Tesis de Licenciatura

Un modelo y algoritmo branch-and-cut para el problema del m-anillo-estrella con capacidades

Hernán Berinsky

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Directora: Dra. Paula Zabala

11 de julio de 2010

Índice general

Agradecimientos

A Paula por haberme dirigido la tesis, transmitirme parte de sus conocimientos y experiencia, y por sobre todo, por su gran predispoción, apoyo y paciencia durante el transcurso de este trabajo.

A los docentes de los departamentos computación y matemática de la FCEyN de la UBA de quienes tuve el gusto de aprender, y que día a día enseñan manteniendo la motivación de muchos alumnos a lo largo de la carrera.

A mis viejos, mi hermano, mis abuelas, mis abuelos que hoy no están pero siguen presentes.

A mis amigos y compañeros Fede, Agus, Claudio y Pablo por compartir largos momentos de estudio y trabajos prácticos. Y necesarios momentos esparcimiento, más largos aún.

A mis amigos que siempre están, Diego, el Nazo, Damián y Andrés.

Resumen

El *Problema del m-anillo-estrella con capacidades*, *CmRSP* (*Capacitated m-Ring-Star Problem*) consiste en encontrar una estructura llamada m-anillo-estrella de costo mínimo que conecte a un conjunto de clientes en red. La estructura está compuesta por conexiones anillo (con costos asociados) que forman m anillos (ciclos) disjuntos de nodos junto con un nodo distinguido (depósito) común a todos los anillos, y conexiones estrella (con costos asociados) que permiten conectar clientes a anillos. Adicionalmente, existen nodos de tránsito, llamados nodos de *Steiner* que pueden ser utilizados opcionalmente en los anillos en casos que permitan reducir costos al conectar clientes a estos nodos con conexiones estrella. Cada anillo, además, tiene una capacidad de clientes, esto quiere decir que no se puede superar un cierto límite de cantidad de clientes en cada anillo, considerando tanto los clientes que forman parte del anillo como aquellos que están conectados a algún nodo del anillo por medio de una conexión estrella. *CmRSP* pertenece a la clase de problemas *NP-Hard*.

Las redes con estructura de m-anillo-estrella se utilizan en redes de comunicación de alcance urbano de fibra óptica. Los cables que llevan las fibras se instalan bajo tierra, y la estructura de anillo permite que la falla en una conexión del anillo, que podría producirse accidentalmente en alguna excavación o rotura de calle por algún tipo de reparación, no afecte la conectividad de la red, asi como también, la falla de algún nodo no afecte al resto de los nodos de la red. Esto se debe a que cada cliente recibe un par de cables de fibra óptica, de manera que es capaz de transmitir en los dos sentidos posibles dentro de su anillo. Por otro lado, las conexiones estrella permiten reducir costos de instalación, considerando que estas conexiones se establecen entre nodos que están físicamente no muy alejados, de manera que sea menor la posibilidad de que ocurra alguna falla en la conexión.

El objetivo de esta tesis es abordar el *Problema del m-anillo-estrella con capacidades*. Se propone un nuevo modelo de *programación entera* y se desarrolla un algoritmo *branch-and-cut* para su resolución exacta. Para ello, se proponen *familias de desigualdades válidas* para el nuevo modelo y se diseñan *algoritmos de separación* de las mismas para utilizar como planos de corte en las relajaciones, se presenta una *heurística inicial* para obtener una cota primal, se analizan distintas *estrategias de branching* y de *recorrido del árbol*. Finalmente se implementa el algoritmo *branch-and-cut* y se analizan los resultados obtenidos.

Abstract

The *Capacitated m-ring-star problem*, *CmRSP*, is the problem of designing a structure named m-ring-star with minimum cost, to connect a set of customers in a network. The m-ring-star is a set of m rings that pass through a distinguished node (depot) shared by all rings, and optionally star connections (with costs) from customers not in rings to rings. Every connection have some cost. Also, the rings can include transit nodes, named *Steiner* nodes, to reduce costs if possible. The number of customers in a ring (included in the ring or connected to the ring using a star connection) is bounded by an upper limit (capacity). *CmRSP* belongs to *NP-Hard* problems's class.

Applications of m-ring-star networks includes design of optical urban networks. Fiber cables could be damaged during street reparations in a urban environment. Each customer receives a pair of fibers to send/receive information in clockwise and counter-clockwise way. So, if some ring connection fails, the network don't lose connectivity. Star connections may be established between non very distant nodes.

The aim of this thesis is to present a new integer programming formulation for the *Capacitated m-ring-star Problem* and develop a *branch-and-cut* algorithm to solve it. To achieve this goal, proposes *families of valid inequalities* for the new model and *separation algorithms* for them to use as cutting planes in linear relaxations, presents an *initial heuristic* to get a primal bound, branching and traversal tree strategies. Then implements the *branch-and-cut* algorithm and analyzes the obtained results.

Capítulo 1

Programación entera

1.1. Introducción

En las últimas décadas, el uso de modelos de programación entera para resolver problemas de optimización combinatoria se incrementó considerablemente. Avances en el estudio de modelos, en el software, hardware y nuevos algoritmos, permitieron un importante avance en la resolución de problemas que pueden formularse como problemas de programación entera, abarcando problemas de diversas áreas como ser problemas de telecomunicaciones, economía, producción, logística, etc., tales como problemas de ruteo de vehículos, de ruteo y diseño de redes de telecomunicaciones, de administración de cadenas de suministro, de asignación de tripulación en lineas aéreas, asignación de horarios, asignación de tareas en máquinas, minimización de desperdicio en el corte materiales, etc. Día a día continúan surgiendo nuevas aplicaciones derivadas de la necesidad de tomar decisiones en muchos ámbitos que requieren de la formulación de nuevos modelos y desarrollo de algoritmos para resolverlos.

En este capítulo se presentan algunos conceptos básicos de programación entera. En la primera sección sobre *formulaciones* se describe la formalización de un problema como problema de *programación entera*. En las secciones de *algoritmos branch-and-bound*, *algoritmos de planos de corte* y *algoritmos branch-and-cut* se describen métodos utilizados que permiten resolver estos modelos, es decir encontrar sus soluciones óptimas. Para una referencia sobre los conceptos de *programación entera* tratados en este trabajo se puede consultar [11].

1.2. Formulaciones

Un problema de *optimización combinatoria (COP)* es un problema que puede formularse como

$$\min_{S \subseteq N}\{\sum_{j \in S} c_j : S \in F\} \tag{1.1}$$

donde $N = \{1, ..., n\}$ es un conjunto finito, $c \in \mathbb{R}^n$ un vector de pesos, y F es un conjunto de subconjuntos de N.

Un problema de *programación entera (IP)* es un problema de la forma

$$\min\{cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^n\} \tag{1.2}$$

donde $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^{1 \times n}$, $b \in \mathbb{R}^{m \times 1}$ y $x \in \mathbb{Z}^{n \times 1}$ un vector columna, cuyas componentes son n variables enteras. La función $f : \mathbb{Z}^{n \times 1} \to \mathbb{R}$, $f(x) = cx$, se llama función objetivo; y el conjunto $\{x \in Z^{n \times 1} : Ax \geq b, x \geq 0\}$, se llama conjunto de soluciones factibles.

Un ejemplo problema de optimización combinatoria modelado como problema de programación entera es el problema de *cubrimiento por conjuntos*. A partir de un mapa con distintas regiones, se debe decidir donde instalar centros de emergencia, de manera que los mismos puedan servir a todas las regiones del mapa al menor costo posible. Cada posible ubicación de los centros de emergencia tiene asociado un costo de instalación y el conjunto de regiones a las que se puede acceder desde el mismo considerando que debe llegar de manera muy rápida a la región que acude (por ejemplo 6 minutos como máximo).

Como problema de optimización combinatoria, puede formularse como

$$\min_{S \subseteq N}\{\sum_{j \in S} c_j : S \in F\} \tag{1.3}$$

donde $M = \{1, ..., m\}$ el conjunto de regiones, $N = \{1, ..., n\}$ el conjunto de las posibles ubicaciones de los centros. $S_j \subseteq M$ el conjunto de regiones a las que puede servir el centro j, con $j \in N$, y c_j el costo de instalación del centro j, y $F = \{T \subseteq N : \cup_{j \in T} S_j = M\}$.

Formulado como problema de programación entera queda

$$\min\{cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^n\} \tag{1.4}$$

donde los elementos de la matriz A, $a_{ij} = 1$ si $i \in S_j$, y $a_{ij} = 0$ en caso contrario, $x_j = 1$ si el centro j se selecciona, y $x_j = 0$ en caso contrario, mientras que $b_i = 1$ y c_j es el costo de instalación del centro j, para cada $i \in \{1, ..., m\}$ y cada $j \in \{1, ..., n\}$. El problema queda formulado como

$$\min\{cx : \sum_{j=1}^{n} a_{ij} x_j \geq 1, i \in \{1, ..., m\}, x \in \{0, 1\}^n\} \tag{1.5}$$

Para un mismo problema pueden existir diferentes maneras de representarlo matemáticamente mediante formulaciones. De la formulación utilizada puede depender el éxito de resolver en forma óptima instancias grandes en un tiempo razonable. Algunas veces, contrariamente a la intuición, puede resultar ventajoso incrementar, en lugar de disminuir, la cantidad de variables o restricciones.

La relajación lineal de un problema de programación entera

$$\text{mín}\{cx : Ax \geq b, x \in \mathbb{Z}_+^n\} \tag{1.6}$$

es el problema de programación lineal

$$\text{mín}\{cx : Ax \geq b, x \in \mathbb{R}_+^n\} \tag{1.7}$$

Cada formulación de programación entera tiene un poliedro asociado $P = \{x \in \mathbb{R}^n : Ax \geq b\}$ con $A \in \mathbb{R}^{mxn}$ y $b \in \mathbb{R}^{mx1}$. La cápsula convexa de un conjunto $X \subseteq \mathbb{R}^n$, $conv(X)$, es el conjunto convexo más pequeño que contiene a X.

Conociendo una descripción de $conv(P)$ mediante un número polinomial (en la cantidad de variables) de desigualdades lineales, se puede resolver el problema como un problema de programación lineal, lo cual es computacionalmente fácil. Desafortunadamente, para la mayoría de los problemas no se ha podido obtener la descripción completa de la cápsula convexa y, en general, el número de restricciones lineales que la caracterizan es exponencial.

El método más simple para resolver un problema de programación entera consiste en enumerar todas las posibilidades, para cada posibilidad enumerada x y calcular cx; luego devolver un valor x cuyo cx es óptimo. Debido a la *explosión combinatoria* esta técnica sólo resulta aplicable a instancias sumamente pequeñas. En las siguientes secciones se describen algoritmos usados en la práctica para resolver problemas de programación entera.

1.3. Algoritmos branch-and-bound

Se mencionó que la utilización del método de enumeración para encontrar una solución óptima de un problema de programación entera, no es aconsejable en la práctica. Para mejorar esta técnica básica, muchas veces es posible descartar vectores del espacio de soluciones, mediante argumentos de dominancia o factibilidad, caracterizando de alguna manera conjuntos de vectores que no forman parte del espacio de soluciones factibles, o que si forman parte del espacio de soluciones factibles pero se conoce alguna cota del valor óptimo de ese conjunto de soluciones que certifica que dentro de ese conjunto no se encontrará un óptimo con un valor mejor respecto a un óptimo ya conocido. Dentro de esta línea, en los años 60

fueron propuestos los algoritmos *Branch-and-Bound*.

Un problema (RP) $z^R = \text{mín}\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ es una *relajación* de (IP) $z = \text{mín}\{c(x) : x \in X, X \subseteq \mathbb{R}^n\}$ si se cumple $X \subseteq T$, y $f(x) \leq c(x)$ para cada $x \in X$. La *relajación lineal* mencionada en la sección anterior es una relajación.
(Observación: en el caso de un problema de maximización, la relajación es el problema de maximización, y debe ser $f(x) \geq c(x)$ para cada $x \in X$).

Además de proveer duales, las relajaciones pueden brindar más información: Si RP es la relajación de un IP, y RP no es factible, entonces IP no es factible. Por otro lado, si x^* es una solución óptima de RP, y se cumple $x^* \in X$ y $f(x^*) = c(x^*)$ entonces x^* es una solución óptima de IP.

Dado un problema de programación entera

$$\text{mín}\{cx : Ax \geq b, x \in X\} \textit{ donde } X \subseteq \mathbb{Z}^n \tag{1.8}$$

Sea z^* el valor óptimo de la función objetivo del problema (de *minimización*); cada solución factible $x \in X$ provee una cota superior $\overline{z} \in \mathbb{R}$ de z^*, es decir, $\overline{z} = cx \geq z^*$; esta cota se llama *cota primal*. Las *cotas duales*, son cotas inferiores $\underline{z} \in \mathbb{R}$ de z^*, $\underline{z} \leq z^*$. Estas cotas pueden obtenerse a partir de relajaciones del problema. Estas relajaciones pueden construirse agrandando el espacio de soluciones factibles, como es el caso de la relajación lineal, y/o reemplazando la función objetivo, como es el caso de la relajación Lagrangeana.

Los algoritmos *branch-and-bound*, utilizan la técnica de *divide & conquer* basándose en que: si $S = \cup_{i=1}^{K} S_i$, entonces $\text{máx}\{cx : x \in S\} = \text{máx}_{1 \leq k \leq K}\{\text{máx}\{cx : x \in S_k\}\}$.

Este esquema puede representarse mediante un árbol cuya raíz corresponde al problema original y sus ramas resultan de la división en partes del espacio de búsqueda. A cada nodo del árbol le corresponde un subproblema que consiste en buscar el óptimo en una parte del espacio de soluciones. Esta descomposición permite descartar nodos del árbol (con sus correspondientes descendientes) de manera temprana, al detectar que bajo ese nodo no se encontrarán mejores soluciones que las que se obtuvieron o las que se pueden obtener por otras ramas del árbol, o bien porque no se encontrarán soluciones factibles bajo esos nodos, como se explica a continuación.

Las información sobre cotas primales y duales en cada nodo del árbol puede utilizarse de la siguiente manera.

Supongamos que se busca la solución de $\text{máx}\{cx : x \in S\}$, $S = S_1 \cup S_2$, y sean $\underline{z}$ y $\overline{z}$ cotas inferior y superior de S, y $\underline{z_i}$ y $\overline{z_i}$ cotas inferior y superior de S_i con $i \in \{1, 2\}$.

Si $b = \underline{z_i} = \overline{z_i} \geq \underline{z}$, es decir que el valor óptimo de de S_i es b (ya que las cotas inferior y superior son iguales), y por otro lado b es una cota inferior de S, ya que en S_i se exhibe que hay una solución factible para S con valor b en su función objetivo, lo cual mejora la cota inferior para S, entonces b puede reemplazarse por $\underline{z_i}$. Esta *poda* en el árbol de búsqueda se llama *poda por optimalidad*. Como consecuencia, hace que no sea necesario seguir analizando descendientes del nodo S_i, ya que se encontró un óptimo utilizando cotas primales y duales, y en segundo lugar, se consiguió reducir el *gap* (diferencia entre cota superior y cota inferior) en el problema padre (en este caso el problema original). El *gap* es cero al encontrarse el valor óptimo, sin embargo puede utilizarse también el *gap* como una medida de tolerancia para un criterio de parada del algoritmo, en la que si $\overline{z} - \underline{z} \leq \epsilon$ para algún $\epsilon > 0$ fijado, entonces se detiene la búsqueda. Esto puede servir en casos en los que la capacidad de respuesta del algoritmo sea más importante que obtener el valor óptimo, bajo una cierta tolerancia.

Otro caso en el que se puede utilizar la información de cotas es en la llamada *poda por cota*. Si $\underline{z_2} > \overline{z_1}$, entonces en S_1 no se encontrará una solución factible con valor mayor que $\underline{z_2}$ mientras que en S_2 se conoce que existe alguna solución factible cuyo valor lo supera. Por lo tanto, no tiene sentido seguir descendiendo por el nodo S_1.

Por último, en el caso que no se encuentre una solución factible (si una relajación en un nodo no tiene una solución factible, tampoco lo tendrá el problema original del nodo), entonces tampoco se continua la búsqueda por esa rama. Esta *poda* se llama *poda por insatisfactibilidad*.

Otra manera de encontrar cotas primales es aplicando heurísticas que encuentren soluciones factibles del problema, ya sea antes de iniciar el algoritmo de búsqueda utilizando la información del problema original (*heurística inicial*) o bien en algunos nodos del árbol utilizando información disponible sobre cotas primales, duales, solución de la relajación y mejor solución encontrada (*heurística primal*). Hay que tener en cuenta que cuanto mejores sean las cotas, será necesario explorar menos nodos. Sin embargo, el cálculo de estas cotas debe lograr un equilibrio entre calidad y esfuerzo en obtenerla. Una cota débil hará que se explore innecesariamente ramas del árbol, pero un procedimiento que brinde buenas cotas a un costo alto puede no justificarse (puede resultar más costoso que explorar muchos nodos del árbol). También se debe decidir sobre como descomponer el problema en distintos subproblemas en cada nodo (*regla de branching*), y como seleccionar el siguiente nodo a explorar (*estrategia de selección*). No hay una combinación de estos factores que resulte mejor para todos los problemas. Es necesario utilizar criterios basados en una combinación de teoría, sentido común y experimentación.

El proceso de *branching* consiste en dividir la región factible anterior en dos o más regiones factibles más pequeñas. Cada nueva región da origen a un nuevo subproblema (nodo hijo), originado por la adición de una nueva restricción al problema del nodo padre. Un requerimiento esencial es que cada solución entera factible del nodo padre pertenezca a, al

menos, uno de los hijos. Estos nuevos subproblemas son agregados a la lista de nodos activos, es decir, aún no explorados. La regla de *branching* más simple consiste en considerar *alguna* variable con valor fraccionario de la solución x^* de la relajación que tiene un valor fraccionario, f, en la solución actual. Si $x_j^* = f$, se puede tomar $S_1 = S \cap \{x : x_j \leq \lfloor f \rfloor\}$ y $S_2 = S \cap \{x : x_j \geq \lceil f \rceil\}$. De esta manera, $S = S_1 \cup S_2$ y $S_1 \cap S_2 = \emptyset$, además x^* no es una solución factible de la relajación lineal de S_1 ni de la relajación lineal S_2, por lo que si la solución óptima de S no es degenerada entonces en estos nuevos nodos se encontrará otra solución, y más aún la solución encontrada como cota primal será mejor ya que $\max\{\overline{z_1}, \overline{z_2}\} < \overline{z}$. Otra estrategia que se aplica es seleccionar una variable x_j fraccionaria tal que $\arg\max_{j \in C}\{\min\{f_j, 1 - f_j\}\}$, donde $f_j = x_j^* - \lfloor x_j^* \rfloor$ y C el conjunto de índices candidatos (de variables con valor fraccionario en x^*).

Respecto de la *estrategia de selección* del siguiente nodo a explorar de la lista de nodos activos, en la práctica hay varios argumentos contradictorios que pueden ser utilizados. Una estrategia sugiere que es conveniente obtener cotas primales lo más pronto posible, con la esperanza de encontrar una buena, ya que estas permiten hacer podas significativas. Bajo este argumento entonces, se sugiere que se haga una *búsqueda en profundidad* para alcanzar lo más pronto posible una solución factible para poder utilizar esa cota primal. Otro argumento en favor a esta estrategia es que una vez resuelta una relajación lineal, partiendo de la base óptima que se obtuvo en un nodo, resolver la relajación lineal de sus nodos hijos que consisten en el mismo problema más una restricción adicional en general no tiene mucho costo. Otra estrategia sugiere que para minimizar la cantidad de nodos a evaluar, es conveniente elegir el nodo activo con mejor la cota (si se está maximizando, seleccionar el nodo s tal que $\overline{z}_s = \max_t \overline{z}_t$); esta regla asegura que nunca se dividirá un nodo cuya cota superior sea menor que el valor óptimo, esta estrategia se llama *mejor nodo primero*. En la práctica se adoptan soluciones que involucran distintas estrategias, como por ejemplo una búsqueda en profundidad inicialmente hasta obtener una solución factible y poder utilizarla de cota primal, luego una estrategia mixta de *mejor nodo primero* y *búsqueda en profundidad*.

Esquema de algoritmo branch-and-bound para el problema $\max\{cx : x \in S \subseteq \mathbb{Z}^n\}$.

---

1. **Inicialización**
   - *a*) $S_0 := S$
   - *b*) $activos := \{S_0\}, \quad z^* := -\infty, \quad x^* :=$ **nulo**
2. **Selección de nodo**
   - *a*) **si** $S = \{\}$ **entonces devolver** x^*
   - *b*) *seleccionar* $S_i \in S$
   - *c*) $S := S \setminus \{S_i\}$

3. **Evaluación**

   a) *resolver la relajación de* S_i

   b) **si** *la relajación no es factible* **entonces** *podar por no factibilidad, e* **ir a** *Selección de nodo*

   c) *sea* z_i *el valor óptimo de la relajación de* S_i *y* x_i *su vector solución*

   d) $\overline{z_i} := z_i$

   e) **si** $\overline{z_i} \leq z^*$ **entonces** *podar por cota, e* **ir a** *Selección de nodo*

   f) **si** x_i *es una solución entera y* $z_i > z^*$ **entonces** $z^* := z^i$, $x^* := x^i$ y *podar por optimalidad e* **ir a** *Selección de nodo*

4. **Separación de problema**

   a) *dividir* S_i *en* S_{i_1}, ...,S_{i_k}

   b) $activos := activos \cup \{S_{i_1}, ..., S_{i_k}\}$

   c) **ir a** *Selección de nodo*

1.4. Algoritmos de planos de corte

Los algoritmos de *planos de corte* fueron originalmente propuestos por Gomory en los 60's [5] como un método general para resolver problemas de programación entera.

Los planos de corte pueden abordarse bajo dos enfoques:

- Con herramientas generales aplicables a cualquier problema de programación entera
  El algoritmo original propuesto por Gomory utiliza como planos de corte desigualdades derivadas del *tableau* óptimo de la relajación lineal, llamados *cortes de Gomory.* Aunque fue demostrado que este algoritmo, bajo ciertas condiciones, termina en un número finito de pasos, en la práctica su convergencia parece ser muy lenta. Por otro lado, la implementación computacional es numéricamente inestable, aunque en la actualidad han sido fortalecidas lográndose buenas implementaciones.
  Posteriormente, se han desarrollado algoritmos que utilizan una variedad de cortes aplicables a cualquier problema de programación entera, como por ejemplo los *cortes disyuntivos, clique, cover,* etc. Cualquiera de las técnicas mencionadas tienen la ventaja de poder ser utilizadas para cualquier problema de programación entera, independientemente de su estructura. Si bien esto es una propiedad deseable en un algoritmo, no siempre brinda la herramienta más adecuada para casos particulares. Un estudio más específico de la estructura particular del problema ayuda a obtener mejores procedimientos.

- Explotando la estructura particular del problema
  En los '70, resurgió el interés por los algoritmos de planos de corte debido al desarrollo de la teoría poliedral. Mediante el estudio de combinatoria poliedral, la intención es reemplazar el conjunto de restricciones de un programa de programación entera por la descripción de la cápsula convexa del conjunto de soluciones factibles. Las desigualdades lineales mínimas necesarias para describir a $conv(S)$ se denominan *facetas*. Si se conoce de forma completa esta descripción, el problema entero puede ser resuelto como un problema de programación lineal. De esta manera, explotando la estructura particular de cada problema, los planos de corte resultarán más efectivos a la hora de cortar soluciones.
  Desafortunadamente no es siempre fácil obtener una descripción de $conv(X)$. En los problemas de clase *NP-hard*, esta descripción requiere una cantidad exponencial de *facetas* a menos que sea $P = NP$. Los algoritmos de planos de corte permiten lidiar de alguna manera con este tipo de problemas en la práctica.

Si se considera el problema de programación entera

$$\text{máx}\{cx : x \in X\} \tag{1.9}$$

donde $X = \{x : Ax \leq b, x \in \mathbb{Z}_+^n\}$

Si $conv(X) = \{x : \tilde{A}x \leq \tilde{b}, x \geq 0\}$, entonces el problema original se puede reformular como

$$\text{máx}\{cx : \tilde{A}x \leq \tilde{b}, x \geq 0\} \tag{1.10}$$

y resolverlo como un problema de programación lineal, ya que la solución óptima de esta última formulaciíon es entera.

Una desigualdad $\pi x \leq \pi_0$ es una *desigualdad válida* para $X \subseteq R^n$ si $\pi x \leq \pi_0$ para cada $x \in X$.

Sea $X = P \cap Z^n$ y F una familia de desigualdades válidas $\pi x \leq \pi_0$, $(\pi, \pi_0) \in F$ para X. En muchos casos, F contiene demasiadas desigualdades, por lo que no es conveniente agregarlas a priori. Además, dada una función objetivo específica, lo que se busca es una buena aproximación de la cápsula convexa cerca de la solución óptima. El algoritmo de planos de corte, reformula de manera automática en cada iteración el problema agregando estas desigualdades como restricciones, en función de la solución obtenida por las relajaciones, en la medida que estas sean útiles.

El *Problema de separación* de un problema de programación entera, consiste en: dado un $x^* \in R^n$, verificar si $x^* \in conv(X)$, y en caso que $x^* \notin conv(X)$ encontrar una desigualdad válida $\pi x \leq \pi_0$ tal que x^* no la cumpla.

1. **Inicialización**

   a) $t := 0, \quad P^0 := P$

2. **Evaluación**

   a) *Resolver* $\overline{z}^t := \max\{cx : x \in P^t\}$, *sea* x^t *una solución óptima*

   b) **si** $x^t \in \mathbb{Z}^n$ **entonces terminar**, x^t *es una solución óptima de IP*

   c) **si** $x^t \notin \mathbb{Z}^n$ **entonces** *resolver el problema de separación para* x^t *y la familia* F

   d) **si** *se encontró una desigualdad* $(\pi^t, \pi_0^t) \in F$ *tal que* $\pi^t x^t > \pi_0^t$ **entonces** $P^{t+1} := P^t \cap \{x : \pi^t x \leq \pi_0^t\}$ *, incrementar* t *e* **ir a** *evaluación* **sino terminar**

Si el algoritmo termina sin encontrar una solución entera, entonces $P^t = P \cap \{x : \pi^i x \leq \pi_0^i, 1 \leq i \leq t\}$ es una formulación más ajustada que puede servir como entrada para un algoritmo *branch-and-bound*.

En la práctica puede ser conveniente agregar varias desigualdades violadas en cada iteración en lugar de una por vez.

El éxito del algoritmo depende en gran medida de la posibilidad y la eficiencia de encontrar desigualdades violadas, que puedan agregarse a la formulación para separar las soluciones fraccionarias.

En un algoritmo de separación debe buscarse un equilibrio entre la calidad de desigualdades que se obtienen y el costo que implica obtenerlas. A veces puede ser conveniente resolver mediante una heurística el problema, es decir, un algoritmo que no siempre encuentre desigualdades válidas violadas a pesar de que existan, si se justifica la ganancia en el tiempo de ejecución total.

1.5. Algoritmos branch-and-cut

En muchas instancias, los dos algoritmos descriptos arriba fallan en la resolución del problema. A comienzos de los '80 se comenzó a aplicar una metodología mixta que conjuga las dos ideas dando origen a los llamados algoritmos *branch-and-cut*. De esta manera se lograron resolver exitosamente instancias de tamaño considerable de una gran cantidad de problemas de programación entera. Uno de los factores que influyen en el fracaso de los algoritmos branch-and-bound es la baja calidad de las cotas obtenidas cuando se aplican relajaciones lineales.

Un algoritmo *branch-and-cut* es un algoritmo *branch-and-bound* en el cual se generan planos de corte a través del árbol de branch-and-bound. Esto significa que, sumado al objetivo de intentar reducir la cantidad de nodos del árbol con las podas del branch-and-bound, se realiza además un mayor trabajo en cada nodo en con el objetivo de obtener formulaciones más ajustadas. De esta manera, el algoritmo de branch-and-bound y de planos de corte se benefician mutuamente, ya que el branch-and-bound puede producir mejores cotas gracias a las desigualdades que se incorporan, al mismo tiempo que el proceso de branching perturba la solución fraccionaria ayudando a los algoritmos de separación.

En la implementación de un algoritmo *branch-and-cut* también influyen factores a tener en cuenta, como la estrategia propia de un algoritmo *branch-and-bound* sumados a las de un algoritmo de planos de corte. Además, se agregan nuevas decisiones tales como, decidir cuando buscar planos de corte, cuantos agregar, etc.

El esquema de un algoritmo *branch-and-cut* es el siguiente para un problema de programación entera.

---

1. **Inicialización**
   $N = \{P\}$ donde P es la formulación de $z = \max\{cx : x \in X\}$
   $\underline{z} = -\infty \quad z^* = nulo$

2. **Selección del próximo nodo**
   Si $N = \{\}$ entonces z^* es el óptimo y el algoritmo termina
   Si no, elegir y borrar un nodo i de N e ir a **Relajación**

3. **Relajación**
   Resolver la relajación lineal de i.

   a) Si no es factible, ir a **Selección del próximo nodo**

   b) Si es factible, x^i es el vector solución y $\overline{z}^i$ es el valor óptimo de i. Si x^i es fraccionaria entonces ir a **Cortes**, caso contrario ir a **Poda**

4. **Cortes**

   a) Si se decide no aplicar cortes, ir a **Poda**

   b) Si se deciden aplicar cortes, entonces intentar cortar la solución x^i.

      1) Si se encuentran cortes, agregar cortes a la formulación P, asignar a i dicha formulación e ir a **Relajación**
      2) Si no se encuentran cortes, ir a **Poda**

5. **Poda**

a) Si $\overline{z}^i \leq \underline{z}$ ir a **Selección del próximo nodo**

b) Si x^i no es fraccionaria, entonces $z^* = \text{máx}\{z^*, z^i\}$, e ir a **Selección del próximo nodo**

c) En otro caso, ir a **Ramificación**

6. **Ramificación**
   Seleccionar una variable de x^i fraccionaria, crear dos nuevos problemas y agregarlos a N, e ir a **Selección del próximo nodo**

Capítulo 2

Formulaciones para el Problema del m-anillo-estrella con capacidades

El *Problema del m-anillo-estrella con capacidades* ha sido abordado con modelos de programación entera por R. Baldacci, M. Dell' Amico y J. Salazar González [3], quienes propusieron dos modelos basados en modelos para el *Problema de Ruteo de Vehículos (VRP)* [10]. El primero se basa en el modelo de flujo de vehículos de dos índices, mientras que el segundo se basa en el modelo de dos flujos de mercaderías. Durante el transcurso del desarrollo de esta tesis, E. Hoshino y C. de Souza [7], abordaron un modelo de programación entera y algoritmo de generación de columnas para su resolución exacta. Por otra parte, A. Mauttone, S. Nesmachnow, A. Olivera y F. Robledo abordan en [8] una resolución heurística proponiendo una meteheurística híbrida, basada en las metaheurísticas *GRASP* y *Tabu Search*.

Si bien la estructura que se busca en el *CmRSP* no es la misma que en el *VRP*, se puede establecer una analogía entre el *CmRSP* y un problema de ruteo de vehículos. En el *VRP* se dispone de m vehículos, cada vehículo tiene capacidad para llevar a lo sumo Q paquetes, y se debe entregar un paquete a cada cliente y los paquetes se consideran indistinguibles; se buscan entonces, m rutas disjuntas en clientes (sin nodos de Steiner y sin conexiones estrella), que minimicen una función de costo.

2.1. Descripción del problema

En esta sección se presenta el *Problema del m-anillo-estrella* como problema de optimización definiendo la estructura *m-anillo-estrella* en términos de teoría de grafos. Para una referencia sobre conceptos y algoritmos en teoría grafos se pueden consultar [1] y [6].

Antes de presentar la definición de m-anillo-estrella de capacidad Q, se puede observar en la Figura 2.1 una representación gráfica que ayuda a entender los distintos componentes

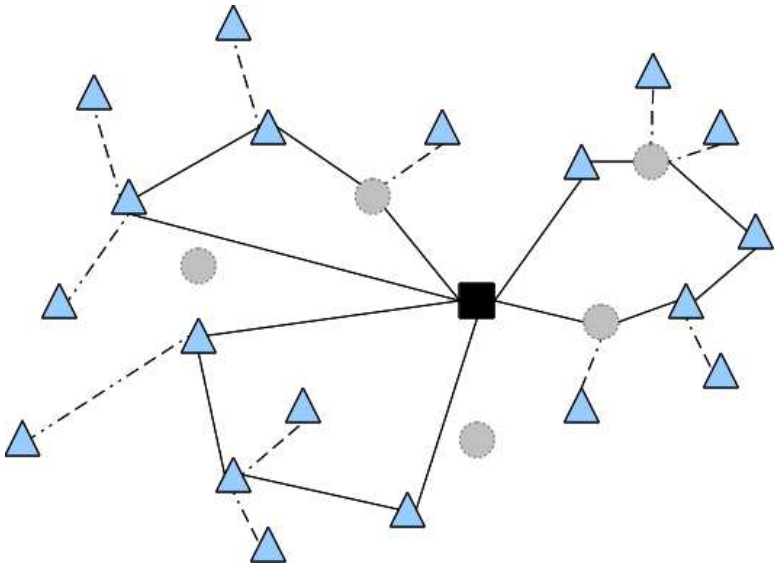

Figura 2.1: *Ejemplo de m-anillo-estrella de 19 clientes, 3 nodos de* Steiner *(en la figura aparecen además 2 nodos de* Steiner *sin utilizar), con 11 conexiones estrella,* $m = 3$ *y capacidad* $Q \geq 7$

que hacen a un m-anillo-estrella. El depósito está representado por un cuadrado, mientras que los clientes están representados con triángulos y los nodos de tránsito (también llamados nodos de *Steiner*) con círculos. Las líneas continuas representan conexiones anillo, mientras que las punteadas representan conexiones estrella.

En la Figura 2.1 se pueden observar 3 anillos (ciclos disjuntos, que comparten el depósito), y conexiones estrella entre clientes que no forman parte de ningún anillo y nodos (clientes o nodos de *Steiner*) que forman parte de un anillo. Todos los clientes deben estar conectados a la red, ya sea incluidos en un anillo o bien conectados a uno (y sólo uno) de ellos por medio de una conexión estrella. Las conexiones estrella se establecen de clientes fuera de un anillo a nodos (cliente o nodos de *Steiner*) que forman parte de un anillo, y no pueden establecerse entre clientes y el depósito. No es el obligatorio el uso de conexiones estrella ni el uso de nodos de *Steiner*, pero se utilizan si permiten reducir el costo de la red. La cantidad de anillos debe ser exactamente m, en este ejemplo es $m = 3$. La capacidad Q de un m-anillo-estrella es la cantidad máxima de clientes que pueden estar conectados a cada anillo-estrella. En este ejemplo se tienen 2 anillo-estrellas conectando 6 clientes y un anillo-estrella conectando a 7 clientes, por lo tanto debe ser $Q \geq 7$. Cada anillo debe tener por lo menos un nodo además del depósito, ya sea un cliente o nodo de *Steiner*.

Sea $G = (V, E \cup A)$ un grafo mixto, donde $V = \{d_0, d_1\} \cup V'$ es el conjunto de nodos, con $V' = U \cup W$, donde U es el conjunto de clientes y W el conjunto de nodos de

Steiner, d_0 es el nodo que representa el depósito y d_1 es una copia del nodo d_0 que se introduce para simplificar el modelo (un anillo queda representado entonces por un camino simple entre el nodo d_0 y el nodo d_1). E es un conjunto de ejes en V con costos de ruteo $c_{vv'} \in \mathbb{R} \quad \forall v \in (V' \cup \{d_0\}) \quad \forall v' \in (V' \cup \{d_1\})$ y A es un conjunto de arcos dirigidos con costos de conexión de U en V' dados por $d_{uv} \in \mathbb{R} \quad \forall u \in U \quad \forall v \in V'$.

Un *anillo-estrella* es un par (R, S), donde R es un subconjunto de ejes en E que representan un anillo, es decir, los ejes forman un camino simple entre el nodo d_0 y el nodo d_1, y S es un subconjunto de arcos de A de la forma (i, j), tales que i es un cliente y j es el extremo de algún eje en R que no sea ni d_0 ni d_1. Un anillo-estrella (R, S) cubre un cliente $u \in U$, si u es extremo de algún eje de R, ó bien S contiene una conexión de u a algún extremo de algún eje de R.

El objetivo del $CmRSP$ es diseñar m anillo-estrella$s$ de manera tal que cada cliente este cubierto exactamente por un anillo-estrella, cada nodo de Steiner sea visitado a lo sumo una vez, cada anillo no cubra más de Q clientes, y la suma de los costos de conexión y ruteo sea mínima.

En las siguientes formulaciones, para $S \subseteq V$, se define $\delta(S) = \{\{i, j\} \in E : i \in S, j \notin S\}$. Si $S = \{i\}$, se utiliza $\delta(i)$ como notación en lugar de $\delta(\{i\})$.

2.2. Modelos de la literatura

2.2.1. Formulación de flujo de vehículos de dos índices

En 2007, Baldacci, Dell' Amico y Salazar González [3], proponen también la siguiente formulación inspirada en la formulación de dos índices del *Problema de Ruteo de Vehículos* [10].

En esta formulación se definen las siguientes variables

Para cada $\{i, j\} \in E$ se define

$$x_{ij} = \begin{cases} 1 & \text{si los nodos } i \text{ y } j \text{ estan conectados en un anillo} \\ 0 & \text{en otro caso} \end{cases}$$

Para cada $(i, j) \in A$ se define

$$z_{ij} = \begin{cases} 1 & \text{si el arco } (i, j) \text{ es una conexión estrella del cliente } i \text{ al nodo } j \text{ de un anillo} \\ 0 & \text{en otro caso} \end{cases}$$

Para cada $j \in W$ se define

$$w_j = \begin{cases} 1 & \text{si el nodo de Steiner } j \text{ aparece en algún anillo} \\ 0 & \text{en otro caso} \end{cases}$$

Para simplificar la formulación, se define que sea $z_{ii} = 1$ si el cliente i forma parte de un anillo. Esto permite utilizar z para limitar la capacidad de clientes en los anillos, ya sean clientes que formen parte del anillo o bien conectados al anillo.

La formulación del problema queda definida como

$$min \sum_{e \in E} c_e x_e + \sum_{(i,j) \in A} d_{ij} z_{ij} \tag{2.1}$$

sujeto a

$$\sum_{e \in \delta(d_0)} x_e = m \tag{2.2}$$

$$\sum_{e \in \delta(d_1)} x_e = m \tag{2.3}$$

$$\sum_{e \in \delta(i)} x_e = 2z_{ii} \quad \forall i \in U \tag{2.4}$$

$$\sum_{e \in \delta(i)} x_e = 2w_i \quad \forall i \in W \tag{2.5}$$

$$\sum_{j \in V'} z_{ij} = 1 \quad \forall i \in U \tag{2.6}$$

$$\sum_{e \in \delta(S)} x_e \geq \frac{2}{Q} \sum_{i \in U} \sum_{j \in S} z_{ij} \quad \forall S \subseteq V' : S \neq \emptyset \tag{2.7}$$

$$x_e \in \{0,1\} \quad \forall e \in E \tag{2.8}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{2.9}$$

$$w_i \in \{0,1\} \quad \forall i \in W \tag{2.10}$$

Las restricciones (2.2) y (2.3) aseguran que m anillos salen del nodo d_0 y m anillos ingresan al nodo d_1 respectivamente. Las familias de restricciones (2.4) y (2.5) exigen que el grado de cada nodo que pertenezca a un anillo $i \in V'$ sea exactamente 2. La familia de restricciones (2.6) exige que cada nodo $i \in U$ (cliente) pertenezca a algun anillo ($z_{ii} = 1$) o bien este conectado en forma directa a algun anillo (para algun $j \in V'$ sea $z_{ij} = 1$). Por último, la familia de restricciones (2.7) imponen que para cualquier $S \subseteq V'$ se requieren al menos $\lceil \frac{1}{Q} \sum_{i \in U} \sum_{j \in S} z_{ij} \rceil$ anillos para visitar a todos los clientes de S (para cada uno de estos anillos se tienen dos ejes incidentes a nodos de S, observar que el depósito no está en S).

2.2.2. Formulación de dos flujos de mercaderías

En 2007, Baldacci, Dell' Amico y Salazar González [3], proponen la siguiente formulación inspirada en la formulación de flujo de mercaderías del *Problema de Ruteo de Vehículos* [10].

En esta formulación se asocia a cada eje $i, j \in E$ dos arcos de sentido opuesto, y se representa por las variables y_{ij} e y_{ji} el flujo que atraviesa por cada uno de estos arcos, y el costo asociado a estos arcos, c_{ij} es el mismo para ambos. En una solución factible, los anillos se representan por las variables y que forman dos caminos de sentido opuesto entre los nodos d_0 y d_1. En el camino desde el nodo d_0 hasta el nodo d_1, la variable y_{ij} representa la cantidad de clientes que restan visitar luego de haber alcanzado el nodo i. En el camino de sentido opuesto, desde el nodo d_1 hasta el nodo d_0, la variable y_{ij} representa la cantidad de clientes que puede visitar desde d_0 hasta j. Por lo tanto, considerando todos los anillos, el flujo total que parte desde el nodo 0 es $|U|$, mientras que el flujo total que parte desde el nodo d_1 es mQ. Cada nodo en el anillo absorbe en ambas direcciones, una cantidad de unidades del flujo igual a la cantidad de clientes conectados al nodo en caso de tratarse de un nodo de *Steiner*, y de uno más la cantidad de clientes conectados al nodo si se trata de un nodo cliente. Por lo tanto, considerando todos los anillos, en una solución factible, en el camino de flujo entre el nodo d_0 y el nodo d_1, parten $|U|$ y llegan 0 unidades a d_1; mientras que en el camino desde d_1 hasta d_0, parten desde d_1, mQ unidades y llegan $mQ - |U|$.

$$z_{ij} = \begin{cases} 1 & \text{si el arco } (i,j) \in A \text{ es una conexión estrella del cliente } i \text{ al nodo } j \text{ de un anillo} \\ 0 & \text{en otro caso} \end{cases}$$

$$w_j = \begin{cases} 1 & \text{si el nodo de Steiner } j \in W \text{ aparece en algún anillo} \\ 0 & \text{en otro caso} \end{cases}$$

$$\text{mín} \frac{1}{Q} \sum_{\{i,j\} \in E} c_{ij}(y_{ij} + y_{ji}) + \sum_{(i,j) \in A} d_{ij} z_{ij} \tag{2.11}$$

sujeto a

$$\sum_{j \in V'} y_{d_0 j} = |U| \tag{2.12}$$

$$\sum_{j \in V'} y_{d_1 j} = mQ \tag{2.13}$$

$$\sum_{j \in V'} y_{j d_1} = 0 \tag{2.14}$$

$$\sum_{j \in V'} y_{j d_0} = mQ - |U| \tag{2.15}$$

$$\sum_{i \in V} y_{ji} + y_{ij} = 2Q z_{jj} \quad \forall j \in U \tag{2.16}$$

$$\sum_{i \in V} y_{ji} + y_{ij} = 2Q w_j \quad \forall j \in W \tag{2.17}$$

$$\sum_{j \in V'} z_{ij} = 1 \quad \forall i \in U \tag{2.18}$$

$$\sum_{i \in V} (y_{ij} - y_{ji}) = 2 \sum_{i \in U} z_{ij} \quad \forall j \in V' \tag{2.19}$$

$$y_{ij} + y_{ji} \in \{0, Q\} \quad \forall \{i, j\} \in E \tag{2.20}$$

$$y_{ij} \geq 0 \quad \forall i, j \in V \tag{2.21}$$

$$z_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{2.22}$$

$$w_j \in \{0, 1\} \quad \forall j \in W \tag{2.23}$$

La restricción (2.12) asegura que el flujo total que parte desde el nodo d_0 es $|U|$, mientras que (2.13) asegura que el flujo total que parte desde el nodo d_1 es mQ (la capacidad total del m-anillo-estrella). La restricción (2.15) asegura que el flujo total que ingresa al nodo d_0 es $mQ - |U|$ (capacidad residual). Las familias de restricciones (2.16) y (2.17), aseguran que cada nodo del anillo el flujo que ingresa más el que sale igual a $2Q$. La familia de restricciones (2.18) asegura que cada cliente este asignado a una anillo. La familia de restricciones (2.19) aseguran la conservación de flujo (la cantidad de flujo que entra a un nodo es igual a la cantidad de flujo que sale). La familia de restricciones (2.19) asegura que la suma de los flujos de ambos sentidos entre cada par de nodos del anillo, es decir, la cantidad de nodos que se visitaron más que pueden visitarse luego de haberse visitado los anteriores es igual a Q.

2.3. Nuevo modelo de programación entera

2.3.1. Formulación de Miller-Tucker-Zemlin para el Problema del Viajante de Comercio

El *Problema del Viajante de Comercio*, *TSP* (*Traveling Salesman Problem*), es uno de los problemas más conocidos de optimización combinatoria. Un vendedor debe visitar n ciudades, partiendo desde una de ellas, visitando exactamente una vez cada ciudad, y regresando luego a la misma ciudad desde donde partió. Si se representa en un grafo completo, cada ciudad con un nodo, el conjunto de itinerarios posibles, que cumple las condiciones, está formado por todos los circuitos hamiltonianos del grafo. El costo de viajar desde la ciudad i a la ciudad j, está dado por c_{ij}. El problema consiste en encontrar un itinerario de costo mínimo. El *TSP*, planteado como problema de decisión [1], pertenece a la clase de problemas *NP-Completo* [4].

La formulación de *Miller-Tucker-Zemlin* (*MTZ*) del *TSP* es la siguiente

$$x_{ij} = \begin{cases} 1 & \text{si luego de visitar la ciudad } i \text{ se visita inmediatamente después la ciudad } j \\ 0 & \text{en otro caso} \end{cases}$$

z_i: cantidad de ciudades visitadas desde la ciudad 1 hasta la ciudad i inclusive

$$\text{mín} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.24}$$

sujeto a

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \quad \forall j \in \{1, ..., n\} \tag{2.25}$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \quad \forall i \in \{1, ..., n\} \tag{2.26}$$

$$z_j \geq (z_i + 1) - n(1 - x_{ij}) \quad \forall i, j \in \{2, ..., n\} \quad i \neq j \tag{2.27}$$

$$z_1 = 1 \tag{2.28}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in \{1, ..., n\} \tag{2.29}$$

[1] Dado un valor B (cota), determinar si existe un itinerario $\pi(1), ..., \pi(n)$ tal que $\sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)} \leq B$

$$z_i \in \{2, ..., n\} \quad \forall i \in \{2, ..., n\} \tag{2.30}$$

La restricciones (2.25) aseguran que el vendedor llegue a cada ciudad exactamente una vez, mientras que (2.26) aseguran que deje cada ciudad exactamente una vez. De esta manera se asegura que se visitan todas las ciudades, y cada una de ellas se visita una única vez. Considerando únicamente estas dos restricciones, no se asegura que existan soluciones factibles compuestas por circuitos disjuntos. El conjunto de restricciones que logran evitar estas soluciones factibles, se las conoce como restricciones de *rompimiento de subtours* [2]. Las restricciones (2.27) y (2.28) aseguran que, si luego de visitar i se visita inmediatamente j, entonces debe ser $z_j \geq z_i + 1$. Esto se debe a que, si luego de visitar i se visita inmediatamente j, entonces $x_{ij} = 1$, con lo cual por (2.27) debe ser $z_j \geq (z_i + 1) - n(1 - x_{ij}) = (z_i + 1)$, y por otro lado, si $x_{ij} = 0$ entonces por (2.27) debe ser $z_j \geq (z_i + 1) - n$, lo cual no agrega nueva información.

Para ver que (2.27), (2.28) y (2.30) no pueden satisfacerse simultáneamente si existen subtours, se observa que la existencia de subtours implica que existen circuitos disjuntos en nodos (al menos dos) [3], y uno de todos los circuitos contiene al nodo 1. Si $< \pi(1), ..., \pi(k), \pi(1) >$ con $1 < k < n$ es un circuito que contiene al nodo 1, entonces por (2.27)

$$\begin{gathered} z_{\pi(1)} - z_{\pi(2)} + n x_{\pi(1)\pi(2)} \leq n - 1 \\ z_{\pi(2)} - z_{\pi(3)} + n x_{\pi(2)\pi(3)} \leq n - 1 \\ \cdots \\ z_{\pi(k-1)} - z_{\pi(k)} + n x_{\pi(k-1)\pi(k)} \leq n - 1 \\ z_{\pi(k)} - z_{\pi(1)} + n x_{\pi(k)\pi(1)} \leq n - 1 \end{gathered}$$

Lo cual implica que $nk \leq (n-1)k$. Por lo tanto, no puede existir más de un circuito en una solución factible. Por otro lado, cada circuito que incluye las n ciudades, tiene una solución factible asociada que cumple que z_i es la cantidad de ciudades visitadas desde la ciudad 1 hasta la ciudad i inclusive.

[2] Existen distintas formulaciones del *TSP* cuyas restricciones consisten de (2.25) y (2.26), y difieren sus restricciones de rompimiento de subtours.

[3] Las restricciones (2.25) y (2.26) aseguran que no existan nodos aislados.

2.3.2. Nuevo modelo para el Problema del m-anillo-estrella con capacidades

Este modelo se basa en una idea similar a la aplicada en la formulación de *Miller-Tucker-Zemlin* para el rompimiento de subtours, tanto para asegurar es que los circuitos contengan al depósito, como para garantizar que no se supere la capacidad de cada anillo-estrella. Con lo cual, en esta formulación un m-anillo-estrella se representa con un grafo dirigido. $V = V' \cup \{d_0, d_1\}$ donde d_0 y d_1 representan el depósito. En la figura 2.2 muestra un m-anillo-estrella, y la figura 2.3 una representación del mismo en esta nueva formulación.

$\forall v \in V' \cup \{d_0\} \quad v' \in V' \cup \{d_1\} \quad v \neq v'$:

$$x_{vv'} = \begin{cases} 1 & \text{si los } v \text{ y } v' \text{ están conectados de manera directa en un anillo} \\ 0 & \text{en otro caso} \end{cases}$$

$\forall u \in U \quad v' \in V' u \neq v'$:

$$y_{uv'} = \begin{cases} 1 & \text{si } u \text{ está conectado a } v' \text{ por una conexión estrella} \\ 0 & \text{en otro caso} \end{cases}$$

z_v: es un número de orden asociado a un nodo $v \in V'$ [4]

f_v: es la cantidad de clientes visitados en un anillo-estrella desde d_0 hasta el nodo v inclusive (incluyendo conexiones estrella), si v forma parte de una conexión anillo

$$\text{mín} \sum_{v \in V} \sum_{v' \in V} c_{vv'} x_{vv'} + \sum_{u \in U} \sum_{v' \in V'} d_{uv'} y_{uv'} \tag{2.31}$$

sujeto a

$$\sum_{v \in V'} x_{d_0 v} = m \tag{2.32}$$

$$\sum_{v \in V'} x_{v d_1} = m \tag{2.33}$$

$$\sum_{v' \in V' \cup \{d_0\}, v' \neq u} x_{v'u} + \sum_{v' \in V', v' \neq u} y_{uv'} = 1 \quad \forall u \in U \tag{2.34}$$

[4] De manera análoga a la variable z de la formulación *Miller-Tucker-Zemlin*, esta variable se utiliza para evitar la existencia de circuitos que no contienen al depósito en las soluciones factibles

$$\sum_{v' \in V' \cup \{d_0\}, v' \neq w} x_{v'w} \leq 1 \quad \forall w \in W \tag{2.35}$$

$$\sum_{v'' \in V' \cup \{d_0\}, v'' \neq v'} x_{v''v'} = \sum_{v'' \in V' \cup \{d_1\}, v'' \neq v'} x_{v'v''} \quad \forall v' \in V' \tag{2.36}$$

$$y_{uv'} \leq \sum_{v \in V' \cup \{d_0\}, v \neq v'} x_{vv'} \quad \forall u \in U \quad \forall v' \in V' \quad u \neq v' \tag{2.37}$$

$$z_{v'} \geq (z_v + 1) - |V'|(1 - x_{vv'}) \quad \forall v, v' \in V' \quad v \neq v' \tag{2.38}$$

$$f_{u'} \geq 1 + \sum_{u \in U, u \neq u'} y_{uu'} \quad \forall u' \in U \tag{2.39}$$

$$f_{w'} \geq \sum_{u \in U} y_{uw'} \quad \forall w' \in W \tag{2.40}$$

$$f_u \geq (f_{v'} + 1 + \sum_{u' \in U, u' \neq u} y_{u'u}) - |U|(1 - x_{v'u}) \quad \forall u \in U \quad \forall v' \in V' \quad u \neq v' \tag{2.41}$$

$$f_w \geq (f_{v'} + \sum_{u' \in U} y_{u'w}) - |U|(1 - x_{v'w}) \quad \forall w \in W \quad \forall v' \in V' \quad w \neq v' \tag{2.42}$$

$$x_{v,v'} \in \{0, 1\} \quad \forall v, v' \in V \quad v \neq v' \tag{2.43}$$

$$y_{u,v'} \in \{0, 1\} \quad \forall u \in U, v' \in V' \quad u \neq v' \tag{2.44}$$

$$1 \leq z_{v'} \leq |V'| \quad \forall v' \in V' \tag{2.45}$$

$$1 \leq f_{v'} \leq Q \quad \forall v' \in V' \tag{2.46}$$

Las restricciones (2.32) y (2.33), aseguran que la cantidad de arcos que salen de d_0 y entran a d_1 sean exactamente m. La familia de restricciones (2.34) asegura que cada cliente $u \in U$ aparece en algún anillo o bien conectado a un anillo por una conexión estrella (debe darse una y solo una de estas dos condiciones). La familia de restricciones (2.35) impone a lo sumo un arco de entrada para cada nodo de *Steiner* (los nodos de *Steiner* pueden no utilizarse). La familia de restricciones (2.36) asegura (junto con (2.34) y (2.35)), que puede haber un arco de entrada a un nodo $v' \in V'$ en algún anillo si y solo si desde ese mismo hay un arco de salida. La familia de restricciones (2.37) asegura que para que exista un arco estrella desde un cliente $u \in U$ a un nodo $v' \in V$, el nodo v' debe aparecer en algún anillo. Las familia de restricciones (2.38) son de rompimiento de subtours, aseguran junto con las cotas de las variables z, que no se formen ciclos (anillos que no pasen por el depósito). Por último las familias de restricciones (2.39), (2.40), (2.41) y (2.42), aseguran junto con las cotas de las variables f, que no se supere la capacidad de los anillos.

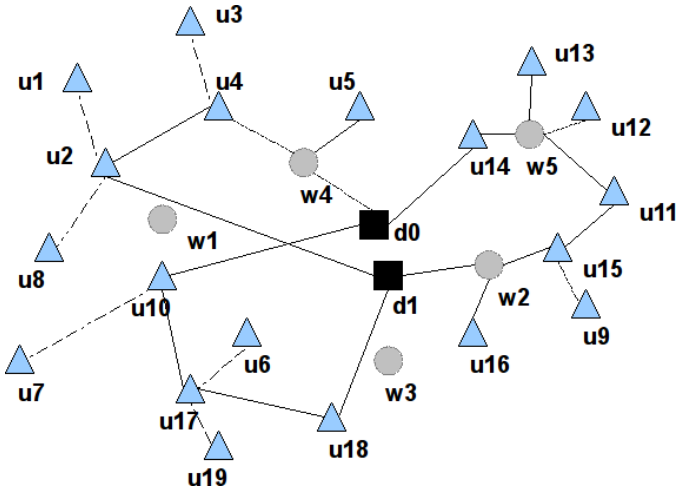

Figura 2.2: *Ejemplo de m-anillo-estrella etiquetado (d_0 y d_1 representan el depósito)*

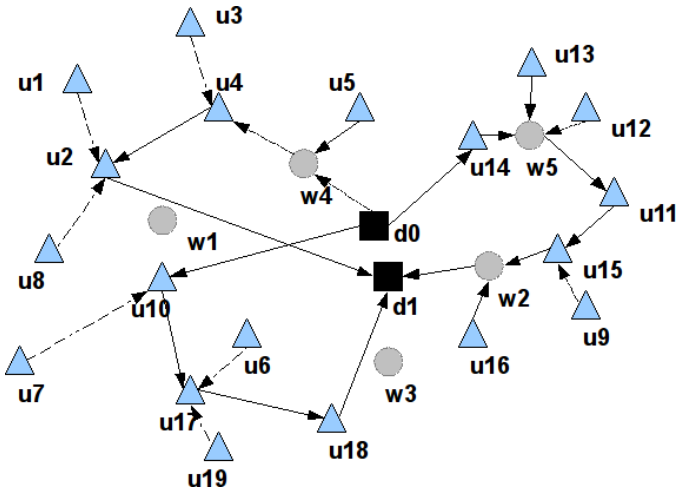

Figura 2.3: *Representación de m-anillo-estrella en la nueva formulación (d_0 y d_1 representan el depósito)*

2.3.3. Rompimiento de simetría

En esta sección se trata el problema de eliminación de soluciones simétricas. Las soluciones simétricas son soluciones factibles del *modelo* de programación entera que representan la misma solución en el *problema real*. Modificar el modelo para evitar la existencia de soluciones simétricas puede ayudar a veces a que un algoritmo que resuelva el modelo revise menos soluciones factibles. Al mismo tiempo, la eliminación de estas soluciones implica agregar restricciones al modelo, asi como también existe la posibilidad de que se incorporen nuevas variables, con lo cual a veces produce un efecto contraproducente dado que la dificultad de resolver el nuevo modelo no se justifica frente a tratar con soluciones simétricas. En algunos modelos es difícil encontrar la manera de romper la simetría (de manera completa), no obstante eliminar solo algunas soluciones simétricas puede también ayudar en algunos casos. En esta sección presentamos alternativas para descartar algunas soluciones simétricas. Se asignará un índice a cada cliente en U y nodo de Steiner en W, de manera que $U = \{u_1, ..., u_{|U|}\}$ y $W = \{w_1, ..., w_{|W|}\}$.

Como se puede observar en la Figura (2.4), utilizando el nuevo modelo presentado en la sección anterior, el siguiente par de m-anillo-estrellas G y G' ($m = 3$), formado por tres anillos sin conexiones estrella, representan soluciones factibles equivalentes.

$$G = \{ < d_0, u_3, u_2, u_7, u_1, d_1 >, < d_0, u_8, u_6, u_4, d_1 >, < d_0, u_5, u_{11}, u_{10}, u_9, d_1 >\} \quad (2.47)$$

$$G' = \{< d_0, u_1, u_7, u_2, u_3, d_1 >, < d_0, u_4, u_6, u_8, d_1 >, < d_0, u_5, u_{11}, u_{10}, u_9, d_1 >\} \quad (2.48)$$

Se puede observar que G y G' representan distintas soluciones factibles del *modelo* ya que se trata de soluciones factibles distintas, pero la misma solución dentro del *problema real* ya que el sentido de los anillos no tiene importancia en el problema real. Y en realidad, considerando estos 3 anillos con sus respectivos nodos, hay en total 2^3 soluciones factibles equivalentes (cada anillo puede orientarse en alguno de los dos sentidos posibles, en general, dada una solución factible hay 2^m maneras de orientar sus anillos).

A continuación se presentan dos maneras de tratar con este problema, en las que se utilizará la siguiente definición. Para cada $v \in V'$

$$\sigma(v) = \begin{cases} i & \text{si } v = u_i \in U \\ |U| + i & \text{si } v = w_i \in W \end{cases}$$

Una manera de evitar el problema en el sentido del anillo es establecer un criterio para establecer que sea factible solo una de todas estas soluciones. Un criterio posible es asignar a cada nodo v el valor constante $\sigma(v)$ entero, que llamamos *índice de* v, distinto para cada nodo, y que para cada anillo, si v y v' son los nodos conectados a d_0 y d_1 deba cumplirse $\sigma(v) \leq \sigma(v')$ (el caso $\sigma(v) = \sigma(v')$ se cumple únicamente cuando $v = v'$ que es el caso de

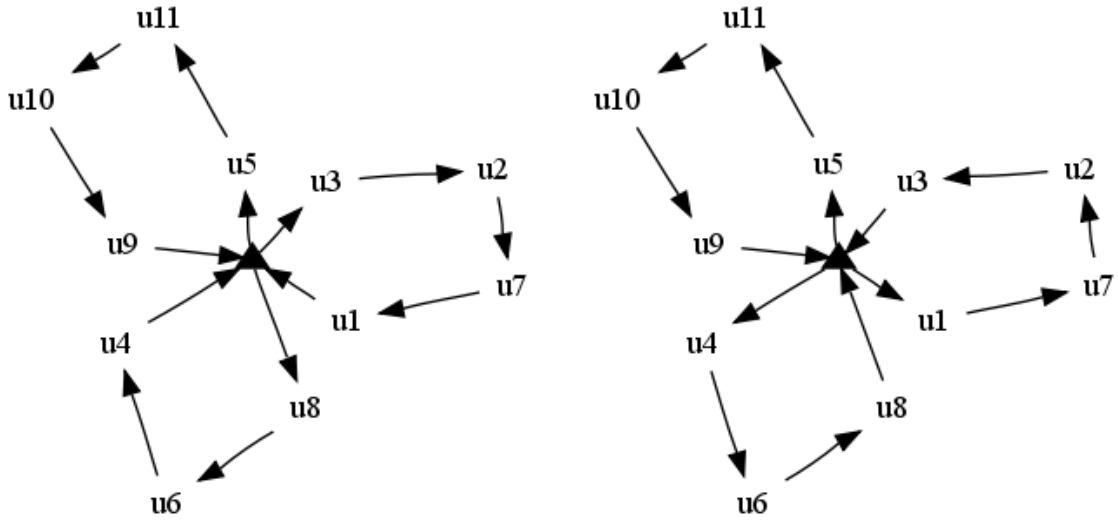

Figura 2.4: *Ejemplo de dos soluciones factibles distintas en el modelo que representan la misma solución en el problema real. El de la izquierda representa a G, y el de la derecha a G′.*

anillos que contienen un único nodo).

Aplicando esta idea, se puede extender el modelo de la siguiente manera.

Se agrega una nueva variable al modelo $p \in Z^{|U|+|W|}$, con $1 \leq p_v \leq |U| + |W|$, para cada $v \in V'$. Y se agregan las siguientes restricciones.

$$p_v \geq \sigma(v) x_{d_0 v} \quad \forall v \in V' \tag{2.49}$$

$$p_v \geq p_{v'} - (|U| + |W|)(1 - x_{v'v}) \quad \forall v, v' \in V' \tag{2.50}$$

$$p_v \leq \sigma(v) + (|U| + |W|)(1 - x_{vd_1}) \quad \forall v \in V' \tag{2.51}$$

Las desigualdades (2.49) aseguran que para cada nodo v conectado a d_0 sea p_v mayor o igual al índice de v (si v no es un nodo conectado a d_0 la desigualdad no agrega información). El efecto que produce, es el de asociar al primer nodo del anillo v (el nodo conectado a d_0) un valor p_v mayor o igual a su índice. Las desigualdades (2.50) aseguran que si (v', v) es un arco de un anillo, entonces el p_v debe ser mayor o igual al índice de v' (si (v', v) no es arco de un anillo, la desigualdad no agrega información). Esto produce que se propague la cota impuesta por la desigualdad anterior por todos los nodos del anillo, es decir, que para cada

nodo v del anillo se cumpla que el valor p_v sea mayor o igual al índice del primer nodo del anillo. Por último, la desigualdad (2.51) asegura que si v está conectado a d_1 en un anillo, entonces se debe cumplir que p_v sea menor o igual que su índice. El efecto producido por esta última familia es asegurar que el último nodo del anillo v (el nodo conectado a d_1) tenga un valor p_v menor igual a su índice. Dado que el valor p_v se propaga desde el primer nodo del anillo, las dos familias de desigualdades anteriores, aseguran que p_v sea mayor o igual al índice del primer nodo del anillo, mientras que esta última familia asegura que sea menor al índice del último nodo del anillo (la igualdad se cumple cuando el anillo tiene un único nodo).

En el caso del ejemplo (2.47), se llega a la siguiente contradicción:
Por la desigualdad (2.49) debe ser $p_{u_3} \geq 3$ $(v = u_3)$, luego por la desigualdad (2.50) deben ser $p_{u_2} \geq 3$ $(v' = u_3, v = u_2)$, $p_{u_7} \geq p_{u_2} \geq 3$ $(v' = u_2, v = u_7)$, $p_{u_1} \geq p_{u_7} \geq p_{u_2} \geq 3$ $(v' = u_7, v = u_1)$. Por la desigualdad (2.51), debe ser $p_{u_1} \leq 1$ $(v = u_1)$. Con lo cual no es una solución del sistema de inecuaciones, ya que debería ser $p_{u_1} \leq 1$ y $p_{u_1} \geq 3$.

El caso del ejemplo (2.48) es el que tiene la orientación de los anillos que cumple con las desigualdades (2.49), (2.50) y (2.51).

Un problema que presentan estas desigualdades, es la introducción de $O(|V'|)$ variables y $O(|V'|^2)$ restricciones, al mismo tiempo que genera nuevas soluciones simétricas. Esto se debe a que para la única orientación que se acepta en los anillos, el valor del vector p en general no es único. Con lo cual, introducir estas nuevas restricciones en el modelo pueden producir un efecto contraproducente, en el sentido de que pueden incrementar la dificultad de resolver el modelo con ellas.

Una desigualdad que permite eliminar algunas de las soluciones simétricas, menos que las anteriores, pero con la ventaja de no introducir nuevas variables en el modelo y sólo introduciendo una única restricción, es la siguiente:

$$\sum_{v \in V'} \sigma(v) x_{d_0 v} \leq \sum_{v \in V'} \sigma(v) x_{v d_1} \tag{2.52}$$

En el caso del ejemplo (2.47), la desigualdad (2.52) no se cumple, ya que el lado izquierdo es 16 y el derecho es 14. Mientras que en el ejemplo (2.47) se cumple la desigualdad (2.52), con el lado izquierdo 10 y el lado derecho 20. Pero también se cumple en la siguiente solución equivalente:

$$G'' = \{< d_0, u_3, u_2, u_7, u_1, d_1 >, < d_0, u_4, u_6, u_8, d_1 >, < d_0, u_5, u_{11}, u_{10}, u_9, d_1 >\} \tag{2.53}$$

en donde el lado izquierdo es 12 y el lado derecho 18.

La orientación de los anillos no es el único problema de simetría que se presenta en el modelo. Dada una solución factible con una orientación fija de los anillos, aparecen soluciones equivalentes que difieren en la asignación de valores de los vectores f y z. En el caso del vector f, un anillo que conecta menos de Q clientes (es decir, que no utiliza toda su capacidad) tiene más de una asignación posible de valores f en sus nodos.

La siguiente desigualdad asegura que para los clientes u conectados a un anillo por medio de un arco estrella, sea $f_u \leq 1$

$$f_u \leq 1 + |U|(1 - y_{uv}) \quad \forall u \in U, \forall v \in V', u \neq v \tag{2.54}$$

Si $y_{uv} = 0$ la desigualdad no agrega información, mientras que si $y_{uv} = 1$ entonces, dado que $f_u \geq 1$ por (2.46), debe ser entonces $f_u = 1$. Esto agrega $O(|V|^2)$ restricciones al modelo.

La siguiente desigualdad asegura que si v se conecta con un arco anillo a v', entonces $z'_v = z_v + 1$.

$$z_{v'} \leq z_v + 1 + |V'|(1 - x_{vv'}) \quad \forall v, v' \in V' \quad v \neq v' \tag{2.55}$$

Si $x_{vv'} = 0$ la desigualdad no agrega información, mientras que si $x_{vv'} = 1$ entonces, dado que $z_{v'} \geq z_v + 1$ por (2.38), debe ser entonces $z_{v'} = z_v + 1$. Esto agrega $O(|V|^2)$ restricciones al modelo.

En los experimentos realizados no se observaron ganancias significativas al agregar al modelo estas desigualdades, ya sea individualmente o en su conjunto.

Capítulo 3

Algoritmo branch-and-cut para el Problema del m-anillo-estrella con capacidades

En este capítulo se evaluan características que hacen a la implementación de un algoritmo branch-and-cut, como heurísticas para obtener cotas primales, relajaciones para obtener cotas duales, desigualdades válidas para utilizar como planos de corte, estrategias de recorrido y selección de nodos del árbol de branch-and-cut y otros detalles de implementación.

3.1. Algoritmo branch-and-bound

En esta sección describimos las principales características de los factores que definen el esquema de branch-and-bound. En general, no hay una elección óptima de cada una de las alternativas del algoritmo válida para cualquier instancia. Las características propias del problema son las que ayudan a determinar una buena elección.

3.1.1. Heurística inicial

A partir de soluciones factibles, en un problema de minimización, pueden obtenerse cotas superiores que permitirán reducir el tamaño del árbol de búsqueda en un algoritmo branch-and-cut. Cuanto más ajustada sea esta cota, más podrá reducirse el tamaño del árbol de búsqueda. Sin embargo, en un contexto de un algoritmo branch-and-cut, es necesario lograr un equilibrio entre el tiempo requerido en obtener las cotas y la calidad de las mismas. Mediante una *heurística inicial* (algoritmo aplicado al problema antes de iniciar el algoritmo de branch-and-cut), se puede producir una solución factible.

Se presenta una *heurística inicial* que consiste en un método de dos fases. En la primera fase, el método busca m clientes de manera de que sea poco probable que cualquier par de

estos comparta un anillo en una solución óptima. La segunda fase construye los m anillo-estrella, cada uno de estos partiendo de los m clientes hallados en la primera fase.

La primera fase tiene como objetivo seleccionar m clientes de manera que el costo de conexión entre los m clientes del conjunto sea alto, al mismo tiempo que no sea bajo el costo de conexión entre cualquier par de estos. Inicialmente se seleccionan dos clientes u_i y u_j tales que c_{u_i,u_j} sea máximo y se incorporan a un conjunto A. Luego, hasta completar m clientes se selecciona de manera iterativa un cliente u' de manera tal que $S(u') = \sum_{u \in A} c_{uu'}$ sea máxima sujeto a que $c_{uu'}$ no sea pequeño para algún $u \in A$ (ya que se busca que u y u' tengan una baja probabilidad de compartir un mismo anillo en una solución óptima). Por ejemplo, si $A = \{u_1, u_2\}$ y se tienen los candidatos u y u' y $c_{uu_1} = 1000$, $c_{uu_2} = 800$, $c_{u'u_1} = 1700$, $c_{u'u_2} = 200$, a pesar de que $c_{uu_1} + c_{uu_2} = 1800$, mientras que $c_{u',u_1} + c_{u',u_2} = 1900$, se busca evitar que se seleccione u' dado que su costo de conexión con u_2 es bajo respecto de los otros costos (con lo cual, deja de ser tan probable que en una solución óptima pertenezcan a anillos distintos). Para ello, se introduce una medida de variabilidad $V(u') = \sum_{u \in A}(c_{u',u} - \overline{c})^2$ con $\overline{c} = \frac{1}{n}\sum_{u \in A} c_{u',u}$. De esta manera, la heurística busca el u' con máximo $S(u')$ entre los u' cuyos $V(u')$ están entre los valores más pequeños (por ejemplo, los primeros $\frac{|U|}{2}$ más pequeños).

En la segunda fase, correspondiente a la construcción del m-anillo-estrella, se generan m pares de conjuntos (R_i, S_i) para cada $i \in \{1, ..., m\}$, donde para cada i, R_i representa el conjunto de arcos de un anillo y S_i el conjunto de arcos estrella. Inicialmente se asignan los m clientes obtenidos en la primer fase a un anillo distinto cada uno, y se definen las variables $primero_i$ y $ultimo_i$ para cada $i \in \{1, ..., m\}$ que representan el primer y último nodo visitado del en el anillo i (inicialmente cada anillo tiene como $primero_i = ultimo_i$ el nodo cliente obtenido en la primer fase). Para cada anillo i con más de un nodo, la variable $segundo_i$ representa el nodo al cual se conecta $primero_i$, es decir $(primero_i, segundo_i) \in R_i$; mientras que $penultimo_i$ representa el nodo que se conecta a $ultimo_i$, es decir $(penultimo_i, ultimo_i) \in R_i$. Luego, mientras queden clientes sin asignar a un anillo, se procede de la siguiente manera:

1. *Selección de cliente no asignado*
   Para cada cliente u no asignado, se analizan para cada anillo i con menos de Q clientes, los siguientes costos de inserción:

   a) $c(u, primero_i) + c(primero_i, segundo_i)$, costo de incorporación del cliente u al anillo i mediante un arco anillo desde el cliente u al cliente $primero_i$.

   b) $c(u, segundo_i) + d(primero_i, segundo_i) - c(primero_i, segundo_i)$, costo de incorporación del cliente u al anillo i, mediante un arco anillo desde el cliente u al cliente $segundo_i$ y modificando el tipo de conexión entre $primero_i$ y $segundo_i$ a una conexión estrella.

   c) $c(u, segundo_i) + d(primero_i, u) - c(primero_i, segundo_i)$, costo de incorporación del cliente u al anillo i, mediante un arco anillo desde el cliente u al cliente $segundo_i$,

eliminando la conexión entre $primero_i$ y $segundo_i$, y conectando $primero_i$ con u mediante un arco estrella.

d) $c(penultimo_i, ultimo_i)+c(ultimo_i, u)$, costo de incorporación del cliente u al anillo i, mediante un arco anillo desde el cliente $ultimo_i$ al cliente u.

e) $c(penultimo_i, u)+d(ultimo_i, penultimo_i)-c(penultimo_i, ultimo_i)$, costo de incorporación del cliente u como último nodo visitado del anillo i, mediante un arco anillo desde el cliente $penultimo_i$ al cliente u y modificando el tipo de conexión entre $ultimo_i$ y $penultimo_i$ a una conexión estrella desde $ultimo_i$ a $penultimo_i$.

f) $c(penultimo_i, u) + d(ultimo_i, u) - c(penultimo_i, ultimo_i)$, costo de incorporación del cliente u como último nodo visitado del anillo i, mediante un arco anillo desde el cliente $penultimo_i$ al cliente u, eliminando la conexión entre $penultimo_i$ y $ultimo_i$, y conectando $ultimo_i$ con u mediante un arco estrella.

Observaciones: los casos $(1b)$ y $(1c)$ sólo se consideran si $primero_i$ no tiene conectados clientes con arcos estrella y el anillo i tiene más de un cliente. De la misma manera, los casos $(1e)$ y $(1f)$ solo se consideran si $ultimo_i$ no tiene conectados clientes con arcos estrella y el anillo i tiene más de un cliente.

Se selecciona un cliente u y anillo i con costo mínimo de inserción.

2. *Actualización de estructura*

Se incorpora el cliente u al anillo i, actualizandose los conjuntos R_i y S_i, y los valores $primero_i$ ó $ultimo_i$ según el caso aplicado:

a) Para el caso $(1a)$, $R_i := R_i \cup \{(u, primero_i)\}$

b) Para el caso $(1b)$, $R_i := R_i \cup \{(u, segundo_i)\} \setminus \{(primero_i, segundo_i)\}$ y $S_i := S_i \cup \{(primero_i, segundo_i)\}$

c) Para el caso $(1c)$, $R_i := R_i \cup \{(u, segundo_i)\} \setminus \{(primero_i, segundo_i)\}$ y $S_i := S_i \cup \{(primero_i, cliente)\}$

d) Para el caso $(1d)$, $R_i := R_i \cup \{(ultimo_i, u)\}$

e) Para el caso $(1e)$, $R_i := R_i \cup \{(penultimo_i, u)\} \setminus \{(penultimo_i, ultimo_i)\}$ y $S_i := S_i \cup \{(ultimo_i, penultimo_i)\}$

f) Para el caso $(1f)$, $R_i := R_i \cup \{(penultimo_i, u)\} \setminus \{(penultimo_i, ultimo_i)\}$ y $S_i := S_i \cup \{(ultimo_i, cliente)\}$

Para los casos $(1a)$, $(1b)$ y $(1c)$ se actualiza $primero_i := u$, mientras que para los casos $(1d)$, $(1e)$ y $(1f)$ se actualiza $ultimo_i := u$.

3.1.2. Selección de variable de branching

La generación del árbol de búsqueda está definida por el proceso de *branching*. En esta etapa, el espacio de soluciones factibles asociado a un nodo se divide en dos o más conjuntos que representan los nuevos nodos (hijos) del árbol. Para llevar a cabo dicha división, se selecciona una variable x fraccionaria cuyo valor es x^* y se crean dos ramas, una correspondiente a $x \leq \lfloor x^* \rfloor$ y $x \geq \lceil x^* \rceil$. La selección de la variable se realiza por una combinación de un esquema de prioridades y un criterio de selección. Las variables se agrupan de acuerdo a su prioridad, y el criterio de selección se aplica sobre el grupo de variables de mayor prioridad que tenga variables fraccionarias.

Los criterios de selección de la variable de branching analizados son los siguientes:

- Por variable fraccionaria: se elige la variable con parte fraccionaria más cercana a $\frac{1}{2}$.
- Por variable fraccionaria: se elige la variable con parte fraccionaria más cercana a 0 ó 1.
- Strong branching: se considera primero un conjunto pequeño de variables con valor fraccionario, para cada una de estas se ramifica y se resulven sus correspondientes relajaciones. Con cada relajación se obtiene una cota inferior por cada rama, y como se está minimizando la mejor de estas será la mayor. Se elige entonces la variable que tiene una ramificación que mejor modifica la cota inferior. En lugar de resolverse las relajaciones de cada variable del conjunto, puede limitarse la resolución a una cantidad máxima de pivoteos en el algoritmo simplex dual de manera de obtener un equilibrio entre la calidad de las cotas inferiores y el tiempo que lleva obtenerlas.
- Pseudo-costos: Los pseudo-costos brindan una manera de estimar la degradación del valor de la función objetivo cuando se fuerza que una variable con valor fraccionario tome un valor entero. La técnica fue introducida en [2]. Para una variable fraccionaria candidata x_k, con parte fraccionaria $f_k = x_k - \lfloor x_k \rfloor$, los pseudo-costos asociados son: $U_k = \frac{\overline{z}_k - z_k^u}{1-f_k}$ $D_k = \frac{\overline{z}_k - z_k^d}{f_k}$, el primero correspondiente a la rama $\lceil x_k \rceil \geq x_k^*$ y el segundo a la rama $\lfloor x_k \rfloor \leq x_k^*$, z_k el óptimo de la relajación actual, z_k^u el óptimo de la relajación en la primer rama y z_k^d el óptimo de la relajación en la segunda rama (cotas duales). Se calcula la degradación de la variable como $D_k f_k + U_k(1 - f_k)$ (este valor se estima) y se selecciona como variable de branching la variable con degradación máxima.

En combinación con estos criterios, se pueden establecer prioridades en las variables o subconjuntos de variables. Esto significa que se seleccionen para el branching variables en orden de prioridad. Por ejemplo, se puede establecer una mayor prioridad para las conexiones de arcos anillo que de arcos estrella. Luego de seleccionarse las variables del conjunto de variables de mayor prioridad que están disponibles para el branching, se aplica alguno de

los criterios de selección anteriores. Además de dichos criterios, se evaluaron los siguintes criterios de selección de variable por prioridad:

- *P0*: Sin prioridades.
- *P1*: Arcos anillo (prioridad 4). Arcos estrella (prioridad 3). Variables z_v (prioridad 2). Variables f_v (prioridad 1).
- *P2*: Arcos anillo conectados a d_0 y arcos arcos anillo con clientes en sus dos extremos (prioridad 5). Arcos anillo con nodos de steiner en algún extremo y arcos anillo conectados a d_1 (prioridad 4). Arcos estrella (prioridad 3). Variables z_v (prioridad 2). variables f_v (prioridad 1).
- *P3*: Arcos anillo conectados a d_0 (prioridad 7). Arcos arcos anillo con clientes en sus dos extremos (prioridad 6). Arcos anillo conectados a d_1 (prioridad 5). Arcos anillo con nodos de steiner en algún extremo (prioridad 4). Arcos estrella (prioridad 3). Variables z_v (prioridad 2). variables f_v (prioridad 1).

En *P1*, *P2* y *P3*, se destaca un aspecto común en la asignación de prioridades. Las prioridades más bajas se asignan a las variables f_v y z_v, ya que por un lado al ser enteras, si comienzan a seleccionarse de manera temprana en el árbol tenderán a producir gran cantidad de ramificaciones (al realizarse ramificaciones binarias, la misma variable puede aparecer en distintos niveles del árbol ramificándose). Por otro lado estas variables son causantes de problemas de simetría, esto quiere decir, que en general existen soluciones factibles distintas que representan la misma solución en el problema real que pueden obtenerse modificando sus valores, por ejemplo, en cada anillo los valores de z_v se numeran de 1 a $|V'|$, si el problema tiene más de un anillo (es decir, $m > 1$), hay más de una manera de asignar esos valores a los z_v del anillo. Además, los valores de f_v de un anillo numeran los nodos del anillo-estrella de 1 a Q, en los anillos con menos de Q clientes se pueden numerar estos valores de otra manera. Con lo cual, es conveniente dejar que estos valores vayan tomando los valores de las relajaciones, ya que en estas variables no importan los valores en sí mismos (tienen coeficiente cero en la función objetivo), sino que estas variables son importantes como control para verificar que una solución no contenga subtours de anillos ni anillo-estrella*s* que excedan su límite de capacidad.

Respecto de las diferencias entre *P1*, *P2* y *P3* son distintos niveles de refinamiento. En *P1* se le da mayor prioridad a la parte de la estructura que es obligatoria (anillo) que de acuerdo a distintas pruebas realizadas es la que contiene la mayor parte de los clientes, esto permite obtener soluciones factibles en menor cantidad de niveles del árbol lo cual mejora las cotas primales (superiores) y puede permitir podar más el árbol, dejando para procesar en niveles más alejados de la raíz en el árbol las conexiones con arcos estrella que no forman parte de la estructura obligatoria y serían utilizados en una solución óptima si reducen el valor de la función objetivo. En *P2* se da un paso más en el mismo sentido, dando prioridad

a arcos que conectan clientes por sobre los que tienen nodos de Steiner y construyendo los anillos priorizando el primero que se conecta al depósito frente al último de manera que puedan setearse valores de algunas variables por las implicaciones lógicas que requieren el camino desde d_0 de un anillo construido parcialmente. En *P3* se da un último paso en el mismo sentido, estableciendo una esquema de prioridades mas refinado que el anterior.

3.1.3. Estrategias de recorrido del árbol

Una vez seleccionada la variable en la etapa de branching, se producen nuevos nodos que se agregan a los que ya estaban activos (aún no explorados). Seleccionar un nodo entre los nodos activos determina la forma en que se recorre el árbol.

Las estrategias analizadas son DFS (búsqueda en profundidad, selecciona el último nodo creado), BestBound (selecciona el nodo con mejor valor en la función objetivo de la relajación lineal), BestEstimate (selecciona el nodo en base a una *estimación* del valor de la función objetivo del problema entero). El paquete CPLEX ofrece dos variantes de la última estrategia.

3.2. Algoritmo de planos de corte

En esta sección se presentan familias de desigualdades válidas para ser utilizadas como planos de corte y algoritmos para resolver el problema de separación asociado a cada familia. Las familias *suma uno* y *anillo ó estrella* son propias de la nueva formulación, mientras que las familias *capacidad*, *conectividad* y *anillo multiestrella* son adaptaciones al nuevo modelo de las utilizadas en el *Problema de ruteo de vehículos con capacidades* (CVRP), descriptas en [10], y también adaptadas en [3] para las formulaciones del *CmRSP* de Baldacci, Dell'Amico y Salazar González.

El objetivo de aplicar planos de corte es obtener relajaciones lineales más ajustadas, de manera de poder conseguir mejores cotas duales (cotas inferiores, ya que se está minimizando). Al mismo tiempo, se debe considerar que la complejidad del algoritmo para resolver el problema de separación debe justificarse de acuerdo al beneficio obtenido gracias a las desigualdades agregadas. Además se debe tener en cuenta la cantidad de desigualdades generadas, ya que agregando una gran cantidad de desigualdades, el tiempo de resolución de las relajaciones se incrementa así como también el espacio utilizado en memoria.

3.2.1. Desigualdades válidas

En esta sección, para U el conjunto de clientes, W el conjunto de nodos de *Steiner*, $V' = U \cup W$, $V = V' \cup \{d_0, d_1\}$ y $S \subseteq V'$ (Observación: S no contiene al depósito), se definen $U(S) = U \cap S$ y $W(S) = W \cap S$.

Desigualdades de capacidad

La desigualdad

$$\sum_{v\in S}\sum_{v'\in V\setminus S} x_{vv'} + \sum_{u\in U(S)}\sum_{v\in V\setminus S} y_{uv} \geq \frac{1}{Q}(\sum_{u\in U(S)}\sum_{v\in V} x_{uv} + \sum_{u\in U}\sum_{v\in S} y_{uv}) \quad \forall S \subseteq V' \tag{3.1}$$

es una desigualdad válida.

Esta familia de desigualdades impone una cota inferior en la cantidad de arcos que deben entrar a un subconjunto $S \subseteq V'$.

Demostración:

Decimos que $u \in U$ se visita en S si $u \in U(S)$ es extremo de una conexión anillo, ó bien $u \in U$ se conecta por una conexión estrella con algún nodo $v \in S$ (Observación: puede suceder que $u \in U(S)$ este conectado por un arco estrella a un nodo $v \in V' \setminus S$, en ese caso u no se visita en S).

Por cada Q clientes $u \in U$ visitados en S, se requiere un anillo para satisfacer la demanda. Por lo tanto, si $v(S) = \sum_{u\in U(S)}\sum_{v\in V} x_{uv} + \sum_{u\in U}\sum_{v\in S} y_{uv}$ es la cantidad de clientes visitados en S, la cantidad de arcos de conexiones anillo que deben tener un extremo fuera de S es al menos $\lceil \frac{v(s)}{Q} \rceil \geq \frac{v(s)}{Q}$. ∎

Desigualdades de conectividad

La desigualdad

$$\sum_{v\in S}\sum_{v'\in V\setminus S} x_{vv'} + \sum_{u\in U(S)}\sum_{v\in V\setminus S} y_{uv} \geq \sum_{v\in V'\cup\{d_1\}} x_{uv} + \sum_{v\in S} y_{uv} \quad \forall u \in U(S), \forall S \subseteq V' \text{ tal que } |U(S)| \geq 1 \tag{3.2}$$

es una desigualdad válida.

Un problema que tienen las *desigualdades de capacidad*, es que si $\sum_{u\in U(S)}\sum_{v\in V} x_{uv} + \sum_{u\in U}\sum_{v\in S} y_{uv} < Q$, es decir, en S se satisface la demanda de menos de Q clientes, entonces el lado derecho de la desigualdad es menor que 1. En estos casos, las *desigualdades de conectividad* pueden aportar una mejor cota.

Demostración:

Si $u \in S$ es extremo de un arco anillo o está conectado por un arco estrella a algún nodo de S, entonces el lado derecho de la desigualdad es 1 y debe existir al menos un arco anillo con un extremo en S y otro extremo en $V \setminus S$. ∎

Desigualdades anillo multiestrella

La desigualdad

$$\sum_{v\in S}\sum_{v'\in V\setminus S} x_{vv'} + \sum_{u\in U(S)}\sum_{v\in V\setminus S} y_{uv} \geq \frac{1}{Q}\Big(\sum_{u\in U(S)}\sum_{v\in V} x_{uv} + \sum_{u\in U}\sum_{v\in S} y_{uv} + \sum_{u\in U(V\setminus S)}\sum_{v\in S} x_{uv}\Big) \quad \forall S \subseteq V' \tag{3.3}$$

es una desigualdad válida.
Estas desigualdades refuerzan de las *desigualdades de capacidad*, en las que además se consideran los clientes fuera de S que se conectan por un arco anillo a un nodo de S. ∎

Demostración:

Por cada Q clientes $u \in U$ ya sea visitados en S ó fuera de S pero conectados a algún nodo de S por un arco anillo, se requiere un anillo para satisfacer la demanda. Por lo tanto, si $v'(S) = \sum_{u\in U(S)}\sum_{v\in V} x_{uv} + \sum_{u\in U}\sum_{v\in S} y_{uv} + \sum_{u\in U(V\setminus S)}\sum_{v\in S} x_{uv}$ es dicha cantidad de clientes, entonces la cantidad de arcos anillo que salen de S es al menos $\lceil \frac{v'(S)}{Q} \rceil \geq \frac{v'(S)}{Q}$. ∎

Desigualdades suma uno

En primer lugar, se presenta la idea que hay detrás de esta desigualdad, y luego se verá la demostración en detalle.

Por un lado, se observa que si tomamos un cliente $u \in U$ de un m-anillo-estrella, podemos afirmar que este cliente tiene exactamente un nodo siguiente, es decir que para algún nodo $v \in V' \bigcup \{d_1\}$ se tiene que $x_{uv} = 1$ en el caso que u forme parte de un anillo ó bien $y_{uv} = 1$ en el caso de que u este conectado a un anillo mediante una conexión estrella, además solo se da uno sólo de estos casos. Esta observación se puede describir con la siguiente igualdad

$$x_{ud_1} + \Big(\sum_{v'\in V', v'\neq u} x_{uv'} + y_{uv'}\Big) = 1 \quad \forall u \in U$$

Ahora si seleccionamos algún otro nodo cliente $u' \in U \quad u' \neq u$, se puede escribir

$$x_{uu'} + y_{uu'} + x_{ud_1} + \Big(\sum_{v'\in V', v'\neq u, v'\neq u'} x_{uv'} + y_{uv'}\Big) = 1 \quad \forall u, u' \in U \quad u \neq u'$$

Por otro lado, observamos que si $u, u' \in U \quad u \neq u'$ y $x_{u'u} + y_{u'u} = 1$ entonces debe ser $x_{uu'} + y_{uu'} = 0$, ya que de otro modo las conexiones formarían un circuito. Si $x_{u'u} + y_{u'u} = 0$, no nos asegura que $x_{uu'} + y_{uu'} = 1$ (podría ser $x_{uu'} + y_{uu'} = 1$ ó bien $x_{uu'} + y_{uu'} = 0$, en este último caso u no es el siguiente de u', ni u' el siguiente de u).

A partir de esta información, podemos decir entonces que

$$x_{u'u} + y_{u'u} = 1 \Rightarrow x_{uu'} + y_{uu'} = 0 \Longleftrightarrow x_{ud_1} + (\sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'}) = 1$$

y que

$$x_{u'u} + y_{u'u} = 0 \Rightarrow x_{ud_1} + (\sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'}) \leq 1$$

De estas implicaciones se desprende que

$$x_{u'u} + y_{u'u} \leq x_{ud_1} + (\sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'}) \quad \forall u \in U$$

Antes de demostrar la desigualdad a partiendo de la nueva formulación, se presenta la siguiente proposición que se utilizará luego.

Proposición 3.1 $x_{ud_1} + \sum_{v' \in V', v' \neq u}(x_{uv'} + y_{uv'}) = 1 \quad \forall u \in U$

Demostración:

Sea $u \in U$, aplicando (2.36) con $v' = u$, se debe cumplir

$$\sum_{v'' \in V' \cup \{d_0\}, v'' \neq u} x_{v''u} = \sum_{v'' \in V' \cup \{d_1\}, v'' \neq u} x_{uv''}$$

Por lo tanto

$$x_{ud_1} + \sum_{v' \in V', v' \neq u} (x_{uv'} + y_{uv'}) = \sum_{v' \in V' \cup \{d_1\}, v' \neq u} x_{uv'} + \sum_{v' \in V', v' \neq u} y_{uv'} = \sum_{v' \in V' \cup \{d_0\}, v' \neq u} x_{v'u} + \sum_{v' \in V', v' \neq u} y_{uv'} = 1$$

Donde en la última igualdad se aplicó (2.34). ∎

Proposición 3.2 $\forall u, u' \in U \quad u \neq u' : x_{uu'} + y_{uu'} = 1 \Rightarrow x_{u'u} + y_{u'u} = 0$

Demostración:

Para que sea $x_{uu'} + y_{uu'} = 1$, debe ser $x_{uu'} = 1$ ó $y_{uu'} = 1$. Se divide la prueba en dos partes, en primer lugar si $x_{uu'} = 1$ se demostrará que $x_{u'u} + y_{u'u} = 0$, y por último si $y_{uu'} = 1$ probaremos que $x_{u'u} + y_{u'u} = 0$.

1. Caso $x_{uu'} = 1$
   Si $x_{uu'} = 1$, quiero ver que $x_{u'u} = 0$ y $y_{u'u} = 0$. Supongamos que $x_{u'u} = 1$ ó $y_{u'u} = 1$.
   Si $x_{uu'} = x_{u'u} = 1$ entonces aplicando (2.38) deben ser

   $$z_{u'} \geq z_u + 1$$

   $$z_u \geq z_{u'} + 1$$

   Pero estas dos desigualdades no pueden cumplirse simultaneamente ya que debería ser $z_u \geq z_{u'} + 1 \geq (z_u + 1) + 1$. Por lo tanto, si $x_{uu'} = 1$ entonces $x_{u'u} = 0$.

   Si $x_{uu'} = 1$, quiero ver que $y_{u'u} = 0$. Aplicando (2.34)

   $$1 = \sum_{v' \in V' \cup \{d_0\}, v' \neq u'} x_{v'u'} + \sum_{v' \in V', v' \neq u'} y_{u'v'}$$

   Por lo tanto, como $x_{uu'} = 1$, debe ser $\sum_{v' \in V', v' \neq u'} y_{u'v'} = 0$ y por lo tanto $\forall v' \in V' v' \neq u' : y_{u'v'} = 0$, en particular para $v' = u$, es $y_{u'u} = 0$.

2. Caso $y_{uu'} = 1$
   Si $y_{uu'} = 1$, quiero ver que $x_{u'u} = 0$ y $y_{u'u} = 0$.
   Aplicando (2.34)

   $$\sum_{v' \in V' \cup \{d_0\}, v' \neq u} x_{v'u} + \sum_{v' \in V', v' \neq u} y_{uv'} = 1$$

   Entonces $\sum_{v' \in V' \cup \{d_0\}, v' \neq u} x_{v'u} = 0$. Por lo tanto $x_{v'u} = 0 \quad \forall v' \in V' \cup \{d_0\}$, en particular $x_{u'u} = 0$.

   Supongamos que $y_{u'u} = 1$. Como $y_{uu'} = 1$, aplicando (2.37) debe ser

   $$1 = y_{uu'} \leq \sum_{v \in V' \cup \{d_0\}, v \neq u'} x_{vu'}$$

Por lo tanto debe ser $\sum_{v\in V'\cup\{d_0\},v\neq u'} x_{vu'} = 1$. Entonces

$$\sum_{v'\in V'\cup\{d_0\},v'\neq u'} x_{v'u'} + \sum_{v'\in V',v'\neq u'} y_{u'v'} = y_{u'u} + \sum_{v'\in V'\cup\{d_0\},v'\neq u'} x_{v'u'} + \sum_{v'\in V',v'\neq u',v'\neq u} y_{u'v'} \geq 2$$

Lo cual contradice (2.34). ∎

La desigualdad

$$x_{u'u} + y_{u'u} \leq x_{ud_1} + \sum_{v'\in V',v'\neq u,v'\neq u'} x_{uv'} + y_{uv'} \quad \forall u,u' \in U \quad u \neq u' \tag{3.4}$$

es una desigualdad válida.

Demostración:

En primer lugar, observaremos que se cumplen

1. $\forall u,u' \in U \quad u \neq u' : 0 \leq x_{ud_1} + \sum_{v'\in V',v'\neq u,v'\neq u'} x_{uv'} + y_{uv'} \leq 1$
2. $\forall u,u' \in U \quad u \neq u' : 0 \leq x_{u'u} + y_{u'u} \leq 1$

De esta manera, se demuestra (3.4) probando que

3. $\forall u,u' \in U \quad u \neq u' : \quad x_{ud_1} + \sum_{v'\in V',v'\neq u,v'\neq u'} x_{uv'} + y_{uv'} = 0 \Rightarrow x_{u'u} + y_{u'u} = 0$
4. $\forall u,u' \in U \quad u \neq u' : \quad x_{u'u} + y_{u'u} = 1 \Rightarrow x_{ud_1} + \sum_{v'\in V',v'\neq u,v'\neq u'} x_{uv'} + y_{uv'} = 1$

1. $\forall u,u' \in U \quad u \neq u' : 0 \leq x_{ud_1} + \sum_{v'\in V',v'\neq u,v'\neq u'} x_{uv'} + y_{uv'} \leq 1$

Sean $u,u' \in U \quad u \neq u'$, por la proposición (3.1)

$$x_{ud_1} + \sum_{v'\in V',v'\neq u,v'\neq u'} x_{uv'} + y_{uv'} \leq x_{ud_1} + \sum_{v'\in V',v'\neq u} x_{uv'} + y_{uv'} = 1$$

2. $0 \leq x_{u'u} + y_{u'u} \leq 1$

   Sean $u, u' \in U \quad u \neq u'$, supongamos que $x_{u'u} + y_{u'u} > 1$, entonces

   $$1 < \sum_{v' \in U', v' \neq u'} x_{u'v'} + y_{u'v'} \leq x_{u'd_1} + \sum_{v' \in V', v' \neq u'} x_{u'v'} + y_{u'v'}$$

   Lo cual contradice la proposición (3.1), ya que $x_{u'd_1} + \sum_{v' \in V', v' \neq u'} x_{u'v'} + y_{u'v'} = 1$

   Para probar los puntos 3 y 4, observemos que
   Sean $u, u' \in U, u \neq u'$, aplicando la proposición (3.1) se tiene

   $$1 = x_{ud_1} + \sum_{v' \in V', v' \neq u} (x_{uv'} + y_{uv'}) = x_{uu'} + y_{uu'} + x_{ud_1} + \sum_{v' \in V', v' \neq u, v' \neq u'} (x_{uv'} + y_{uv'})$$

   Por lo tanto $x_{uu'} + y_{uu'} = 1 - (x_{ud_1} + \sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'})$

   Con lo cual $x_{ud_1} + \sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'} = 0$ si y sólo si $x_{uu'} + y_{uu'} = 1$

   Aplicando la proposición (3.2), se demuestran 3 y 4 de la siguiente forma

3. $x_{ud_1} + \sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'} = 0 \Rightarrow x_{uu'} + y_{uu'} = 1 \Rightarrow x_{u'u} + y_{u'u} = 0$

4. $x_{u'u} + y_{u'u} = 1 \Rightarrow x_{uu'} + y_{uu'} = 0 \Rightarrow x_{ud_1} + \sum_{v' \in V', v' \neq u, v' \neq u'} x_{uv'} + y_{uv'} = 1$ ∎

Desigualdades anillo ó estrella

La desigualdad

$$y_{uv'} + x_{v'u} + x_{uv'} \leq 1 - \sum_{v \in V', v \neq v', v \neq u} y_{uv} \quad \forall u \in U \quad \forall v' \in V' \quad u \neq v' \tag{3.5}$$

es una desigualdad válida.

Demostración:

$$y_{uv'} + x_{v'u} + x_{uv'} \leq 1 - \sum_{v \in V', v \neq v', v \neq u} y_{uv} \iff x_{v'u} + x_{uv'} \leq 1 - \sum_{v \in V', v \neq u} y_{uv}$$

1. Caso $\sum_{v \in V', v \neq u} y_{uv} = 0$
   Quiero ver que $x_{v'u} + x_{uv'} \leq 1$. Supongo que $x_{v'u} + x_{uv'} > 1$, con lo cual deben ser $x_{v'u} = x_{uv'} = 1$.

   Por (2.38)

   $$x_{v'u} = 1 \Rightarrow z_u \geq z_{v'} + 1$$

   $$x_{uv'} = 1 \Rightarrow z_{v'} \geq z_u + 1$$

   Con lo cual se llega a la siguiente contradicción: $z_u \geq z_{v'} + 1 \geq z_u + 1 + 1$, de haber supuesto que $x_{v'u} = x_{uv'} = 1$.

   Por lo tanto, debe ser $x_{v'u} + x_{uv'} \leq 1$.

2. Caso $\sum_{v \in V', v \neq u} y_{uv} > 0$

   Por (2.34) debe ser $\sum_{v \in V' \cup \{d_0\}} x_{vu} = 0$, entonces $x_{v'u} = 0$. Aplicando (2.36) $\sum_{v \in V' \cup \{d_1\}} x_{uv} = \sum_{v \in V' \cup \{d_0\}} x_{vu} = 0$, entonces $x_{uv'} = 0$. ∎

3.2.2. Algoritmos de separación

En esta sección se describen algoritmos de separación para algunas de las desigualdades válidas. En un algoritmo de planos de corte, estos algoritmos de separación tienen un papel decisivo para que el método tenga éxito. Las familias de desigualdades válidas para un problema suelen tener una cantidad demasiado grande de desigualdades, y si bien incorporando una mayor cantidad de éstas se obtiene una región de factibilidad del problema relajado más ajustada, requerirá mayores recursos de memoria. Un algoritmo de separación debe encontrar para una solución óptima fraccionaria de una relajación, desigualdades válidas dentro de las familias de desigualdades válidas de la sección anterior que corten la solución de la relajación. En la práctica suelen aplicarse heurísticas para encontrar estos cortes en un tiempo razonable, la desventaja que tiene la aplicación de heurísticas de separación es que de existir alguna desigualdad válida de alguna familia que corte la solución fraccionaria, la heurística no brinda ninguna garantía que pueda encontrarla, por lo que se utilizaría un subconjunto de desigualdades válidas de la familia.

Para la familia de *desigualdades de capacidad*, puede aplicarse la siguiente heurística. A partir de una solución fraccionaria x^* y su grafo asociado $G = (V, E \cup A)$, consideramos el grafo $G' = (V, E \cup A')$ con $A' = A \setminus (\{(d_0, v) : v \in V'\} \cup \{(v, d_1) : v \in V'\})$. De esta manera,

en G' cualquier par de nodos que pertenezcan a un mismo anillo-estrella fraccionario formaran parte de una misma componente conexa de G', mientras que cualquier par de nodos que no pertenezca a un mismo anillo-estrella fraccionario forma parte de distintas componentes conexas en G'. Para cada componente conexa con más de Q clientes, la desigualdad de la familia de *desigualdades de capacidad* con S el conjunto de nodos de la componente corta la solución fraccionaria, solo en algunos casos por tratarse de una heurística. Para obtener cada componente, se puede aplicar un algoritmo que tome los nodos v de la solución tales que $x^*_{d_0,v} > 0$ y obtenga los nodos y cantidad de clientes alcanzables desde v con un algoritmo de recorrido de grafos (por ejemplo búsqueda en profundidad).

En la práctica las desigualdades de capacidad, conectividad y anillo-multiestrella, no demostraron tener suficiente eficiencia como la familia de desigualdades forma $\sum_{v\in S}\sum_{v'\in V\setminus S} x_{vv'} + \sum_{u\in U(S)}\sum_{v\in V\setminus S} y_{uv} \geq k$, donde $k = \lceil \frac{U(S)}{Q} \rceil$, ya que estas últimas permiten obtener cotas más ajustadas. En adelante, se mencionarán a estas desigualdades como *desigualdades de capacidad.*

Para la familia de desigualdades *suma uno*, en la demostración de (3.4) se observó que

$$x_{uu'} + y_{uu'} = 1 - (x_{ud_1} + \sum_{v'\in V', v'\neq u, v'\neq u'} x_{uv'} + y_{uv'})$$

Con lo cual, el algoritmo de separación consiste en buscar $u, u' \in U \quad u \neq u'$ tales que $x_{u'u} + y_{u'u} > 1 - (x_{uu'} + y_{uu'})$. Los pares (u, u') hallados violan la desigualdad (3.4), ya que

$$x_{u'u}+y_{u'u} > 1-(x_{uu'}+y_{uu'}) = 1-1+(x_{ud_1}+\sum_{v'\in V', v'\neq u, v'\neq u'} x_{uv'}+y_{uv'}) = x_{ud_1}+\sum_{v'\in V', v'\neq u, v'\neq u'} x_{uv'}+y_{uv'}$$

Para las familias de desigualdades *anillo ó estrella*, pueden verificarse para cada $u \in U$, $v \in V'$ con $u \neq v$ si (3.5) corta la solución de la relajación, y luego aplicar como cortes las desigualdades que no se cumplan.

3.3. Algoritmo branch-and-cut

Una vez analizados los factores relacionados a los algoritmos de *branch-and-bound* y *branch-and-cut*, queda por definir cuando aplicar cortes y cuantos cortes aplicar.

Una vez resuelta la relajación lineal en un nodo del árbol, se debe decidir si se generan cortes o se procede a realizar el branching. Es de esperar que los planos de corte ayuden a mejorar las cotas, permitiendo asi podar mejor las ramas del árbol. Sin embargo, los algoritmos de separación (por los cuales se buscan las desigualdades violadas) y la resolución

de la relajación tienen un costo, lo que hace que las decisiones de cuando y por cuantas iteraciones aplicar un algoritmo de planos de corte antes de realizar un *branching* es un factor crucial en la performance del algoritmo. Utilizamos para esto los parámetros *IPC* y *skip factor*. El *IPC* limita la cantidad de iteraciones que se realizan del algoritmo de planos de corte en cada nodo del árbol, y *skip factor* indica la frecuencia con la que se aplica planos de corte.

Capítulo 4

Experiencia computacional

En este capítulo se detallan las experiencias computacionales realizadas. El propósito de estas experiencias es por un lado identificar cuales alternativas resultan más favorables para el algoritmo branch-and-cut (BC). Por otro lado, evaluar instancias para configuraciones fijas de clientes, nodos de Steiner y depósito con sus costos asociados variando los valores de m y Q, con el fin de identificar en que medida afectan estos valores al tiempo de resolución.

Para ello se construyeron instancias de prueba y se utilizaron también las instancias evaluadas por R. Baldacci, M. Dell' Amico y J. Salazar González [3]. Las pruebas computacionales se realizaron en una SUN UltraSparc III 1 Ghz con 2 Gb de RAM utilizando las librerías del paquete CPLEX 10.1.

4.1. Instancias

Los experimentos computacionales se realizaron sobre distintas clases de instancias.

La primer clase, que llamamos E, son instancias simétricas euclideanas que se construyeron generando $|U|+|W|+1$ puntos con distribución uniforme en el cuadrado $[0,100]\times[0,100]$. De los puntos generados, uno de ellos representa el depósito, mientras que los restantes se distribuyen en $|U|$ clientes y $|W|$ nodos de Steiner. Los costos se tomaron como la distancia euclideana de los nodos que involucran (como consecuencia, los costos de los arcos son simétricos, $c_{vv'} = c_{v'v}$ y $d_{vv'} = d_{v'v}$).

Para cada par $(|U|,|W|) \in \{8,9,10,11,12,13,14\} \times \{0,2,4\}$ se generaron 20 instancias ($20 \times 7 \times 3 = 420$ instancias en total), y cada una de ellas fue probada para distintos valores (m,Q), de acuerdo a la cantidad de clientes de cada instancia. El criterio utilizado para asignar distintos valores de (m,Q) fue el siguiente: $m \in \{2,3,4,5\}$, luego dado un m fijo, se tomaron los posibles Q de manera tal que la demanda no pueda satisfacerse por menos de m anillos (es decir, que sea $Q < \overline{Q} = \lceil \frac{|U|}{m-1} \rceil$) y por otro lado, asegurar que el valor de

Q alcance a satisfacer toda la demanda (es decir, que sea $Q \geq \underline{Q} = \lceil \frac{|U|}{m} \rceil$). Por lo tanto, los valores (m, Q) se toman del conjunto $\{(m,Q) : m \in \{2,3,4,5\} \wedge \lceil \frac{|U|}{m} \rceil \leq Q < \lceil \frac{|U|}{m-1} \rceil\}$. La combinación de valores de m y Q, afectan en la densidad (cantidad de clientes en promedio) por anillo-estrella. Con este conjunto de pruebas se analizará el comportamiento de los distintos algoritmos para instancias de distintos tamaños, y al mismo tiempo, como afecta en cada instancia la variación de los valores m y Q.

La segunda clase de instancias, que llamamos *BDS*, son las que fueron evaluadas por R. Baldacci, M. Dell' Amico y J. Salazar González usando los algoritmos presentados en [3]. Estas instancias se construyeron a partir de las instancias de TSPLIB, tomando el primer nodo como depósito, los siguientes $\lfloor \alpha(n-1) \rfloor$ nodos como clientes y los restantes como nodos de Steiner, para los valores de $\alpha \in \{0.25,0.50,0.75,1.0\}$. Los costos se tomaron como la distancia euclideana entre los nodos. Estas instancias se evaluaron con $m \in \{3,4,5\}$ y $Q = \lceil \frac{|U|}{0,9m} \rceil$, para distintas combinaciones de 26 y 51 nodos (considerando clientes y nodos de *Steiner*).

4.2. Estrategia de recorrido del árbol y selección de variables

En esta sección se evalua la combinación de distintos criterios de selección de variable de branching y estrategias de recorrido del árbol. El objetivo es determinar que combinación resulta más favorable para la resolución del modelo planteado y del tipo de instancias que se están evaluando.

En la *Tabla B&B* se presentan los tiempos de resolución, y el gap obtenido, sobre las instancias *BDS*. En la Figura 4.1 se pueden ver los promedios de tiempos y %gaps de los datos de la tabla para cada una de las combinaciones. Las combinaciones de parámetros aplicados son:

- Los criterios de selección de variables, *vars*, descriptos en (3.1.2):
  - máxima inviabilidad (MI), se elige la variable con parte fraccionaria más cercana a $\frac{1}{2}$
  - pseudo-costos (PC)
  - pseudo-costos reducidos (RPC), un algoritmo similar a pseudo-costos que provee CPLEX, en el que se busca aproximar sus valores, computacionalmente menos intensiva
  - strong-branching (SB)

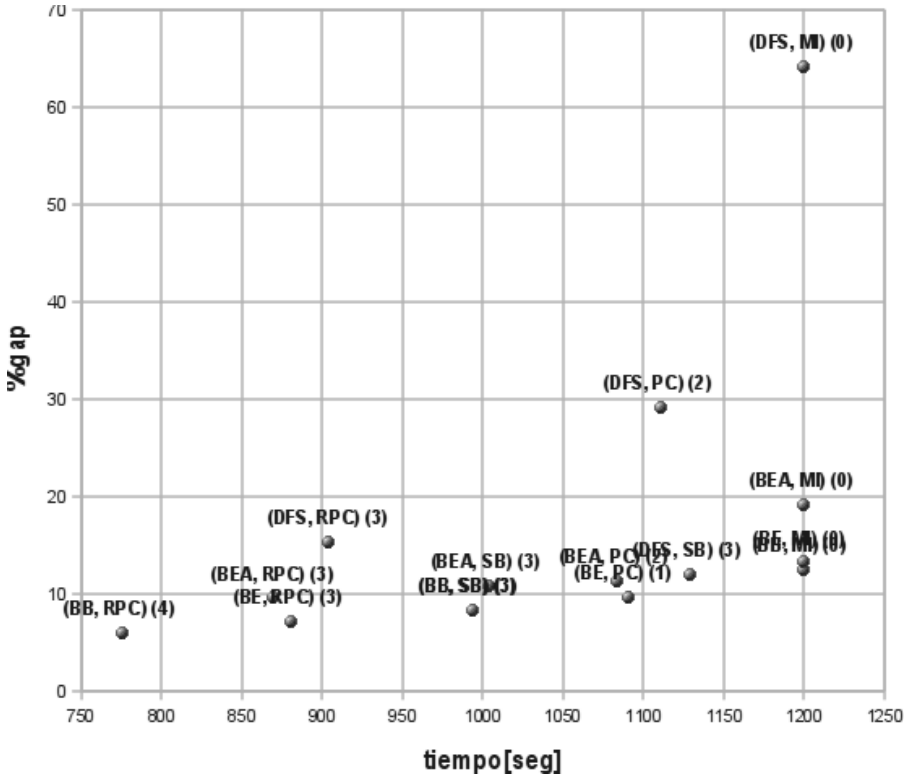

Figura 4.1: *Promedios de tiempo y %gap. Entre parentesis aparecen la cantidad de instancias resueltas (gap = 0.0 %) en 1200 segundos para cada combinación*

- Las estrategias de recorrido del árbol, *nods*, descriptos en (3.1.3):
  - búsqueda en profundidad (*DFS*)
  - mejor cota (*BB*)
  - mejor estimación (*BE*)
  - mejor estimación alternativa (*BEA*), una variante de mejor estimación que provee CPLEX, computacionalmente menos intensiva.
- Los esquemas de prioridades *P0*, *P1*, *P2* y *P3* descriptos en (3.1.2)

Los valores presentados en la tabla son los siguientes:

t_{P_X}: el tiempo de resolución, expresado en segundos, aplicando el criterio de prioridades P_X (* si superó el tiempo límite de 1200 segundos)

g_{P_X}: el porcentaje de gap obtenido aplicando el criterio de prioridades P_X (* si no se encontró una solución factible, y 0 si se comprobó que la solución es óptima)

nod_{P_X}: la cantidad de nodos explorados del árbol aplicando el criterio de prioridades P_X

Se puede observar que, salvo en algunos pocos casos, el criterio de prioridades no produce un cambio significativo en el resultado obtenido. Las excepciones que se observan son en los casos:

- Para nods=BEA y vars=MI
  - n=26, m=3, Q=7, $|U| = 18$, $|W| = 7$, donde aplicando el criterio de prioridades P1 se obtuvo, para el mismo tiempo de resolución un gap de un 10 % mejor que aplicando los demás criterios.
  - n=26, m=3, Q=10, $|U| = 25$, $|W| = 0$, donde aplicando el criterio de prioridades P2 se obtuvo el valor óptimo, para el mismo tiempo de resolución, mientras que los restantes criterios finalizó la ejecución con un gap de un 8.18 %, 13.18 % y 98.50 %.
  - n=26, m=5, Q=4, $|U| = 18$, $|W| = 7$, donde aplicando el criterio de prioridades P1 se obtuvo para el mismo tiempo de resolución, un gap de aproximadamente un 10 % mejor que aplicando los otros criterios.

- Otros casos
  - n=26, m=3, Q=, $|U| = 18$, $|W| = 7$, nods=BEA, vars=SB, donde aplicando el criterio de prioridades P3 se obtuvo el valor óptimo, para el mismo tiempo de resolución, mientras que los restantes criterios finalizó la ejecución con un gap de un 17.54 %.
  - n=26, m=3, Q=7, $|U| = 18$, $|W| = 7$, nods=DFS, vars=MI, donde aplicando el criterio de prioridades P0 se obtuvo, para el mismo tiempo de resolución, un gap de 16.34 %, mientras que los restantes criterios finalizó la ejecución con un gap de un 96.67 %.
  - n=26, m=3, Q=10, $|U| = 25$, $|W| = 0$, nods=BE, vars=PC, donde aplicando el criterio de prioridades P0 se obtuvo el valor óptimo, para el mismo tiempo de resolución, mientras que los restantes criterios finalizó la ejecución con un gap de un 27.86 %.
  - n=26, m=4, Q=5, $|U| = 18$, $|W| = 7$, nods=BE, vars=SB, donde aplicando el criterio de prioridades P1 se obtuvo el valor óptimo, para el mismo tiempo de resolución, mientras que los restantes criterios finalizó la ejecución con un gap de un 18 %.
  - n=26, m=4, Q=7, $|U| = 25$, $|W| = 0$, nods=BB, vars=RPC, donde aplicando los criterio de prioridades P0 y P3 se obtuvo el valor óptimo en 254 segundos y menos de 1 segundo respectivamente, mientras que los restantes criterios finalizó la ejecución a los 1200 segundos con un gap de un 9.94 %.

Respecto de los parámetros de selección de variable de branching (*vars*) y selección del siguiente nodo a (*nods*), se observa que la combinación de estos parámetros que resultaron efectivos en mayor medida respecto de otras combinaciones de parámetros son con vars=BB y nods=RPC.

Las instancias utilizadas en estas pruebas son las *BDS*, los valores que figuran en cada prueba de n, $|U|$, m y Q se corresponden con las instancias ei$l$n.tsp.m.$|U|$.Q.A.BDS.cmrsp. Cada prueba corresponde a una instancia y una combinación de parámetros vars y nods.

n	m	Q	$\|U\|$	$\|W\|$	nods	vars	t_{P0}	g_{P0}	nod_{P0}	t_{P1}	g_{P1}	nod_{P1}	t_{P2}	g_{P2}	nod_{P2}	t_{P3}	g_{P3}	nod_{P3}
26	3	5	12	13	DFS	RPC	235	0.00	46700	234	0.00	46700	234	0.00	46700	235	0.00	46700
26	3	5	12	13	BEA	PC	1121	0.00	220500	1116	0.00	220500	1116	0.00	220500	1118	0.00	220500
26	3	5	12	13	BB	SB	595	0.00	15600	598	0.00	15600	596	0.00	15600	603	0.00	15600
26	3	5	12	13	BE	PC	*	3.49	236700	*	3.58	234100	*	3.57	234700	*	3.46	237400
26	3	5	12	13	DFS	SB	1111	0.00	29500	1108	0.00	29500	1109	0.00	29500	1115	0.00	29500
26	3	5	12	13	BEA	MI	*	14.74	289400	*	14.74	292400	*	14.74	293700	*	14.74	294500
26	3	5	12	13	BB	PC	*	3.47	237400	*	3.54	235500	*	3.53	235500	*	3.49	236700
26	3	5	12	13	BEA	RPC	143	0.00	26600	143	0.00	26600	143	0.00	26600	143	0.00	26600
26	3	5	12	13	BE	MI	*	9.41	243900	*	9.41	243900	*	9.41	244200	*	9.41	243600
26	3	5	12	13	DFS	PC	*	10.61	245800	*	10.61	244900	*	10.61	245500	*	10.61	244900
26	3	5	12	13	BE	RPC	169	0.00	32400	169	0.00	32400	169	0.00	32400	171	0.00	32400
26	3	5	12	13	BB	MI	*	9.41	243400	*	9.41	244600	*	9.41	244600	*	9.41	243900
26	3	5	12	13	BEA	SB	590	0.00	15600	593	0.00	15600	594	0.00	15600	593	0.00	15600
26	3	5	12	13	BB	RPC	170	0.00	32400	172	0.00	32400	170	0.00	32400	169	0.00	32400
26	3	5	12	13	DFS	MI	*	*	170600	*	*	170200	*	*	171300	*	*	172000
26	3	5	12	13	BE	SB	597	0.00	15600	597	0.00	15600	601	0.00	15600	598	0.00	15600
26	3	7	18	7	BB	MI	*	14.38	168300	*	14.38	168300	*	14.38	168300	*	14.38	167800
26	3	7	18	7	BEA	SB	*	17.54	14900	*	17.54	14900	*	17.54	14900	1199	0.00	14900
26	3	7	18	7	BB	RPC	*	9.79	112600	*	9.80	112400	*	9.80	111900	*	9.80	112300
26	3	7	18	7	DFS	MI	*	16.34	157800	*	99.67	153500	*	99.67	153800	*	99.67	153800
26	3	7	18	7	BE	SB	*	10.50	11100	*	10.49	11300	*	10.45	11800	*	10.44	11900
26	3	7	18	7	DFS	RPC	*	19.50	151700	*	19.50	151300	*	19.50	151100	*	19.50	151400
26	3	7	18	7	BEA	PC	*	14.66	134700	*	14.66	134000	*	14.66	133500	*	14.66	133400
26	3	7	18	7	BB	SB	*	10.45	11800	*	10.48	11300	*	10.48	11300	*	10.49	11200
26	3	7	18	7	BE	PC	*	10.98	127700	*	10.98	127300	*	10.98	127300	*	10.99	126500
26	3	7	18	7	DFS	SB	*	19.58	19500	*	19.58	19400	*	19.58	19500	*	19.58	19400
26	3	7	18	7	BEA	MI	*	27.99	209200	*	17.15	202200	*	27.99	212300	*	27.99	212500
26	3	7	18	7	BB	PC	*	10.93	133800	*	10.94	133400	*	10.95	131800	*	10.94	132600
26	3	7	18	7	BEA	RPC	*	14.37	112900	*	14.37	113000	*	14.37	112100	*	14.37	112700
26	3	7	18	7	BE	MI	*	15.75	160000	*	15.75	161100	*	15.75	161200	*	15.75	161800
26	3	7	18	7	DFS	PC	*	44.92	228200	*	44.92	224700	*	44.92	225300	*	44.92	227200
26	3	7	18	7	BE	RPC	*	9.73	118300	*	9.74	117600	*	9.73	117600	*	9.75	116800

Tabla B&B (1 de 5)

n	m	Q	$\|U\|$	$\|W\|$	nods	vars	t_{P0}	g_{P0}	nod_{P0}	t_{P1}	g_{P1}	nod_{P1}	t_{P2}	g_{P2}	nod_{P2}	t_{P3}	g_{P3}	nod_{P3}
26	3	10	25	0	BE	MI	*	6.40	123700	*	6.40	124900	*	6.40	124900	*	6.40	124900
26	3	10	25	0	DFS	PC	1199	0.00	146400	*	27.86	146700	*	27.86	146300	*	27.86	146700
26	3	10	25	0	BE	RPC	*	4.16	104600	*	4.17	103900	*	4.16	104200	*	4.17	103900
26	3	10	25	0	BB	MI	*	6.40	123400	*	6.40	125200	*	6.40	125200	*	6.40	124600
26	3	10	25	0	BEA	SB	*	9.52	13000	*	9.52	13000	*	9.52	13000	*	9.52	13100
26	3	10	25	0	BB	RPC	*	4.16	104800	*	4.16	104200	*	4.16	104200	*	4.16	104200
26	3	10	25	0	DFS	MI	*	*	104600	*	*	104800	*	*	104600	*	*	104300
26	3	10	25	0	BE	SB	*	5.70	10200	*	5.70	10100	*	5.70	10100	*	5.70	10100
26	3	10	25	0	DFS	RPC	*	18.77	124700	*	18.77	124200	*	18.77	124700	*	18.77	124700
26	3	10	25	0	BEA	PC	*	11.11	114500	*	11.11	113700	*	11.11	113800	*	11.11	113300
26	3	10	25	0	BB	SB	*	5.70	10100	*	5.70	10100	*	5.70	10100	*	5.70	10100
26	3	10	25	0	BE	PC	*	5.00	109000	*	5.00	109200	*	5.00	109400	*	5.01	108900
26	3	10	25	0	DFS	SB	*	11.11	16300	*	11.11	16300	*	11.11	16300	*	11.11	16300
26	3	10	25	0	BEA	MI	*	8.18	111300	*	13.18	111200	1199	0.00	111700	*	98.50	111700
26	3	10	25	0	BB	PC	*	5.00	109300	*	5.00	109200	*	5.00	109200	*	5.00	109600
26	3	10	25	0	BEA	RPC	*	6.71	111100	*	6.71	110900	*	6.71	111100	*	6.71	111100
26	4	4	12	13	BE	PC	216	0.00	48300	215	0.00	48300	216	0.00	48300	216	0.00	48300
26	4	4	12	13	DFS	SB	730	0.00	20300	732	0.00	20300	733	0.00	20300	733	0.00	20300
26	4	4	12	13	BEA	MI	*	12.84	330100	*	12.84	332800	*	12.84	334700	*	12.84	333200
26	4	4	12	13	BB	PC	217	0.00	48300	217	0.00	48300	216	0.00	48300	217	0.00	48300
26	4	4	12	13	BEA	RPC	126	0.00	25000	127	0.00	25000	126	0.00	25000	126	0.00	25000
26	4	4	12	13	BE	MI	*	9.13	268100	*	9.13	269200	*	9.12	269400	*	9.13	267300
26	4	4	12	13	DFS	PC	395	0.00	87000	396	0.00	87000	394	0.00	87000	395	0.00	87000
26	4	4	12	13	BE	RPC	144	0.00	27700	143	0.00	27700	143	0.00	27700	143	0.00	27700
26	4	4	12	13	BB	MI	*	9.13	267700	*	9.13	269100	*	9.12	269300	*	9.13	267700
26	4	4	12	13	BEA	SB	344	0.00	9700	345	0.00	9700	344	0.00	9700	345	0.00	9700
26	4	4	12	13	BB	RPC	143	0.00	27700	142	0.00	27700	144	0.00	27700	143	0.00	27700
26	4	4	12	13	DFS	MI	*	14.47	360000	*	14.47	360000	*	14.47	360400	*	14.47	358800
26	4	4	12	13	BE	SB	318	0.00	9100	317	0.00	9100	318	0.00	9100	317	0.00	9100
26	4	4	12	13	DFS	RPC	227	0.00	44500	227	0.00	44500	227	0.00	44500	227	0.00	44500
26	4	4	12	13	BEA	PC	228	0.00	50200	228	0.00	50200	227	0.00	50200	227	0.00	50200
26	4	4	12	13	BB	SB	317	0.00	9100	318	0.00	9100	317	0.00	9100	317	0.00	9100

Tabla B&B (2 de 5)

n	m	Q	$\|U\|$	$\|W\|$	nods	vars	t_{P0}	g_{P0}	nod_{P0}	t_{P1}	g_{P1}	nod_{P1}	t_{P2}	g_{P2}	nod_{P2}	t_{P3}	g_{P3}	nod_{P3}
26	4	5	18	7	DFS	RPC	*	33.09	156200	*	33.09	156300	*	33.09	157100	*	33.09	157100
26	4	5	18	7	BEA	PC	*	21.02	155700	*	21.02	155400	*	21.02	155000	*	21.02	156600
26	4	5	18	7	BB	SB	*	18.01	11600	*	18.01	11600	*	18.01	11600	*	18.01	11600
26	4	5	18	7	BE	PC	*	19.39	158300	*	19.40	158200	*	19.39	158900	*	19.39	159400
26	4	5	18	7	DFS	SB	*	22.04	21800	*	22.04	21600	*	22.04	21800	*	22.04	21800
26	4	5	18	7	BEA	MI	*	22.00	204100	*	22.00	202200	*	22.00	201500	*	22.00	202200
26	4	5	18	7	BB	PC	*	19.40	158000	*	19.39	158900	*	19.39	158500	*	19.41	156600
26	4	5	18	7	BEA	RPC	*	19.32	99800	*	19.32	99400	*	19.32	99600	*	19.32	99800
26	4	5	18	7	BE	MI	*	22.81	148700	*	22.81	146900	*	22.81	146600	*	22.81	146800
26	4	5	18	7	DFS	PC	*	39.65	180100	*	39.65	180000	*	39.65	178800	*	38.71	177900
26	4	5	18	7	BE	RPC	*	14.46	104300	*	14.47	103600	*	14.47	103600	*	14.47	103500
26	4	5	18	7	BB	MI	*	17.35	140800	*	22.81	147400	*	22.81	147000	*	22.81	146700
26	4	5	18	7	BEA	SB	*	22.08	15000	*	22.08	14800	*	22.08	15000	*	22.08	15000
26	4	5	18	7	BB	RPC	*	14.46	104300	*	14.52	100300	*	14.47	103600	*	14.47	103400
26	4	5	18	7	DFS	MI	*	99.72	153100	*	99.72	153700	*	99.72	154200	*	99.72	154300
26	4	5	18	7	BE	SB	*	18.00	11600	1199	0.00	11600	*	18.00	11600	*	18.00	11600
26	4	7	25	0	BB	MI	*	12.98	130000	*	12.98	129300	*	12.98	129600	*	12.98	129600
26	4	7	25	0	BEA	SB	*	12.29	13400	*	12.29	13400	*	12.29	13400	*	12.29	13400
26	4	7	25	0	BB	RPC	254	0.00	85600	*	9.94	87100	*	9.94	86900	0	0.00	0
26	4	7	25	0	DFS	MI	*	*	120300	*	*	119600	*	*	119600	*	*	120100
26	4	7	25	0	BE	SB	*	10.85	10800	*	10.85	10800	*	10.85	10800	*	10.85	10800
26	4	7	25	0	DFS	RPC	*	22.75	158300	*	22.75	158700	*	22.75	157900	*	22.75	158000
26	4	7	25	0	BEA	PC	*	12.25	120700	*	12.25	121100	*	12.25	121300	*	12.25	121800
26	4	7	25	0	BB	SB	*	10.85	10900	*	10.85	10900	*	10.85	10800	*	10.85	10900
26	4	7	25	0	BE	PC	*	9.77	108000	*	9.76	108700	*	9.76	108300	*	9.76	108300
26	4	7	25	0	DFS	SB	*	13.75	17900	*	14.44	17900	*	13.75	17900	*	14.44	17900
26	4	7	25	0	BEA	MI	*	17.13	178800	*	17.13	177700	*	17.13	178100	*	17.13	178700
26	4	7	25	0	BB	PC	*	9.76	108600	*	9.76	108700	*	9.76	108600	*	9.76	108700
26	4	7	25	0	BEA	RPC	*	12.89	91100	*	12.89	90700	*	12.89	90700	*	12.89	91000
26	4	7	25	0	BE	MI	*	12.98	129900	*	12.98	129900	*	12.98	129200	*	12.98	129900
26	4	7	25	0	DFS	PC	*	19.80	157200	*	19.80	157400	*	19.80	156700	*	19.80	155800
26	4	7	25	0	BE	RPC	*	9.94	87000	*	9.94	87100	*	9.94	87300	*	9.94	87800

Tabla B&B (3 de 5)

n	m	Q	$\|U\|$	$\|W\|$	nods	vars	t_{P0}	g_{P0}	nod_{P0}	t_{P1}	g_{P1}	nod_{P1}	t_{P2}	g_{P2}	nod_{P2}	t_{P3}	g_{P3}	nod_{P3}
26	5	3	12	13	BE	MI	*	10.30	251700	*	10.31	250800	*	10.32	248300	*	10.32	249300
26	5	3	12	13	DFS	PC	*	20.39	270300	*	20.39	269000	*	20.39	272400	*	20.39	271600
26	5	3	12	13	BE	RPC	411	0.00	74000	412	0.00	74000	415	0.00	74000	413	0.00	74000
26	5	3	12	13	BB	MI	*	10.30	251300	*	10.32	249200	*	10.31	250100	*	10.31	249700
26	5	3	12	13	BEA	SB	897	0.00	21000	896	0.00	21000	896	0.00	21000	898	0.00	21000
26	5	3	12	13	BB	RPC	411	0.00	74000	412	0.00	74000	414	0.00	74000	412	0.00	74000
26	5	3	12	13	DFS	MI	*	24.53	338600	*	24.53	342500	*	24.53	343000	*	24.53	343100
26	5	3	12	13	BE	SB	829	0.00	19600	829	0.00	19600	830	0.00	19600	828	0.00	19600
26	5	3	12	13	DFS	RPC	473	0.00	88100	476	0.00	88100	476	0.00	88100	475	0.00	88100
26	5	3	12	13	BEA	PC	*	8.71	239600	*	8.71	240000	*	8.71	240800	*	8.71	241100
26	5	3	12	13	BB	SB	834	0.00	19600	830	0.00	19600	829	0.00	19600	832	0.00	19600
26	5	3	12	13	BE	PC	*	4.18	240800	*	4.16	242500	*	4.16	241900	*	4.16	242500
26	5	3	12	13	DFS	SB	1119	0.00	27000	1120	0.00	27000	1119	0.00	27000	1121	0.00	27000
26	5	3	12	13	BEA	MI	*	16.61	324000	*	16.61	322400	*	16.61	309700	*	16.61	320300
26	5	3	12	13	BB	PC	*	4.18	240100	*	4.15	242500	*	4.15	242500	*	4.15	242500
26	5	3	12	13	BEA	RPC	356	0.00	61500	354	0.00	61500	354	0.00	61500	355	0.00	61500
26	5	4	18	7	BE	PC	*	21.81	159800	*	21.81	159800	*	21.81	158800	*	21.81	159800
26	5	4	18	7	DFS	SB	*	24.86	20500	*	24.86	20500	*	24.86	20500	*	24.86	20500
26	5	4	18	7	BEA	MI	*	32.27	200600	*	23.51	189700	*	31.00	199200	*	32.27	201400
26	5	4	18	7	BB	PC	*	21.81	159500	*	21.81	159800	*	21.81	159400	*	21.81	159500
26	5	4	18	7	BEA	RPC	*	20.00	107600	*	20.00	107300	*	20.00	107600	*	20.00	107600
26	5	4	18	7	BE	MI	*	19.86	149300	*	19.86	148300	*	19.86	148600	*	19.87	140600
26	5	4	18	7	DFS	PC	*	28.30	198000	*	28.30	197800	*	28.30	194600	*	28.30	198400
26	5	4	18	7	BE	RPC	*	15.73	109200	*	15.75	108700	*	15.74	109100	*	15.73	109200
26	5	4	18	7	BB	MI	*	19.86	150000	*	19.86	149100	*	19.86	148000	*	19.86	148300
26	5	4	18	7	BEA	SB	*	21.67	13700	*	21.67	13700	*	21.67	13700	*	21.67	13600
26	5	4	18	7	BB	RPC	*	15.73	109200	*	15.73	109300	*	15.74	108900	*	15.73	109400
26	5	4	18	7	DFS	MI	*	99.63	160400	*	99.63	160400	*	99.63	160300	*	99.63	159200
26	5	4	18	7	BE	SB	*	16.77	11100	*	16.77	11100	*	16.77	11100	*	16.77	11100
26	5	4	18	7	DFS	RPC	*	25.63	194000	*	25.63	192500	*	25.63	195200	*	25.63	195000
26	5	4	18	7	BEA	PC	*	23.51	162800	*	23.51	161600	*	23.51	162000	*	23.51	162300
26	5	4	18	7	BB	SB	*	16.77	11100	*	16.77	11100	*	16.77	11100	*	16.77	11100

Tabla B&B (4 de 5)

n	m	Q	$\|U\|$	$\|W\|$	nods	vars	t_{P0}	g_{P0}	nod_{P0}	t_{P1}	g_{P1}	nod_{P1}	t_{P2}	g_{P2}	nod_{P2}	t_{P3}	g_{P3}	nod_{P3}
26	5	6	25	0	DFS	RPC	*	18.15	134700	*	18.15	134500	*	18.15	134500	*	18.15	134700
26	5	6	25	0	BEA	PC	*	10.54	111000	*	10.54	110700	*	10.54	110500	*	10.54	111500
26	5	6	25	0	BB	SB	*	12.15	9800	*	12.15	9800	*	12.15	9800	*	12.15	9800
26	5	6	25	0	BE	PC	*	11.17	114400	*	11.17	114100	*	11.17	114500	*	11.17	115200
26	5	6	25	0	DFS	SB	*	15.93	16800	*	15.93	16800	*	15.93	16700	*	15.93	16600
26	5	6	25	0	BEA	MI	*	20.31	164900	*	20.31	164900	*	20.31	163800	*	20.31	164900
26	5	6	25	0	BB	PC	*	11.17	114400	*	11.17	114300	*	11.17	114500	*	11.17	115100
26	5	6	25	0	BEA	RPC	*	12.19	87600	*	12.19	90100	*	12.19	90000	*	12.19	90700
26	5	6	25	0	BE	MI	*	11.84	126200	*	11.84	126700	*	11.84	126300	*	11.84	126300
26	5	6	25	0	DFS	PC	*	98.32	106000	*	98.32	106000	*	98.32	106400	*	98.32	106400
26	5	6	25	0	BE	RPC	*	9.22	94400	*	9.21	94600	*	9.21	95100	*	9.21	94600
26	5	6	25	0	BB	MI	*	11.84	126800	*	11.84	126600	*	11.84	125900	1199	0.00	126200
26	5	6	25	0	BEA	SB	*	14.23	11200	*	14.23	11000	*	14.23	11000	*	14.23	11400
26	5	6	25	0	BB	RPC	*	9.21	94600	*	9.21	94600	*	9.22	94100	*	9.21	95100
26	5	6	25	0	DFS	MI	*	22.67	209700	*	22.67	210100	*	22.67	209100	*	22.67	209700
26	5	6	25	0	BE	SB	*	12.15	9800	*	12.15	9800	*	12.15	9800	*	12.15	9800

Tabla B&B (5 de 5)

4.3. Familias de planos de corte

En esta sección se evaluan las distintas familias de cortes con sus algoritmos de separación, con el objetivo de establecer que combinación de estas familias resulta favorable para la resolución del modelo planteado, para las instancias BDS.

En la *Tabla B&C* se observa que la combinación de las familias de cortes *capacidad* y *suma uno* resultan en promedio más favorables frente a las otras alternativas.

Las figuras 4.2, 4.2 y 4.2 presentan para cada combinación de familias de cortes (C: capacidad, A: anillo o estrella y U: suma uno), un resúmen con el tiempo promedio de resolución, la cantidad de instancias resueltas en 1800 segundos y el %gap promedio.

4.4. Parámetros de branch-and-cut

Una vez analizadas las estrategias de recorrido y generación del árbol, y la eficiencia para distintas combinaciones de desigualdades válidas, y las familias de cortes que se utilizarán, el siguiente paso es analizar cuando aplicar cortes y por cuantas iteraciones.

Una vez resulta la relajación lineal de un nodo del árbol, se debe decidir si se generan cortes o bien se procede a realizar el branching. Es de esperar que los planos de corte ayuden a mejorar las cotas y esto permita podar ramas del árbol. Pero al mismo tiempo, el proceso de búsqueda de desigualdades violadas y la posterior resolución de la relajación tiene un costo. Por lo tanto, es dentro del esquema de branch-and-cut, se debe encontrar un equilibrio entre estas dos posibilidades. Para tratar esta decisión se utilizan los parámetros *IPC* y *skip factor*.

Dada la efectividad de los cortes analizada en la sección anterior, para el nodo raíz no aplicaremos estos parámetros, permitiendo un máximo de 30 iteraciones. Los valores de estos parámetros no son fáciles de determinar.

Los resultados obtenidos, que se presentan en la tabla *Parámetros de cortes*, no muestran una evidencia, en general, en favor de alguna combinación de valores de *IPC* y *skip factor*. Esto puede deberse a la cantidad de cortes producidos para cada familia de desigualdades en relación a la cantidad de nodos explorados como puede verse en la otra tabla. Esto puede producir que al reducirse el valor de *IPC* o *skip factor*, estos cortes se postergan realizandose en nodos descendientes a los que se aplicarían con un mayor valor de *IPC* o menor valor de *skip factor*.

nod_{SL}: cantidad de nodos explorados, sin límite en la cantidad de iteraciones por nodo (IPC)

t_{SL}: tiempo de resolución, sin límite en la cantidad de iteraciones por nodo

gap_{SL}: gap obtenido, sin límite en la cantidad de iteraciones por nodo (si el gap es cero, se comprobó que la solución obtenida es óptima)

nod_k: cantidad de nodos explorados, con $IPC = k$

t_k: tiempo de resolución, con $IPC = k$

gap_k: gap obtenido, con $IPC = k$

c_{SL}: cantidad de cortes de la familia *capacidad* aplicados, sin límite en la cantidad de iteraciones por nodo (IPC)

n	$\|U\|$	$\|W\|$	m	Q	nod_C	t_C	nod_A	t_A	nod_U	t_U	nod_{CA}	t_{CA}	nod_{CU}	t_{CU}	nod_{AU}	t_{AU}	nod_{CAU}	t_{CAU}
26	12	13	3	5	1015	14	27424	134	48958	231	856	14	954	14	27424	136	856	14
26	12	13	4	4	315	6	18010	87	21908	105	481	9	192	4	18010	87	481	9
26	12	13	5	3	245	4	74830	406	51707	284	271	5	230	4	74830	405	271	5
26	18	7	3	7	1467	30	179494	*	181518	*	2014	38	3660	77	181777	*	2014	39
26	18	7	4	5	2332	51	151291	*	160909	*	5780	119	5341	111	150382	*	5780	120
26	18	7	5	4	5346	100	156611	*	147625	*	10925	222	15357	361	156585	*	10925	221
26	25	0	3	10	464	11	160077	*	173862	*	1750	35	1373	26	168930	*	1517	27
26	25	0	4	7	2568	57	136515	*	137996	*	2326	47	2731	61	129632	*	3340	84
26	25	0	5	6	1861	35	149721	*	143216	*	2257	50	2351	51	144198	*	849	21
51	12	38	3	5	12776	1280	112556	*	118958	*	19089	*	14936	1290	118523	*	19136	*
51	12	38	4	4	7587	922	112966	*	121360	*	12535	*	5180	542	106301	*	12932	*
51	12	38	5	3	2897	424	104601	*	113429	*	3373	341	6150	776	104324	*	3373	348
51	25	25	3	10	16266	*	102124	*	110621	*	15048	*	14287	*	104324	*	12553	*
51	25	25	4	7	13224	*	101063	*	105822	*	11348	*	11075	*	102181	*	9862	*
51	25	25	5	6	10040	*	100841	*	102341	*	13532	*	10946	*	100243	*	10814	*

Tabla B&C

nod_C/t_C: cantidad de nodos explorados/tiempo de resolución aplicando *cortes capacidad*

nod_A/t_A: cantidad de nodos explorados/tiempo de resolución aplicando *cortes anillo ó estrella*

nod_U/t_U: cantidad de nodos explorados/tiempo de resolución aplicando *cortes suma uno*

nod_{CA}/t_{CA}: cantidad de nodos explorados/tiempo de resolución aplicando *cortes capacidad* y *cortes anillo ó estrella*

nod_{CU}/t_{CU}: cantidad de nodos explorados/tiempo de resolución aplicando *cortes capacidad* y *cortes suma uno*

nod_{AU}/t_{AU}: cantidad de nodos explorados/tiempo de resolución aplicando *cortes anillo ó estrella* y *cortes suma uno*

nod_{CAU}/t_{CAU}: cantidad de nodos explorados/tiempo de resolución aplicando *cortes capacidad*, *cortes anillo ó estrella* y *cortes suma uno*

El tiempo de resolución se indica en segundos. En los casos que aparece *, indica que no se resolvió la instancia en 1800 segundos.

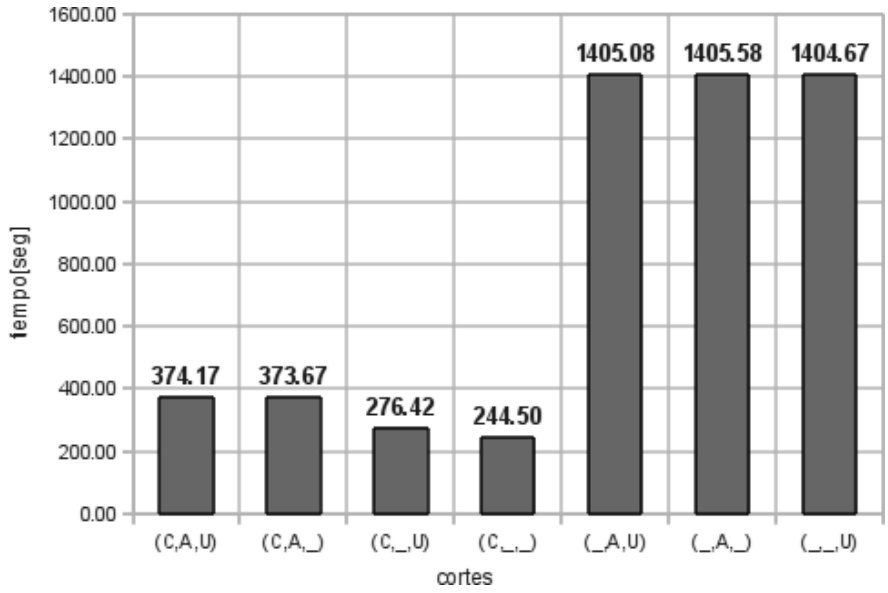

Figura 4.2: *Tiempo promedio para cada combinación de familias de cortes*

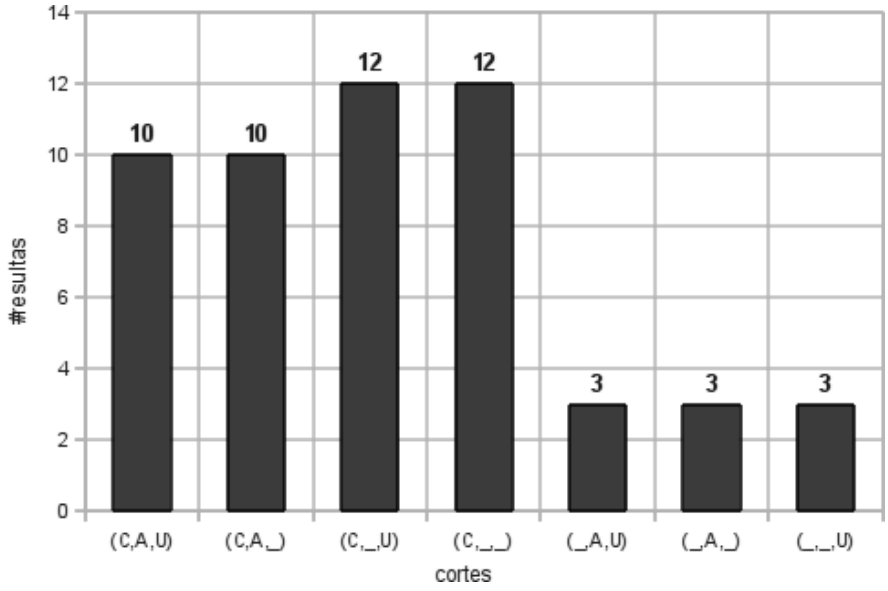

Figura 4.3: *Cantidad de instancias resueltas para cada combinación de familias de cortes*

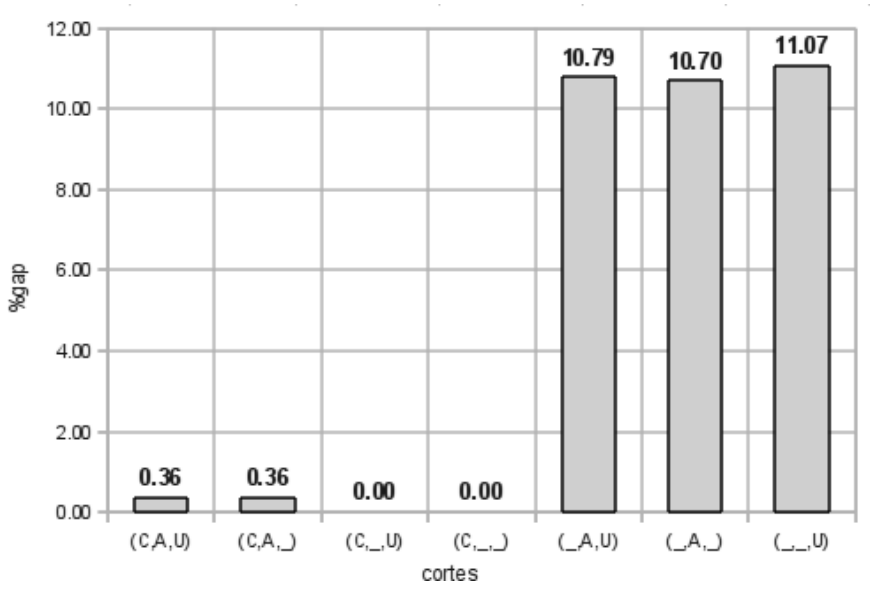

Figura 4.4: *%Gap promedio para cada combinación de familias de cortes*

u_{SL}: cantidad de cortes de la familia *suma uno* aplicados, sin límite en la cantidad de iteraciones por nodo (IPC)

c_k: cantidad de cortes de la familia *capacidad* aplicados, con $IPC = k$

u_k: cantidad de cortes de la familia *suma uno* aplicados, con $IPC = k$

n	m	Q	sf	nod_{SL}	t_{SL}	gap_{SL}	nod_1	t_1	gap_1	nod_2	t_2	gap_2	nod_3	t_3	gap_3	nod_4	t_4	gap_4
26	3	5	1	954	22	0.00	594	16	0.00	954	26	0.00	954	26	0.00	954	25	0.00
26	3	5	2	1156	28	0.00	1104	30	0.00	1040	27	0.00	1156	26	0.00	1156	29	0.00
26	3	5	3	1659	28	0.00	1124	24	0.00	775	15	0.00	1659	27	0.00	1659	27	0.00
26	3	5	4	2045	43	0.00	2045	42	0.00	2045	39	0.00	2045	42	0.00	2045	42	0.00
26	3	7	1	3660	126	0.00	3660	129	0.00	3660	129	0.00	3660	130	0.00	3660	130	0.00
26	3	7	2	4294	122	0.00	4294	124	0.00	4294	125	0.00	4294	124	0.00	4294	124	0.00
26	3	7	3	4850	149	0.00	2735	92	0.00	4850	152	0.00	4850	155	0.00	4850	151	0.00
26	3	7	4	3915	118	0.00	6667	217	0.00	3377	94	0.00	3915	120	0.00	3915	117	0.00
26	3	10	1	1373	43	0.00	1373	43	0.00	1373	42	0.00	1373	45	0.00	1373	44	0.00
26	3	10	2	1536	39	0.00	1536	42	0.00	1536	40	0.00	1536	43	0.00	1536	39	0.00
26	3	10	3	1050	34	0.00	1130	32	0.00	1920	46	0.00	1050	37	0.00	1050	36	0.00
26	3	10	4	1575	43	0.00	1575	45	0.00	1575	46	0.00	1575	46	0.00	1575	44	0.00
26	4	4	1	192	8	0.00	203	9	0.00	111	5	0.00	201	9	0.00	176	7	0.00
26	4	4	2	581	13	0.00	663	17	0.00	981	23	0.00	581	14	0.00	581	15	0.00
26	4	4	3	572	13	0.00	478	11	0.00	497	12	0.00	173	6	0.00	650	14	0.00
26	4	4	4	270	7	0.00	674	16	0.00	987	22	0.00	646	15	0.00	270	6	0.00
26	4	5	1	5341	183	0.00	6613	242	0.00	2348	102	0.00	4582	182	0.00	3601	128	0.00
26	4	5	2	3586	137	0.00	5981	161	0.00	4083	141	0.00	3213	119	0.00	3586	140	0.00
26	4	5	3	5916	210	0.00	5402	169	0.00	11537	358	0.00	5916	200	0.00	5916	199	0.00
26	4	5	4	4122	147	0.00	6975	232	0.00	4122	145	0.00	4122	140	0.00	4122	140	0.00
26	4	7	1	2731	99	0.00	2526	86	0.00	2595	95	0.00	2731	99	0.00	2731	102	0.00
26	4	7	2	3702	122	0.00	3702	125	0.00	3702	121	0.00	3702	122	0.00	3702	123	0.00
26	4	7	3	8733	382	0.00	8733	374	0.00	8733	386	0.00	8733	392	0.00	8733	385	0.00
26	4	7	4	3026	95	0.00	3026	98	0.00	3026	98	0.00	3026	94	0.00	3026	94	0.00
26	5	3	1	230	6	0.00	245	9	0.00	194	6	0.00	235	7	0.00	232	7	0.00
26	5	3	2	664	17	0.00	506	15	0.00	490	13	0.00	783	18	0.00	413	11	0.00
26	5	3	3	521	17	0.00	632	11	0.00	476	13	0.00	390	12	0.00	507	12	0.00
26	5	3	4	839	17	0.00	873	18	0.00	673	15	0.00	599	14	0.00	733	17	0.00
26	5	4	1	11665	477	0.00	11665	474	0.00	11665	476	0.00	11665	472	0.00	11665	473	0.00
26	5	4	2	9495	394	0.00	13553	453	0.00	9495	400	0.00	9495	397	0.00	9495	405	0.00
26	5	4	3	13034	552	0.00	21516	925	0.00	13034	547	0.00	13034	545	0.00	13034	552	0.00
26	5	4	4	18275	583	0.00	18275	584	0.00	18275	584	0.00	18275	589	0.00	18275	581	0.00
26	5	6	1	2351	87	0.00	2351	85	0.00	2351	86	0.00	2351	86	0.00	2351	88	0.00
26	5	6	2	1472	47	0.00	1581	59	0.00	1472	47	0.00	1472	45	0.00	1472	42	0.00
26	5	6	3	3685	137	0.00	2930	92	0.00	3398	104	0.00	4146	159	0.00	3685	138	0.00
26	5	6	4	2222	63	0.00	2222	59	0.00	2222	57	0.00	2222	59	0.00	2222	60	0.00

Tabla Parámetros de cortes

n	m	Q	sf	c_{SL}	u_{SL}	c_1	u_1	c_2	u_2	c_3	u_3	c_4	u_4
26	3	5	1	100	10	128	6	100	10	100	10	100	10
26	3	5	2	114	6	151	10	123	10	114	6	114	6
26	3	5	3	77	4	85	14	84	4	77	4	77	4
26	3	5	4	86	12	86	12	86	12	86	12	86	12
26	3	7	1	137	36	137	36	137	36	137	36	137	36
26	3	7	2	116	48	116	48	116	48	116	48	116	48
26	3	7	3	114	44	123	44	114	46	114	44	114	44
26	3	7	4	124	38	122	52	91	40	124	38	124	38
26	3	10	1	90	46	90	46	90	46	90	46	90	46
26	3	10	2	84	32	84	32	84	32	84	32	84	32
26	3	10	3	58	38	65	34	84	40	58	38	58	38
26	3	10	4	49	44	49	44	49	44	49	44	49	44
26	4	4	1	54	2	74	6	59	2	63	2	57	2
26	4	4	2	110	8	118	4	137	8	110	8	110	8
26	4	4	3	107	0	84	4	78	4	65	4	105	2
26	4	4	4	47	4	70	2	118	2	92	4	47	4
26	4	5	1	252	70	184	62	269	48	185	58	202	62
26	4	5	2	202	36	176	50	196	42	168	44	202	36
26	4	5	3	189	58	272	62	157	70	189	58	189	58
26	4	5	4	129	58	203	82	129	68	129	58	129	58
26	4	7	1	151	56	224	60	153	58	151	56	151	56
26	4	7	2	126	58	126	58	126	58	126	58	126	58
26	4	7	3	169	92	169	92	169	92	169	92	169	92
26	4	7	4	97	42	97	42	97	42	97	42	97	42
26	5	3	1	79	2	107	0	83	0	83	0	85	0
26	5	3	2	113	4	133	2	112	6	125	6	104	2
26	5	3	3	139	6	86	18	122	6	88	6	118	6
26	5	3	4	103	4	101	0	79	0	95	4	110	2
26	5	4	1	179	78	179	78	179	78	179	78	179	78
26	5	4	2	142	80	157	68	142	80	142	80	142	80
26	5	4	3	208	66	200	76	208	66	208	66	208	66
26	5	4	4	175	80	175	82	175	80	175	80	175	80
26	5	6	1	187	70	187	70	187	70	187	70	187	70
26	5	6	2	90	40	118	50	90	40	90	40	90	40
26	5	6	3	125	66	120	64	158	74	166	66	125	66
26	5	6	4	73	46	73	46	73	46	73	46	73	46

4.5. Heurística inicial

En esta sección se evalua la efectividad de la heurística inicial. Para ello se comparan la cantidad de nodos explorados, el tiempo de resolución y el gap obtenido en instancias *BDS*, aplicando la heurística y sin aplicar la heurística. Además se presenta el gap obtenido por la heurística inicial para tener una idea de la efectividad de la misma.

En las figuras 4.5 y 4.6 se pueden ver la cantidad de nodos explorados y el tiempo de resolución de las instancias evaluadas aplicando la heurística inicial vs la resolución sin aplicar la heurística inicial.

Los valores que se presentan en la *Tabla Heurística inicial* son los siguientes:

nod: cantidad de nodos explorados sin aplicar la heurística inicial

tiempo: tiempo de resolución sin aplicar la heurística inicial

gap: gap obtenido sin aplicar la heurística inicial

gap_{ini_H}: gap (inicial) obtenido al aplicar la heurística inicial

nod_H: cantidad de nodos explorados aplicando la heurística inicial

$tiempo_H$: tiempo de resolución aplicando la heurística inicial

gap_H: gap obtenido aplicando la heurística inicial

n	m	Q	$\|U\|$	$\|W\|$	nod	$tiempo$	gap	gap_{ini_H}	nod_H	$tiempo_H$	gap_H
26	3	5	12	13	954	23	0.00	35.07	696	18	0.00
26	4	4	12	13	192	7	0.00	30.05	434	16	0.00
26	5	3	12	13	230	7	0.00	35.13	366	12	0.00
26	3	7	18	7	3660	124	0.00	32.71	5424	212	0.00
26	4	5	18	7	5341	181	0.00	38.03	5563	234	0.00
26	5	4	18	7	11665	458	0.00	35.82	5609	243	0.00
26	3	10	25	0	1373	42	0.00	36.45	747	33	0.00
26	4	7	25	0	2731	99	0.00	37.54	1101	41	0.00
26	5	6	25	0	2351	86	0.00	31.60	1179	50	0.00

Debido a la rápida resolución de instancias con pocos clientes (los casos de 12 clientes y algunos casos de 18 clientes) sin aplicar la heurística inicial, no se justifica la aplicación de la misma. En instancias con mayor cantidad de clientes, los resultados son favorables, con un %gap inicial promedio menor al 35 %.

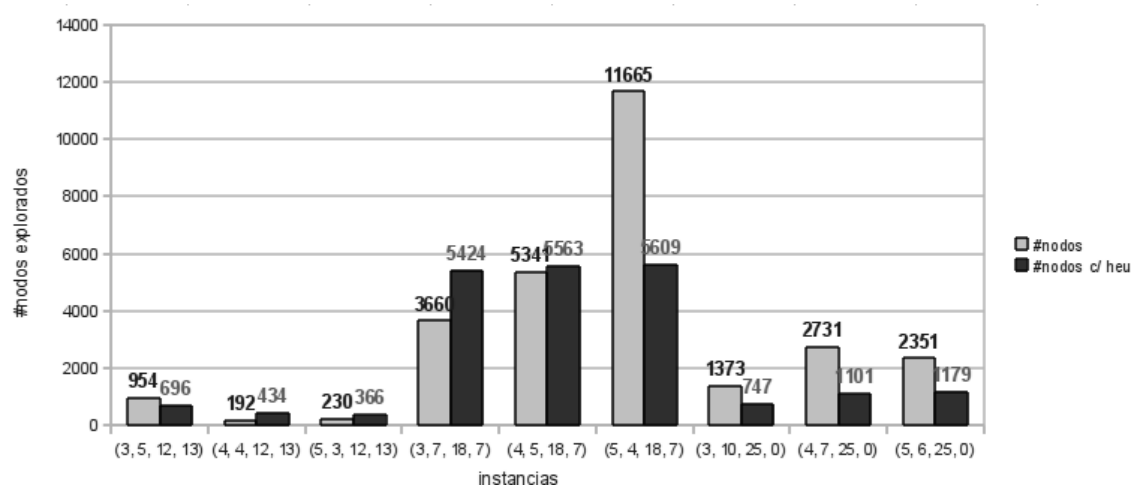

Figura 4.5: *Cantidad de nodos explorados en la resolución sin aplicar la heurística vs resolución aplicando la heurística inicial*

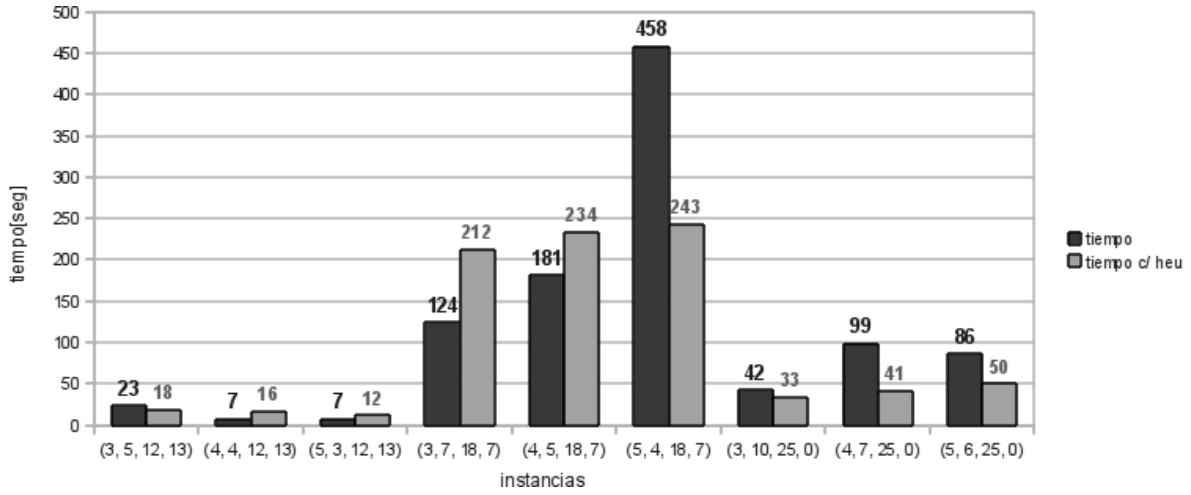

Figura 4.6: *Tiempo de resolución sin aplicar la heurística vs resolución aplicando la heurística inicial*

4.6. Variación de parámetros del problema

Para una configuración fija de clientes, nodos de *Steiner* y depósito, fijando matrices de costos de conexiones anillo y estrella, se obtienen distintos tiempos de ejecución que resultan de variar los parámetros m y Q del problema. Las instancias E permiten realizar este análisis debido a la cantidad de instancias construidas para cada configuración. Se compara también el algoritmo *BC* frente a la resolución de CPLEX con sus parámtros por defecto.

Se muestra en la siguiente tabla los resultados agrupados por n ($n = |U| + |W|$). Como es de esperar, a medida que aumenta n, aumentan t_{prom} y $\%gap$, mientras que $\%res$ disminuye.

n	tot	res_{CPX}	$\%res_{CPX}$	t_{CPX}	$\%gap_{CPX}$	res_{BC}	$\%res_{BC}$	t_{BC}	$\%gap_{BC}$
8	120	120	100.00	1.18	0.00	120	100.00	0.50	0.00
9	140	140	100.00	6.14	0.00	140	100.00	0.76	0.00
10	280	280	100.00	12.12	0.00	280	100.00	1.56	0.00
11	300	290	96.67	55.43	0.35	299	99.67	13.83	0.08
12	460	436	94.78	100.19	0.52	460	100.00	12.93	0.00
13	500	449	89.80	123.38	1.05	499	99.80	28.90	0.01
14	560	402	71.79	190.97	3.14	552	98.57	55.09	0.18
15	360	250	69.44	275.12	3.63	360	100.00	64.12	0.00
16	400	176	44.00	228.01	8.17	375	93.75	189.05	0.62
17	200	60	30.00	372.03	11.55	174	87.00	171.30	1.35
18	220	52	23.64	370.63	13.68	161	73.18	274.09	3.90

En la siguiente tabla se presentan los resultados agrupados por $|U|$ y $|W|$ (se muestra además el valor de n). Se puede observar que para un valor de n fijo, el tiempo de ejecución aumenta cuanto mayor es la cantidad de clientes. Es decir, agregar clientes a una instancia afecta a t_{prom} en mayor medida que agregar nodos de Steiner.

Si bien el tiempo promedio de resolución de BC se acerca cada vez más al de CPLEX en la tabla, se debe tener en cuenta que este tiempo está considerado sobre las instancias resueltas. Con lo cual, en el análisis, el tiempo promedio de resolución debe ser tenido en cuenta junto con la cantidad o el porcentaje de instancias resueltas al comparar los algoritmos.

Los valores presentados en la tabla son los siguientes:

tot: cantidad de instancias a resolver

res_{CPX}: cantidad de instancias resueltas por CPLEX

res_{BC}: cantidad de instancias resueltas por BC

res_{CPX}: porcentaje de instancias resueltas por CPLEX

res_{BC}: porcentaje de instancias resueltas por BC

t_{CPX}: promedio de tiempo, expresado en segundos, de resolución con CPLEX (considerando las instancias resueltas)

t_{BC}: promedio de tiempo, expresado en segundos, de resolución con BC (considerando las instancias resueltas)

$\%gap_{CPX}$: promedio de gap obtenido por CPLEX (el valor es cero si se encontró que la solución es óptima)

$\%gap_{BC}$: promedio de gap obtenido por BC (el valor es cero si se encontró que la solución es óptima)

n	$\|U\|$	$\|W\|$	tot	res_{CPX}	$\%res_{CPX}$	t_{CPX}	$\%gap_{CPX}$	res_{BC}	$\%res_{BC}$	t_{BC}	$\%gap_{BC}$
8	8	0	120	120	100.00	1.18	0.00	120	100.00	0.50	0.00
9	9	0	140	140	100.00	6.14	0.00	140	100.00	0.76	0.00
10	8	2	120	120	100.00	10.21	0.00	120	100.00	1.67	0.00
10	10	0	160	160	100.00	13.56	0.00	160	100.00	1.49	0.00
11	9	2	140	140	100.00	18.47	0.00	140	100.00	2.36	0.00
11	11	0	160	150	93.75	89.92	0.66	159	99.38	23.92	0.15
12	8	4	120	119	99.17	80.26	0.09	120	100.00	5.17	0.00
12	10	2	160	154	96.25	85.34	0.26	160	100.00	8.41	0.00
12	12	0	180	163	90.56	128.77	1.04	180	100.00	22.13	0.00
13	9	4	140	130	92.86	76.72	0.81	140	100.00	42.94	0.00
13	11	2	160	149	93.13	141.87	0.74	160	100.00	13.93	0.00
13	13	0	200	170	85.00	142.85	1.46	199	99.50	31.06	0.03
14	10	4	160	141	88.13	164.18	0.98	160	100.00	18.18	0.00
14	12	2	180	129	71.67	167.52	3.49	180	100.00	59.13	0.00
14	14	0	220	132	60.00	242.48	4.42	212	96.36	79.52	0.45
15	11	4	160	131	81.88	196.65	2.09	160	100.00	62.87	0.00
15	13	2	200	119	59.50	361.50	4.86	200	100.00	65.13	0.00
16	12	4	180	82	45.56	191.62	9.30	166	92.22	178.28	0.77
16	14	2	220	94	42.73	259.74	7.25	209	95.00	197.59	0.50
17	13	4	200	60	30.00	372.03	11.55	174	87.00	171.30	1.35
18	14	4	220	52	23.64	370.63	13.68	161	73.18	274.09	3.90

En la siguiente tabla se presenta para cada valor de $|U|$, distintas combinaciones de (m,Q). Se observa que, disminuye la densidad promedio de clientes por anillo $\frac{|U|}{mQ}$ (para un

valor de m fijo, a medida que aumenta aumenta Q), disminuye también t_{prom}.

$\|U\|$	m	Q	$\frac{\|U\|}{mQ}$	tot	res_{CPX}	$\%res_{CPX}$	t_{CPX}	$\%gap_{CPX}$	res_{BC}	$\%res_{BC}$	t_{BC}	$\%gap_{BC}$
8	2	4	1.00	60	60	100.00	73.65	0.00	60	100.00	4.88	0.00
8	2	5	0.80	60	59	98.33	40.05	0.18	60	100.00	3.07	0.00
8	2	6	0.67	60	60	100.00	12.70	0.00	60	100.00	1.77	0.00
8	2	7	0.57	60	60	100.00	4.25	0.00	60	100.00	0.70	0.00
8	3	3	0.89	60	60	100.00	38.45	0.00	60	100.00	2.67	0.00
8	4	2	1.00	60	60	100.00	13.53	0.00	60	100.00	1.58	0.00
9	2	5	0.90	60	58	96.67	83.66	0.57	60	100.00	42.02	0.00
9	2	6	0.75	60	58	96.67	35.36	0.41	60	100.00	10.67	0.00
9	2	7	0.64	60	59	98.33	11.27	0.24	60	100.00	6.97	0.00
9	2	8	0.56	60	59	98.33	3.69	0.09	60	100.00	4.28	0.00
9	3	3	1.00	60	57	95.00	61.79	0.34	60	100.00	18.10	0.00
9	3	4	0.75	60	59	98.33	31.12	0.23	60	100.00	24.40	0.00
9	5	2	0.90	60	60	100.00	4.57	0.00	60	100.00	1.05	0.00
10	2	5	1.00	60	50	83.33	159.98	1.56	60	100.00	23.73	0.00
10	2	6	0.83	60	56	93.33	155.88	0.68	60	100.00	10.20	0.00
10	2	7	0.71	60	59	98.33	112.64	0.03	60	100.00	7.93	0.00
10	2	8	0.63	60	60	100.00	31.85	0.00	60	100.00	4.68	0.00
10	2	9	0.56	60	60	100.00	18.80	0.00	60	100.00	2.10	0.00
10	3	4	0.83	60	57	95.00	124.44	0.37	60	100.00	13.50	0.00
10	4	3	0.83	60	56	93.33	49.59	0.44	60	100.00	7.17	0.00
10	5	2	1.00	60	57	95.00	38.23	0.22	60	100.00	5.55	0.00
11	2	6	0.92	60	51	85.00	255.00	2.31	60	100.00	41.97	0.00
11	2	7	0.79	60	51	85.00	131.86	1.55	59	98.33	16.92	0.41
11	2	8	0.69	60	57	95.00	147.86	0.63	60	100.00	10.20	0.00
11	2	9	0.61	60	58	96.67	62.12	0.20	60	100.00	33.95	0.00
11	2	10	0.55	60	60	100.00	21.65	0.00	60	100.00	2.37	0.00
11	3	4	0.92	60	45	75.00	219.56	2.66	60	100.00	58.57	0.00
11	3	5	0.73	60	57	95.00	154.68	0.78	60	100.00	20.63	0.00
11	4	3	0.92	60	51	85.00	169.24	1.16	60	100.00	83.88	0.00

$\|U\|$	m	Q	$\frac{\|U\|}{mQ}$	tot	res_{CPX}	$\%res_{CPX}$	t_{CPX}	$\%gap_{CPX}$	res_{BC}	$\%res_{BC}$	t_{BC}	$\%gap_{BC}$
12	2	6	1.00	60	33	55.00	200.45	6.86	58	96.67	111.14	0.35
12	2	7	0.86	60	39	65.00	223.15	5.71	58	96.67	90.59	0.48
12	2	8	0.75	60	41	68.33	171.61	4.48	60	100.00	54.07	0.00
12	2	9	0.67	60	46	76.67	112.76	3.11	60	100.00	40.93	0.00
12	2	10	0.60	60	53	88.33	65.08	0.85	60	100.00	12.25	0.00
12	2	11	0.55	60	59	98.33	54.41	0.16	60	100.00	2.67	0.00
12	3	4	1.00	60	29	48.33	298.86	8.98	56	93.33	195.45	0.62
12	3	5	0.80	60	39	65.00	150.51	4.61	58	96.67	69.76	0.25
12	4	3	1.00	60	35	58.33	273.57	6.72	56	93.33	195.29	0.60
13	2	7	0.93	60	24	40.00	388.67	9.96	54	90.00	92.76	0.77
13	2	8	0.81	60	27	45.00	384.26	8.35	56	93.33	95.73	0.61
13	2	9	0.72	60	29	48.33	283.24	6.53	59	98.33	67.97	0.13
13	2	10	0.65	60	42	70.00	250.29	3.55	59	98.33	23.27	0.13
13	2	11	0.59	60	51	85.00	112.43	1.48	60	100.00	22.45	0.00
13	2	12	0.54	60	57	95.00	132.16	0.72	60	100.00	27.80	0.00
13	3	5	0.87	60	21	35.00	335.57	10.39	57	95.00	181.54	0.58
13	3	6	0.72	60	35	58.33	375.23	5.46	58	96.67	119.90	0.11
13	4	4	0.81	60	28	46.67	268.57	7.37	55	91.67	88.33	1.28
13	5	3	0.87	60	35	58.33	292.29	5.76	55	91.67	146.96	1.01
14	2	7	1.00	60	11	18.33	337.55	12.89	51	85.00	232.53	2.32
14	2	8	0.88	60	14	23.33	374.14	11.38	53	88.33	195.58	1.75
14	2	9	0.78	60	16	26.67	297.88	9.19	56	93.33	171.82	0.88
14	2	10	0.70	60	26	43.33	294.69	6.23	58	96.67	106.76	0.35
14	2	11	0.64	60	40	66.67	282.33	3.43	58	96.67	73.45	0.53
14	2	12	0.58	60	51	85.00	194.10	1.28	60	100.00	127.82	0.00
14	2	13	0.54	60	56	93.33	80.84	0.48	60	100.00	12.67	0.00
14	3	5	0.93	60	10	16.67	558.90	15.61	42	70.00	218.14	3.93
14	3	6	0.78	60	20	33.33	515.65	9.51	51	85.00	240.43	1.70
14	4	4	0.88	60	17	28.33	459.94	12.53	47	78.33	302.06	3.29
14	5	3	0.93	60	17	28.33	286.94	10.44	46	76.67	346.41	3.04

4.7. Comparación CPLEX vs BC

Finalmente, se observa una gran diferencia en los tiempos de resolución en favor del algoritmo desarrollado. En la siguiente tabla se muestra una comparación, sobre las instancias *BDS*, de la resolución con CPLEX (*CPX*) (con los parámetros de branch-and-bound que mejor resultaron, junto a las familias de cortes que activadas por defecto) y el algoritmo de branch-and-cut desarrollado para *Problema del m-anillo-estrella con capacidades* (*BC*).

n	$\|U\|$	$\|W\|$	m	Q	nod_{CPX}	t_{CPX}	$\%gap_{CPX}$	nod_{BC}	t_{BC}	$\%gap_{BC}$
26	12	13	3	5	26600	143	0.0	954	14	0.0
26	12	13	4	4	25000	126	0.0	192	4	0.0
26	12	13	5	3	61500	356	0.0	230	4	0.0
26	18	7	3	7	112900	*	14.37	3660	77	0.0
26	18	7	4	5	99800	*	19.32	5341	111	0.0
26	18	7	5	4	107600	*	20.0	15357	361	0.0
26	25	0	3	10	111100	*	6.71	1373	26	0.0
26	25	0	4	7	91100	*	12.89	2731	61	0.0
26	25	0	5	6	87600	*	12.19	2351	51	0.0
51	12	38	3	5	112100	*	12.26	14936	1290	0.0
51	12	38	4	4	113100	*	12.98	5180	542	0.0
51	12	38	5	3	112100	*	14.70	6150	776	0.0
51	25	25	3	10	110800	*	15.48	14287	*	2.73
51	25	25	4	7	109200	*	15.76	11348	*	3.06
51	25	25	5	6	111900	*	15.92	10842	*	8.95

nod_{CPX}/t_{CPX}: cantidad de nodos explorados/tiempo de resolución con CPLEX

nod_{BC}/t_{BC}: cantidad de nodos explorados/tiempo de resolución con BC

El tiempo de resolución se indica en segundos. En los casos que aparece *, indica que no se resolvió la instancia en 1800 segundos.

$\%gap_{CPX}$: %gap obtenido por CPX

$\%gap_{BC}$: %gap obtenido por BC

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se abordó la resolución del *Problema del m-anillo-estrella con capacidades* utilizando modelos de programación entera.

Se propuso una nueva formulación para modelar este problema. Se analizó el problema de simetría de soluciones en el modelo propuesto y se propusieron alternativas para tratar el problema. Luego se realizaron las primeras experiencias computacionales sobre un conjunto de instancias conocidas (*BDS*) de entre 26 y 51 nodos, y sobre otro conjunto de instancias de entre 8 y 18 nodos con distintas cantidades de anillo-estrellas y capacidades. En estas pruebas no se observaron mejoras en el tiempo de resolución con las alternativas propuestas para la eliminación de soluciones simétricas.

Luego, con el objetivo de desarrollar un algoritmo *branch-and-cut* que permita mejorar el tiempo de resolución de las instancias, se estudiaron los distintos factores que hacen a los algoritmos de *branch-and-bound* y *planos de corte*.

En primer lugar se diseñó una heurística inicial para obtener una cota primal de manera rápida que permitió obtener un gap inicial de alrededor de un 35 % en las instancias *BDS*, mejorando en la práctica el tiempo de resolución en las instancias probadas con más de 25 clientes. En instancias con menor cantidad de clientes no se justificó la aplicación de esta heurística, dado que el tiempo de resolución de las instancias resultó en general menor que el tiempo de ejecución de la heurística más el tiempo de resolución de la instancia partiendo de la solución inicial brindada por la misma.

Por otro lado, se analizó la combinación de distintos criterios de selección se variables y recorrido del árbol de *branch-and-bound*, en donde se obtuvo en promedio, un mejor tiempo de resolución con la aplicación de *psuedo-costos reducidos* para la selección de variable y la variante de *mejor estimación alternativa* provista por CPLEX para el recorrido del árbol. El algoritmo con esta configuración de parámetros, junto con la aplicación de cortes por defecto provistos por CPLEX (*CPX*), se utilizó luego para compararlo con el algoritmo *branch-and-*

cut desarrollado para este problema en particular.

Por otra parte, se buscaron *desigualdades válidas* y se diseñaron *algoritmos de separación* para las mismas para utilizar como *planos de corte* en las relajaciones. En las pruebas realizadas, el mejor tiempo de resolución se obtuvo, en promedio, con la aplicación de los algoritmos de separación correspondientes a las *desigualdades de capacidad* en combinación con las *desigualdades suma uno*.

A partir de los resultados obtenidos en las experiencias computacionales anteriores, se analizaron los parámetros *IPC* y *skip factor*, propios del algoritmo *branch-and-cut*, pero no pudo desprenderse de este análisis una combinación de éstos que resulte favorable frente a los demás. Esto puede llegar a explicarse mirando los cortes aplicados de cada familia. La cantidad de cortes producidos en cada nodo por las *desigualdades suma uno* fueron muy pocos y para distintos valores de *IPC* la cantidad total de cortes aplicados fue la misma en muchos casos. Al disminuir el *IPC*, muchos cortes pudieron aplicarse en nodos descendientes de los cuales se habrían aplicado con un mayor *IPC*. En cuanto a las *desigualdades de capacidad*, la cantidad total de cortes fue pequeña en relación a la cantidad de nodos explorados, con lo cual un incremento del *skip factor* también pudo haber implicado en muchos casos, postergar la aplicación de cortes. Con lo cual, la decisión tomada, fue no saltear en ningún nodo la búsqueda de desigualdades válidas, ni limitar la cantidad de cortes por nodo.

Finalmente, se realizaron pruebas comparativas sobre las instancias *BDS* con algoritmo de branch-and-cut provisto por CPLEX (*CPX*) y el algoritmo de branch-and-cut desarrollado para *Problema del m-anillo-estrella con capacidades* en particular (*BC*). En la comparación se obtuvieron resultados notoriamente favorables para el algoritmo *BC*.

Este trabajo deja lugar para futuros estudios. De la experimentación resultó que para una instancia y un valor fijo de m, las instancias llevan mayor tiempo de resolución a medida que Q se reduce (hasta reducirse a su valor mínimo $Q = \lceil \frac{|U|}{m} \rceil$). Dado que al aumentar el valor de Q dejando fijos todos los demás parámetros en una instancia, se obtiene una relajación del problema, sería interesante usar este hecho en el contexto de un algoritmo branch & bound ó branch & cut, asi como también diseñar heurísticas para construir soluciones factibles para el valor de Q requerido partiendo de una solución que no es válida para Q pero si es válida para algún valor mayor que Q.

Además sería interesante investigar otro tipo de heurísticas y el uso de metaheurísticas, tanto para construir una solución inicial, como para obtener cotas primales a partir de una solución factible. Por otra parte, los factores que mayor influencia positiva tuvieron en el tiempo de resolución de las instancias son los criterios de selección de variable de branching y recorrido del árbol, y las familias de planos de corte con sus algoritmos de separación. Con lo cual investigar y diseñar estrategias de branching y recorrido del árbol de branch & bound específicas para este problema con esta nueva formulación, asi como también buscar

otras familias de desigualdades válidas para esta formulación y sus respectivos algoritmos de separación ayudaría también podría ayudar a mejorar los tiempos de resolución de las instancias y/o reducir el gap. La desigualdad de capacidad fue la más eficiente en las pruebas que se hicieron y el algoritmo de separación es heurístico, sería interesante contar con algún algoritmo exacto de complejidad polinomial o bien mejorar la heurística para que sea capaz de encontrar más desigualdades válidas violadas.

Bibliografía

[1] R. Ahuja, T. Magnanti, J. Orlin. *Network Flows: Theory, Algorithms and Applications*, Pretince Hall.

[2] M. Benichou, J. Gauthier, P. Girodet, G. Hehntges, G. Ribiere, O. Vincent. *Experiments in Mixed-Integer Linear Programming*, Mathematical Programming 1, 76-94, 1971.

[3] R. Baldacci, M. Dell'Amico, J. Salazar González. *The Capacitated m-Ring-Star Problem*, Operations Research Vol. 55, No. 6, November-December 2007, pp. 1147-1162, INFORMS.

[4] M. Garey, D. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Series of Books in the Mathematical Sciences, W. H. Freeman.

[5] R. Gomory. *Outline of an algorithm for integer solution to linear programs*, Bulletin American Mathematical Society 64, 275-278, 1958.

[6] J. Gross, J. Yellen. *Graph Theory and Its Applications*, Discrete Mathematics and Its Applications, CRC Press.

[7] E. Hoshino, C. Souza. *Column Generation Algorithms for the Capacitated m-Ring-Star Problem*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 5092/2008, June 2008.

[8] A. Mauttone, S. Nesmachnow, A. Olivera, F. Robledo. *A hybrid metaheuristic algorithm to solve the Capacitated m-Ring Star Problem*, 2007, International Network Optimization Conference.

[9] H. D. Sherali, P. J. Driscoll. *On Tightening the Relaxations of Miller-Tucker-Zemlin Formulations for Assymetric Traveling Salesman Problems*, Operations Research Vol. 50, No. 4, July-August 2002, pp. 656-669, INFORMS.

[10] P. Toth, D. Vigo. *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[11] L. Wolsey. *Integer Programming*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons.