos de hiperaristas de cardinal 3. Por el lado del empaquetamiento, generaremos los tamaños de los ítems de tal manera que, en promedio, entren justo 3 ítems en cada contenedor. Esto debería devenir en restricciones *cover* de tamaño 4. A diferencia de las instancias utilizadas en la sección de heurísticas, el parámetro de tamaño de ítems recientemente descripto lo realizaremos solamente sobre la mitad de los ítems de la instancia. Esto lo hacemos así ya que, si pusiéramos dicho tamaño para todos los ítems de la instancia, entonces sería esperable que se generen todas las restricciones *cover* de 4 ítems, lo cual es demasiado abrupto en dos sentidos. En primer lugar serían demasiadas restricciones de 4 ítems para cuando se combine, por ejemplo, con un hipergrafo de conflictos de densidad 10 %. Además, el hecho de que todos los ítems tengan un tamaño tan considerable, hace que sea muy fácil calcular cotas duales basadas estrictamente en empaquetamiento que hagan que las instancias generadas resulten triviales.

De esta manera, generamos instancias con 22, 24, 26, 28, 30 y 32 ítems. El hipergrafo de conflictos tendrá una densidad que variará entre 10 %, 25 %, 50 %, 75 % y 90 %, con sus hiperaristas generadas de manera aleatoria. Los contenedores tendrán una capacidad fija de 5040. Mientras que los ítems, tendrán tamaños asociados de dos maneras diferentes. La mitad de los ítems tendrá un tamaño generado aleatoriamente de manera uniforme en el intervalo [1380, 1980]. La otra mitad de los ítems tendrá un tamaño fijo de 252. Con esto se busca que la primera mitad de ítems genere un número considerable de *covers* de cardinal 4, mientras que la segunda mitad de los ítems solamente impondría una restricción en el caso de querer asignar demasiados juntos.

Al igual que en la experimentación 2-3-4, por cada combinación de parámetros, se generaron 5 instancias diferentes para obtener resultados más robustos.

A continuación se muestran los resultados obtenidos para la experimentación con este tipo de instancias. Al igual que en los casos anteriores, lo primero que reportaremos es el tiempo de corrida en segundos utilizado por cada enfoque diferente sobre cada una de las instancias, teniendo como límite una hora de corrida. Se contemplan 6 versiones del algoritmo. *BaseAS*, *BaseASC*, *BaseASB* y *BaseASCB* con las mismas especificaciones utilizadas en las experimentaciones anteriores. A dichos cuatro enfoques, le sumamos en este caso *BaseASCo*, que corresponde a la versión *BaseAS* más la utilización de las desigualdades *extended cover* descriptas y *BaseASCBCo*, que corresponde análogamente a agregar dichos cortes a la versión *BaseASCB*. Cabe destacar que para realizar estos experimentos, se han deshabilitado los cortes *cover* que ofrece el *solver* por defecto, con el fín de poder visualizar de manera controlada el impacto que genera la inclusión de esta familia.

En las tablas presentadas podemos extraer conclusiones en dos grandes direcciones. Por un lado, podemos ver cómo los componentes analizados para los tipos de instancias 3-regulares y 2-3-4 nuevamente se adaptan satisfactoriamente al observar los tiempos de ejecución. En este nuevo tipo de instancias, otra vez se observa que la versión que no utiliza ni cortes poliedrales ni *branching* presenta tiempos mucho más elevados que las variantes donde sí se utilizan las diferentes mejoras. Si comparamos, por ejemplo, contra la versión *BaseASCB*, encontramos instancias, tal como la número 0 de densidad 90 % y 22 ítems, en donde *BaseAS* consume todo el tiempo disponible sin poder resolver la instancia, mientras que *BaseASCB* llega a demostrar la optimalidad del resultado en casi un sexto del tiempo límite proporcionado. En mayor o menor medida, en todos los casos podemos ver que los enfoques en los que se utilizan cortes o *branching* son ampliamente superiores a la versión *BaseAS*. En esta misma dirección, podemos nuevamente ratificar los resultados intermedios que se obtienen al incluir o solamente los cortes, o solamente el *branching*. Tanto *BaseASC* como *BaseASB* vuelven a mostrarse muy superiores en desempeño con respecto a *BaseAS*, pero con no tan buenos resultados como los obtenidos por *BaseASCB*. Esto ratifica la idea de que la mejora en la performance proviene de una buena interacción entre los componentes por sobre la acción de cada componente por separado. Resulta interesante ver cómo las conclusiones obtenidas en instancias 3-regulares que presentan doble particularidad, en cuanto a que no tienen en cuenta restricciones de empaquetamiento explicitamente y que tiene un cardinal de hiperarista fijo, son luego generalizadas a los tipos de instancia 2-3-4 y las instancias donde aparte de conflictos hay también restricciones por los tamaños de los ítems. Más allá de que todavía quedan sin analizar infinidad de combinaciones posibles para la creación de instancias, los resultados presentados muestran un nivel de robustez y generalización muy satisfactorio de las diferentes componentes propuestas.

En otro eje de discusión, podemos analizar lo que sucedió con los cortes *cover*. Dichos cortes fueron utilizados muchas veces durante la ejecución de los algoritmos, ya que durante las corridas pudimos observar que el *solver* hacía uso de esta familia de desigualdades. Sin embargo, esta utilización de los cortes no se tradujo en una mejora notable en los tiempos de cómputo para resolver las instancias. Si observamos los resultados obtenidos por los enfoques con cortes *cover*, *BaseASCo* y *BaseASCBCo*, y los comparamos con la versiones análogas sin los cortes, podemos notar que las diferencias son muy pequeñas. No existen grandes variaciones en los tiempos de corrida de manera general, más que en alguna que otra instancia en particular. En general los tiempos reportados son bastante parecidos y en ninguna de las comparaciones hay un enfoque

Cuadro 6.39: Instancias Bin Packing - 22 ítems

%	Algoritmo	0	1	2	3	4
10	BaseAS	**0.01**	0.02	**0.01**	0.02	**0.01**
	BaseASCo	**0.01**	0.02	0.02	0.02	**0.01**
	BaseASC	**0.01**	0.02	**0.01**	0.02	**0.01**
	BaseASB	0.02	0.02	0.02	0.02	0.02
	BaseASCB	0.02	0.02	**0.01**	0.02	0.02
	BaseASCBCo	**0.01**	0.02	0.02	0.02	**0.01**
25	BaseAS	**0.03**	0.03	**0.03**	**0.03**	0.03
	BaseASCo	0.04	0.03	**0.03**	**0.03**	0.03
	BaseASC	0.04	0.03	0.04	0.04	0.03
	BaseASB	**0.03**	0.03	**0.03**	**0.03**	0.03
	BaseASCB	0.04	0.03	**0.03**	**0.03**	0.03
	BaseASCBCo	0.04	0.03	**0.03**	**0.03**	0.03
50	BaseAS	0.07	18.19	**0.07**	6.1	0.06
	BaseASCo	0.07	10.95	**0.07**	10.84	0.06
	BaseASC	0.07	5.63	0.08	6	0.06
	BaseASB	0.07	3.72	0.08	3.48	0.06
	BaseASCB	0.07	3.08	0.08	3.39	0.06
	BaseASCBCo	0.07	**3.03**	**0.07**	**2.77**	0.06
75	BaseAS	6.71	5.14	5.81	4.3	395.36
	BaseASCo	6.51	8.65	11.47	6.81	450.21
	BaseASC	4.34	2.89	3.39	3.14	229.35
	BaseASB	3.26	2.73	3.31	2.89	162.62
	BaseASCB	2.45	**1.96**	**2.29**	2.01	**61.72**
	BaseASCBCo	**2.28**	2.38	3.29	**1.89**	62.87
90	BaseAS	3600	67.16	105.21	2589.6	117.57
	BaseASCo	3600	78.32	117.19	2972.37	122.96
	BaseASC	2642.25	15.53	30.85	1490.65	48.5
	BaseASB	2557.45	37.5	58.24	1477.38	74.74
	BaseASCB	**645.53**	10.66	7.73	613.97	**7.12**
	BaseASCBCo	664.46	**10.55**	**6.48**	**532.65**	8.31

claramente ganador. Posiblemente las variaciones que hace que a veces un enfoque gane por una pequeña fracción de tiempo y a veces pierda, se den por razones relacionadas a cuestiones casi azarosas de cómo cada enfoque término recorriendo un árbol de búsqueda que, posiblemente, en esencia es muy parecido.

Para reforzar el análisis presentado mostraremos las cotas duales obtenidas por cada enfoque en las instancias más difíciles, las que no fueron resueltas en el tiempo límite. De esta manera, buscamos obtener un segundo eje importante para analizar el desempeño de cada uno de los algoritmos.

Con los resultados mostrados en las tablas correspondientes a las cotas duales podemos ver que el análisis realizado se refuerza. En un primer escalafón tenemos a las versiones *BaseAS* y *BaseASCo* que son las que obtienen las cotas duales de peor calidad. En casi todas las instancias se encuentran bastante por debajo del resto de las variantes y entre ellas no se observan diferencias. Si bien la versión *BaseASCo* utiliza una cantidad alta de cortes *cover* durante su ejecución, es posible que la cantidad de puntos alternativos con la misma cota dual sea tal que los cortes de dicha familia no sean suficientes para lograr producir una cota dual más elevada.

En un segundo orden tenemos a las versiones *BaseASB* y *BaseASC*. Ambos enfoques logran llegar a cotas duales de mayor calidad con respecto a los enfoques sin cortes ni *branching*, pero en casi todas las instancias se los puede ver por debajo de los enfoques que combinan ambas compo-

Cuadro 6.40: Instancias Bin Packing - 24 ítems

%	Algoritmo	0	1	2	3	4
10	BaseAS	0.02	0.02	0.02	0.02	0.02
	BaseASCo	0.02	0.02	0.02	0.02	0.02
	BaseASC	0.02	0.02	0.02	0.02	0.02
	BaseASB	0.02	0.02	0.02	0.02	0.02
	BaseASCB	0.02	0.02	0.02	0.02	0.02
	BaseASCBCo	0.02	0.02	0.02	0.02	0.02
25	BaseAS	**0.04**	0.04	**0.04**	0.05	0.05
	BaseASCo	0.05	0.04	**0.04**	**0.04**	**0.04**
	BaseASC	**0.04**	0.04	**0.04**	**0.04**	**0.04**
	BaseASB	0.05	0.04	**0.04**	**0.04**	**0.04**
	BaseASCB	**0.04**	0.04	0.05	0.05	**0.04**
	BaseASCBCo	**0.04**	0.04	**0.04**	**0.04**	**0.04**
50	BaseAS	31.7	21.67	79.62	20.17	23.3
	BaseASCo	69.24	38.16	114.3	15.84	33.63
	BaseASC	10.34	7.45	9.78	4.91	12.16
	BaseASB	7.9	4.88	6.52	6.61	9.4
	BaseASCB	5.92	**3.68**	4.03	**3.56**	**5.03**
	BaseASCBCo	**5.88**	4.01	**3.84**	4.19	5.08
75	BaseAS	1206.14	444.52	499.5	363.23	461.47
	BaseASCo	868.45	264.43	402.45	336.2	619.51
	BaseASC	283.28	412.55	305.98	493.87	315.2
	BaseASB	228.51	95.62	112.79	93.64	118.18
	BaseASCB	64.59	46.84	56.49	50.49	60.81
	BaseASCBCo	**61.56**	**45.18**	**54.44**	**50.38**	**57.04**
90	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	1886.53	1245.45	1518.1	1913.16	2340.23
	BaseASB	3600	2142.34	2096.54	2186.33	2582.98
	BaseASCB	378.76	**227.9**	**322.87**	**276.93**	198.16
	BaseASCBCo	**338.82**	255.8	358.96	315.81	**186.24**

nentes. En general, si bien en los tiempos de ejecución pudimos ver comportamientos parecidos de estas dos versiones, al ver las cotas duales podemos notar algunas diferencias. La versión *BaseASC* logra mejores cotas duales y esto sucede porque al estar reportando las instancias más difíciles que no pudieron ser resueltas hasta la optimalidad, todas las instancias mostradas presentan un alto porcentaje de conflictos, lo cual es un ambiente ideal para la utilización de los cortes clique.

Por último, podemos ver que las mejores cotas duales se obtienen nuevamente con los enfoques que combinan cortes y *branching*. Entre *BaseASCB* y *BaseASCBCo* no se observan diferencias muy notorias. Nuevamente, es posible que las diferencias observadas se encuentren más ligadas a eventuales recorridos del árbol de búsqueda que a una relación de dominancia de un enfoque sobre el otro.

Por los resultados observados, concluímos en que en las diversas instancias se puede observar satisfactoriamente cómo los componentes desarrollados para el algoritmo branch-and-cut funcionaron correctamente, mejorando el desempeño sobre las instancias propuestas. En algunos casos bajando considerablemente el tiempo necesario para resolver las instancias hasta la optimalidad, y en otros casos logrando resolver instancias que por los enfoques más directos no habían podido ser resueltas en el tiempo límite planteado.

Cuadro 6.41: Instancias Bin Packing - 26 ítems

%	Algoritmo	0	1	2	3	4
10	BaseAS	0.03	0.03	0.03	0.03	0.03
	BaseASCo	0.03	0.04	0.03	0.03	0.03
	BaseASC	0.03	0.03	0.03	0.03	0.03
	BaseASB	0.03	0.03	0.03	0.03	0.03
	BaseASCB	0.03	0.03	0.03	0.03	0.03
	BaseASCBCo	0.03	0.03	0.03	0.03	0.03
25	BaseAS	80.3	16.95	72.17	183.82	34.2
	BaseASCo	142.79	27.38	85.56	280.54	53.39
	BaseASC	37.74	10.97	18.45	29.82	15.15
	BaseASB	36.54	15.26	23.23	28.44	26.7
	BaseASCB	**15.6**	**6.32**	**13.22**	**14.59**	**9.22**
	BaseASCBCo	19.56	10.43	18.5	18.29	16.7
50	BaseAS	19.21	2.89	14.97	18.71	10.1
	BaseASCo	10.74	3.9	11.86	17.91	17.26
	BaseASC	8.03	2.93	4.42	11.63	6.13
	BaseASB	6.09	4.05	5.91	9.83	6.4
	BaseASCB	**4.21**	**2.78**	**3.81**	**5.86**	**3.64**
	BaseASCBCo	4.48	4.04	6.14	6.2	3.77
75	BaseAS	1541.68	592.39	1602.33	2206.81	3600
	BaseASCo	2364.51	1020.8	1883.83	1972.92	3467.06
	BaseASC	3589.8	691.64	929.33	394.34	**1586.94**
	BaseASB	394.02	139.32	197.95	234.5	3600
	BaseASCB	**207.73**	**67.89**	75.43	**73.54**	2281.4
	BaseASCBCo	381.2	70.54	**73.23**	78.79	2077.1
90	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
	BaseASCBCo	3600	3600	3600	3600	3600

Cuadro 6.42: Instancias Bin Packing - 28 ítems

%	Algoritmo	0	1	2	3	4
10	BaseAS	**0.05**	0.05	0.05	**0.04**	0.04
	BaseASCo	1.17	0.05	**0.04**	**0.04**	0.04
	BaseASC	**0.05**	0.05	**0.04**	**0.04**	0.04
	BaseASB	0.13	0.05	**0.04**	**0.04**	0.04
	BaseASCB	**0.05**	0.05	0.05	0.05	0.04
	BaseASCBCo	**0.05**	0.05	0.05	**0.04**	0.04
25	BaseAS	0.1	**0.08**	0.1	0.09	0.08
	BaseASCo	**0.09**	**0.08**	0.1	0.09	0.08
	BaseASC	0.1	0.09	0.1	0.09	0.08
	BaseASB	0.1	**0.08**	**0.08**	**0.08**	0.08
	BaseASCB	**0.09**	0.12	0.1	**0.08**	0.08
	BaseASCBCo	0.1	**0.08**	0.1	0.09	0.08
50	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	415.86	580.89	719.26	569.99	644.78
	BaseASB	893.7	747.69	1052.41	971.09	1128.9
	BaseASCB	**228.07**	**186.35**	**247.92**	**244.18**	**239.2**
	BaseASCBCo	264.73	223.29	296.25	261.97	264.5
75	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	**3523.93**	3600	3600	3600
	BaseASCBCo	3600	3354.22	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
	BaseASCBCo	3600	3600	3600	3600	3600

Cuadro 6.43: Instancias Bin Packing - 30 ítems

%	Algoritmo	0	1	2	3	4
10	BaseAS	0.17	**0.05**	0.06	0.06	0.06
	BaseASCo	0.12	0.07	**0.05**	0.06	0.06
	BaseASC	**0.05**	**0.05**	**0.05**	0.06	0.06
	BaseASB	0.15	0.07	**0.05**	0.06	0.06
	BaseASCB	**0.05**	**0.05**	**0.05**	0.06	0.06
	BaseASCBCo	**0.05**	**0.05**	**0.05**	0.06	0.06
25	BaseAS	**0.11**	0.11	**0.1**	0.11	0.11
	BaseASCo	0.12	0.11	**0.1**	0.11	0.11
	BaseASC	0.12	0.11	0.11	0.11	0.11
	BaseASB	**0.11**	0.11	0.11	0.11	0.11
	BaseASCB	0.12	0.11	0.11	0.11	0.11
	BaseASCBCo	0.12	0.11	**0.1**	0.11	0.11
50	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	798.95	1203.07	1499.9	1197.87	874.22
	BaseASB	1585.9	1322.53	1613.53	1089.16	1165.08
	BaseASCB	**381.69**	**335.24**	**435.62**	**320.9**	**294.31**
	BaseASCBCo	410.69	355.77	501.53	369.86	328.39
75	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
	BaseASCBCo	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
	BaseASCBCo	3600	3600	3600	3600	3600

Cuadro 6.44: Instancias Bin Packing - 32 ítems

%	Algoritmo	0	1	2	3	4
10	BaseAS	0.23	0.08	0.06	0.12	**0.07**
	BaseASCo	0.15	0.08	0.06	0.12	0.08
	BaseASC	**0.08**	0.08	0.06	0.12	0.08
	BaseASB	**0.08**	0.08	0.06	0.12	**0.07**
	BaseASCB	**0.08**	**0.07**	0.06	0.12	0.08
	BaseASCBCo	**0.08**	**0.07**	**0.05**	0.12	0.08
25	BaseAS	0.17	0.23	3600	**0.17**	0.18
	BaseASCo	**0.16**	0.23	3600	0.18	0.18
	BaseASC	0.17	0.23	2468.33	0.18	0.18
	BaseASB	0.17	**0.21**	3600	0.18	**0.17**
	BaseASCB	0.17	0.22	**1192.52**	0.18	0.18
	BaseASCBCo	**0.16**	0.24	2238.84	0.18	0.18
50	BaseAS	3600	0.57	3600	0.55	3600
	BaseASCo	3600	0.63	3600	0.55	1301.98
	BaseASC	3600	0.64	3600	0.49	**54.87**
	BaseASB	3600	0.63	3600	0.47	60.98
	BaseASCB	**531.7**	**0.56**	3600	0.48	130
	BaseASCBCo	3600	0.74	**1029**	**0.44**	388.71
75	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
	BaseASCBCo	3600	3600	3600	3600	3600
90	BaseAS	3600	3600	3600	3600	3600
	BaseASCo	3600	3600	3600	3600	3600
	BaseASC	3600	3600	3600	3600	3600
	BaseASB	3600	3600	3600	3600	3600
	BaseASCB	3600	3600	3600	3600	3600
	BaseASCBCo	3600	3600	3600	3600	3600

Cuadro 6.45: Cotas duales - 26 ítems

%	Algoritmo	0	1	2	3	4
90	BaseAS	6	6	6	6	6
	BaseASCo	6	6	6	6	6
	BaseASC	7	7	7	7	7
	BaseASB	6	7	6.8	7	6.1
	BaseASCB	**7.3**	**7.3**	**7.4**	**7.3**	**7.3**
	BaseASCBCo	**7.3**	**7.3**	7.3	**7.3**	**7.3**

Cuadro 6.46: Cotas duales - 28 ítems

%	Algoritmo	0	1	2	3	4
75	BaseAS	6	6	6	6	6
	BaseASCo	6	6	6	6	6
	BaseASC	6.4	6.4	6.2	6.1	6.4
	BaseASB	6	6.3	6	6	6
	BaseASCB	**7**	**8**	**7**	**7**	**7**
	BaseASCBCo	**7**	**8**	**7**	**7**	**7**
90	BaseAS	6	6	6	6	6
	BaseASCo	6	6	6	6	6
	BaseASC	7	7	7	7	7
	BaseASB	6.1	6.2	6.2	6	6
	BaseASCB	**7.3**	**7.4**	**7.4**	**7.3**	**7.4**
	BaseASCBCo	**7.3**	**7.4**	**7.4**	**7.3**	7.3

Cuadro 6.47: Cotas duales - 30 ítems

%	Algoritmo	0	1	2	3	4
75	BaseAS	6	6	6	6	6
	BaseASCo	6	6	6	6	6
	BaseASC	6	6.1	6.1	6.2	6
	BaseASB	6	6	6	6	6
	BaseASCB	**6.7**	**6.8**	**7**	**7**	**7**
	BaseASCBCo	6.6	6.7	**7**	**7**	6.8
90	BaseAS	6	6	6	6	6
	BaseASCo	6	6	6	6	6
	BaseASC	7	7	7	7	7
	BaseASB	6.8	6.2	6	6.1	7
	BaseASCB	**7.3**	7.2	**7.3**	7.3	**7.3**
	BaseASCBCo	7.2	**7.3**	**7.3**	**7.4**	7.2

Cuadro 6.48: Cotas duales - 32 ítems

%	Algoritmo	0	1	2	3	4
75	BaseAS	7	7	6	7	7
	BaseASCo	7	7	6	7	7
	BaseASC	7	7	6.1	7	7
	BaseASB	7	7	7	7	7
	BaseASCB	7	7	**6.8**	7	7
	BaseASCBCo	7	7	**6.8**	7	7
90	BaseAS	7	7	6	7	7
	BaseASCo	7	7	6	7	7
	BaseASC	7	7	**7**	7	7
	BaseASB	7	7	**7**	7	7
	BaseASCB	**7.3**	**7.2**	**7**	7	**7.2**
	BaseASCBCo	**7.3**	**7.2**	**7**	**7.2**	7

Capítulo 7

Conclusiones

7.1. Conclusiones generales

En este trabajo estudiamos el Problema de Empaquetamiento con Conflictos Generalizados haciendo uso de técnicas de Programación Lineal Entera. Nuestro objetivo fue lograr un análisis teórico del problema y de las instancias de aplicación, siempre con el propósito final de desarrollar un algoritmo robusto y eficiente.

En primer lugar, se desarrollaron heurísticas iniciales para el problema con el objetivo de obtener soluciones de calidad, en tiempos reducidos, que sirviesen como un componente importante en un posterior algoritmo exacto. En este segmento, quedó clara la dominancia de las metaheurísticas de búsqueda local presentadas por sobre las heurísticas constructivas. La capacidad del *Simulated Anealling* para explorar una porción considerable del espacio de búsqueda en un tiempo muy acotado resultó muy superadora por sobre las heurísticas que generan un tipo específico de soluciones. Sobre este capítulo nos parece interesante remarcar el rol que tuvo la redefinición del concepto de conflictividad de un ítem. La noción clásica asociada a los conflictos de a pares, no resultaba suficiente para comprender la relación que existe entre los diferentes órdenes de restrictividad que imponenen conjuntos de diferentes tamaños.

En el siguiente capítulo abocamos nuestro estudio al aspecto poliedral. Esto se realizó no sobre el modelo original formulado al presentar el problema, sino sobre un modelo que incluyó nuevas desigualdades con el objetivo de reducir simetrías. En este aspecto se consiguieron buenos resultados, pudiendo caracterizar de forma completa el sistema minimal de ecuaciones del poliedro, y luego derivando variadas familias de desigualdades válidas. Muchas de las familias presentadas definen facetas del poliedro asociado bajo ciertas hipótesis y se presentaron las demostraciones correspondientes. En este capítulo nos pareció interesante cómo la interacción entre dos tipos de restricciones de diferente natureleza (restricción de empaquetamiento y conflictos de conjuntos de ítems), en realidad parecen comportarse como un solo tipo de restricción unificada, que son las restricciones de hiperarista en el hipergrafo de conflictos. Las desigualdades *cover* que predican sobre los tamaños de los ítems, son en realidad análogas a las restricción de conjunto independiente instanciadas particularmente en la hiperarista que definen los ítems del *cover*. De esta manera, no se hizo una distinción particular de la familia de desigualdades *cover* en el estudio poliedral, ya que desde el punto de vista teórico serían análogas a las presentadas, pero sí se las utilizó explícitamente en el aspecto práctico ya que las traducciones mencionadas entre las familias de desigualdades no son inocuas en los costos computacionales de resolución.

Dada la estrecha relación de nuestro problema con el Problema de Coloreo en Hipergrafos, no resulta sorpresivo ver que muchas de las desigualdades tienen parecidos con desigualdades válidas para el Problema de Coloreo en Grafos. Es interesante ver cómo varias de las desigualdades presentadas son generalizaciones de situaciones que en el problema correspondiente a grafos son más particulares por solamente tener conflictos de a pares de ítems. Es así que, por ejemplo, una desigualdad válida como la de conjunto independiente para grafos, debió generalizarse en dos

aspectos para poder ser utilizada en nuestro problema. Por un lado, si bien la definición de conjunto independiente es la misma, ahora esta noción no solamente está teniendo en cuenta las hiperaristas explícitas del hipergrafo, sino también las restricciones provenientes de los tamaños de los ítems. Por otro lado, se debió introducir el concepto de $\bar{r}$, el cardinal del conjunto independiente máximo de la vecindad de un ítem i al ya utilizar i. Este concepto no tiene cabida en el Problema de Coloreo en Grafos ya que, al ser todas aristas de a pares de ítems, la incorporación de un ítem automáticamente anula la utilización de cualquiera de sus vecinos, por lo que $\bar{r}$ siempre es 0.

Al ejemplo dado, se le suman otras desigualdades que hacen uso de las hiperaristas con tamaño mayor a 2, como las que predican sobre la intersección de hiperaristas o sobre ciclos generados de diferentes maneras (*loose cycles* y *tight cycles*). Todas estas son situaciones no explotables en el Problema de Coloreo de Grafos ya que, al ser todas aristas de tamaño 2, no existe la misma variabilidad de situaciones.

Por último, en el siguiente capítulo se combinaron los resultados poliedrales, junto con las heurísticas desarrolladas y otras componentes específicas como preprocesamientos, más cortes y una regla de *branching* particular. Con estos variados aspectos se logró desarrollar un algoritmo exacto que demostró un comportamiento gradualmente superador con la inclusión de los diferentes componentes. En conclusión final, se logró pasar de un modelo básico que no lograba resolver la gran mayoría de las instancias presentadas, a un algoritmo mucho más robusto y eficiente. Robusto ya que mostró adaptarse correctamente a los diferentes tipos de instancias propuestas. Eficiente ya que se logró mover el umbral de los tamaños de instancias resolubles, y se bajó considerablemente el tiempo de cómputo en las instancias que ya eran resolubles anteriormente.

Más allá de los buenos resultados obtenidos a nivel experimental, algo que nos parece interesante remarcar sobre el tratamiento en la práctica del problema es la interiorización realizada sobre las características particulares de las instancias a resolver. A lo largo del trabajo, y en particular en el capítulo 6, se logró un análisis de las instancias para entender cómo en este problema que interaccionan dos tipos de restricciones, se pueden encontrar situaciones en donde ambas restricciones cumplen un rol importante y no sucede que un tipo es subsumido por el otro.

Finalmente, en este trabajo se observa nuevamente de forma clara que al tratarse de problemas computacionalmente difíciles, si bien las herramientas de propósito general tienen un gran desempeño, se necesita de un estudio particular del problema para poder lograr algoritmos más eficientes.

7.2. Trabajo futuro

Nuestro trabajo sobre el Problema de Empaquetamiento con Conflictos Generalizados deja abiertas varias líneas interesantes para una futura investigación:

- Mejoras de cotas primales: en las instancias utilizadas las heurísticas iniciales mostraron buenos resultados, muchas veces alcanzando el valor óptimo de las instancias. Esto hizo que no fuera provechosa la inversión de tiempo en el desarrollo de heurísticas primales para el algoritmo exacto. Posiblemente, al exponer el algoritmo a instancias de características distintas, sea útil avanzar sobre el aspecto de las heurísticas primales para lograr una mayor eficiencia.

- Estudio poliedral: para la realización del estudio poliedral se utilizaron diferentes fuentes. Se desarrollaron desigualdades válidas basadas en la semántica de las variables pertinentes, se utilizaron desigualdades de problemas relacionados y se utilizó el software PORTA [7] para generar descripciones completas del poliedro en instancias de tamaño pequeño. Si bien se logró una buena cantidad de resultados poliedrales, es claro que la descripción de las desigualdades válidas no es completa y sería interesante continuar la investigación generando mayor conocimiento en esta dirección.

- Otras componentes experimentales: a lo largo del desarrollo del algoritmo exacto se obviaron ciertas componentes específicas por diferentes motivos, como por ejemplo por no mostrar

un buen comportamiento en experimentos preliminares. Sin embargo, sería útil repasar qué otras componentes sí podrían tener un impacto positivo sobre el algoritmo presentado. En particular, la regla de *branching* propuesta mostró un buen comportamiento, sobre todo al asociarse con algunos cortes que propiciaban la fijación de variables. En este aspecto, un punto para una experimentación futura sería tener en cuenta un reordenamiento de los ítems del problema, con el fin de interactuar con los cortes y el *branching* propuesto para lograr una mayor fijación de variables.

- Particularizaciones: si bien el objetivo de este trabajo fue desarrollar un algoritmo lo más robusto posible, el problema presentado es de una generalidad importante por lo que una posible línea de investigación futura podría ser la particularización del problema en ciertas situaciones. Por ejemplo, sería interesante observar si es posible desarrollar mejoras específicas para ciertos tipos de hipergrafos particulares.

Bibliografía

[1] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Oper. Res. Lett.*, 33(1):42–54, 2005.

[2] B. Baker and E. Coffman Jr. Mutual exclusion scheduling. *Theoretical Computer Science*, 162:225–243, 1996.

[3] C. Basnet and J. Wilson. Heuristic for determining the number of warehouses for storing non-compatible products. *International Transactions in Operational Research*, 12:527–538, 2005.

[4] D. Beasley, D. Bull, and R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[5] O. Beaumont, N. Bonichon, P. Duchon, and H. Larchevêque. Distributed approximation algorithm for resource clustering. *Lecture Notes in Computer Science*, 5058:61–73, 2008.

[6] C. Berge. *Hypergraphs.* North-Holland, 1989.

[7] T. Christof and A. Löbel. *PORTA: Polyhedron representation transformation algorithm*, 1997.

[8] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. *Combinatorial Optimization*, 1:315–338, 1979.

[9] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods*, 3(1):66–76, 1982.

[10] E. Coffman, M. Garey, and D. Johnson. Approximation algorithms for bin-packing — an updated survey. *Algorithm Design for Computer System Design*, 1984.

[11] F. Croce and D. Oliveri. Scheduling the italian football league: An ilp-based approach. *Comput. Oper. Res.*, 33(7), 2006.

[12] H. Crowder, E. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[13] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[14] B. Dilkina and W. Havens. The u.s. national football league scheduling problem. *IAAI EMERGING APPLICATIONS*, 1:814–819, 2004.

[15] S. Elhedhli, L. Li, M. Gzara, and J. Naoum-Sawaya. A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 23(3):404–415, 2011.

[16] L. Epstein and A. Levin. On bin packing with conflicts. *Lecture Notes in Computer Science*, 4368:160–173, 2007.

[17] F. Gardi. *Ordonnancement avec exclusion mutuelle par un graphe d'intervalles ou d'une classe apprentée : complexité et algorithmes.* PhD thesis, Université de la Méditerranée - Aix-Marseille II, 2005.

[18] F. Gardi. Mutual exclusion scheduling with interval graphs or related classes, part i. *Discrete Applied Mathematics*, 157(1):19 – 35, 2009.

[19] M. Garey, R. Graham, D. Johnson, and A. Chi-Chih Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory, Series A*, 21(3):257 – 298, 1976.

[20] M. Garey and D. Johnson. Approximation algorithms for bin packing problems: A survey. *Analysis and Design of Algorithms in Combinatorial Optimization*, 1981.

[21] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1990.

[22] M. Gendreau, G. Laporte, and F. Semet. Heuristics and lower bounds for the bin packing problem with conflicts. *Comput. Oper. Res.*, 31(3):347–358, 2004.

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

[24] R. Gomory. Outline of an algorithm for integer solutions to linear program. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[25] T. Gonzalez. *Handbook of Approximation Algorithms and Metaheuristics.* Chapman & Hall/CRC, 2007.

[26] M. Grötschel, M. Junger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6):1195–1220, 1984.

[27] M. Grötschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169 – 197, 1981.

[28] M. Grötschel and M. Padberg. On the symmetric travelling salesman problem I: inequalities. *Math. Program.*, 16(1):265–280, 1979.

[29] M. Grötschel and M. Padberg. On the symmetric travelling salesman problem II: lifting theorems and facets. *Math. Program.*, 16(1):281–302, 1979.

[30] C. Holland, J. Levis, R. Nuggehalli, B. Santilli, and J. Winters. Ups optimizes delivery routes. *Interfaces*, 47(1):8–23, 2017.

[31] IBM. Cplex optimization studio 12.7.

[32] K. Jansen. An approximation scheme for bin packing with conflicts. *Lecture Notes in Computer Science*, 1432:35–46, 1998.

[33] K. Jansen and S. Öhring. Approximation algorithms for time constrained scheduling. *Information and Computation*, 132(2):85 – 108, 1997.

[34] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

[35] R. Kalfakakou, S. Katsavounis, and K. Tsouros. Minimum number of warehouses for storing simultaneously compatible products. *International Journal of Production Economics*, 81–82:559 – 564, 2003.

[36] K. Kaparis and A. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical Programming*, 124:69 – 91, 2010.

[37] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:307–395, 1984.

[38] L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[39] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[40] G. Laporte and S. Desroches. Examination timetabling by computer. *Computers and OR*, 11, 1984.

[41] M. Maiza and M. Radjef. Heuristics for solving the bin-packing problem with conflicts. *Applied Mathematical Sciences*, page 1752, 2011.

[42] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319, 2003.

[43] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., 1990.

[44] A. Fernandes Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 22(3):401–415, 2010.

[45] I. Méndez-Díaz and P. Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2):159 – 179, 2008.

[46] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1988.

[47] T. Oncan, K. Altinel, and G. Laporte. Invited review: A comparative analysis of several asymmetric traveling salesman problem formulations. *Comput. Oper. Res.*, 36(3):637–654, 2009.

[48] M. Padberg. On the facial structure of set packing polyhedra. *Math. Program.*, 5(1):199–215, 1973.

[49] M. Padberg and G. Rinaldi. An efficient algorithm for the minimum capacity cut problem. *Math. Program.*, 47:19–36, 1990.

[50] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[51] C. Ribeiro. Sports scheduling: Problems and applications. *International Transactions in Operational Research*, 19(1-2):201–226, 2012.

[52] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2013.

[53] G. Sárközy. Improved monochromatic loose cycle partitions in hypergraphs. *Discrete Math.*, 334:52–62, 2014.

[54] M. Savelsbergh. A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research*, 45:831–841, 1997.

[55] E. Triki, Y. Collette, and P. Siarry. A theoretical study on the behaviour of simulated annealing leading to a new cooling schedule. *Eur J Oper Res*, 166:77—92, 2005.