Curso 1995/96
CIENCIAS Y TECNOLOGÍAS

JUAN JOSÉ SALAZAR GONZÁLEZ

Optimización combinatoria poliédrica: problemas de rutas-localización

Directores
PAOLO TOTH
MATTEO FISCHETTI

SERVICIO DE PUBLICACIONES
UNIVERSIDAD DE LA LAGUNA

SOPORTES AUDIOVISUALES E INFORMÁTICOS
Serie Tesis Doctorales

a mis padres

Agradecimientos

Al Prof. Paolo Toth y al Prof. Matteo Fischetti debo mucho más que sólo la dirección y contenido de esta memoria. De cada uno de ellos he recibido, además de la constante e inmejorable motivación de un *ideal Profesor*, el incondicional apoyo de un *singular Amigo*.

Agradezco al Ing. Alberto Caprara, co-autor del Capítulo 8, y al Dr. Daniele Vigo las numerosas ayudas prestadas durante mis estancias en Bolonia (Italia).

Agradezco de manera muy especial al Prof. Carlos González Martín, tanto por iniciarme en el estudio de la Optimización Combinatoria Poliédrica, como por haber sido mi tutor durante mi Tercer Ciclo en la Universidad de La Laguna. Le agradezco también la lectura de una versión preliminar de esta memoria, y las notables mejoras que ha sugerido.

Agradezco al Prof. Joaquín Sicilia Rodríguez y a la Prof. María Teresa Ramos Domínguez el haber hecho posible mi contacto inicial con el Prof. Paolo Toth.

Agradezco al *Departamento de Estadística, Investigación Operativa y Computación* (Universidad de La Laguna), al *Dipartimento di Elettronica, Informatica e Sistemistica* (Universidad de Bolonia, Italia), al *Dipartimento di Elettronica e Informatica* (Universidad de Padua, Italia), y a una larga lista de profesores y amigos por sus aportaciones a mis conocimientos.

Esta memoria ha sido parcialmente subvencionada por el Vicerrectorado de Investigación de la Universidad de la Laguna, por la Dirección General de Universidades e Investigación del Gobierno de Canarias, por el M.U.R.S.T. (Italia) a través del "Progetto Finalizzato Trasporti II", por la Comunidad Económica Europea a través de sus proyectos científicos ERBCHRXCT930087 y ERBCIPDCT940623, por la Sociedad de Estadística e Investigación Operativa, y por la Fundación Ramiro Melendreras.

Juan José Salazar González.

Contenido

Prólogo

Esta memoria reúne algunos trabajos realizados sobre *Problemas de Rutas-Localización*, en el contexto de la *Optimización Combinatoria Poliédrica*. Los primeros capítulos se encuadran dentro de la *Teoría de Poliedros*, mientras que en los últimos se proponen algoritmos de *Ramificación y Corte* para diversos problemas combinatorios. Los problemas considerados están estrechamente relacionados con el clásico *Problema del Viajante de Comercio* (TSP), donde se cambia la condición de que el ciclo deba ser Hamiltoniano, por otra condición relacionada con la selección apropiada de ciertos nodos. Precisamente esta mezcla entre selección de nodos y ruta óptima justifica la clasificación de la familia como *problemas de rutas-localización*). Los dos últimos capítulos proponen dos nuevas aplicaciones reales, y sus relaciones con el TSP no deben buscarse tanto en los propios enunciados de los problemas, como en la metodología que se propone para sus resoluciones.

El Capítulo 1 establece la notación básica, al tiempo que enuncia los resultados necesarios en el resto de la memoria. También sitúa en su última sección los problemas combinatorios que luego se analizan, indicando algunos problemas abiertos que podrían ser objeto de futuros trabajos en esta misma línea.

El Capítulo 2 introduce una inmediata variante del TSP, sin estudios específicos precedentes en la literatura, pero sin embargo de gran importancia al ser la relajación inmediata de notables problemas combinatorios. Dicha variante es el *Problema del Ciclo* (CP), y consiste en la búsqueda de un ciclo de costo mínimo en un grafo, que asumimos no dirigido. Nuestras contribuciones consisten en las descripciones de algunas nuevas familias que definen facetas de su polítopo. También proponemos un procedimiento de reforzamiento o de elevación de coeficientes que genera facetas del polítopo del CP a partir de facetas del polítopo del TSP, y otro procedimiento que permite extender facetas de polítopos del CP pequeños a mayores.

El Capítulo 3 introduce una variante del TSP que se conoce como *Problema del Viajante de Comercio Generalizado*, en dos versiones diferentes que se representan por GTSP y E-GTSP, respectivamente. Ambas presentan los nodos del grafo divididos en grupos (*clusters*), y buscan un ciclo simple de costo mínimo. En el GTSP tal ciclo deberá visitar *al menos* un nodo de cada cluster, mientras que en el E-GTSP tal ciclo deberá visitar

exactamente un nodo de cada cluster. Para cada uno de los correspondientes polítopos describimos familias de desigualdades que definen facetas y procedimientos de elevación de coeficientes.

El Capítulo 4 aprovecha las desigualdades introducidas en el capítulo anterior para proponer un algoritmo de ramificación y corte para el GTSP y el E-GTSP, afrontando satisfactoriamente los correspondientes problemas de separación. También se implementa una particular fase inicial que genera una interesante familia de desigualdades para comenzar la fase de hiperplanos de corte, y se proponen eficientes heurísticos. El algoritmo en su globalidad es contrastado sobre varios problemas de la literatura, mostrando una muy alta eficiencia en relación con algoritmos anteriores.

El Capítulo 5 aborda un problema fundamental dentro de los problemas de rutas de vehículos. En este problema, cada nodo del grafo tiene asociado un premio, y cada arco un costo. Dado un costo límite, se busca un ciclo simple con un costo no superior al costo límite, pero que visite los nodos que le producen un premio total máximo. El problema recibe el nombre de *Problema de la Orientación* (OP), y ha sido considerado extensamente en la literatura. Nosotros proponemos el primer algoritmo de tipo ramificación y corte, y demostramos su eficiencia sobre una amplia gama de ejemplos de la literatura. En nuestro trabajo llegamos a resolver problemas con hasta 500 nodos en tiempos de cálculo razonables. El éxito fundamental se debe a una nueva familia de desigualdades que se proponen, llamadas *desigualdades condicionales*. La separación de éstas se basa en una enumeración sobre el grafo soporte de las soluciones fraccionarias, y ello se ha desvelado como un fundamental heurístico, al tiempo que como un generador de interesantes cortes. Estos cortes no son válidos para todo el polítopo del OP, pero sí para soluciones mejores que la mejor solución heurística del momento. El algoritmo que se propone, aunque del tipo ramificación y corte, es al tiempo una variante ya que cuando acaba la fase de ramificación y corte queda la resolución de otro OP generalmente poco denso.

El Capítulo 6 afronta el clásico *Problema de Rutas de Vehículos con Capacidades* (CVRP). Aportamos nuestras muy positivas experiencias usando la formulación como problema de multi-flujos (o de tres índices). Aunque esta formulación es conocida para modelizar problemas generales de rutas de vehículos, su utilización práctica ha sido siempre descartada en el CVRP. Ello ha sido motivado por la hipótesis de igualdad en la capacidad de los vehículos, lo que lleva a pensar en la no-necesidad de distinguirlos. Sin embargo, en este capítulo mostramos como la distinción entre las rutas de los vehículos presenta notables ventajas al proporcionar una descomposición de las soluciones del CVRP en soluciones de problemas de ciclos. Actuando individualmente sobre estos otros problemas hemos logrado alcanzar mejores cotas inferiores que las propuestas en otros trabajos precedentes.

El Capítulo 7 es una nueva aplicación de la Optimización Combinatoria Poliédrica en Estadística. El problema se conoce como *Problema de Supresión de Celdas* (CSP), y

aparece cuando un organismo pretende publicar una tabla estadística pero debe primero suprimir las celdas cuya publicación revelaría información privada. De antemano se conocen esas celdas sensibles (*supresiones primarias*), y los rangos de valores que todo adversario debería creer posibles para tales celdas. El CSP se plantea elegir las otras celdas no sensibles (*supresiones secundarias*) que también deberán ocultarse para garantizar tales niveles de privacidad. Nosotros aportamos un simple modelo matemático del CSP como un problema de Programación Lineal Entera, observando que se trata de un caso particular de un problema de *suma de ciclos* (*sum-of-circuits*). También describimos varias familias de desigualdades que fortalecen su relajación lineal, afrontamos sus respectivos problemas de separación, y proponemos un eficaz algoritmo heurístico que explota la información de soluciones fraccionarias. Con todo ello hemos implementado un algoritmo de ramificación y corte para su resolución exacta, y mostramos nuestras experiencias sobre diversos problemas, tanto reales como generados aleatoriamente. A título meramente indicativo, nuestra propuesta resuelve hasta la optimalidad todos los problemas de una librería procedente del *Netherlands Central Bureau* en menos de 6 segundos sobre un PC 486 / 50 Mhz; algunos de ellos precisaban varios miles de segundos de un ordenador *SUN Sparc 1+* ([G92]).

El Capítulo 8 es una nueva aplicación de la Optimización Combinatoria Poliédrica planteada por ingenieros, que se ocupan del desarrollo de Bases de Datos en la Universidad de Bolonia (Italia). Dadas unas posibles cuestiones de clientes, y una información que puede ser indexada según varios campos, el *Problema de Selección de Indices* (GUFLP) plantea elegir qué índices conviene construir de manera que se minimicen ciertos costes de respuesta y mantenimiento. El problema cuenta con la dificultad de que algunas cuestiones pueden ser respondidas utilizando más de un índice, es decir, configuraciones de índices. Nosotros formulamos matemáticamente este problema como una generalización del clásico *Problema de Localización sin Capacidades* (*Uncapacitated Facility Location Problem*, UFLP), estudiamos su estructura poliédrica viéndolo como un caso particular del problema del empaquetamiento de conjuntos (*Set Packing*, SP), y proponemos un algoritmo de ramificación y corte para su resolución. La clave de este algoritmo es la separación exacta de las desigualdades peña y de las desigualdades ciclo impar particularmente reforzadas. El algoritmo propuesto resultó bastante eficaz sobre las dos familias distintas de ejemplos que consideramos.

Los resultados principales de esta memoria han sido presentados en diversos congresos internacionales, y parcialmente publicados en [FST94], [ESI95], [FS95], [FST95a], [FST95b], [FST95c], [CS96a], [CS96b], y [FST96].

Capítulo 1

Introducción General

El objetivo de esta memoria es presentar nuevos estudios teóricos y computacionales de algunos problemas en *Optimización Combinatoria*. La metodología utilizada se enmarca en la *Combinatoria Poliédrica*. Así, algunos capítulos estudian el polítopo asociado a un problema combinatorio (siguiendo pautas de la *Teoría de Poliedros*), y otros proponen un algoritmo de resolución para su resolución (básicamente de tipo *ramificación y corte*). Este capítulo establece las definiciones y resultados básicos que se presuponen en el resto de la memoria. La última sección introduce los problemas que se tratan en detalle en los restantes capítulos, mencionando también otros problemas no tratados y que motivan futuros trabajos análogos a los realizados en esta memoria.

1.1 Introducción

La *Optimización* trata los problemas de encontrar el máximo o el mínimo de una cierta función objetivo en un cierto domino X. Las teorías clásicas de Optimización (Cálculo Diferencial, Cálculo en Variedades, Teoría del Control Óptimo, ...) afrontan los problemas cuando el domino X es infinito (y la función objetivo tiene determinadas características específicas). La *Optimización Combinatoria*, por su parte, es la rama de la Optimización que afronta los problemas con un dominio X finito, conocidos como *problemas combinatorios*.

Desde el punto de vista de la Matemática Clásica, los problemas combinatorios son triviales, ya que asume como válida la técnica *general* de la Enumeración Total, esto es, del examen exhaustivo de todas y cada una de las posibles soluciones. Es fácil decir "elegimos el mejor de este número finito de posibilidades". Ahora bien, desde el punto de vista de la Matemática Moderna (más preocupada por dar soluciones a problemas

actuales de la Economía, Técnica, Química, ..., y en tal sentido más confundida con la Ingeniería Informática), los problemas combinatorios no son para nada triviales, sino que, muy al contrario, suponen grandes retos. En efecto, la enumeración total de las 100! posibles permutaciones que puede seguir un brazo mecánico al tener que perforar 100 puntos distintos de un chasis durante una cadena de montaje, agota sin lugar a dudas la paciencia de cualquier usuario (aún usando el más potente de los ordenadores actuales). La Optimización Combinatoria busca, para cada uno de sus problemas, una mejor alternativa práctica de resolución frente a la (normalmente no-factible) técnica general de Enumeración Total, aprovechando para ello la *particular* estructura combinatoria de cada dominio X.

Los orígenes históricos de la Optimización Combinatoria se deben a problemas en Economía relativos a la planificación y administración de operaciones, y el uso eficiente de recursos. Pronto se abordaron más aplicaciones técnicas, modelizándose como problemas combinatorios, tales como secuenciación de máquinas, planificación de producción, diseño y localización de factorías. Hoy vemos que los problemas combinatorios están por *todas partes*: inversión de capital, diseño de campañas de ventas, localización de plantas industriales, estudios de códigos genéticos, clasificación de plantas y animales, diseño de nuevas moléculas, asignación controlada de ondas de radios, construcción de compiladores y bases de datos, trazado de redes de comunicación robustas, posicionamiento de satélites, diseño y producción de circuitos VLSI, tamaño de flotas de camiones y planificación del transporte, desarrollo de sistemas de transportes de masas y secuenciación de autobuses y trenes, asignación de trabajadores a tareas tales como pilotos a aviones, embalaje de mercancías, codificación de información, etc. La lista podría ser interminable. Incluso en áreas como deportes, arqueología o psicología, se está utilizando la Optimización Combinatoria para responder a importantes cuestiones.

Son varias las ramas que, desde la simbiosis Matemáticas - Informática, están trabajando para el éxito de la Optimización Combinatoria. Entre ellas destacamos la *Combinatoria Poliédrica*, en cuyo marco se realizan la aportaciones de esta memoria.

La Combinatoria Poliédrica es la aplicación del *Algebra Lineal* (más concretamente, de la *Teoría de Sistemas Lineales* y la *Teoría de Poliedros*) a la Optimización Combinatoria, usando como puente de enlace la *Programación Matemática* (más concretamente, la *Programación Lineal*). Su objetivo inicial era el desarrollo de algoritmos para problemas combinatorios mediante el estudio de un poliedro generado por sus posibles soluciones X (debidamente representado en un espacio donde la función objetivo es lineal). En cierto modo podemos afirmar que la Combinatoria Poliédrica nace a mitad del siglo XX con el desarrollo del algoritmo del simplex para la Programación Lineal. Sin embargo, su objetivo actual se ha extendido cubriendo también otras técnicas similares de resolución, y sobre todo bastantes resultados teóricos puramente combinatorios ("min-max theorems", "polarity", "blocking/antiblocking",...). Precisamente esto último, junto con el hecho

de que cada problema es tratado de modo aislado para aprovechar específicamente su estructura combinatoria, hace que la Combinatoria Poliédrica sea un extenso campo de investigación, lleno de inmensidad de resultados y cuestiones abiertas, difícilmente resumibles en este capítulo. Además, ya en 1985 decía Grötschel [G85]:

> "The subject of polyhedral combinatorics has grown so explosively in the recent twenty years that it is virtually impossible to write an (annotated) bibliografy of this field aiming at a high degree of completeness".

Nosotros nos limitaremos en las siguientes Secciones 1.2–1.5 a citar brevemente algunos conceptos y resultados básicos de Complejidad Algorítmica, Teoría de Grafos y Teoría de Poliedros. No pretendemos que sean una introducción general a sus correspondientes temas, sino sólo una base para establecer la notación específica que se usa a lo largo de esta memoria. Para profundizar en detalles, o para un estudio general de la Combinatoria Poliédrica, recomendamos los trabajos de Bachem y Grötschel [BG82], de Pulleyblank [P82, P91], o de Grötschel y Lovász [GL93], y los libros de texto de Schrijver [S86], de Grötschel, Lovász y Schrijver [GLS88], o de Nemhauser y Wolsey [NW88]. También en esta misma línea se recomienda el texto en elaboración de Cook, Cunninghan, Pulleyblank y Schrijver [CCPS95], aún no publicado. La Sección 1.6 finaliza este capítulo describiendo (también brevemente) el contexto de la Optimización Combinatoria en el que surgen los problemas tratados en los restantes capítulos de este trabajo.

1.2 Conceptos básicos en Complejidad Algorítmica

Un *problema* es la descripción de una situación, más una pregunta sobre ésta. La descripción de la situación está en función de ciertos parámetros ρ que se llaman *datos de entrada ("input")*, y la respuesta puede representarse mediante unas variables σ que constituyen los llamados *datos de salida ("output")*, o propiamente *respuesta*. Tanto σ como ρ deberán estar codificados como cadenas de caracteres de un cierto alfabeto Σ (es decir, elementos del cierre de Kleen Σ^*). En tal sentido, los problemas se pueden representar como conjuntos $\{(\rho, \sigma)\}$ en el producto $\Sigma^* \times \Sigma^*$. Para cada ρ fijo, cuando el subconjunto $\{(\rho, \sigma)\}$ es no vacío se conoce como *ejemplo ("instance")* del problema, aunque en ocasiones se confunde con el propio término "problema". Se llama *tamaño* del ejemplo definido por los datos de entrada ρ a la longitud de esta cadena (es decir, al número de caracteres iguales o distintos para codificar ρ con el alfabeto Σ).

Un *autómata* es un mecanismo capaz de realizar ciertas operaciones elementales de modo automático. Pensemos en un ordenador. Un *programa* es un conjunto de instrucciones o pasos específicamente diseñado para hacer funcionar un autómata. Un *algoritmo* para resolver un problema (mediante un autómata) es un programa que conduce

al autómata en un número finito de pasos a unos datos de salida concretos σ, a partir de los datos de entrada concretos ρ de manera que (ρ, σ) es un elemento del problema. Para disponer de una medida de la "eficiencia" de un algoritmo en la resolución de un problema, a cada uno se le asigna la siguiente función f sobre el espacio de los números naturales, conocida como *complejidad del algoritmo*: $f(n)$ es el máximo de la cantidad de pasos que precisa el algoritmo para resolver ejemplos $\{(\rho, \sigma)\}$ del problema con tamaño $\leq n$. Ahora bien, en general, este valor es muy difícil de evaluar con precisión, es extremadamente dependiente de $|\Sigma|$, y la finalidad es sólo la comparación entre algoritmos. Por ello, se suele simplificar el cálculo buscando una función $g(n)$ más simple que $f(n)$ pero con su mismo tipo de crecimiento (por ejemplo n^3 o 2^n o $n\,log(n)$). En tal caso se dice que su complejidad es *de orden* $g(n)$, y se representa por $O(g(n))$. Un algoritmo se dice *polinomial* cuando su complejidad es $O(g(n))$ con g acotado superiormente mediante un polinomio en n, cualquiera que sea el cardinal del alfabeto σ que se considere. En caso contrario se dice que el algoritmo es *exponencial*. Una subfamilia particular dentro de los algoritmos exponenciales la forman los algoritmos que son polinomiales sólo sobre alfabetos con un único elemento, que se conocen como algoritmos *pseudo-polinomiales*.

Un problema se llama *polinomial* cuando se conoce un algoritmo polinomial que lo resuelva. La familia de los problemas polinomiales se conoce como *clase* $\mathcal{P}$. Un problema se dice $\mathcal{NP}$ (o que está en la *clase* $\mathcal{NP}$) cuando existe un algoritmo polinomial que confirma la pertenencia al problema de cualquier elemento (ρ, σ) que efectivamente lo sea. Tales algoritmos se llaman *certificados de factibilidad*. Es evidente que $\mathcal{P} \subseteq \mathcal{NP}$, y la *gran* pregunta de la Complejidad Algorítmica es "¿$\mathcal{NP} \subseteq \mathcal{P}$?".

Un problema se llama $\mathcal{NP}$*-difícil* si dicho problema es $\mathcal{NP}$ y para cada problema de la clase $\mathcal{NP}$ existe un algoritmo polinomial que transforma cualquier ejemplo de éste en un ejemplo de aquél. Por tanto configura una subclase de $\mathcal{NP}$ (representada por $\mathcal{NPC}$) que contiene sus problemas más difíciles. Se conoce que tal subclase es no vacía, y que dentro destacan por su simplicidad precisamente muchos problemas combinatorios.

Dentro de la clase $\mathcal{NPC}$ se puede a su vez distinguir entre los problemas $\mathcal{NP}$-difíciles *en sentido débil*, y los problemas $\mathcal{NP}$-difíciles *en sentido fuerte*, según que se conozca un algoritmo pseudo-polinomial o no, respectivamente, para su resolución. Los problemas que se tratan en esta memoria pertenecen a este último subgrupo, esto es, son problemas $\mathcal{NP}$-difíciles en sentido fuerte (véase la Figura 1.1).

1.3 Conceptos básicos de Teoría de Grafos

Un *grafo* $G = (V, E)$ es un par de conjuntos finitos, donde los elementos de V se llaman *nodos* (o *vértices*) y los elementos de E se llaman *arcos* (o *aristas*). Cada arco e está asociado con un par de nodos $[u, v]$, y se dice que e es *incidente* a u y v. Según que el

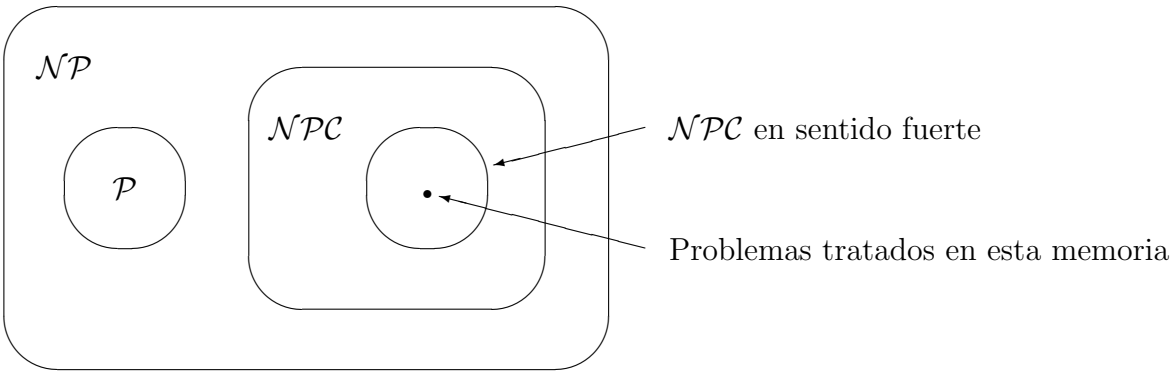


Figura 1.1: Complejidad Algorítmica y los problemas tratados en esta memoria.

par de nodos asociado con un arco se considere o no dirigido, el grafo se dice *dirigido* o *no-dirigido*, respectivamente. Durante esta memoria trabajaremos fundamentalmente con grafos no-dirigidos, y por ello en la presente sección sólo introducimos conceptos para éstos. Además, asumimos (salvo cuando se dice explícitamente lo contrario) que E no tiene más de un arco asociado con un mismo par de nodos, ni un arco asociado con un par de nodos iguales *(bucle)*. Un grafo se dice *completo* cuando, fijado V, el conjunto E contiene todos los posibles arcos. Dado $V' \subseteq V$, representamos por $E(V')$ al conjunto de todos los arcos cuyos dos nodos incidentes están en V'. Un *corte* es el conjunto de todos los arcos del grafo que tienen exactamente un nodo incidente en un conjunto V'. Un grafo se dice *bipartido* si existe un corte que contiene todos sus arcos.

Un *subgrafo* $G' = (V', E')$ de un grafo $G = (V, E)$ es otro grafo tal que $V' \subseteq V$ y $E' \subseteq E$. Dado $V' \subseteq V$, se llama *subgrafo inducido* por V' al grafo $G' = (V', E(V'))$. Un subgrafo que sea además un grafo completo se llama *peña ("clique")*. Un subgrafo que tenga $V' = V$ se llama *generador ("spanning")*.

Un *camino* es un conjunto de arcos ordenables en una secuencia $e_1, e_2, \ldots, e_{k-1}, e_k$, y de modo que dos arcos consecutivos comparte un mismo nodo incidente. Si u es el otro nodo incidente de e_1, no en e_2, y v el otro de e_k, no en e_{k-1}, entonces u y v se llaman los *nodos extremos* del camino, y se dice que el camino *conecta* u con v, y que *visita* los nodos incidentes a sus arcos. El número k se llama *longitud* del camino. Un grafo se dice *conexo* si para cada par de nodos existe un camino que los conecta. Los subgrafos conexos maximales de un grafo se llaman *componentes conexas*. Un grafo se dice *k-conexo* si para cada para de nodos existen k caminos que los unen, y sus nodos intermedios forman conjuntos disjuntos. Un nodo se llama *punto de articulación* cuando su eliminación de V aumenta el número de componentes conexas del grafo. Un arco se llama *puente* cuando su eliminación de E aumenta el número de componentes conexas del grafo.

Un *ciclo* es un camino cuyos nodos extremos coinciden en uno mismo. Asumimos que en los ciclos tienen longitud ≥ 3. Un ciclo se dice *simple* si no contiene ciclos de menor

longitud. Un ciclo se dice *Hamiltoniano* si es simple y de longitud $|V|$, es decir, visita una y sólo una vez cada uno de los nodos del grafo. Estos también reciben el nombre de *tours*, dejando la denominación de *subtours* para los ciclos simples no Hamiltonianos. Un ciclo que contiene una y sólo una vez cada uno de los arcos del grafo se llama *Euleriano*.

Un *bosque* es un conjunto de arcos que no contienen ciclos. Se llama *árbol* cuando además todo subgrafo que lo contenga sea conexo, Un árbol se dice *generador* si todos los subgrafos que lo contengan son generadores.

1.4 Conceptos básicos de Teoría de Poliedros

Dado E un conjunto finito, denotamos por $\mathbb{R}^E$ el espacio lineal de los $|E|$-vectores reales x con una componente x_e para todo $e \in E$. Sea $\alpha \in \mathbb{R}^E$ $(\alpha \neq 0)$ y $\alpha_0 \in \mathbb{R}$. El *semiespacio* definido por la desigualdad $\alpha x \leq \alpha_0$ es el conjunto $\{x' \in \mathbb{R}^E : \alpha x' \leq \alpha_0\}$. El *hiperplano* definido por la desigualdad $\alpha x \leq \alpha_0$ es el conjunto $\{x' \in \mathbb{R}^E : \alpha x' = \alpha_0\}$.

Sea $X \subseteq \mathbb{R}^E$ un conjunto finito. Los elementos de X se dicen *linealmente independientes* si la igualdad $\sum_{x \in X} \lambda_x x = 0$ con $\lambda_x \in \mathbb{R}$ $(x \in X)$ implica $\lambda_x = 0$ para todo $x \in X$. Los elementos de X se dicen *afínmente independientes* si las igualdades $\sum_{x \in X} \lambda_x x = 0$ con $\lambda_x \in \mathbb{R}$ $(x \in X)$ y $\sum_{x \in X} \lambda_x = 0$ implican $\lambda_x = 0$ para todo $x \in X$. El siguiente resultado relaciona ambos conceptos.

Lema 1.4.1 *Sea E un conjunto finito y X un subconjunto finito de $\mathbb{R}^E$. Son equivalentes las siguientes afirmaciones:*

1. *X son afínmente independientes,*
2. *dado un $w \in \mathbb{R}^E$, $\{x - w : x \in X\}$ son afínmente independientes,*
3. *dado un $x' \in X$, $\{x - x' : x \in X \setminus \{x\}\}$ son linealmente independientes.*

La *envolvente lineal* de X (o *espacio generado* por X) es el conjunto de puntos de la forma $\sum_{x \in X} \lambda_x x$, donde $\lambda_x \in \mathbb{R}$ para todo $x \in X$, y sus elementos se llaman *combinaciones lineales* de X. Adicionalmente se puede exigir, o bien $\sum_{x \in X} \lambda_x = 1$, o bien $\lambda_x \geq 0$ para todo $x \in X$, o bien ambas hipótesis. En el primer caso, el conjunto se llama *envolvente afín* y sus elementos *combinaciones afines*. En el segundo caso, el conjunto se llama *envolvente cónica* (o *cono generado* por X, que representaremos por $cone(X)$) y sus elementos *combinaciones cónicas*. En el tercer caso, el conjunto se llama *envolvente convexa* (que representaremos por $conv(X)$) y sus elementos *combinaciones convexas*.

Un *poliedro* es la intersección de un número finito de semiespacios, y por tanto es representable a través de un sistema lineal finito $Ax \leq b$. Uno de los resultados básicos de la Combinatoria Poliédrica es el Teorema siguiente, debido a Weyl y Minkowski.

Teorema 1.4.1 *Para todo poliedro* $P = \{x \in \mathbb{R}^E : Ax \leq b\}$ *existen conjuntos finitos* $X, Y \subseteq \mathbb{R}^E$ *tales que* $P = conv(X) + cone(Y)$. *Recíprocamente, para todo par de conjuntos* $X, Y \subseteq \mathbb{R}^E$ *existe un sistema lineal finito* $Ax \leq b$ *tal que* $conv(X) + cone(Y) = \{x \in \mathbb{R}^E : Ax \leq b\}$.

Un *polítopo* es un poliedro acotado, es decir, un poliedro P para el que existen $l, u \in \mathbb{R}^E$ tales que $P \subseteq \{x \in \mathbb{R}^E : l \leq x \leq u\}$. El Teorema 1.4.1 nos da una definición alternativa de polítopo como la envolvente convexa de un conjunto finito de puntos X, al tiempo que enuncia un interesante corolario: Dado un conjunto finito X de soluciones posibles, su envolvente convexa se puede representar mediante un sistema lineal finito $Ax \leq b$. Esto es un resultado fundamental si tenemos presente las potentes herramientas que nos vienen de la Programación Lineal. Queda pendiente el problema de determinar un "buen" sistema lineal. Recordemos que en Optimización Combinatoria no bastan los teoremas de existencia.

Se llama *dimensión* de un poliedro P, y se denota por $\dim(P)$, al mayor número de puntos afinmente independientes en P menos 1. Esta definición se ha forzado así para que coincida con su significado intuitivo: un punto tenga dimensión 0, una semirrecta 1, un triángulo 2, un cubo 3, ... El siguiente resultado muestra una útil conexión entre la dimensión de P y el mayor número n_l de puntos linealmente independientes en P.

Lema 1.4.2 *Para cualquier* $P \subseteq \mathbb{R}^E$,

1. *si 0 es combinación afín de* P, *entonces* $\dim(P) = n_l$,
2. *si 0 no es combinación afín de* P, *entonces* $\dim(P) = n_l - 1$.

Un poliedro se dice *de dimensión máxima* (o *llena*) en $\mathbb{R}^E$ si $\dim(P) = |E|$. Esto significa que P no está contenido en ningún hiperplano, y que por tanto cualquier representación suya carece de ecuaciones.

Una desigualdad $\alpha x \leq \alpha_0$ se dice *válida* para un punto x^* (o que el punto *satisface* o *verifica* la desigualdad) si $\alpha x^* \leq \alpha_0$; en caso contrario se dice que la desigualdad está *violada* por el punto. La desigualdad se dice *válida* para un poliedro P si la desigualdad es válida para todos los puntos de P, esto es, si P está contenido en el semiespacio que ella define. Una desigualdad se dice *soporte* para P si es válida y existe algún punto de P en el hiperplano que ella define. El conjunto de puntos de P que están en el hiperplano

definido por una desigualdad soporte se llama *cara* de P. Adicionalmente, y por convenio, se llaman también caras al poliedro mismo P y al poliedro vacío $\emptyset$, si bien éstas se conocen como *caras impropias* para distinguirlas de las anteriores, que se llaman *caras propias* de P. Una *faceta* de P es una cara propia maximal. Se dice que una desigualdad $\alpha x \leq \alpha_0$ *induce* o *define una cara* F de P si es una desigualdad válida para P y $F = \{x \in P : \alpha x = \alpha_0\}$ es una cara de P. Cuando $\alpha x \leq \alpha_0$ induce una cara F de P y $\alpha' x \leq \alpha'_0$ induce otra cara F' de P, si $F \subset F' \subset P$ entonces se dice que $\alpha x \leq \alpha_0$ *domina* a $\alpha' x \leq \alpha'_0$. *Fortalecer* una desigualdad significa encontrar otra a partir de ella que la domina. Por definición, las desigualdades que inducen facetas no están dominadas por ninguna otra desigualdad, y por tanto no pueden fortalecerse.

Desde la perspectiva de determinar un sistema lineal $Ax \leq b$ para representar un poliedro dado como combinaciones convexas de vectores en X, las desigualdades que definen facetas son las más importantes. En efecto, un sistema lineal con igualdades y desigualdades que determina un poliedro se dice *minimal* cuando

1. no es posible prescindir de ninguna igualdad o desigualdad sin ampliar el poliedro,
2. no es posible convertir ninguna desigualdad en igualdad sin restringir el poliedro.

La importancia que antes hemos dado a las desigualdades que definen facetas se basa en el resultado siguiente.

Teorema 1.4.2 *Sea P un poliedro no vacío representado por las igualdades $Ax \leq b$ y por las desigualdades $Cx = d$. Entonces $Ax \leq b, Cx = d$ es un sistema lineal minimal para P si y sólo si*

1. *las filas de B son linealmente independientes, y*
2. *cada desigualdad en $Ax \leq b$ define una faceta de P.*

Este resultado también garantiza la unicidad (salvo multiplicaciones por un escalar positivo) de las desigualdades que definen una faceta de un poliedro de dimensión máxima, y por tanto que no hay peligro de tener dos desigualdades claramente diferentes que definan la misma faceta.

Esta relevancia de las facetas en la búsqueda de sistemas lineales minimales, exige mejores caracterizaciones de ellas, y en tal sentido resulta útil el siguiente teorema.

Teorema 1.4.3 *Sea F una cara propia de $P = \{x \in \mathbb{R}^E : Ax \leq b, Cx = d\}$. Las siguientes afirmaciones son equivalentes:*

1. *F es una faceta de P;*
2. *$\dim(F) = \dim(P) - 1$;*
3. *si $\alpha x \leq \alpha_0$ y $\alpha' x \leq \alpha_0$ son dos desigualdades que inducen la faceta F, entonces existe $\delta \in \mathbb{R}$ positivo y $\lambda \in \mathbb{R}^d$ tal que $\alpha' = \delta\alpha + \lambda C$ y $\alpha'_0 = \delta\alpha_0 + \lambda d$.*

Ambas caracterizaciones constituyen los dos métodos fundamentales para demostrar que una desigualdad dada $\alpha x \leq \alpha_0$ induce una faceta de un poliedro P. El primero (número 2 en el Teorema 1.4.3) se conoce como *método directo*, y persigue demostrar que existen $\dim(P)$ puntos afinmente independientes que satisfacen $\alpha x = \alpha_0$. El segundo (número 3 en el Teorema 1.4.3) se conoce como *método indirecto*, y se basa en que las facetas están definidas por una única desigualdad, salvo combinaciones de las ecuaciones que definen el poliedro y salvo multiplicaciones por un escalar positivo. Durante esta memoria seguiremos principalmente el método directo, aunque en el Capítulo 2 también usaremos el método indirecto.

Supongamos que una desigualdad $\alpha x \leq \alpha_0$ define una cara F de un poliedro P. Supongamos que hemos encontrado otra desigualdad $\alpha' x \leq \alpha'_0$ que induce la misma cara que $\alpha x \leq \alpha_0$, diferente en el sentido de la caracterización 3 del Teorema 1.4.3, y que por tanto $\alpha x \leq \alpha_0$ no define una faceta. Entonces es posible (en ocasiones) encontrar un oportuno ϵ tal que $(\alpha + \epsilon\alpha')x \leq (\alpha_0 + \epsilon\alpha'_0)$ induce una cara de P de dimensión $\geq \dim(F) + 1$, es decir, que domina a $\alpha x \leq \alpha_0$. Los Capítulos 2, 3 y 6 ilustran esta idea aportando nuevos resultados.

En tal línea trabajan los conocidos *teoremas de elevación* (del inglés *"lifting theorems"*), afrontando el problema de generar facetas para un poliedro P a partir de facetas de una cara F suya. Uno de los más básicos (y que será utilizado en los Capítulos 2 y 3) es el siguiente resultado, base del *procedimiento el elevación secuencial* introducido por Padberg [P75].

Teorema 1.4.4 *Sea P un poliedro cuyos puntos extremos tienen componentes 0 o 1, $e^* \in E$ una componente cualquiera tal que $F = \{x \in P : x_{e^*} = 0\}$ es una cara no vacía. Si $\sum_{e \in E \setminus \{e^*\}} \alpha_e x_e \leq \alpha_0$ define una faceta de F, entonces*

$$\alpha_{e^*} x_e + \sum_{e \in E \setminus \{e^*\}} \alpha_e x_e \leq \alpha_0$$

con

$$\alpha_{e^*} := \alpha_0 - \max \left\{ \sum_{e \in E \setminus \{e^*\}} \alpha_e x_e \ : \ x \in P \text{ y } x_{e^*} = 1 \right\}$$

define una faceta de P.

No queremos terminar esta sección dejando en el lector la falsa idea de que las desigualdades que definen facetas son imprescindibles para la resolución de un problema. Como se ha dicho, lo son si se busca un sistema lineal minimal para la envolvente convexa generada por las posibles soluciones de un problema combinatorio, y en tal sentido conviene adelantar que para la mayor parte (esto es, para los $\mathcal{NP}$-difíciles) se cree que nunca se tendrá (aunque este pesimismo desaparecería si $\mathcal{NP} = \mathcal{P}$). La realidad es que por el momento sólo se puede aspirar a descubrir algunas de las facetas, describiendo así parcialmente el poliedro. Sin embargo, ni tan siquiera esto es en ocasiones tarea fácil, y muchas veces inútil pensando sólo a la resolución del problema. El Capítulo 5 es prueba clara de esto. En él se usan con éxito desigualdades que en ocasiones, ni tan siquiera son válidas para todo P, sino sólo para aquella parte de P que contienen puntos extremos mejores respecto a uno conocido.

1.5 Conceptos básicos de Combinatoria Poliédrica

La base inicial para la resolución de un problema combinatorio mediante la Combinatoria Poliédrica es la representación de sus miembros como vectores (típicamente vectores incidentes 0-1), y la construcción de un modelo matemático de programación lineal entera. Este modelo generalmente ya contiene interesantes desigualdades lineales que delimitan el poliedro P en cuestión. Sin embargo, no son suficientes, y un estudio "ad hoc" (como los que se realizan en esta memoria) son necesarios para describir nuevas familias de desigualdades lineales. Ahora bien, no todo está resuelto cuando se tienen tales familias, incluso en el caso más optimista de disponer de una familia $\mathcal{F}$ de desigualdades que describa completamente a P. En efecto, luego queda saber manipularlas en el interior de un algoritmo que funcione en la práctica, dando soluciones a problemas reales. Una primera dificultad la plantea el siguiente problema, conocido como el *problema de separación*:

> Dada una familia de desigualdades $\mathcal{F}$ y un punto $x^* \in \mathbb{R}^E$, demostrar que x^* satisface todas las desigualdades de $\mathcal{F}$, o por el contrario, determinar una desigualdad de $\mathcal{F}$ que x^* no satisface.

Este problema es la clave en el desarrollo de un algoritmo de hiperplanos de corte en el que se desea optimizar una función lineal sobre un poliedro P definido por un sistema lineal $\mathcal{F}$. En efecto, el inmediato intento de tratar de resolver el problema lineal (LP) con *todas* las desigualdades de $\mathcal{F}$ fracasa en la práctica ante el elevado número $|\mathcal{F}|$. Un algoritmo de hiperplanos de corte procede entonces creando un LP inicial con sólo *algunas* desigualdades $\mathcal{F}' \subset \mathcal{F}$, e invoca luego iterativamente el siguiente procedimiento

1. resolver el LP, y sea x^* su solución óptima;

2. resolver el problema de separación con x^*, parando cuando se concluye que x^* satisface $\mathcal{F}$, o introduciendo en el LP la desigualdad que x^* no satisface.

Evidentemente, es fundamental disponer de un potente algoritmo que resuelva los LP's que se proponen. Ahora bien, no menos importante, tanto desde un punto de vista teórico como práctico, es el algoritmo que afronta el problema de separación. La importancia teórica queda acentuada con la siguiente equivalencia entre *separar* y *optimizar*:

> Sea P un polítopo definido por una familia de desigualdades $\mathcal{F}$. El problema de optimización de una función lineal sobre P es resoluble en tiempo polinomial si y sólo si el problema de separar un punto de $\mathcal{F}$ es polinomial

(Véase Schrijver [S86], o Grötschel, Lovász y Schrijver [GLS88], o Nemhauser y Wolsey [NW88] para más detalles de este otro resultado fundamental en Combinatoria Poliédrica.)

A nivel práctico, el número de iteraciones que precisa la técnica de hiperplanos de corte depende estrictamente de cómo se resuelva el problema de separación, y más concretamente de qué desigualdad se proponga para su incorporación al LP cuando x^* no pertenece al poliedro P generado por $\mathcal{F}$.

Por otra parte, conviene observar que en los problemas de Optimización Combinatoria normalmente no se dispone de un sistema lineal completo $\mathcal{F}$ que describa su poliedro, sólo de un sistema lineal parcial $\mathcal{F}'$ que describe completamente a un poliedro P^0 (ligeramente) mayor que P. Sin embargo, en muchos casos la anterior técnica de hiperplanos de corte sirve igualmente ya que en diversas ocasiones el punto final x^* de P^0 también está en P. Los análisis combinatorios que se exponen en los siguientes capítulos de este trabajo ilustran extensamente esta idea. No obstante, claramente, una descripción parcial no garantizará siempre tal éxito con todos los ejemplos del problema, y por tanto el correspondiente algoritmo de hiperplanos de corte (donde la separación atiende sólo a $\mathcal{F}'$) debe ser completado con algo más.

La conclusión anterior no es una consecuencia únicamente debida al hecho de disponer sólo de descripciones parciales. De hecho, aún teniendo una descripción lineal completa $\mathcal{F}$ de P, puede resultar conveniente detener el proceso iterativo sin esperar a que lo ordene la separación. Este es el caso de ver el problema combinatorio como un problema de Programación Lineal Entera, y se usan los cortes de Chvátal-Gomory como una descripción completa. En efecto, durante la aplicación de una técnica de hiperplanos de corte puede suceder que el valor objetivo de los puntos x^* no cambie durante bastante iteraciones consecutivas (digamos que durante m iteraciones el valor de la función objetivo no se altera en ϵ unidades). Este fenómeno, característico de muchos problemas combinatorios, se conoce como *tailing-off*, y una de las formas de afrontarlo es provocando la interrupción del anterior proceso iterativo.

Cuando detenemos la fase de hiperplanos de corte sin que x^* se corresponda con una de las soluciones posibles del problema combinatorio, entonces se dispone de un potente LP sobre el que probar las distintas posibilidades que le faltan a x^* para ser factible. Así, se dispone una herramienta que proporciona una buena *cota* en el marco de un algoritmo de *Ramificación y Acotación ("Branch and Bound")*. Esta es la idea básica del trabajo, por ejemplo, de Grötschel y Holland [GH91] para la resolución del clásico *Problema del Viajante de Comercio* (TSP). Otra idea más elaborada es la de re-aplicar el procedimiento de los hiperplanos de corte cada vez que el Algoritmo de Ramificación y Acotación deba calcular la *cota*. Esta otra idea se conoce como *Ramificación y Corte ("Branch and Cut")*, y aparece fundamentalmente como tal en el trabajo de Padberg y Rinaldi [PR87] sobre el TSP. En los últimos años han aparecido muchos otros interesantes resultados también sobre otros problemas combinatorios, tales como, por ejemplo, el realizado por Hoffman y Padberg [HP93] sobre la planificación de una flota de aviones. El corazón de esta técnica es precisamente resolver la separación sobre una familia de desigualdades de manera que se mantengan válidas para los poliedros "hijos" que se tratan durante el examen de los nodos del árbol de ramificación (como sucede con las desigualdades que definen facetas del poliedro original). No obstante, son muchos los otros componentes que colaboran conjuntamente al buen funcionamiento de un algoritmo de ramificación y corte. Valga como ejemplo el disponer de algoritmos heurísticos que, aprovechando la información proporcionada por soluciones exactas de relajaciones, generen buenas soluciones posibles del problema original. Puesto que dichas soluciones proceden generalmente de la resolución de un LP, se llaman *heurísticos basados en el LP ("LB-based heuristics")*. Esta memoria aporta nuevas aplicaciones del algoritmo de Ramificación y Corte a la Optimización Combinatoria en sus Capítulos 4, 5, 6, 7 y 8.

Terminamos esta sección observando que no es tarea nada fácil el hacer funcionar en la práctica esta idea general de un algoritmo de Ramificación y Corte, y que muy por el contrario, son numerosas las dificultades que deben ser resueltas. Dejando a un lado las grandes cuestiones de determinar (para cada problema) los "buenos" cortes y reglas de ramificación, citaremos a título ilustrativo dos de tales dificultades:

- No siempre es posible tratar con LP's que contienen todas las variables (por ejemplo, si $|E| > 10.000$), y aún siéndolo, no siempre aconsejable para acelerar la resolución del mismo LP. De este modo, resulta conveniente gestionar externamente el control de las variables que en cada iteración posiblemente interesan tener en cada LP. Esto se conoce como fase de *"pricing"*.

- No suele ser posible (ni recomendable) mantener en el LP todas las restricciones que han sido propuestas por la separación. Suele convenir crear una estructura de datos externa, llamada *piscina ("pool")*, que en forma comprimida contiene tales restricciones, mientras que en el LP se mantienen sólo aquellas que, según algún conveniente criterio, se consideran útiles. Así, a cualquier llamada a la fase de sep-

aración deberá preceder una garantía de que no existen desigualdades en la piscina no verificadas por la solución actual del LP. Para ello, tras la resolución de cada LP se realiza una búsqueda exhaustiva de las mismas, y se incorporan las que se localicen. Así se evita la regeneración repetida de una misma desigualdad.

En los casos donde es extremadamente difícil demostrar la optimalidad, estos algoritmos pueden también presentarse como "buenos" *heurísticos*, en el sentido de que producen soluciones próximas a las óptimas con una cierta cualidad garantizada. Esta garantía procede de la resolución interna de *relajaciones* del problema original, esto es, de problemas más fáciles que el original creados eliminando o debilitando algunas de las restricciones de partida.

1.6 Algunos problemas combinatorios

La Optimización Combinatoria cubre una enorme familia de problemas, según hemos indicado al inicio de este Capítulo. En esta memoria se aborda la resolución de algunos problemas combinatorios encuadrables fundamentalmente dentro de la particular subfamilia conocida como *problemas de rutas-localización ("routing-location problems")*. En esta subfamilia se afrontan contemporáneamente el problema de la elección de puntos especiales *(problema de localización)*, con el diseño de rutas óptimas que los recorran *(problema de rutas)*, y en tal sentido contiene a su vez una cantidad considerable de problemas de gran relevancia tanto práctica como teórica. La importancia práctica es clara en una sociedad basada en comunicaciones y distribuciones. La importancia teórica se fundamenta en que algunos de sus problemas se consideran como *problemas test* para el desarrollo y prueba de técnicas generales, al tiempo que son objetivos claves de, por ejemplo, la Complejidad Algorítmica.

Consideremos una red de conexiones, modelizada a través de un grafo $G = (V, E)$ donde cada arco $e \in E$ tiene un costo asociado c_e, y (quizás) cada nodo $v \in V$ un premio asociado p_v. Dentro de los problemas $\mathcal{NP}$-difíciles, el más simple y conocido es el *Problema del Viajante de Comercio (Travelling Salesman Problem*, TSP), en el que se busca un circuito Hamiltoniano en G de costo mínimo. Este problema ha sido extensamente tratado, según prueba el libro [LLRS85] que detalla su "estado del arte" hasta 1985.

El TSP ha sido el primer problema donde la Combinatoria Poliédrica ha demostrado con éxito lo que puede aportar a la Optimización Combinatoria. Han sido varios los trabajos que han contribuido a ello, pero merecen destacarse de forma particular los de Padberg y Rinaldi [PR87, PR90, PR91] y de Grötschel y Holland [GH91] cuyos algoritmos de Combinatoria Poliédrica permitieron resolver ejemplos con entre 17 y 2392 ciudades.

Algunos de ellos fueron generados aleatoriamente; otros proceden de aplicaciones geográficas o industriales. El clave del éxito fue una mejor comprensión del polítopo del TSP: además de las antes conocidas desigualdades de *eliminación de subtour (subtour elimination constraints)*, y de *2-emparejamiento (2-matching constraints)*, se aportaron las desigualdades *peine (comb inequalities)* y las *árbol peña (clique tree inequalities)*. Y cada una acompañada de eficaces algoritmos (algunos heurísticos) para afrontar su correspondiente problema de separación. Pero además, no se puede menospreciar también el gran papel que en todo ello ha jugado la aparición de mejores programas resolvedores en Programación Lineal, y así los mejores resultados para el TSP están en las *CPLEX newsletter* donde Applegate, Bixby, Chvátal y Cook han publicado recientemente la obtención de pruebas de optimalidad para ejemplos con 3038 y 7397 ciudades (véase, por ejemplo, [CCPS95, página 258]). Ultimamente se han aportado nuevas familias de facetas del polítopo del TSP (véase por ejemplo Rinaldi y Naddef [NR93]), pero aún no se ha afrontado convenientemente el (difícil) problema de separarlas, y por tanto no hay nuevos resultados computacionales. Fischetti y Toth [FT92] también han mostrado la gran herramienta que ofrece la Combinatoria Poliédrica para la versión asimétrica del TSP.

A partir del TSP aparecen distintas variantes que tratan de incorporar o cambiar alguna restricción, bien con objetivos teóricos, bien con la finalidad de crear un problema combinatorio más ajustados a ciertos problemas reales. Algunas de ellas son:

Problema del Ciclo *(Cycle Problem*, CP): Consideremos un grafo no-dirigido, y un costo asociado con cada arco. El CP busca un ciclo simple en el grafo, no necesariamente Hamiltoniano, de costo mínimo.

Se trata de un problema clave que subyace en numerosos problemas de rutas-localización, como se observa en varios capítulos de este trabajo. En el Capítulo 2 se realiza un estudio de su polítopo: se realizan demostraciones inéditas sobre familias de desigualdades que definen facetas, y se aporta una nueva técnica que permite extender facetas del polítopo del TSP al polítopo del CP.

Problema del Viajante de Comercio Generalizado *(Generalized Travelling Salesman Problem*, GTSP): Consideremos un grafo no-dirigido donde el conjunto de nodos está dividido en grupos *("clusters")*, y cada arco tiene un costo asociado. El GTSP busca un ciclo simple que, con costo mínimo, visite cada cluster al menos una vez. Otra versión del problema (llamada E-GTSP) busca un ciclo simple que, con costo mínimo, visite cada cluster exactamente una vez.

Ambas variantes de este problema proceden de aplicaciones prácticas, por ejemplo, en localización de buzones de correo en una ciudad, o de nodos principales para crear una red de comunicaciones. El Capítulo 3 aporta un estudio poliédrico original de ambos, y el Capítulo 4 aplica tales resultados proponiendo un algoritmo de

ramificación y corte que, debidamente experimentado, se ha revelado como una buena técnica de resolución del GTSP.

Problema de la Orientación *(Orienteering Problem*, OP): Consideremos un grafo no-dirigido donde cada arco tiene asociado un costo, y cada nodo un premio. El OP busca un ciclo simple que, con un costo no superior a un determinado costo límite, visite los nodos que le conlleven la recolección del máximo premio posible.

Este problema nace en el contexto de la localización de un depósito que optimice la posterior entrega de mercancía a clientes, aunque también es aplicable en otros contextos. Además, es un problema clave como relajación de numerosos problemas de rutas de vehículos. Una característica práctica bastante extendida es que el ciclo deba pasar necesariamente por un nodo dado, ya que representa la ruta de un vehículo que originalmente está en el depósito, y donde al final debe volver. El Capítulo 5 propone un algoritmo del tipo ramificación y corte, basados en nuevas y fuertes desigualdades no necesariamente válidas para todo el poliedro.

Problema de Rutas de Vehículos con Capacidades *(Capacitated Vehicle Routing Problem*, CVRP): Sea $G = (V, E)$ un grafo, donde existe un nodo particular 1 llamado depósito. Cada arco e tiene un costo c_e, y cada nodo v distinto del depósito representa un cliente con una cierta demanda q_v. Inicialmente en el depósito hay m vehículos de capacidad idéntica Q. El CVRP se plantea la búsqueda de al máximo m ciclos simples con el menor costo posible, que pasen por el depósito, que todo cliente esté visitado por exactamente un ciclo, y que la suma de las demandas de los clientes asignados a cada ciclo no exceda la capacidad Q.

Este es un clásico problema de Optimización Combinatoria. Ha sido extensamente tratado desde numerosos puntos de vista. Nosotros aportamos en el Capítulo 6 nuevas experiencias con un modelo clásico, rechazado de antemano por otros investigadores. Nuestra experiencias muestran que el modelo es, por el contrario, muy prometedor. La base de ello radica en la posibilidad de descomponer la solución del modelo en soluciones de problemas de ciclos, y volcar entonces los recientes avances poliédricos.

Indudablemente, son muchos los otros problemas extrechamente relacionados con el TSP. A título de ejemplos citamos:

Problema de Recolección de Premios *(Prize-Collecting Travelling Salesman Problem*, PCTSP): Consideremos un grafo no-dirigido donde cada arco tiene asociado un costo, y cada nodo un premio. El PCTSP busca un ciclo simple que, visitando clientes que al menos le proporcionen un premio límite, tenga el menor costo posible.

Este problema es una especie de problema dual del OP, si bien tiene un poliedro más fácil de estudiar, ya que la restricción del premio límite mínimo mantiene una estructura combinatoria más fácil de considerar que la restricción del costo límite máximo. De hecho, aquélla es un caso particular de ésta. Un estudio poliédrico específico de la versión asimétrica del PCTSP aparece en Balas [B89, B93a].

Problema del Ciclo que Recubre *(Covering Tour Problem*, CTP): Consideremos un grafo donde los nodos están divididos en dos conjuntos ($V = U \cup W$), y donde cada arco tiene una distancia asociada. Sea c_0 una distancia pre-establecida. EL CTP busca un ciclo simple en el subgrafo inducido por U, con distancia mínima, y de manera que para todo nodo de W, la distancia de éste a alguno de U visitado por el ciclo no supere c_0.

Este problema es un caso particular del GTSP. Un estudio específico ha sido realizado en Gendreau, Laporte y Semet [GLS96]

Problema de Viajante de Comercio que Reposta *(Refuelling Travelling Salesman Problem*, RTSP): Consideremos un grafo donde los nodos están divididos en dos conjuntos ($V = U \cup W$), y donde cada arco tiene una distancia asociada. Sea c_0 una distancia pre-establecida. El RTSP busca un ciclo Hamiltoniano en el grafo, con distancia mínima, y de manera que todos los caminos que tal ciclo contiene en el subgrafo inducido por W tengan distancia no superior a c_0.

El RTSP aparece en aplicaciones donde un viajante (avión, tren, máquina de perforar, respectivamente) debe visitar un conjunto de nodos (aeropuertos, estaciones de trenes, puntos potenciales de perforación, respectivamente) pero no pudiendo visitar en modo continuado excesivos nodos en U sin pasar por algún nodo en W, donde se dispone de servicios especiales (repostar carburante, controlar su buen funcionamiento, enfriar la punta de perforación, respectivamente). El caso particular donde la limitación sobre cada camino no es en base a su distancia, sino al número de arcos que lo forman (o nodos en U que contiene), fue propuesto originalmente por Balas en una comunicación personal. No se conoce ningún trabajo sobre el RTSP en literatura, por lo que abre otro punto para futuras investigaciones.

También partiendo del ATSP aparecen interesantes variantes, tanto aplicaciones directas de problemas de rutas, como de problemas de secuenciación de tareas. Entre ellas citamos a título ilustrativo:

Problema del Repartidor *(Deliveryman Problem*, DP): Consideremos un grafo G dirigido y un costo asociado a cada arco. Entre los nodos hay uno especial, llamado depósito; los restantes se llaman clientes. El DP busca un circuito Hamiltoniano que minimice el costo medio del camino que lleva desde el depósito a un cliente.

TSP Dependiente del Tiempo *(Time-Dependent Travelling Salesman Problem*, TDTSP): Consideremos un grafo G dirigido, y un vector de $|V|$ costos asociado a cada arco. Entre los nodos hay uno especial, llamado depósito; los restantes se llaman clientes. El DP busca un circuito Hamiltoniano de costo mínimo, entendiendo que el costo de un arco es c_e^k si el arco e ocupa la posición k-ésima en el circuito.

TSP con Tiempos de Ventana *(Time-Window Travelling Salesman Problem*, TWTSP): Consideremos un grafo G dirigido y un tiempo asociado con cada arco. Entre los nodos hay uno especial, llamado depósito; los restantes se llaman clientes. Supongamos que cada cliente tiene asociado un tiempo de apertura y un tiempo de cierre. El TWTSP busca un circuito Hamiltoniano de manera que el coste del camino en el circuito desde el depósito hasta el cliente se mantenga entre los dos tiempos que éste tiene asociado.

TSP con Restricciones de Precedencia *(Travelling Salesman Problema with Precedence Constraint*, TSP-PC): Consideremos un grafo G dirigido, una función de costo definida sobre los arcos, y una relación $\mathcal{R}$ de precedencia sobre los nodos. El TSP-PC busca un circuito Hamiltoniano de costo mínimo que respete las relaciones de precedencia entre los nodos.

El Capítulo 7 afronta un problema combinatorio relacionado con el cubrimiento de un conjunto de arcos por ciclos, nacido en el contexto de la publicación de Tablas Estadísticas Públicas. Cuando una empresa pública o privada se plantea la publicación de ciertos datos, normalmente está vinculada por un compromiso de no revelar información específica de individuos particulares, y que asegura la privacidad de éstos. En tal sentido, durante el proceso de publicación de tales datos en forma de tablas estadísticas, aparecen algunas celdas que necesariamente se deben ocultar *(celdas sensibles)*. Sin embargo, la existencia de algunas relaciones entre los datos (como las sumas marginales por filas y columnas) produce que típicamente no baste con esto, y necesariamente otras celdas de la tabla deberán ser también ocultadas. El problema de elegir estas otras celdas de manera que se garanticen ciertos niveles deseados de protección, pero se minimice al mismo tiempo la información no publicada por estas *supresiones secundarias*, es un problema combinatorio llamado *Problema de Supresión de Celdas (Cell Suppression Problem*, CSP). El Capítulo 7 pretende clarificar estas ideas, y propone una técnica de resolución poliédrica que supera notablemente cualquier técnica precedente.

En el Capítulo 8 introducimos una generalización del clásico *Problema de Localización sin Capacidades (Uncapacitated Facility Location Problem*, UFLP). Esta generalización ha sido motivada por un problema práctico que aparece durante el diseño físico de bases de datos. La buena administración de una base de datos depende directamente de la estructura de acceso considerada, y en tal sentido es clave la resolución óptima del llamado *Problema de Selección de Indices (Index Selection Problem*, ISP). El Capítulo 8 modeliza este problema como un problema combinatorio, poniéndolo como una generalización

del UFLP. Al tiempo también propone un algoritmo de ramificación y corte que se ha manifestado como bastante eficaz en experiencias computacionales con ejemplos del ISP.

Capítulo 2

Problema del Ciclo en Grafos No-Dirigidos

El *Problema del Ciclo* (CP) es una variante del Problema del Viajante de Comercio (TSP), donde en lugar de pedir que un ciclo "pase exactamente una vez" por cada una de las ciudades, se exige que "pase al máximo una vez". Cuando no se establece ninguna hipótesis sobre los costos asociados a los arcos, el CP es un problema $\mathcal{NP}$-difícil que subyace como relajación de numerosos problemas combinatorios. Desde un punto de vista teórico presenta un polítopo bien definido como la envolvente convexa de puntos extremos fáciles de generar (los ciclos simples); sin embargo, de su definición mediante un sistema de desigualdades se conoce poco. En este capítulo realizamos un estudio poliédrico del CP sobre grafos no dirigidos, demostrando que las desigualdades de la clásica formulación lineal entera del CP definen facetas, extendiendo las *desigualdades peine* del TSP al CP, y proponiendo un procedimiento de elevación para obtener facetas del CP a partir de facetas del TSP.

2.1 Introducción

Dado un grafo no-dirigido $G = (V, E)$ y un costo c_e asociado con cada arco $e \in E$, se sabe que el problema de encontrar un ciclo (simple o no) de G teniendo costo total mínimo es resoluble en tiempo polinomial. En efecto, un ciclo óptimo (si lo hay) puede encontrarse tras calcular un camino mínimo desde i hasta j en el grafo $(V, E \setminus [i, j])$, para cada arco $[i, j] \in E$. (No consideraremos ciclos de longitud menor de 3 ya que éstos son trivialmente analizables.) Este problema tendrá solución óptima (es decir, existirá un ciclo de costo total mínimo) si y sólo si existen ciclos en G (en otro caso, el problema sería no-factible),

pero ninguno de ellos tiene un costo total negativo (en otro caso, el problema sería no-acotado). Además, en tales casos se obtiene una solución óptima entre los ciclos simples (esto es, ciclos que visitan exactamente un nodo).

Si se busca explícitamente un *ciclo simple* de G con costo total mínimo, entonces el problema pasa a ser $\mathcal{NP}$-difícil. En efecto, un ejemplo del estándar Problema del Viajante de Comercio (TSP, para abreviar) puede reducirse a un ejemplo del anterior problema simplemente restando una constante positiva grande al costo de cada arco. De este modo, si tuviésemos un algoritmo polinomial para resolver el problema inicial, tendríamos también un algoritmo polinomial para el TSP. Al problema inicial se le conoce como *Problema del Ciclo* (CP, para abreviar). Balas [B93b] expone un análisis poliédrico de la versión asimétrica de este problema. Este trabajo ha sido distribuido como [FST94], y contemporáneamente a ha aparecido también el trabajo de Bauer [B94] que persigue los mismos objetivos, si bien [B94] presenta un análisis más arduo por el espacio de variables en el que lo realiza.

Sea $\mathcal{C}$ la colección de todos los ciclos simples de G, y sea $x^T \in I\!R^E$ el vector característico de un ciclo $T \in \mathcal{C}$ (es decir, la componente e-ésima x_e^T es 1 si el arco e está en T, y 0 en otro caso). Si $n := |V|$ entonces definimos el *Cono del CP* como

$$C_n := cone\{x^T : T \in \mathcal{C}\} = \left\{ \sum_{T \in \mathcal{C}} \lambda_T x^T : \lambda_T \geq 0 \text{ para todo } T \in \mathcal{C} \right\},$$

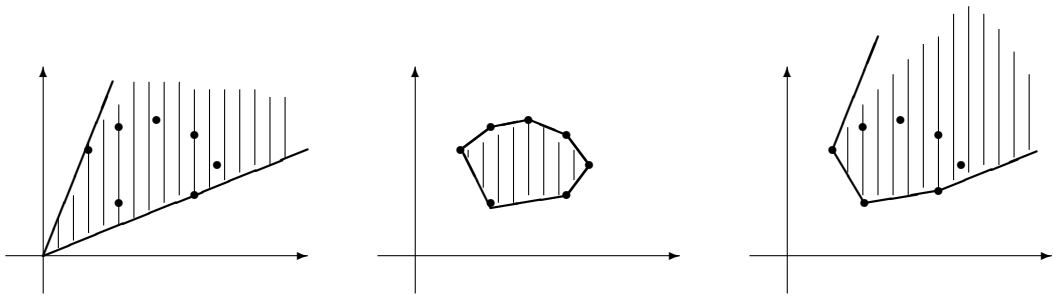
y el *polítopo del CP* como

$$P_n := conv\{x^T : T \in \mathcal{C}\} = \left\{ \sum_{T \in \mathcal{C}} \lambda_T x^T : \begin{array}{c} \lambda_T \geq 0 \text{ para todo } T \in \mathcal{C}, \\ \text{y } \sum_{T \in \mathcal{C}} \lambda_T = 1 \end{array} \right\}.$$

El primer problema introducido equivale a minimizar cx en $x \in P_n + C_n$, y el segundo problema introducido es equivalente a minimizar cx en $x \in P_n$. (Véase la Figura 2.1 (a), (b) y (c) para una visión geométrica de C_n, P_n y $P_n + C_n$, respectivamente.) Dado que el último es un problema $\mathcal{NP}$-difícil, es muy improbable determinar un sistema lineal que describa completamente a P_n. Como se ha indicado, no ocurre igual con el último poliedro, y sin embargo, hasta el momento, tampoco se conoce ninguna descripción completa para $P_n + C_n$.

Sea $S \subseteq V$, entonces $\delta(S)$ representa el conjunto de los arcos incidentes con exactamente un nodo de S, y $E(S)$ es el conjunto de los arcos con ambos nodos incidentes en S, es decir

$$\begin{aligned} \delta(S) &:= \{[i,j] : i,j \in S\}, \\ E(S) &:= \{[i,j] : i \in S, j \notin S\}. \end{aligned}$$



(a) C_n. (b) P_n. (c) $P_n + C_n$.

Figura 2.1: Poliedros asociados con el CP.

Por simplicidad, escribiremos $\delta(v)$ en lugar de $\delta(\{v\})$ para $v \in S$, y $x(T)$ en lugar de $\sum_{e \in T} x_e$ para $T \subseteq E$.

Seymour [S79] dió la siguiente descripción lineal completa para C_n:

$$x(\delta(S) \setminus \{e\}) - x_e \geq 0 \qquad \text{para todo } S \subset V \text{ y } e \in \delta(S), \tag{2.1}$$

$$x_e \geq 0 \qquad \text{para todo } e \in E. \tag{2.2}$$

Coullard y Pulleyblank [CP89] estudiaron conos similares. Nuestro objetivo es estudiar propiamente el polítopo P_n. Asumiremos que $G = (V, E)$ es un grafo completo.

2.2 El polítopo del CP cuando $|V| \leq 4$

Cuando $n = 3$ entonces P_n es sólo 1 punto (con lo que $dim(P_n) = 0$), descrito por $x_e = 1$ para todo $e \in E$.

Cuando $n = 4$ entonces P_n es la envolvente convexa de 7 puntos afinmente independientes (el Teorema 2.3.1 es también aplicable en este caso), con lo que $\dim(P_n) = 6$. Así, una descripción lineal completa viene dada por los hiperplanos determinados por cada conjunto de 6 de estos puntos, esto es

$$x(T) \geq 2 \qquad \text{para todo ciclo Hamiltoniano } T \text{ en } G, \tag{2.3}$$

$$x(\delta(v)) \leq 2 \qquad \text{para todo } v \in V. \tag{2.4}$$

La desigualdad $x(E) \geq 3$ define la cara impropia P_n cuando $n = 3$, y coincide con la mitad de la suma de las desigualdades (2.3) que definen facetas de P_n. Por tanto, en ningún caso $x(E) \geq 3$ define una faceta de P_n.

2.3 El polítopo del CP cuando $|V| \geq 5$

Teorema 2.3.1 $dim(P_n) = |E|$.

Demostración. En efecto, si $\alpha x^T = \gamma$ para todo $T \in \mathcal{C}$ entonces $\alpha_{[i,j]} = \alpha_{[i,k]} + \alpha_{[k,j]}$ para todas las ternas i, j, k de nodos distintos, y por tanto $\alpha_e = 0$ para todo $e \in E$, y $\gamma = 0$. □

Buscando una descripción lineal entera de los puntos extremos de P_n, podemos concluir que éstos están en correspondencia biyectiva con las soluciones x de

$$x\left(\delta(S) \setminus (\delta(i) \cup \delta(j))\right) - x\left((\delta(i) \cap E(S)) \cup (\delta(j) \cap E(V \setminus S)) \cup [i,j]\right) \geq -2$$

$$S \subset V, i \in S, j \in V \setminus S \tag{2.5}$$

$$x(E) \geq 3 \tag{2.6}$$

$$x_e \in \{0,1\} \qquad e \in E. \tag{2.7}$$

Es decir, $P_n = conv\{x \in I\!R^E : (2.5) - (2.7) \text{ se verifica }\}$. Ahora bien, tratando de justificar esta afirmación, y buscando al tiempo una simplicidad de notación, observemos lo siguiente.

En general, estudiar un polítopo en un espacio afín donde resulta de dimensión máxima, es mejor que estudiarlo en un espacio afín donde el polítopo no tiene esta propiedad; esto se basa en que en este último caso la misma faceta puede aparecer en diversas formas todas equivalentes. Sin embargo, a fin de simplificar la notación y unir el polítopo del CP con el polítopo del TSP, para este problema particular, nosotros creemos más adecuado estudiar P_n en un espacio lineal de mayor dimensión. Consideraremos una variable adicional y_v para cada $v \in V$ asumiendo el valor 1 si el nodo v no está visitado, y 0 en otro caso. Obsérvese que el vector de variables y representa el vector característico de los nodos *no* visitados. Aunque inicialmente parecería más natural utilizar la variable complementaria $\bar{y}_v := 1 - y_v$, hemos optado por usar y_v para mostrar una notación más próxima al TSP.

En el nuevo espacio, la formulación (2.5)–(2.7) se convierte ahora en

$$x(\delta(v)) + 2\,y_v = 2 \qquad v \in V \tag{2.8}$$

$$x(\delta(S)) \geq 2(y_i + y_j - 1) \qquad S \subset V, i \in S, j \in V \setminus S \tag{2.9}$$

$$y(V) \leq |V| - 3 \tag{2.10}$$

$$x_e \in \{0,1\} \qquad e \in E \tag{2.11}$$

$$y_v \in \{0,1\} \qquad v \in V \tag{2.12}$$

donde, si $S \subseteq V$, entonces definimos $y(S) := \sum_{v \in S} y_v$. Ahora es más fácil observar que las ecuaciones (2.8) imponen que todo nodo visitado tiene 2 arcos incidentes, y todo nodo

no-visitado tiene 0 arcos incidentes. Las desigualdades (2.9) imponen que no exista un subtour completamente contenido en un subconjunto S, si fuera de éste existe un nodo j con $y_j = 0$, esto es, visitado. La restricción 2.10 impone que los ciclos factibles usen al menos 3 arcos.

Con algún exceso de notación, $P_n \equiv conv\{(x,y) \in I\!R^{E\cup V} : (2.8) - (2.12) \text{ se verifica }\}$. Con más exactitud, el polítopo de la izquierda equivale a la proyección del polítopo de la derecha en el subespacio de las variables x_e. Tal y como se dice en Balas [B93b], cuando un poliedro Π viene proyectado en un subespacio, las facetas de Π no se proyectan necesariamente en facetas del poliedro proyección. Ahora bien, en nuestro caso particular, si ocurre así puesto que la correspondencia entre ambos polítopos es una restricción de una aplicación lineal f entre los espacios vectoriales $I\!R^E$ y $I\!R^{E\cup V}$ (definida como $f(z) = (x,y)$ con $x_e = z_e$ para todo $e \in E$ y $y_v = 1 - \sum_{e\in\delta(v)} z_e/2$ para todo $v \in V$). Por lo tanto $\{z^1, \ldots, z^k\}$ son linealmente independientes en $I\!R^E$ si y sólo si $\{f(z^1), \ldots, f(z^k)\}$ son linealmente independientes en $I\!R^{E\cup V}$. Por ello a partir de ahora nosotros no distinguiremos entre ambos polítopos.

Estudiemos ahora la estructura poliédrica de P_n. Nótese que la estructura facial de P_n está claramente relacionada con la del polítopo Q del TSP, con quien coincide cuando se imponen las ecuaciones adicionales $y_v = 0$ para todo $v \in V$, es decir

$$Q = P_n \cap \{(x,y) \in I\!R^{E\cup V} : y_v = 0 \text{ para todo } v \in V\}.$$

Siguiendo una idea que se expone también en el Capítulo 3, para unir ambos polítopos definamos el polítopo intermedio

$$P(F) := P_n \cap \{(x,y) \in I\!R^{E\cup V} : y_v = 0 \text{ para todo } v \in F\}$$

cualquiera que sea $\emptyset \subseteq F \subseteq V$. Por definición, $P(V) = Q$ y $P(\emptyset) = P_n$.

Lema 2.3.1 *Para todo $F \subseteq V$,* $\dim(P(F)) = |E| - |F|$.

Demostración. Dado que el sistema de ecuaciones de $P(F)$ incluye el sistema linealmente independientes de las ecuaciones $y_v = 0$ para todo $v \in F$, es claro que $\dim(P(F)) \leq |E| - |F|$. Demostraremos ahora por inducción sobre $\rho := |V \setminus F|$ la existencia de $|E| - |F| + 1$ puntos extremos de $P(F)$ afínmente independientes, con lo que tendremos la otra desigualdad.

Cuando $\rho = 0$, es decir, cuando $F = V$, la afirmación es cierta dado que $P(F)$ se corresponde con el polítopo del TSP (véase, por ejemplo, Grötschel y Padberg [GP85]).

Asumamos ahora que la afirmación es verdad para $\rho = k \geq 0$ y demostrémosla para $\rho' = k + 1$, es decir, para un conjunto de nodos $F \subset V$ tal que $|V \setminus F| = k + 1$.

Consideremos $v \in V \setminus F$ y definamos $F' = F \cup \{v\}$. Debido a la hipótesis de inducción, existen $|E| - |F'| + 1$ puntos afinmente independientes pertenecientes a $P(F')$, y por tanto a $P(F)$. Como $|F'| = |F|+1$ necesitamos un punto extremo adicional de $P(F)$. Tal punto existe y se corresponde con cualquier ciclo Hamiltoniano en el subgrafo inducido por $V \setminus \{v\}$ (recordemos que se asume $n \geq 4$), más el nodo aislado v. □

Re-encontramos ahora el Teorema 2.3.1 como un corolario del resultado previo. Otra inmediata consecuencia es que la eliminación de un nodo de cualquier conjunto de nodos F no vacío incrementa la dimensión de $P(F)$ en exactamente una unidad. Por tanto cualquier desigualdad que defina una faceta para $P(F)$ puede elevarse de una forma simple de manera que induzca una faceta para $P(F \setminus \{v\})$.

Lema 2.3.2 *Sea $F \subseteq V$ y $u \in F$. Sea*

$$\sum_{e \in E} \alpha_e x_e + \sum_{v \notin F} \beta_v y_v \leq \gamma$$

cualquier desigualdad definiendo una faceta para $P(F)$. Entonces la desigualdad elevada

$$\sum_{e \in E} \alpha_e x_e + \sum_{v \notin F} \beta_v y_v + \beta_u y_u \leq \gamma$$

es una desigualdad que induce una faceta de $P(F \setminus \{u\})$, donde β_u se define como

$$\beta_u := \gamma - \max \left\{ \sum_{e \in E} \alpha_e x_e + \sum_{v \notin F} \beta_v y_v \ : \ (x, y) \in P(F \setminus \{u\}), \ \textit{con } y_u = 1 \right\}.$$

Demostración. La afirmación es consecuencia del Teorema de elevación secuencial de Padberg descrito, por ejemplo, en Grötschel y Padberg [GP85]. □

El Lema 2.3.2 lleva a un procedimiento de elevación de desigualdades que definen facetas de P_n a partir de desigualdades que definen facetas para Q. Para alcanzar tal objetivo se debe elegir una *secuencia de elevación* de nodos $\{v_1, v_2, \ldots, v_n\}$, y el resultado se aplica iterativamente para derivar una faceta de $P(\{v_{t+1}, \ldots, v_n\})$ a partir de una faceta de $P(\{v_t, \ldots, v_n\})$ para $t = 1, \ldots, n$. Distintas secuencias de elevación pueden llevar a distintas facetas de P_n a partir de una misma faceta de Q. A continuación usamos este procedimiento de elevación para analizar la estructura facial de P_n.

Comenzamos con las restricciones de no-negatividad.

Teorema 2.3.2 *1. La desigualdad $x_e \geq 0$ define una faceta de P_n para cada $e \in E$.*

2. La desigualdad $y_v \geq 0$ define una faceta de P_n para cada $v \in V$.

Demostración. La primera afirmación es una consecuencia directa del Lema 2.3.2, puesto que $x_e \geq 0$ define una faceta del polítopo del TSP y cada secuencia de elevación produce $\beta_v = 0$ para todo $v \in V$. La segunda es consecuencia del Lema 2.3.1 puesto que la cara de P_n inducida por $y_v \geq 0$ es $P(\{v\})$. □

Si un nodo v no es visitado por una solución factible (x, y) (es decir, si $y_v = 1$) entonces no puede utilizarse ninguno de sus arcos incidentes $e \in \delta(v)$ (es decir, $x_e = 0$). Ello lleva a la siguiente desigualdad válida para P_n:

$$x_e + y_v \leq 1 \quad \text{para todo } v \in V \text{ y } e \in \delta(v). \tag{2.13}$$

Tal desigualdad coincide con una restricción del tipo (2.1) con $|S| = 1$ (o $|V \setminus S| = 1$) cuando se proyectan fuera la variable y_v mediante la ecuación (2.8). Veamos que definen facetas de P_n.

Teorema 2.3.3 *La desigualdad $x_e + y_v \leq 1$ define una faceta de P_n para todo $v \in V$ y $e \in \delta(v)$.*

Demostración. El Teorema 2.3.1 nos proporciona $|E \setminus \delta(v)| + 1$ puntos afinmente independientes que no visitan el nodo v (esto es, con $y_v = 1$ y $x_e = 0$). Nos resta obtener otros $|\delta(v)| - 1$ puntos afinmente independientes usando el arco e (esto es, con $y_v = 0$ y $x_e = 1$). Pero esto es una fácil labor ya que existe un ciclo simple que pasa por el arco e y por cualquier otro arco de $\delta(v) \setminus \{e\}$. Además, ellos también forman, todos juntos, un conjunto afinmente independiente. □

Corolario 2.3.1 *1. La desigualdad $x_e \leq 1$ no define una faceta de P_n para ningún $e \in E$.*

2. La desigualdad $y_v \leq 1$ no define una faceta de P_n para ningún $v \in V$.

Demostración. Evidente ya que cada desigualdad en (2.13) domina estrictamente las restricciones de cota superior $x_e \leq 1$ y $y_v \leq 1$. □

Continuando el análisis de las desigualdades (2.1), cuando $|S| = 2$ (o $|v \setminus S| = 2$) ellas no definen facetas de P_n, ya que coinciden con la suma de dos desigualdades (2.13).

Corolario 2.3.2 *La desigualdad $x(\delta(S) \setminus \{e\}) - x_e \geq 0$ ($e \in \delta(S)$) no define una faceta de P_n cuando $|S| = 2$ o $|V \setminus S| = 2$.*

Demostración. Supongamos $S = \{u, v\}$ y $e = [v, j]$ con $j \in V \setminus S$. Entonces

$$x\left(\delta(S) \setminus \{[u, j]\}\right) - x_{[v,j]} =$$

$$x(\delta(u)) + x(\delta(v)) - 2\, x_{[u,v]} - 2\, x_{[v,j]} =$$

$$(x(\delta(u) \setminus \{f\}) - x_f) + (x(\delta(v) \setminus \{f\}) - x_e)$$

con $f \in \delta(u)$ y $e \in \delta(v)$. □

Cuando $3 \leq |S| \leq |V| - 3$ (y por tanto $n \geq 6$), tampoco siempre (2.1) es una desigualdad que define una faceta de P_n.

Corolario 2.3.3 *La desigualdad $x(\delta(S) \setminus \{e\}) - x_e \geq 0$ ($e \in \delta(S)$) no define una faceta de P_n cuando $|S| = 3$ y $n = 6$.*

Demostración. Supongamos $V = \{u, v, w, i, j, k\}$, $S = \{u, v, w\}$ y $e = [v, j]$. La prueba viene de observar que todo punto extremo de P_n que satisface con igualdad la restricción del enunciado, visita necesariamente alguno de los nodos en S, y por tanto satisface con igualdad

$$x(\delta(v) \setminus \{e\}) + x(\delta(j) \setminus \{e\}) \leq 2.$$

Por lo tanto, los puntos extremos de P_n en el hiperplano $x(\delta(S) \setminus \{e\}) - x_e \geq 0$ también están sobre otro hiperplano diferente. □

En la Sección 2.4 veremos un procedimiento que aprovecha el razonamiento de la demostración anterior para generar una desigualdad que define una faceta de P_n a partir de aquélla que no la define. Finalmente demostraremos que las restantes desigualdades en (2.1) si definen facetas de P_n.

Teorema 2.3.4 *La desigualdad $x(\delta(S) \setminus \{e\}) - x_e \geq 0$ ($e \in \delta(S)$) define una faceta de P_n cuando $S \subset V$, $3 \leq |S| \leq |V| - 3$ y $n \geq 7$.*

Demostración. Consideremos un hipotético hiperplano $\sum_{e \in E} \alpha_e x_e \leq \alpha_0$ que contenga todos los puntos que satisfacen $x(\delta(S) \setminus \{e\}) - x_e = 0$. Veamos que tal hiperplano es necesariamente del tipo $\beta x(\delta(S) \setminus \{e\}) - \beta x_e \leq 0$, para algún $\beta > 0$.

Sin pérdida de generalidad podemos asumir que $|S| \geq 4$. Esto nos garantiza que para todo $f \in E(S)$ se tiene que $\alpha_f = 0$, ya que cualquier terna triángulo $\{f, f', f''\}$ en $E(S)$ se deduce $\alpha_f = \alpha_{f'} + \alpha_{f''}$. Esto mismo conlleva que $\alpha_0 = 0$.

Supongamos que $e = [v, j]$ con $v \in S, j \notin S$. Para cada par de nodos $i, k \in V \setminus S$ $(i, k \not= j)$, se deduce que $\alpha_{[i,j]} = \alpha_{[j,k]} + \alpha_{[i,k]}$, $\alpha_{[j,k]} = \alpha_{[i,j]} + \alpha_{[i,k]}$, y además dado que el

triángulo $\{[i,j],[j,k],[i,k]\}$ es un punto que del hiperplano, $\alpha_{[i,j]} + \alpha_{[j,k]} + \alpha_{[i,k]} = 0$. Y de tal sistema se concluye que $\alpha_g = 0$ para todo $g \in E(V \setminus S)$.

Finalmente, si definimos por $\beta := -\alpha_e$, los puntos triángulos $\{e, f, h\}$ con $f \in E(S)$ conllevan $\alpha_h = \beta$ para todo $h \in \delta(S) \cap \delta(j)$, y los triángulos $\{e, g, h\}$ con $g \in E(V \setminus S)$ conllevan $\alpha_h = \beta$ para todo $h \in \delta(S) \setminus \delta(j)$.

La validez del hiperplano inicial confirma la no-negatividad de β. □

Estudiemos ahora las restricciones (2.9). (obsérvese que siempre podemos asumir que $|S| \leq \lfloor |V|/2 \rfloor$ debido a que S y $V \setminus S$ producen la misma cara de P_n.) Cuando $|S| = 1$ las restricciones (2.9) se reducen a las restricciones de no-negatividad $y_v \geq 0$ para $v \in V$; así ellas definen facetas de P_n debido al Teorema 2.3.2. Cuando $|S| = 2$ las restricciones (2.9) se pueden escribir como $x_e + y_v - y_w \leq 1$ para algún $e \in \delta(v) \setminus \delta(w)$; así ellas están dominadas por miembros de las desigualdades (2.13), y no definen facetas de P_n. Cuando $3 \leq |S| \leq |V| - 3$ las restricciones (2.9) inducen facetas de P_n como muestra el siguiente teorema, resultado de aplicar el procedimiento de elevación a las *restricciones de eliminación de subtour* (SEC's) del TSP (véase, por ejemplo, Grötschel y Padberg [GP85]).

Teorema 2.3.5 *La desigualdad* $x(E(S)) + y(S \setminus \{i\}) - y_j \leq |S| - 1$ *define una faceta de* P_n *para cada* $S \subset V$ *con* $3 \leq |S| \leq |V| - 3$, $i \in S$, $j \in V \setminus S$.

Demostración. Usamos el Lema de elevación 2.3.2. Sea $\{v_1, \ldots, v_n\}$ cualquier secuencia de elevación tal que $v_t \in S$ para $t = 1, \ldots, |S|$, $v_{|S|} = i$, $v_t \in V \setminus S$ para $t = |S| + 1, \ldots, n$, y $v_n = j$. La desigualdad $x(E(S)) + \sum_{v \in V} \beta_v y_v \leq |S| - 1$ induce una faceta para el polítopo del TSP para cualquier elección de los coeficientes β_v. Iterativamente calculamos los coeficientes de elevación $\beta_{v_t} = 1$ para $t = 1, \ldots, |S| - 1$, $\beta_{v_{|S|}} = 0$, $\beta_{v_t} = 0$ para $t = |S| + 1, \ldots, n - 1$, y $\beta_{v_n} = -1$. □

Ahora generalizaremos las *desigualdades peine* del TSP (véase Grötschel y Padberg [GP85] para la definición y propiedades de estas desigualdades que inducen facetas en el TSP). Un *peine* es una familia $(H, T_1, \ldots, T_s)$ de $s + 1$ subconjuntos de nodos tales que:

1. $s \geq 3$ y un número impar;
2. $T_j \cap T_k \neq \emptyset$ para $1 \leq j < k \leq s$;
3. $T_j \cap H \neq \emptyset$ y $T_j \setminus H \neq \emptyset$ para $j = 1, \ldots, s$.

Teorema 2.3.6 *Sea $(H, T_1, \ldots, T_s)$ un peine. Para $j = 1, \ldots, s$ sean a_j y b_j nodos en $T_j \cap H$ y $T_j \setminus H$, respectivamente. Entonces la siguiente desigualdad*

$$(x(E(H)) + y(H)) + \sum_{j=1}^{s} (x(E(T_j)) + y(T_j \setminus \{a_j, b_j\})) \leq (|H| - 1) + \sum_{j=1}^{s} (|T_j| - 1) - \frac{s-1}{2} \tag{2.14}$$

define una faceta para el P_n.

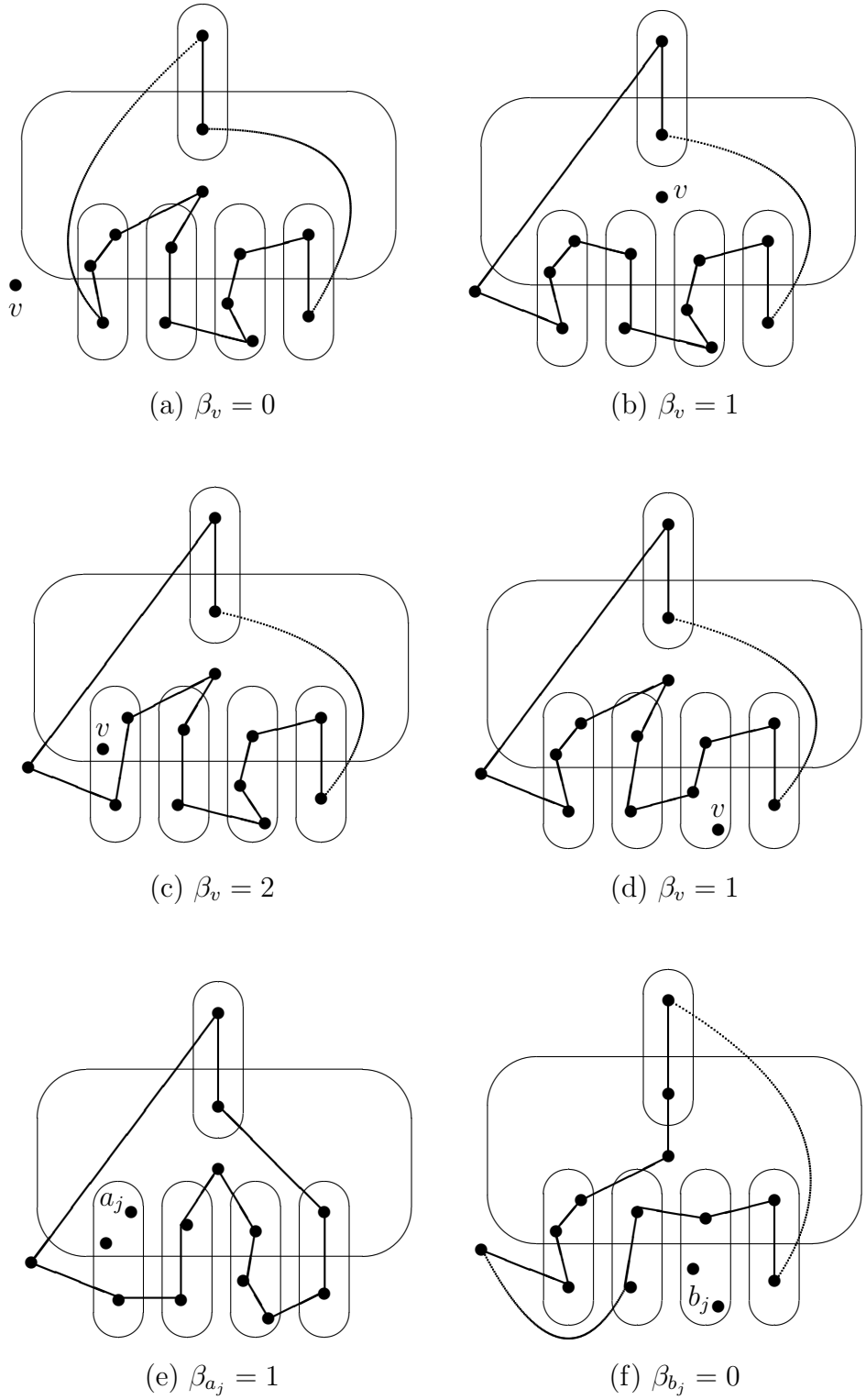
Demostración. Como en la demostración del Teorema 2.3.5, comencemos considerando una secuencia de elevación $\{v_1, \ldots, v_t\}$ de V donde a_j (resp., b_j) sigue a todos los otros nodos en $T_j \cap H$ (resp., $T_j \setminus H$). Los coeficientes $\beta_v = 0$ para $v \in V \setminus (H \cup T_1 \cup \ldots \cup T_k)$ y $\beta_v = 1$ para $v \in H \setminus (T_1 \cup \ldots \cup T_k)$ se calculan fácilmente. La correcteza de los coeficientes $\beta_v = 2$ para $v \in (T_j \cap H) \setminus \{a_j\}$, y $\beta_v = 1$ para $v \in (T_j \setminus H) \setminus \{b_j\}$, puede controlarse con ayuda de las Figuras 2.2a y 2.2b, respectivamente.

En cuanto al valor de β_{a_j} observemos que a_j es el último nodo de la secuencia de elevación dentro de $T_j \cap H$, es decir, cuando se calcula β_{a_j} debemos tener presente la posibilidad de que un ciclo simple evite todos los nodos en $T_j \cap H$. Ello lleva a $\beta_{a_j} = 1$ por cuanto nosotros ya hemos calculado el valor $\beta_v = 2$ para $v \in (T_j \cap H) \setminus \{a_j\}$; véase la Figura 2.2e. De forma similar, cuando se calcula β_{b_j} debemos considerar la posibilidad de tener no cubiertos todos los nodos en $T_j \setminus H$. Ello lleva a $\beta_{b_j} = 0$ debido al ciclo simple que se presenta en la Figura 2.2f. □

Como ya se ha indicado, en la misma línea todas las desigualdades que definen facetas para el polítopo del TSP producen desigualdades que definen facetas para el polítopo del CP, sin más que aplicar el procedimiento del Lema 2.3.2. Ahora bien, ello lleva intrínseca la difícil tarea de calcular los coeficientes β_u que no se conocen de antemano, y que deben ser calculados resolviendo un particular CP. Resulta pues de gran utilidad desarrollar otro procedimiento alternativo de elevación, y esto lo haremos en la próxima sección. La inspiración para ello aparece en la demostración del siguiente resultado referente a las desigualdades (2.10), que ahora sí definen facetas de P_n. Como veremos, otra forma alternativa de alcanzar la estructura poliédrica de P_n diferente de aquella que parte del polítopo del TSP (donde todos los nodos tienen $y_v = 0$), será partir del propio P_n para un n menor (donde casi todos los nodos tienen $y_v = 1$), y proceder igualmente por inducción.

2.4 Un procedimiento de elevación

En lugar de relacionar la estructura facial de P_n con aquélla del bien conocido polítopo Q del TSP (debido a que Q es una relajación de P_n), ahora estamos interesados en describir



(a) $\beta_v = 0$ (b) $\beta_v = 1$

(c) $\beta_v = 2$ (d) $\beta_v = 1$

(e) $\beta_{a_j} = 1$ (f) $\beta_{b_j} = 0$

Figura 2.2: Ciclos simples ajustados para la demostración del Teorema 2.3.6.

un procedimiento de elevación inductivo sobre n para producir una faceta de P_n. Para ello, introducimos antes algunas definiciones básicas.

Definición 2.4.1 *Sea $\alpha x \leq \beta y + \gamma$ una desigualdad válida para P_n, y v un nodo arbitrario pero fijo. Denotamos con $\mathcal{H}(\alpha, \beta, \gamma) := P_n \cap \{(x, y) \in \mathbb{R}^{E \cup V} : \alpha x + \beta y = \gamma\}$ la cara de P_n inducida por $\alpha x \leq \beta y + \gamma$. La* cara v-restricción *inducida por $\alpha x \leq \beta y + \gamma$ es la cara $\mathcal{H}_v(\alpha, \beta, \gamma) := \mathcal{H}(\alpha', \beta', \gamma')$ donde $\alpha'_e = \alpha_e$ para $e \in E \setminus \delta(v)$, $\alpha'_e = 0$ para $e \in \delta(v)$, $\beta'_u = \beta_u$ para $u \in V \setminus \{v\}$, $\beta'_v = 0$, y $\gamma' = \gamma - \beta_v$. El* grafo v-compatible *de $\alpha x \leq \beta y + \gamma$ es el grafo $G_v(\alpha, \beta, \gamma) = (V \setminus \{v\}, E^*)$ con $[u, w] \in E^*$ si y sólo si existe $(x, y) \in \mathcal{H}(\alpha, \beta, \gamma)$ con $x_{[v,u]} = x_{[v,w]} = 1$. El* rango *de un grafo se define como el rango de su matriz de incidencia arco-nodo, esto es, el número de sus nodos menos el número de sus componentes bipartitas conexas. El grafo se llama de* rango máximo *cuando su matriz de incidencia arco-nodo es de rango máximo, esto es, cuando $G_v(\alpha, \beta, \gamma)$ tiene un ciclo impar en cada componente conexa.*

Lema 2.4.1 *Para cada desigualdad válida $\alpha x \leq \beta y + \gamma$ de P_n y cada nodo $v \in V$, la dimensión de $\mathcal{H}(\alpha, \beta, \gamma)$ es mayor o igual a la dimensión de la cara v-restricción, más el rango de su grafo v-compatible.*

Demostración. Sea Z una matriz en la que cada fila es un punto extremo de $\mathcal{H}(\alpha, \beta, \gamma)$. Puesto que $\mathcal{H}(\alpha, \beta, \gamma)$ está contenido en un hiperplano que no pasa por el origen (por ejemplo, aquel inducido por (2.10)), un subconjunto de filas de Z será afínmente independiente si y sólo si éste es linealmente independiente. Así la dimiensión de $\mathcal{H}(\alpha, \beta, \gamma)$ coincide con el rango de Z menos 1. Ahora Z puede verse dividida en

$$Z = \begin{bmatrix} Z_{11} & 0 & 1 \\ Z_{21} & Z_{22} & 0 \end{bmatrix},$$

donde la última columna se corresponde con la variable y_v, y las columnas de Z_{22} se corresponden con las variables x_e para $e \in \delta(v)$. Entonces el rango de Z es al menos la suma del rango de Z_{11} más el rango de $[Z_{22}\ 1]$. Por construcción, el rango de Z_{11} es la dimensión de la cara v-restricción inducida por $\alpha x \leq \beta y + \gamma$, más 1. En cuanto a Z_{22}, observamos que cada una de sus filas contiene exactamente 2 unos y (salvo filas repetidas) puede verse como una matriz de incidencia arco-nodo del grafo v-compatible $G_v(\alpha, \beta, \gamma)$ asociado con $\alpha x \leq \beta y + \gamma$. Además, la última columna de $[Z_{22}\ 1]$ es una combinación lineal (con coeficientes $1/2$) de las otras columnas. Esto concluye la demostración. □

Corolario 2.4.1 *Sea $\alpha x \leq \beta y + \gamma$ una desigualdad válida para P_n y sea v un nodo tal que la cara v-restricción inducida por $\alpha x \leq \beta y + \gamma$ es una faceta de P_{n-1}. Entonces $\alpha x \leq \beta y + \gamma$ define una faceta de P_n si y sólo si su grafo v-compatible asociado, $G_v(\alpha, \beta, \gamma)$, es de rango máximo.*

Estas conclusiones proponen demostraciones alternativas para los teoremas de la Sección 2.3 que no expondremos por considerarlas ahora redundantes. Ilustraremos, sin embargo, la sencillez de esta nueva metodología de demostraciones sobre un interesante resultado, nada trivial de demostrar por la vía clásica.

Teorema 2.4.1 *Si $n \geq 5$ entonces $y(V) \leq |V| - 3$ define una faceta de P_n.*

Demostración. Si $n = 5$, entonces los puntos extremos que verifican $y(V) = |V| - 3$ son los triángulos en E, de los que hay uno asociado con cada arco $e = [u, v] \in E$ (aquél con $y_u = y_v = 1$), y son además todos afinmente independientes (el determinante de sus vectores característicos es no nulo). Para concluir la demostración basta aplicar ahora el Corolario 2.4.1, observando que el grafo v-compatible asociado es un grafo completo. □

Como se muestra en el Lema 2.4.1, el rango del grafo v-compatible $G_v(\alpha, \beta, \gamma)$ asociado con una desigualdad $\alpha x \leq \beta y + \gamma$ dada juega un papel centra cuando se analiza la estructura poliédrica de P_n. Desgraciadamente, determinar si un arco está o no presente en $G_v(\alpha, \beta, \gamma)$ exige la construcción de una cierta solución del CP (x, y) con $\alpha x + \beta y = \gamma$, por lo que se trata de un problema $\mathcal{NP}$-difícil en general. En la práctica, uno sólo está interesado en encontrar condiciones suficientes para la pertenencia de un arco en $G_v(\alpha, \beta, \gamma)$.

Sea $\alpha x \leq \beta y + \gamma$ una desigualdad. Para $v \in V$, denotamos por

$$\Delta(v) := \{[i, j] \in E \setminus \delta(v) \ : \ \beta_v + \alpha_{ij} = \alpha_{iv} + \alpha_{jv}\}$$

el conjunto de los arcos *ajustados* para v. Recordemos que una cara $\mathcal{H}$ de P_n se llama *trivial* cuando $\mathcal{H} \subseteq \{(x, y) \in I\!R^{E \cup V} \ : \ x_e = 0\}$ para algún $e \in E$; *no-trivial* en otro caso.

Lema 2.4.2 *Sea $v \in V$, y $\alpha x \leq \beta y + \gamma$ una desigualdad válida definiendo una cara v-restricción no-trivial de P_n. Entonces $G_v(\alpha, \beta, \gamma)$ contiene todos los arcos en $\Delta(v)$.*

Demostración. Sea $[i, j]$ cualquier arco en $\Delta(v)$. A partir de la hipótesis de no-trivialidad, existe un punto extremo $(x, y) \in \mathcal{H}(\alpha, \beta, \gamma)$ que usa el arco $[i, j]$. Si este punto visita v la afirmación se verifica trivialmente; si no es así, entonces comenzando con esta solución construimos un nuevo punto $(\tilde{x}, \tilde{y})$ sustituyendo el arco $[i, j]$ por $[i, v]$ y $[j, v]$. Entonces $\alpha\tilde{x} + \beta\tilde{y} = \alpha x + \beta y + (\alpha_{iv} + \alpha_{jv} - \alpha_{ij} - \beta_v) = \alpha x + \beta y$, ya que $[i, j] \in \Delta(v)$. Así pues, $(\tilde{x}, \tilde{y})$ está en $\mathcal{H}(\alpha, \beta, \gamma)$ y visita v. Se concluye así que $[i, j] \in G_v(\alpha, \beta, \gamma)$. □

Del Corolario 2.4.1 también podemos derivar otros útiles resultados. Sea $\alpha x \leq \beta y + \gamma$ una desigualdad válida para P_n y sea v cualquier nodo tal que la cara v-restricción inducida

por $\alpha x \leq \beta y + \gamma$ es una faceta de P_{n-1}. Si $\alpha x \leq \beta y + \gamma$ no define una faceta de P_n, entonces el corolario anterior dice que una componente conexa de $G_v(\alpha, \beta, \gamma)$ es bipartita. Mostraremos cómo pueden cambiarse los coeficientes α_e ($e \in \delta(v)$) para producir una desigualdad "más fuerte", definiendo una cara de P_n de mayor dimensión que antes. Para ello, sea $S \subseteq V \setminus \{v\}$ el conjunto de nodos de una componente bipartita de $G_v(\alpha, \beta, \gamma)$, dividido en entre los conjuntos S_1 y S_2 tales que $|S_1| \geq |S_2|$. Entonces cada $(x, y) \in \mathcal{H}(\alpha, \beta, \gamma)$ verifica la ecuación:

$$\bar{\alpha}x := \sum_{e \in \delta(S_1) \cap \delta(v)} x_e - \sum_{e \in \delta(S_2) \cap \delta(v)} x_e = 0.$$

Claramente, $\bar{\alpha}x \leq 0$ no es una desigualdad válida para P_n, puesto que está violada por cualquier solución del CP que use los dos arcos $[i, v]$ y $[j, v]$, donde i es un nodo cualquiera elegido en S_1, y j está elegido arbitrariamente en $V \setminus (S_2 \cup \{v, i\})$ (la existencia de estos dos nodos deriva de las hipótesis $|S_1| \geq |S_2|$ y $n \geq 5$). Se sigue que $\bar{\alpha}x = 0$ no es una combinación lineal de las ecuaciones (2.8). Además, no es una combinación lineal de (2.8) y $\alpha x + \beta y = \gamma$ ya que la cara v-restricción inducida por $\bar{\alpha}x \leq 0$ es P_{n-1}, mientras que se ha asumido que la cara v-restricción inducida por $\alpha x \leq \beta y + \gamma$ es una cara propia.

Consideremos ahora la desigualdad elevada

$$(\alpha + \epsilon\bar{\alpha})x + \beta y \leq \gamma, \tag{2.15}$$

donde $\epsilon \geq 0$ es un parámetro real. Nótese que $\epsilon = 0$ lleva a la desigualdad válida $\alpha x \leq \beta y + \gamma$, mientras que cuando ϵ tiende a $+\infty$ la desigualdad (2.15) tiende a la desigualdad no-válida $\bar{\alpha}x \leq 0$. Sea ϵ^* el máximo valor de ϵ tal que (2.15) es válida. Este valor puede calcularse como

$$\epsilon^* = \min \left\{ \frac{\gamma - \alpha x - \beta y}{\bar{\alpha}x} \ : \ (x, y) \in P_n,\ \bar{\alpha}x > 0 \right\},$$

es decir

$$\epsilon^* = \min \left\{ \epsilon^1, \epsilon^2 \right\},$$

donde

$$\begin{aligned} \epsilon^1 &:= \frac{1}{2} \min \left\{ \Delta_e \ : \ e \in E(S_1) \right\}, \\ \epsilon^2 &:= \min \left\{ \Delta_e \ : \ e \in \delta(S_1) \setminus \delta(\{v\} \cup S_2) \right\}, \end{aligned}$$

y para todo $[i, j] \in E \setminus \delta(v)$ con $i \in S_1$ y $j \notin S_2$,

$$\Delta_{[i,j]} := \min \left\{ \gamma - \alpha x - \beta y \ : \ (x, y) \in P_n,\ x_{[i,v]} = x_{[j,v]} = 1 \right\} (\geq 0).$$

Para cualquier $\epsilon \in [0, \epsilon^*]$, $\mathcal{H}(\alpha + \epsilon\bar{\alpha}, \beta, \gamma)$ es una cara propia de P_n (debido a que (2.8), $\alpha x + \beta y = \gamma$, y $\bar{\alpha}x = 0$ son ecuaciones linealmente independientes) que contienen

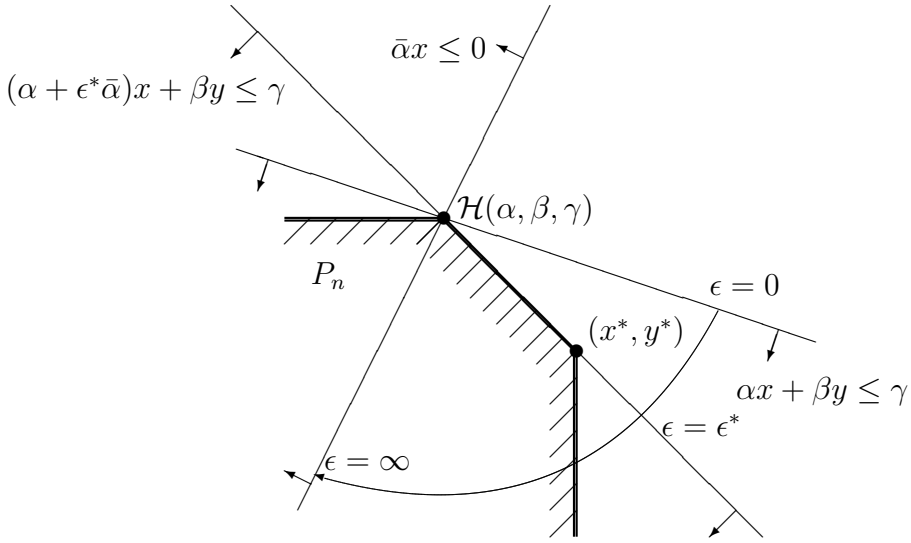


Figura 2.3: Geometría del procedimiento de elevación.

a $\mathcal{H}(\alpha, \beta, \gamma)$. Además, $\mathcal{H}(\alpha + \epsilon^*\bar{\alpha}, \beta, \gamma)$ contiene también un punto $(x^*, y^*) \in P_n$ tal que $\bar{\alpha}x^* > 0$ (es decir, el punto que determina el valor ϵ^*), lo que prueba que $\dim(\mathcal{H}(\alpha + \epsilon^*\bar{\alpha}, \beta, \gamma)) \geq \dim(\mathcal{H}(\alpha, \beta, \gamma)) + 1$. La geometría del procedimiento de elevación anterior se ilustra en la Figura 2.3.

Iterando el procedimiento anterior podemos obtener una desigualdad que induce una faceta para P_n a partir de una que define una faceta en P_{n-1}. Esto motiva el estudio específico de P_n para pequeños valores de n (digamos $n = 6$ o $n = 7$).

Capítulo 3

Problema del Viajante de Comercio Generalizado: Su Poliedro

El Problema del Viajante de Comercio Generalizado simétrico (GTSP) es una variante del clásico Problema del Viajante de Comercio simétrico (TSP), en el que los nodos están divididos en grupos (*clusters*) y el viajante tiene que visitar al menos un nodo de cada cluster. Una versión diferente de este problema, llamada E-GTSP, aparece cuando se debe visitar exáctamente un nodo de cada cluster. Tanto el GTSP como el E-GTSP son problemas $\mathcal{NP}$-difíciles, y tienen aplicaciones prácticas tanto en problemas generales de rutas como de secuenciación. En este capítulo modelizamos GTSP y E-GTSP como problemas de programación lineal entera, y estudiamos la estructura poliédrica de sus correspondientes polítopos. En el Capítulo 4 se utilizan los resultados que aquí se muestran para diseñar un algoritmo de Ramificación y Corte que encuentra y demuestra la solución óptima de problemas de la literatura con hasta 442 nodos.

3.1 Introducción

A menudo, los Problemas de Rutas y Secuenciación requieren la determinación de secuencias óptimas que verifiquen un determinado conjunto de restricciones. El más conocido de tales problemas es el clásico *Problema del Viajante de Comercio* (TSP), donde se plantea la búsqueda de un ciclo Hamiltoniano de costo mínimo sobre un grafo determinado. Este problema ha sido extensamente estudiado en las últimas décadas; véase el libro editado por Lawler, Lenstra, Rinnooy Kan, y Shmoys [LLRS85] para los últimos resultados sobre el TSP hasta 1985.

En varias aplicaciones se permite que un ciclo factible visite solo un subconjunto de nodos del grafo, elegidos según un criterio pre-especificado. Por ejemplo, en el *Problema de la Recolección de Premios* (*Prize Collecting TSP*) cada nodo tiene un premio asociado, y un vendedor busca un ciclo de costo mínimo para cubrir un subconjunto de nodos cuyo premio total no sea menor a un valor predeterminado. Este problema ha sido estudiado, entre otros, por Balas [B89, B93a], y Fischetti y Toth [FT88].

En este capítulo consideramos una versión diferente (tampoco Hamiltoniana) del clásico TSP, conocido en la literatura como el *Problema del Viajante de Comercio Generalizado* (*Generalized Travelling Salesman Problem*) (GTSP), que puede definirse como sigue. Consideremos un grafo no dirigido completo $G = (N, E)$ con $N := \{1, \ldots, n\}$ su conjunto de nodos, y $E := \{\, [i, j] \,:\, i, j \in N, i \neq j\}$ su conjunto de arcos. Consideremos, además, una partición propia $C_1, \ldots, C_m$ de N dada, donde los subconjuntos C_h se llamarán *clusters*. Asumiremos que $m \geq 5$ a lo largo del capítulo. Sea c_e el *costo* asociado con cada arco $e \in E$. Un ciclo se llama *factible* si y sólo si es simple y visita al menos un nodo de cada cluster. El GTSP se plantea encontrar un ciclo factible $T \subset E$ cuyo costo global $\sum_{e \in T} c_e$ sea mínimo. Tal problema conlleva dos decisiones:

i) elegir un subconjunto de nodos $S \subseteq N$, tal que $|S \cap C_h| \geq 1$ para todo $h = 1, \ldots, m$;

ii) encontrar un ciclo Hamiltoniano de costo mínimo en el subgrafo de G inducido por S.

Una versión distinta del problema, llamada E-GTSP en lo que sigue, aparece cuando se impone la restricción adicional de que se debe visitar Exactamente un nodo de cada cluster. Nótese que GTSP y E-GTSP son equivalentes cuando los costos satisfacen la desigualdad triangular, es decir, $c_{ij} \leq c_{ik} + c_{kj}$ para toda tripleta de nodos (i, j, k).

GTSP y E-GTSP son útiles modelos para problemas que suponen simultáneamente las decisiones de selección y secuenciación, tales como por ejemplo en problemas de localización-ruta. Encuentran aplicaciones prácticas en compañías dedicadas al reparto de mercancía con múltiples locales de almacenamiento, en la secuenciación de ficheros de ordenador, rutas de clientes a través de agencias gubernamentales, selección de aeropuertos y rutas de planes turísticos, secuenciación de manofacturas flexibles, distribución postal, y diseño de redes de ordenador; véase Noon [N88], y Noon y Bean [NB91].

Ambos GTSP y E-GTSP son claramente $\mathcal{NP}$-difíciles, puesto que se reducen al TSP cuando $m = n$, esto es, $|C_h| = 1$ para todo h. Han sido estudiados, entre otros, por Laporte y Nobert [LN83], Salazar [S92], y Sepehri [S91]; la versión asimétrica ha sido investigada en Laporte, Mercure y Nobert [LMN87], y Noon y Bean [NB91].

Ahora se introduce la notación principal usada posteriormente. Para cada $S \subseteq N$, sea

$$\begin{aligned} E(S) &:= \{\, [i,j] \in E \,:\, i \in S\,,\, j \in S\}, \\ \delta(S) &:= \{\, [i,j] \in E \,:\, i \in S\,,\, j \notin S\}, \\ \mu(S) &:= |\{h \,:\, C_h \subseteq S\}|, \\ \eta(S) &:= |\{h \,:\, C_h \cap S \neq \emptyset\}|. \end{aligned}$$

Para $v \in N$ escribimos $\delta(v)$ en lugar de $\delta(\{v\})$, y denotamos por $C_{h(v)}$ el cluster conteniendo v. También definimos

$$W := \{v \in N \,:\, |C_{h(v)}| = 1\}.$$

Un modelo de programación lineal entera para GTSP es el siguiente. Sea $x_e = 1$ si el arco $e \in E$ aparece en la solución óptima, $x_e = 0$ en otro caso. Además, sea $y_v = 1$ si el nodo $v \in N$ es visitado, $y_v = 0$ en otro caso. GTSP entonces equivale a

$$v(GTSP) := \min \sum_{e \in E} c_e\, x_e \tag{3.1}$$

sujeto a

$$\sum_{e \in \delta(v)} x_e = 2\, y_v \qquad \text{para } v \in N \tag{3.2}$$

$$\sum_{v \in C_h} y_v \geq 1 \qquad \text{para } h \,=\, 1, \ldots, m \tag{3.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\,(y_i + y_j - 1) \qquad \text{para} \quad \begin{array}{l} S \subset N,\, 2 \leq |S| \leq n-2, \\ i \in S,\, j \in N \setminus S \end{array} \tag{3.4}$$

$$x_e \in \{0,1\} \qquad \text{para } e \in E \tag{3.5}$$

$$y_v \in \{0,1\} \qquad \text{para } v \in N. \tag{3.6}$$

Las restricciones (3.2) requieren que el número de aristas incidentes con un nodo sea o 2 (si v es visitado) o 0 (otro caso). Las restricciones (3.3) imponen que al menos un nodo de cada cluster sea visitado. Las desigualdades (3.4) son las restricciones de conexión, imponiendo que cada corte separando dos vértices visitados (i y j) debe ser atravesado al menos dos veces.

Las restricciones (3.4) son válidas siempre que uno busca un ciclo (no necesariamente Hamiltoniano); su contrapartida en el caso asimétrico ha sido estudiada por Balas [B89, B93a] para el Problema de la Recolección de Premios. Para los problemas GTSP y E-GTSP se pueden derivar de éstas, otras desigualdades más fuertes que explotan la particular estructura de estos problemas (véase la Sección 3.2).

Un modelo matemático para el E-GTSP se obtiene sustituyendo en (3.1)–(3.6) las desigualdades (3.3) con las igualdades

$$\sum_{v \in C_h} y_v = 1 \quad \text{para } h = 1, \ldots, m. \tag{3.7}$$

Denotamos por P, $P^=$, y Q a los polítopos de los *GTSP*, *E-GTSP*, y *TSP*, respectivamente, definidos como

$$P := \text{ conv}\{(x,y) \in I\!R^{E \cup N} : \text{(3.2), (3.3), (3.4), (3.5), (3.6) se verifican}\},$$
$$P^= := P \cap \{(x,y) \in I\!R^{E \cup N} : \text{(3.7) se verifica}\},$$

y

$$Q := P \cap \{(x,y) \in I\!R^{E \cup N} : y_v = 1 \text{ para todo } v \in N\}.$$

Claramente, $P^=$ y Q son caras de P. Estas caras son disjuntas cuando $m < n$, mientras que para $m = n$ los tres polítopos P, $P^=$, y Q coinciden.

En este capítulo analizamos la estructura facial de P y de $P^=$. En la Sección 3.2 introducimos un teorema general que permite elevar cualquier faceta de Q hacia una faceta de P. Este resultado se usa para derivar familias de desigualdades que inducen facetas para P relativas a las restricciones de *eliminación de subtour* y *peine*. En la Sección 3.3 analizamos $P^=$ y discutimos los casos en los que las desigualdades de la Sección 3.2 inducen facetas para $P^=$. Nuestro análisis se basa en un resultado general que une la estructura poliédrica de $P^=$ a aquélla del polítopo del E-GTSP asociado con un ejemplo obtenido eliminando un nodo de un cluster con más de un vértice (cluster no simple). Esto nos permite inductivamente reducir el análisis poliédrico de $P^=$ a aquél de Q. También describimos un procedimiento para elevar facetas de Q a facetas de $P^=$.

Los resultados teóricos aquí descritos se usan en el Capítulo 4 para diseñar un algoritmo de ramificación y corte que resuelve exactamente ejemplos con hasta 442 nodos. El algoritmo se basa en procedimientos de separación exactos y heurísticos para las principales familias de desigualdades que se analizan en este capítulo. Los análisis computacionales que allí se muestran nos permiten asegurar la substancial mejora que las desigualdades que aquí se estudian producen sobre la relajación lineal del modelo (3.1)–(3.6).

Para favorecer la simplicidad, en lo que sigue no distinguiremos entre una solución del GTSP (o del E-GTSP) y su vector característico.

3.2 Facetas del polítopo del GTSP

En esta sección estudiamos el polítopo del GTSP, P. La estructura facial de P está claramente relacionada con aquélla del polítopo del TSP, Q, que aparece cuando se imponen

las ecuaciones adicionales $y_v = 1$ para todo $v \in N$. A fín de unir estos dos polítopos, definamos los polítopos intermedios

$$P(F) := P \cap \{(x, y) \in I\!R^{E \cup N} \, : \, y_v = 1 \text{ para todo } v \in F\},$$

donde $\emptyset \subseteq F \subseteq N$. Por definición, $P(N) = Q$ y $P(\emptyset) = P$.

Nuestro primer objetivo es determinar la dimensión de $P(F)$ para cualquier F. Esto nos lleva a estudiar el sistema de ecuaciones que determina $P(F)$. Tal sistema incluye las $|N|$ ecuaciones linealmente independientes (3.2), más las ecuaciones que fijan las variables

$$y_v = 1 \qquad \text{para todo } v \in F \cup W, \tag{3.8}$$

donde W ha sido definido en la Sección 3.1. Pero además, no existe ninguna otra ecuación linealmente independiente que sea verificada por todos los puntos de $P(F)$, tal y como se deduce del siguiente resultado.

Teorema 3.2.1 *Para todo $F \subseteq N$, $dim(P(F)) = |E| - |F \cup W|$.*

Demostración. Claramente $\dim(P(F)) \leq |E| - |F \cup W|$ puesto que $P(F) \subset I\!R^{E \cup N}$ y las $|N| + |F \cup W|$ ecuaciones válidas (3.2) y (3.8) son linealmente independientes. Afirmamos que existen $|E| - |F \cup W| + 1$ puntos afinmente independientes en $P(F)$. Demostrado esto tendremos que $\dim(P(F)) \geq |E| - |F \cup W|$, y por tanto el enunciado del teorema. La demostración de nuestra afirmación se realiza por inducción sobre el cardinal de F.

Cuando $|F| = n$ la afirmación es verdad, puesto que $P(F)$ se corresponde con el polítopo del TSP (véase, por ejemplo, Grötschel y Padberg [GP85]).

Asumamos ahora que la afirmación es correcta para $|F| = \sigma$, y consideremos cualquier conjunto de nodos F' con $|F'| = \sigma - 1$. Sea v cualquier nodo fuera de F', y definamos $F := F' \cup \{v\}$. Por la hipótesis de inducción, existen $|E| - |F \cup W| + 1$ puntos afinmente independientes pertenecientes a $P(F)$ y por tanto también a $P(F')$. Si $v \in W$ entonces $|F \cup W| = |F' \cup W|$, y hemos terminado. En otro caso, $|F \cup W| = |F' \cup W| + 1$, es decir, necesitamos un punto adicional. Tal punto existe siempre, y se corresponde con un ciclo Hamiltoniano en el subgrafo inducido por $N \setminus \{v\}$. □

Corolario 3.2.1 *$dim(P) = |E| - |W|$.*

Según el Teorema 3.2.1, dado un conjunto no vacío $F \subseteq N$ y cualquier $v \in F$ tenemos lo siguiente: si $v \in W$ entonces $\dim(P(F \setminus \{v\})) = \dim(P(F))$, en otro caso $\dim(P(F \setminus \{v\})) = \dim(P(F)) + 1$. En otras palabras, la eliminación de un nodo de F incrementa la dimensión de $P(F)$ en, como mucho, una unidad. Como consecuencia, cualquier desigualdad que defina una faceta para $P(F)$ puede ser elevada de una forma simple hasta definir también una faceta para $P(F \setminus \{v\})$.

Teorema 3.2.2 *Sea $F \subseteq N$ y $u \in F$. Sea*

$$\sum_{e \in E} \alpha_e \, x_e \, + \sum_{v \in N} \beta_v \, (1 - y_v) \, \geq \, \gamma$$

cualquier desigualdad definiendo una faceta para $P(F)$. Entonces la desigualdad elevada

$$\sum_{e \in E} \alpha_e \, x_e \, + \sum_{v \in N \setminus \{u\}} \beta_v \, (1 - y_v) \, + \, \tilde{\beta}_u \, (1 - y_u) \, \geq \, \gamma$$

es válida y define una faceta para $P(F \setminus \{u\})$, donde $\tilde{\beta}_u$ es un valor arbitrario si $u \in W$, mientras que

$$\tilde{\beta}_u = \gamma - \min \left\{ \sum_{e \in E} \alpha_e \, x_e \, + \sum_{v \in N \setminus \{u\}} \beta_v \, (1 - y_v) \, : \, (x, y) \in P(F \setminus \{u\}), \quad y \;\; y_u = 0 \right\}$$

se verifica cuando $u \notin W$.

Demostración. La afirmación es una consecuencia del conocido teorema de elevación secuencial (*sequential lifting theorem*, Padberg [P75]), descrito por ejemplo en Grötschel y Padberg [GP85]. □

El Teorema 3.2.2 lleva a un procedimiento de elevación útil para derivar desigualdades que definen facetas para el polítopo del GTSP a partir de aquéllas del polítopo del TSP. Para ello debemos elegir cualquier *secuencia de elevación* de los nodos, digamos $\{v_1, \ldots, v_n\}$, e iterativamente derivamos una faceta de $P(\{v_{t+1}, \ldots, v_n\})$ a partir de una faceta de $P(\{v_t, \ldots, v_n\})$ para $t = 1, \ldots, n$. Distintas secuencias de elevación pueden producir distintas facetas.

Ahora usaremos el procedimiento de elevación para analizar la estructura facial de P. Comenzaremos con las restricciones de no-negatividad.

Teorema 3.2.3 *Las desigualdades $x_e \geq 0$ definen una faceta de P para todo $e \in E$.*

Demostración. Es consecuencia directa del Teorema 3.2.2, ya que $x_e \geq 0$ define una faceta del polítopo del TSP y cada secuencia de elevación produce $\tilde{\beta}_v = 0$ para todo $v \in N$. □

Nótese que las desigualdades $y_v \geq 0$ están dominadas por las desigualdades válidas $y_v \geq x_e$ para $e \in \delta(v)$, por lo que ellas no definen facetas.

Ahora analizaremos las restricciones de cota superior sobre las variables, y las restricciones (3.3).

Teorema 3.2.4 *La desigualdad $x_e \leq 1$ define una faceta de P si y sólo si $e \in E(W)$.*

Demostración. Sea $e = [u, v]$. Si $u, v \in W$ entonces la afirmación es consecuencia del Teorema 3.2.7 eligiendo $S = \{u, v\}$. En otro caso $x_e \leq 1$ está dominado por $x_e \leq y_u$ (si $u \notin W$) o por $x_e \leq y_v$ (si $v \notin W$). □

Teorema 3.2.5 *La desigualdad $y_v \leq 1$ define una faceta de P si y sólo si $v \notin W$.*

Demostración. Es suficiente observar que la cara de P inducida por $y_v \leq 1$ coincide con $P(F)$ cuando $F := \{v\}$, con lo que la afirmación se sigue del Teorema 3.2.1. □

Teorema 3.2.6 *La desigualdad $\sum_{v \in C_h} y_v \geq 1$ no define una faceta de P para ningún $h = 1, \ldots, m$.*

Demostración. Debido a (3.2), la restricción (3.3) es equivalente a

$$\sum_{e \in \delta(C_h)} x_e + 2 \sum_{e \in E(C_h)} x_e \geq 2.$$

Por lo tanto (3.3) está dominada por la desigualdad válida $\sum_{e \in \delta(C_h)} x_e \geq 2$ cuando $E(C_h) \neq \emptyset$, mientras que para $E(C_h) = \emptyset$ (es decir, cuando $|C_h| = 1$) define la cara impropia P. □

Ahora aplicaremos el procedimiento de elevación a las bien conocidas *restricciones de eliminación de subtour* del TSP (en su forma de corte):

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{para } S \subset N,\ 2 \leq |S| \leq n - 2.$$

Nótese que la desigualdad anterior no cambia cuando S se reemplaza por $N \setminus S$, con lo que podemos asumir sin pérdida de generalidad $|S| \leq \lfloor n/2 \rfloor$. Las restricciones de eliminación de subtour definen facetas del polítopo del TSP (véase Grötschel y Padberg [GP85]).

Teorema 3.2.7 *Sea $S \subset N$ con $2 \leq |S| \leq n-2$. La siguiente* restricción de Eliminación de Subtour *(GSEC) es válida y define una faceta para P:*

$$\sum_{e \in \delta(S)} x_e + \tilde{\beta}_i (1 - y_i) + \tilde{\beta}_j (1 - y_j) \geq 2 \quad \textit{para } i \in S,\ j \in N \setminus S,$$

donde

$$\tilde{\beta}_i := \begin{cases} 2 & \textit{si } \mu(S) = 0, \\ 0 & \textit{en otro caso;} \end{cases}$$

$$\tilde{\beta}_j := \begin{cases} 2 & \textit{si } \mu(N \setminus S) = 0, \\ 0 & \textit{en otro caso;} \end{cases}$$

véase la Sección 3.1 para la definición de $\mu(\cdot)$.

Demostración. Usemos el Teorema 3.2.2 de elevación. Sea $\{v_1, \ldots, v_n\}$ cualquier secuencia de elevación tal que $v_{n-1} = i$ y $v_n = j$. La desigualdad $\sum_{e \in \delta(S)} x_e + \sum_{v \in N} \beta_v (1 - y_v) \geq 2$ define una faceta para el polítopo del TSP, cualquiera que sea la elección de β. Iterativamente calculamos los coeficientes de elevación $\tilde{\beta}_{v_t} = 0$ para $t = 1, \ldots, n-2$. Para $\tilde{\beta}_{v_{n-1}}$, obtenemos $\tilde{\beta}_{v_{n-1}} = 2$ si existe una solución factible del GTSP que no visite nodos en S (es decir, si no existe $C_h \subseteq S$); $\tilde{\beta}_{v_{n-1}} = 0$ en otro caso. Análogamente, $\tilde{\beta}_{v_n} = 2$ si no existe $C_h \subseteq N \setminus S$; $\tilde{\beta}_{v_n} = 0$ en otro caso. □

El teorema anterior produce las tres familias siguientes de desigualdades que definen facetas para el GTSP. Sea $S \subset N$ tal que $2 \leq |S| \leq n - 2$.

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{para } \mu(S) \neq 0, \mu(N \setminus S) \neq 0, \tag{3.9}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\, y_i \qquad \text{para } \mu(S) = 0, \mu(N \setminus S) \neq 0, i \in S, \tag{3.10}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\,(y_i + y_j - 1) \qquad \text{para } \mu(S) = \mu(N \setminus S) = 0, i \in S, j \in N \setminus S. \tag{3.11}$$

Intercambiando si es preciso el papel de S y $N \setminus S$, podemos asumir siempre que las desigualdades (3.9) y (3.11) se escriben para $S \subset N$ tal que $|S| \leq \lfloor n/2 \rfloor$. Lo mismo se mantiene para las desigualdades (3.10) eligiendo $i \in N \setminus S$ cuando $\mu(S) \neq 0$ y $\mu(N \setminus S) = 0$.

Nótese que (3.10) son también válidas (pero sin definir facetas) cuando $\mu(S) \neq 0$. Análogamente, las desigualdades (3.11) se verifican para cualquier $S \subset N$ y coinciden con (3.4).

Utilizando las ecuaciones (3.2), las anteriores desigualdades se pueden reescribir como:

$$\sum_{e \in E(S)} x_e \leq \sum_{v \in S} y_v - 1 \qquad \text{para } \mu(S) \neq 0, \mu(N \setminus S) \neq 0, \tag{3.12}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{v \in S \setminus \{i\}} y_v \qquad \text{para } \mu(S) = 0, \mu(N \setminus S) \neq 0, i \in S, \tag{3.13}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{v \in S \setminus \{i\}} y_v - y_j + 1 \qquad \text{para } \mu(S) = \mu(N \setminus S) = 0, i \in S, j \in N \setminus S. \tag{3.14}$$

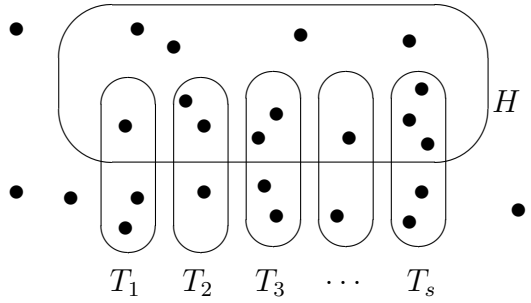


Figura 3.1: Un peine.

Esta forma de las restricciones presenta la ventaja de tener pocos coeficientes no-nulos (suponiendo que $|S| \leq \lfloor n/2 \rfloor$), por lo que es mejor para ser incorporada en un procedimiento de hiperplanos de corte.

Cuando $|S| = 2$ aparecen casos particulares de GSEC's, llevando a

$$x_e \leq y_v \quad \text{para } v \in N,\ e \in \delta(v). \tag{3.15}$$

Finalmente consideramos las desigualdades *peine* del TSP. Un peine es una familia $C = (H, T_1, \ldots, T_s)$ de $s+1$ subconjunto de nodos, donde $s \geq 3$ es un entero impar; véase la Figura 3.1 para una ilustración. H se llama el *mango* de C, mientras que $T_1, \ldots, T_s$ se llaman *dientes*. Además, deben satisfacer las siguientes condiciones:

i) $T_1, \ldots, T_s$ son disjuntos dos a dos;

ii) $T_j \cap H \neq \emptyset$ y $T_j \setminus H \neq \emptyset$ para $j = 1, \ldots, s$.

Se define el *tamaño* de C como $\sigma(C) := |H| + \sum_{j=1}^{s}(|T_j| - 1) - (s+1)/2$.

La *desigualdad peine* asociada con C es

$$\sum_{e \in E(H)} x_e + \sum_{j=1}^{s} \sum_{e \in E(T_j)} x_e \leq \sigma(C), \tag{3.16}$$

es válida y define una faceta para el TSP (véase Grötschel y Padberg [GP85]). Es bien sabido que intercambiando el papel de H y de $N \setminus H$ se obtiene una formulación equivalente de (3.16).

Comenzando con (3.16), podemos obtener análogas desigualdades induciendo facetas para el GTSP mediante el uso del Teorema 3.2.2 de elevación (trivialmente modificado para tratar con desigualdades de tipo "$\leq$").

Teorema 3.2.8 *Consideremos $C = (H, T_1, \ldots, T_s)$ un peine. Para $j = 1, \ldots, s$, sea a_j un nodo en $T_j \cap H$ si $\mu(T_j \cap H) = 0$, $a_j = 0$ (un valor ficticio) en otro caso; y sea b_j un nodo en $T_j \setminus H$ si $\mu(T_j \setminus H) = 0$, $b_j = 0$ en otro caso. Entonces la siguiente* desigualdad peine generalizada *es válida y define una faceta para P:*

$$\sum_{e \in E(H)} x_e + \sum_{j=1}^{s} \sum_{e \in E(T_j)} x_e + \sum_{v \in N} \tilde{\beta}_v (1 - y_v) \leq \sigma(C), \tag{3.17}$$

donde $\tilde{\beta}_v = 0$ para todo $v \in N \setminus (H \cup T_1 \cup \ldots \cup T_s)$, $\tilde{\beta}_v = 1$ para todo $v \in H \setminus (T_1 \cup \ldots \cup T_s)$, y para $j = 1, \ldots, s$:

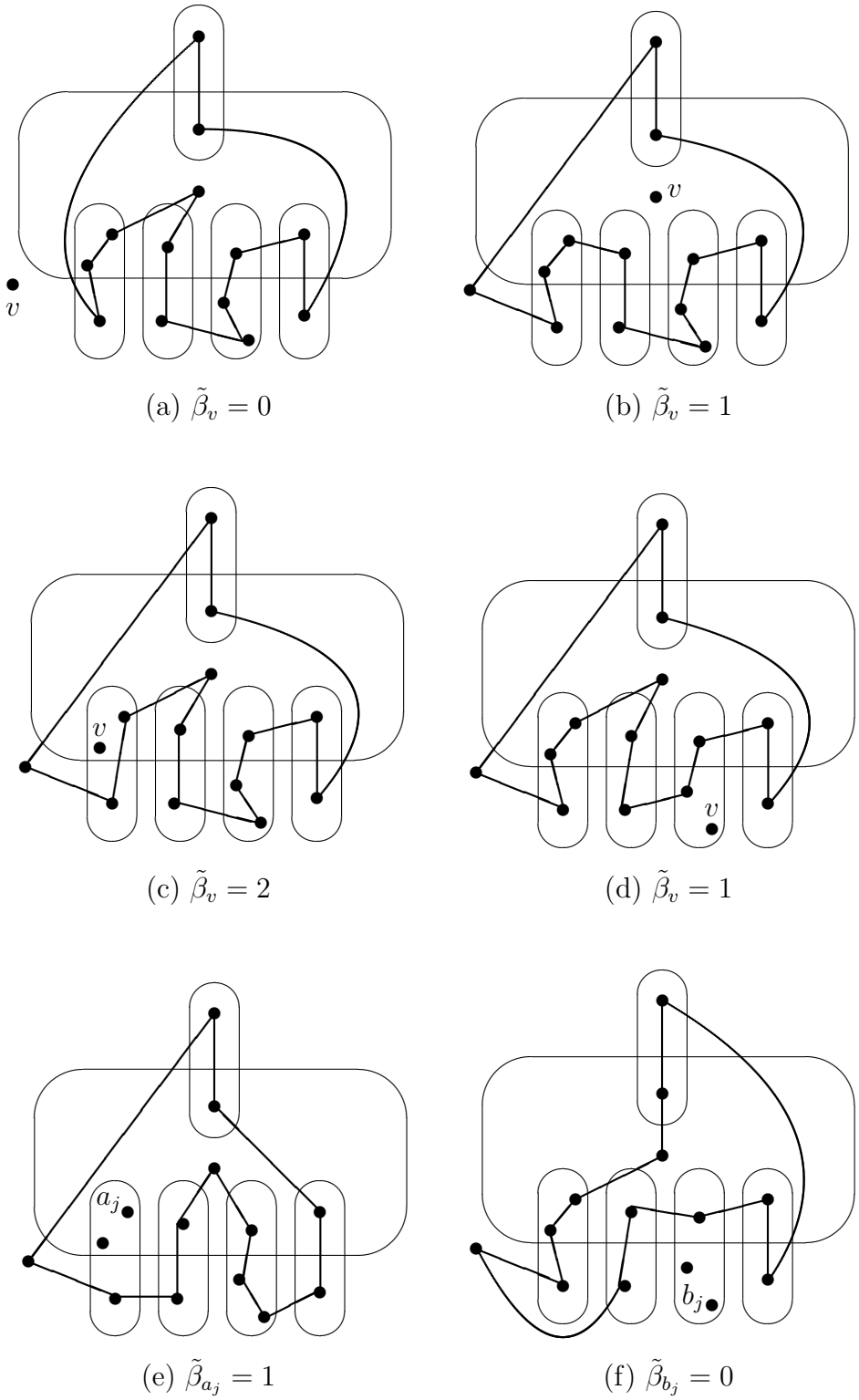
$$\begin{aligned}
\tilde{\beta}_v &= 2 \qquad \text{para } v \in T_j \cap H,\ v \neq a_j;\\
\tilde{\beta}_{a_j} &= 1 \qquad \text{si } a_j \neq 0;\\
\tilde{\beta}_v &= 1 \qquad \text{para } v \in T_j \setminus H,\ v \neq b_j;\\
\tilde{\beta}_{b_j} &= 0 \qquad \text{si } b_j \neq 0.
\end{aligned}$$

Demostración. Consideremos cualquier secuencia de elevación para los nodos $v \in N$ en los que cada $a_j \neq 0$ (resp., $b_j \neq 0$) sigue a todos los otros nodos en $T_j \cap H$ (resp., $T_j \setminus H$). Los coeficientes $\tilde{\beta}_v = 0$ para $v \in N \setminus (H \cup T_1 \cup \ldots \cup T_s)$ y $\tilde{\beta}_v = 1$ para $v \in H \setminus (T_1 \cup \ldots \cup T_s)$ son fácilmente calculables (las Figuras 3.2a y 3.2b muestran, respectivamente, soluciones del GTSP que verifican estas condiciones con igualdad). La correcteza de los coeficientes $\tilde{\beta}_v = 2$ para $v \in (T_j \cap H) \setminus \{a_j\}$, y $\tilde{\beta}_v = 1$ para $v \in (T_j \setminus H) \setminus \{b_j\}$, puede comprobarse con ayuda de las Figuras 3.2c y 3.2d, respectivamente.

Para los valores de $\tilde{\beta}_{a_j}$ cuando $a_j \neq 0$, observamos que a_j es el último nodo de la secuencia de elevación dentro de $T_j \cap H$, es decir cuando se calcula $\tilde{\beta}_{a_j}$ se debe tener presente la posibilidad que tiene una solución del GTSP de evitar todos los nodos en $T_j \cap H$. Esto lleva a $\tilde{\beta}_{a_j} = 1$ puesto que nosotros ya hemos calculado los valores $\tilde{\beta}_v = 2$ para $v \in (T_j \cap H) \setminus \{a_j\}$; véase la Figura 3.2e. De modo similar, cuando se calcula $\tilde{\beta}_{b_j}$ para $b_j \neq 0$ debemos considerar la posibilidad de tener no cubiertos todos los nodos en $T_j \setminus H$. Esto produce el valor $\tilde{\beta}_{b_j} = 0$, debido a la solución diseñada en la Figura 3.2f. □

3.3 Facetas del polítopo del E-GTSP.

Ahora abordaremos la estructura poliédrica del polítopo del E-GTSP, $P^=$. Este polítopo es claramente una cara de P, por lo que todas las desigualdades definiendo facetas para P estudiadas en la Sección 3.2 son también válidas (aunque no necesariamente definen facetas) para $P^=$.



(a) $\tilde{\beta}_v = 0$ (b) $\tilde{\beta}_v = 1$

(c) $\tilde{\beta}_v = 2$ (d) $\tilde{\beta}_v = 1$

(e) $\tilde{\beta}_{a_j} = 1$ (f) $\tilde{\beta}_{b_j} = 0$

Figura 3.2: Ajustadas soluciones del GTSP para la demostración del Teorema 3.2.8 ($\sigma(C) = 11$).

Puesto que en el E-GTSP debe visitarse exactamente un nodo de cada cluster, podemos eliminar los arcos internos a los clusters, y re-definir el conjunto de arcos como

$$E := \{[i,j] : i \in N, j \in N \setminus C_{h(i)}\}.$$

Debido a esta reducción, las restricciones (3.7) son equivalentes a

$$\sum_{e \in \delta(C_h)} x_e = 2 \quad \text{para todo } h = 1, \ldots, m, \tag{3.18}$$

y un modelo simplificado para el E-GTSP puede obtenerse ahora reemplazando las restricciones (3.4) y (3.6) por las dos familias de restricciones:

$$\sum_{e \in E(S)} x_e \leq r - 1 \qquad \text{para } S = \cup_{i=1}^{r} C_{l_i} \text{ y } 3 \leq r \leq m-3, \tag{3.19}$$

$$\sum_{e \in \delta(C_l) \cap \delta(w)} x_e \leq y_w \qquad \text{para } l = 1, \ldots, m \text{ y } w \in N \setminus C_l, \tag{3.20}$$

respectivamente. Las desigualdades (3.19) se llamarán *Restricciones de Eliminación de Subtour Generalizadas Básicas* (GSEC's Básicas), y las (3.20) se llamarán *desigualdades abanico*. Ambas son casos particulares de las GSEC's (3.12) debido a (3.7). En efecto, (3.19) son equivalentes a las GSEC's (3.12) escritas para $S = \cup_{i=1}^{r} C_{l_i}$, donde $\sum_{v \in S} y_v = r$ ya que $\sum_{v \in C_{l_i}} y_v = 1$ para $i = 1, \ldots, r$. Análogamente, (3.20) proceden de (3.12) cuando $S = C_l \cup \{w\}$, ya que $E(S) = \delta(C_l) \cap \delta(w)$ y $\sum_{v \in S} y_v = \sum_{v \in C_l} y_v + y_w = 1 + y_w$. Nótese que las restricciones (3.20) dominan a (3.15).

Al igual que en la sección previa, nos proponemos relacionar la estructura facial de $P^=$ con aquélla del bien estudiado polítopo del TSP, Q, incluso a pesar de que Q no es una relajación de $P^=$. Para ello, introducimos algunas definiciones elementales.

Definición 3.3.1 *Dada una desigualdad válida $\alpha x \leq \beta y + \gamma$ para $P^=$, denotamos por $\mathcal{H}(\alpha, \beta, \gamma) := P^= \cap \{(x,y) \in I\!R^{E \cup N} : \alpha x = \beta y + \gamma\}$ la cara de $P^=$ inducida por $\alpha x \leq \beta y + \gamma$. Sea $v \in N \setminus W$ un nodo arbitrario pero fijo, y denotamos por $P_v^=$ el polítopo del E-GTSP asociado con el subgrafo de G inducido por $N \setminus \{v\}$. La* v-restricción *de $\alpha x \leq \beta y + \gamma$ es la desigualdad obtenida de $\alpha x \leq \beta y + \gamma$ eliminando las variables y_v y x_e para todo $e \in \delta(v)$, es decir, la proyección de $\alpha x \leq \beta y + \gamma$ en el espacio $\{(x,y) \in I\!R^{E \cup N} : y_v = 0\,,\, x_e = 0$ para $e \in \delta(v)\}$. El* grafo v-compatible *de $\alpha x \leq \beta y + \gamma$ es el grafo $G_v^* = (N \setminus C_{h(v)}, E^*)$ con $[u,w] \in E^*$ si y sólo si existe $(x,y) \in \mathcal{H}(\alpha, \beta, \gamma)$ con $x_{[v,u]} = x_{[v,w]} = 1$. El* rango *de un grafo se define como el rango de su matriz de incidencia arco-nodo, es decir, como el número de sus nodos menos el número de sus componentes conexas bipartitas; véase, por ejemplo, Nemhauser y Wolsey [NW88, página 76]. El grafo se dice que es de* rango lleno *cuando su matriz de incidencia arco-nodo es de rango lleno, es decir, cuando cada componente conexa tiene un ciclo impar.*

Lema 3.3.1 *Para cada desigualdad válida $\alpha x \leq \beta y + \gamma$ para $P^=$ y cada nodo $v \in N \setminus W$ la dimensión de $\mathcal{H}(\alpha, \beta, \gamma)$ es mayor o igual a la dimensión de la cara de $P_v^=$ inducida por su v-restricción, más el rango de su grafo v-compatible.*

Demostración. Sea X la matriz en la que cada fila es un punto extremo de $\mathcal{H}(\alpha, \beta, \gamma)$. Puesto que $\mathcal{H}(\alpha, \beta, \gamma)$ contiene un hiperplano que no pasa por el origen (esto es, aquél inducido por (3.7) para $h = 1$), un subconjunto de filas de X es afinmente independiente si y sólo si es linealmente independiente. Así la dimensión de $\mathcal{H}(\alpha, \beta, \gamma)$ coincide con el rango de X menos 1. Ahora, X puede dividirse en

$$X = \begin{bmatrix} X_{11} & 0 & 0 \\ X_{21} & X_{22} & 1 \end{bmatrix},$$

donde la última columna se corresponde con la variable y_v, y las columnas de X_{22} se corresponden con las variables x_e para $e \in \delta(v)$. Entonces el rango de X es, al menos, la suma del rango de X_{11} más el rango de $[X_{22}\ 1]$. Por construcción, el rango de X_{11} es la dimensión de la cara de $P_v^=$ inducida por la v-restricción de $\alpha x \leq \beta y + \gamma$, más 1. Para X_{22}, observemos que cada una de sus filas contiene exactamente dos 1's y (salvando las filas repetidas) puede verse como la matriz incidencia arco-nodo del grafo v-compatible G_v^* asociado con $\alpha x \leq \beta y + \gamma$. Además, la última columna de $[X_{22}\ 1]$ es una combinación lineal (con coeficientes 1/2) de las restantes columnas. Esto demuestra la afirmación. □

El Lema 3.3.1 nos permite extender algunos conocidos resultados del polítopo del TSP al caso del E-GTSP usando inducción sobre

$$\rho = \sum_{h=1}^{m} (|C_h| - 1) = n - m. \tag{3.21}$$

Como se demuestra en el Lema 3.3.1, el rango del grafo v-compatible G_v^* asociado con una desigualdad dada $\alpha x \leq \beta y + \gamma$ juega un papel central en el análisis de la estructura poliédrica de $P^=$. Desgraciadamente, determinar si un arco está o no presente en G_v^* exige la construcción de una cierta solución del E-GTSP (x, y) con $\alpha x = \beta y + \gamma$, por lo que se trata de un problema $\mathcal{NP}$-difícil en general. En la práctica, uno está más interesado en encontrar condiciones suficientes para la existencia de una arco en G_v^*. A continuación describimos una de estas condiciones, relacionada con el trabajo de Naddef y Rinaldi [NR93] para el TSP Gráfico, y con el trabajo de Balas y Fischetti [BF92, BF93] para el TSP Asimétrico.

Definición 3.3.2 *Una ecuación $\alpha x \leq \beta y + \gamma$ se denomina* Tight-Triangular *(TT, para*

abreviar) cuando para todo $v \in N$ se tiene

$$\beta_v = \max\{\alpha_{iv} + \alpha_{jv} - \alpha_{ij} \ : \ [i,j] \in E \setminus \delta(C_{h(v)})\}.$$

Para $v \in N$, denotamos por

$$\Delta(v) := \{[i,j] \in E \setminus \delta(C_{h(v)}) \ : \ \beta_v = \alpha_{iv} + \alpha_{jv} - \alpha_{ij}\}$$

el conjunto de arcos tight *para v.*

Recordemos que una cara $\mathcal{H}$ de $P^=$ se llama *trivial* cuando $\mathcal{H} \subseteq \{(x,y) \in I\!R^{E \cup N} : x_e = 0\}$ para algún $e \in E$; *no-trivial* en otro caso.

Lema 3.3.2 *Sea $v \in N \setminus W$, y $\alpha x \leq \beta y + \gamma$ una desigualdad TT válida para $P^=$ cuya v-restricción define una cara no-trivial de $P^=_v$. Entonces G^*_v contiene todos los arcos en $\Delta(v)$.*

Demostración. Sea $[i,j]$ cualquier arco en $\Delta(v)$. Consideremos cualquier solución del E-GTSP en $\mathcal{H}(\alpha,\beta,\gamma)$, digamos (x,y), que use el arco $[i,j]$ y visite el cluster $C_{h(v)}$ a través de los arcos $[a,w]$ y $[w,b]$, digamos, donde $w \in C_{h(v)} \setminus \{v\}$. La existencia de (x,y) está garantizada por la hipótesis de no-trivialidad. Comenzando con eta solución, construimos un nuevo punto $(\tilde{x},\tilde{y})$ reemplazando los tres arcos $[a,w]$, $[b,w]$ y $[i,j]$, por $[i,v]$, $[j,v]$ y $[a,b]$. Entonces $\alpha\tilde{x} - \beta\tilde{y} = \alpha x - \beta y + (\alpha_{iv} + \alpha_{jv} - \alpha_{ij} - \beta_v) - (\alpha_{aw} + \alpha_{bw} - \alpha_{ab} - \beta_w)$, donde $\alpha_{aw} + \alpha_{bw} - \alpha_{ab} - \beta_w \leq 0$ puesto que $\alpha x \leq \beta y + \gamma$ es una desigualdad TT, y $\alpha_{iv} + \alpha_{jv} - \alpha_{ij} - \beta_v = 0$ puesto que $[i,j] \in \Delta(v)$. Se sigue entonces que $\alpha\tilde{x} - \beta\tilde{y} \geq \alpha x - \beta y = \gamma$, es decir, $(\tilde{x},\tilde{y}) \in \mathcal{H}(\alpha,\beta,\gamma)$. Esto prueba que $[i,j]$ pertenece al grafo G^*_v, como se pretendía. □

Obsérvese sin embargo que G^*_v puede contener arcos no en $\Delta(v)$, como se muestra, por ejemplo, en la demostración del Teorema 3.3.8, caso (b).

Ahora estamos listos para estudiar la estructura facial de $P^=$.

Teorema 3.3.1 *$dim(P^=) = |E| - m$.*

Demostración. Claramente, $\dim(P^=) \leq |E| - m$ puesto que las ecuaciones (3.2) y (3.7) son linealmente independientes. Por ello, resta sólo demostrar que la dimensión de la cara (impropia y no-trivial) $\mathcal{H}(0,0,0)$ inducida por $0x \leq 0y + 0$ no es menor que $|E| - m$. Usamos inducción sobre ρ, según se define en (3.21).

Cuando $\rho = 0$ tenemos el caso TSP estándar, y la afirmación es correcta.

Asumamos ahora que la afirmación es correcta para $\rho = \bar{\rho}$, y consideremos cualquier ejemplo del E-GTSP con $\rho = \bar{\rho} + 1$. Entonces existe un nodo $v \in N \setminus W$. Debido al Lema 3.3.1, tenemos $\dim(\mathcal{H}(0,0,0)) \geq d_1 + d_2$, donde d_1 es la dimensión de la cara de $P_v^=$ inducida por la v-restricción de $0x \leq 0y + 0$, y d_2 es el rango del grafo de v-compatibilidad G_v^* asociado con $0x \leq 0y + 0$. Por la hipótesis de inducción, $d_1 \geq |E \setminus \delta(v)| - m$, resta así demostrar que $d_2 = |\delta(v)| = |N \setminus C_{h(v)}|$, es decir, que G_v^* es de rango lleno. Pero esto se sigue trivialmente del Lema 3.3.2, puesto que $\Delta(v)$ contiene todos los arcos en $E \setminus \delta(C_{h(v)})$ y, por lo tanto, G_v^* es conexo y contiene un ciclo impar (recordemos que se asume que $m \geq 5$). □

Ahora analizaremos las restricciones de no-negatividad. Debido a (3.20), $y_v \geq 0$ no define una faceta para ningún $v \in N$. Sin embargo, $x_e \geq 0$ si lo hace para todo $e \in E$.

Teorema 3.3.2 *La desigualdad $x_e \geq 0$ define una faceta (trivial) de $P^=$ para todo $e \in E$.*

Demostración. La demostración sigue una línea similar a la del Teorema 3.3.1, y se basa en inducción sobre ρ. El punto clave es demostrar que para algún $v \in N \setminus W$ el grafo de v-compatibilidad G_v^* asociado con $-x_e \leq 0$ es de rango lleno. Para ello demostramos que G_v^* contiene un árbol generador (es decir, es conexo) y un ciclo impar.

Recordemos que se asume $m \geq 5$. Supongamos sin pérdida de generalidad que $e = [1,2]$, y sea $v \notin \{1,2\} \cup W$ un nodo cualquiera. Pueden ocurrir dos casos:

(a) $v \in C_{h(1)} \cup C_{h(2)}$: Nótese que $x_e = 0$ se verifica necesariamente para todo $(x,y) \in P^=$ con $y_v = 1$. Se sigue pues que G_v^* contiene todos los arcos en $E \setminus \delta(C_{h(v)})$.

(b) $v \notin C_{h(1)} \cup C_{h(2)}$: En este caso, un árbol generador G_v^* se obtiene tomando los arcos $[1,u]$ para todo $u \notin C_{h(1)} \cup C_{h(v)}$, y $[2,u]$ para todo $u \in C_{h(1)} \setminus \{1\}$. Además, existe un arco $[2,u]$ para cualquier $u \notin C_{h(1)} \cup C_{h(2)} \cup C_{h(v)}$, por lo que G_v^* contiene el ciclo impar $\{[1,2],[2,u],[1,u]\}$.

El resultado está así demostrado. □

Ahora abordaremos las GSEC's (3.12)–(3.14). Estas desigualdades son claramente válidas para $P^=$, pero no todas ellas definen facetas. Demostraremos que (3.13) y (3.14) están dominadas por (3.12). Además, demostraremos que una desigualdad (3.12) define una faceta de $P^=$ si y sólo si se verifica una de las siguientes condiciones:

(i) $S \subseteq W$ y $|S| = 2$,

(ii) $S = C_l \cup \{w\}$ para algún $w \in N \setminus (C_l \cup W)$,

(iii) $\eta(S) \geq 3$ y $\eta(N \setminus S) \geq 3$,

donde $\eta(\cdot)$ ha sido definido en la Sección 3.1.

Teorema 3.3.3 *Las GSEC's (3.13) y (3.14) no definen facetas de* $P^{=}$.

Demostración. Cualquier GSEC (3.14) puede ser escrita como

$$\sum_{e \in E(S)} x_e - \sum_{v \in S \setminus \{i\}} y_v + y_j - 1 \leq 0.$$

Si $C_{h(i)} = C_{h(j)}$, entonces $y_i + y_j \leq 1$, por lo que la desigualdad (3.14) es una desigualdad debilitada de $\sum_{e \in E(S)} x_e - \sum_{v \in S} y_v \leq 0$ la cual, a su vez, está dominada estrictamente por la ecuación

$$\frac{1}{2} \sum_{v \in S} (\sum_{e \in \delta(v)} x_e - 2y_v) = 0.$$

En otro caso (3.14) está dominado por la GSEC (3.13) escrita para $S' := S \setminus C_{h(j)}$, es decir por

$$\sum_{e \in E(S')} x_e - \sum_{v \in S' \setminus \{i\}} y_v \leq 0,$$

puesto que

$$\sum_{e \in E(S')} x_e \geq \sum_{e \in E(S)} x_e - \sum_{v \in C_{h(j)} \cap S} \sum_{e \in \delta(v)} x_e = \sum_{e \in E(S)} x_e - 2 \sum_{v \in C_{h(j)} \cap S} y_v,$$

$$- \sum_{v \in S' \setminus \{i\}} y_v = - \sum_{v \in S \setminus \{i\}} y_v + \sum_{v \in C_{h(j)} \cap S} y_v,$$

y

$$0 \geq \sum_{v \in C_{h(j)} \cap S} y_v + y_j - 1.$$

Análogamente, una GSEC (3.13) se puede expresar como

$$\sum_{e \in E(S)} x_e - \sum_{v \in S \setminus \{i\}} y_v \leq 0,$$

y está dominada por la GSEC (3.12) escrita para $S' := S \cup C_{h(i)}$, es decir por

$$\sum_{e \in E(S')} x_e - \sum_{v \in S'} y_v + 1 \leq 0,$$

puesto que $E(S) \subset E(S')$ implica

$$\sum_{e \in E(S')} x_e \geq \sum_{e \in E(S)} x_e,$$

y

$$-\sum_{v \in S'} y_v + 1 = -\sum_{v \in C_{h(i)}} y_v + 1 - \sum_{v \in S' \setminus C_{h(i)}} y_v = -\sum_{v \in S \setminus C_{h(i)}} y_v \geq -\sum_{v \in S \setminus \{i\}} y_v.$$

□

Ahora estudiamos algunos casos particulares de GSEC's (3.12).

Teorema 3.3.4 *Las restricciones de cota $x_e \leq 1$ definen una faceta de $P^=$ cuando $e \in E(W)$.*

Demostración. Sin pérdida de generalidad sea $e = [1, 2]$. Como en la demostración de los Teoremas 3.3.1 y 3.3.2, es suficiente demostrar que, para cualquier $v \notin W$ dado, el grafo v-compatible G_v^* asociado con $x_e \leq 1$ es conexo y tiene un ciclo impar. En efecto, sea $i, j \notin C_{h(v)} \cup \{1, 2\}$ tal que $C_{h(i)} \neq C_{h(j)}$; entonces G_v^* contiene los arcos $[1, u]$ para todo $u \notin C_{h(v)} \cup \{1, 2\}$, más los arcos $[i, u]$ para $u \in \{2, j\}$. □

El teorema anterior muestra que la restricción de cota $x_{[i,j]} \leq 1$ define una faceta siempre que $i, j \in W$; en otro caso, está dominada por la desigualdad abanico $\sum_{e \in \delta(C_{h(j)}) \cap \delta(i)} x_e \leq y_i$ (si $i \notin W$), o $\sum_{e \in \delta(C_{h(i)}) \cap \delta(j)} x_e \leq y_j$ (si $j \notin W$). Además, las restricciones cota $y_v \leq 1$ nunca definen una faceta de $P^=$ debido a las ecuaciones (3.7).

Teorema 3.3.5 *La desigualdad abanico (3.20) define una faceta de $P^=$ cuando $w \notin W$.*

Demostración. Nótese que la w-restricción de (3.20) es $0x \leq 0y+0$ y por tanto define una faceta de $P_w^=$ con dimensión $|E \setminus \delta(w)| - m$ en virtud del Teorema 3.3.1. Demostraremos que el grafo de w-compatibilidad G_w^* asociado con (3.20) es conexo, esto es, el rango de G_w^* no es menor que $|\delta(w)| - 1$. En efecto, sea $i \in C_l$ y $j \in N \setminus (C_l \cup C_{h(w)})$; es entonces fácil ver que G_w^* contiene los arcos $[i, u]$ para todo $u \notin C_l \cup C_{h(w)}$, y $[j, u]$ para todo $u \in C_l \setminus \{i\}$. Debido al Lema 3.3.1, la cara inducida por (3.20) tiene dimensión no menor que $|E| - m - 1$, de lo que se sigue la afirmación puesto que claramente esta cara es propia. □

Teorema 3.3.6 *La desigualdad GSEC (3.12) define una faceta de $P^=$ cuando tanto S como $N \setminus S$ tocan, al menos, 3 clusters cada uno, es decir, cuando $\eta(S) \geq 3$ y $\eta(N \setminus S) \geq 3$.*

Demostración. Al igual que en las demostraciones anteriores, es suficiente demostrar que, para cualquier nodo $v \in N \setminus W$ dado, el grafo de v-compatibilidad G_v^* asociado con (3.12) es conexo y tiene un ciclo impar. Mediante un intercambio de S con $N \setminus S$ si fuese preciso, podemos sin perdida de generalidad asumir que $v \in N \setminus S$. Además, sean 1, 2 y 3 nodos distintos tales que $C_{h(i)} \neq C_{h(j)}$ para todo $i, j \in \{1, 2, 3, v\}$, $i \neq j$, con $1, 2 \in N \setminus S$, y $C_{h(3)} \subset S$. Claramente, (3.12) es una desigualdad TT. Demostraremos que $\Delta(v)$ contiene un árbol generador $N \setminus C_{h(v)}$, más un ciclo impar. En efecto, $\Delta(v)$ contiene todos los arcos $[1, u]$ con $u \notin C_{h(v)} \cup C_{h(1)}$, y todos los arcos $[2, u]$ con $u \in C_{h(1)} \setminus \{1\}$, más el arco $[2, 3]$ que pertenece al ciclo impar $\{[1, 2], [2, 3], [1, 3]\}$. Debido al Lema 3.3.2, esto implica que G_v^* es de rango lleno, y confirma la afirmación. □

Corolario 3.3.1 *La GSEC Básica (3.19) define una faceta de $P^=$ cuando $3 \leq r \leq m-3$.*

Resta estudiar las GSEC's (3.12) no cubiertas por los Teoremas 3.3.5–3.3.6. En realidad, estas desigualdades no definen facetas puesto que $E(S)$ inducen un grafo bipartito, y por tanto pueden obtenerse como suma de ciertas desigualdades abanico según muestra el siguiente resultado.

Teorema 3.3.7 *La GSEC (3.12) no define una faceta de $P^=$ cuando $C_l \subset S \subseteq C_l \cup C_h$ y $|S \cap C_h| \geq 2$, para algunos clusters C_l y C_h distintos.*

Demostración. Basta observar que (3.12) puede obtenerse en este caso como suma de la ecuación $\sum_{v \in C_l} y_v = 1$ y la desigualdad abanico $\sum_{e \in \delta(C_l) \cap \delta(w)} x_e \leq y_w$ para todo $w \in S \cap C_h$, cada uno de los cuales define una faceta diferente de $P^=$. □

Corolario 3.3.2 *La GSEC Básica (3.19) no define una faceta de $P^=$ cuando $r = 2$ y $|S| \geq 3$.*

Corolario 3.3.3 *La desigualdad abanico (3.20) no define una faceta de $P^=$ cuando $w \in W$ y $|C_l| \geq 2$.*

Finalmente analizamos las desigualdades peine generalizadas (3.17).

Teorema 3.3.8 *La desigualdad peine generalizada (3.17) define una faceta de $P^=$ cuando $a_j = b_j = 0$ para cada diente T_j $(j = 1, \ldots, s)$.*

Demostración. Una vez más, es suficiente mostrar que para un cierto $v \in N \setminus W$, el grafo de v-compatibilidad G_v^* asociado con (3.17) es conexo y tiene un ciclo impar. Mediante el intercambio de H y $N \setminus H$ si fuese preciso, podemos siempre elegir $v \in N \setminus H$. Además, sin pérdida de generalidad asumiremos que $v \notin T_1 \cup T_2$, $C_{h(1)} \subseteq T_1 \setminus H$, $C_{h(2)} \subseteq T_2 \setminus H$, y $C_{h(3)} \subseteq T_3 \cap H$. Pueden ocurrir dos caso:

(a) $v \notin \cup_{j=3}^{s} T_j$: Puesto que (3.17) es una desigualdad TT, en virtud del Lema 3.3.2 basta observar que $\Delta(v)$ contiene un árbol generador de $N \setminus C_{h(v)}$ (por ejemplo, aquél con los arcos $[1, u]$ para $u \notin C_{h(v)} \cup T_1$, y $[2, w]$ para $w \in T_1 \setminus \{1\}$), más un ciclo impar (por ejemplo, $\{[1, 2], [1, 3], [2, 3]\}$).

(b) $v \in \cup_{j=3}^{s} T_j$: Sea $v \in T_3$. Todos los arcos $[w, u] \in E \setminus \delta(C_{h(v)})$ con $w \in T_3 \cap H$ y $u \in N \setminus H$ fácilmente se ve que pertenecen a $\Delta(v)$, con lo que también al conjunto de arcos de G_v^*. Obsérvese sin embargo que el grafo $(N \setminus C_{h(v)}, \Delta(v))$ no es de rango lleno, ya que todos sus nodos en $H \setminus T_3$ son nodos aislados. Por ello debemos usar un argumento "ad hoc" para concluir que G_v^* es de rango lleno, como se precisa. En efecto, Las soluciones del E-GTSP diseñadas en la Figura 3.3 satisfacen (3.17) con igualdad, y muestran que G_v^* contiene todos los arcos de $[3, u] \in E \setminus \delta(C_h)$ con $u \in H \setminus T_3$, más el arco $[1, w]$ para cada $w \in S := N \setminus (H \cup T_1 \cup \ldots \cup T_s)$ si $S \neq \emptyset$, o $w = 2$ en otro caso. Así G_v^* es conexo, y contiene el ciclo impar $\{[1, w], [1, 3], [w, 3]\}$.

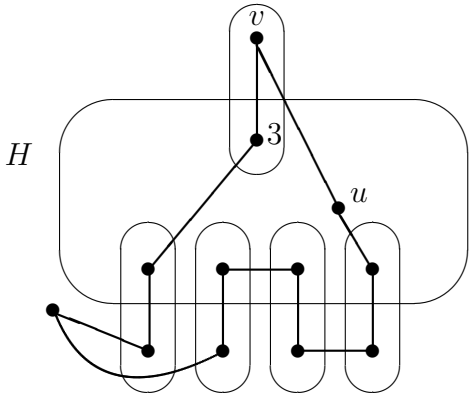
□

Concluimos la sección con la descripción de un procedimiento de elevación para el E-GTSP, que nos permite extender cualquier faceta del polítopo del TSP a una faceta de $P^=$.

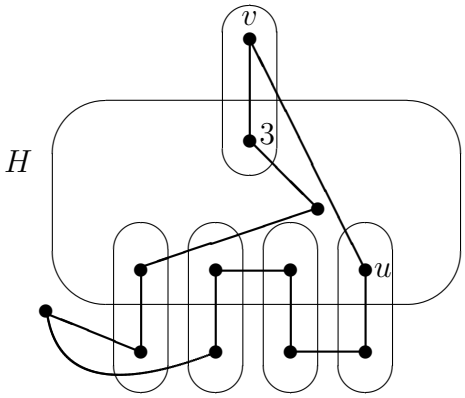
Supongamos que $\alpha x \leq \beta y + \gamma$ define una cara propia de $P^=$, y sea v un nodo cualquiera de $N \setminus W$ tal que la v-restricción de $\alpha x \leq \beta y + \gamma$ define una faceta de $P_v^=$. Debido al Lema 3.3.1, $\alpha x \leq \beta y + \gamma$ define una faceta de $P^=$ si su grafo de v-compatibilidad asociado, G_v^*, es de rango lleno. Asumamos ahora que éste no sea el caso, esto es, que alguna componente conexa de G_v^* es bipartita. Mostraremos como pueden modificarse los coeficientes α_e, $e \in \delta(v)$, de manera que se genere una desigualdad "más fuerte", definiendo una cara propia de $P^=$ con mayor dimensión. Para ello, sea $S \subseteq N \setminus C_{h(v)}$ el conjunto de nodos de una componente bipartita de G_v^*, con nodos partidos entre S_1 y S_2 tales que $\eta(S_1) \geq \eta(S_2)$. Entonces cada $(x, y) \in \mathcal{H}(\alpha, \beta, \gamma)$ verifica la ecuación:

$$\bar{\alpha} x := \sum_{e \in \delta(S_1) \cap \delta(v)} x_e - \sum_{e \in \delta(S_2) \cap \delta(v)} x_e = 0.$$

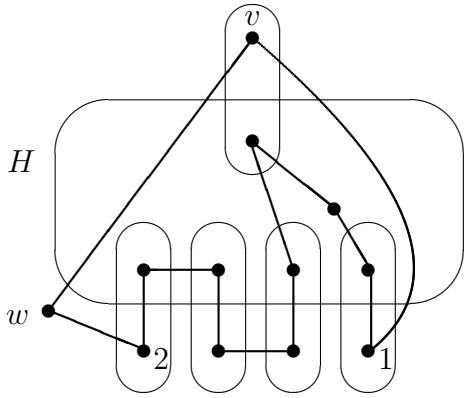
Claramente, $\bar{\alpha} x \leq 0$ no es una desigualdad válida para $P^=$, ya que es violada por cualquier solución del E-GTSP que use los dos arcos $[i, v]$ y $[j, v]$, donde i se elige arbitrariamente



(a) G_v^* contiene el arco $[3, u]$ para todo $u \in H \setminus (\cup_{j=1}^{s} T_j)$.



(b) G_v^* contiene el arco $[3, u]$ para todo $u \in (H \setminus T_3) \cap (\cup_{j=1}^{s} T_j)$.



(c) G_v^* contiene el arco $[1, w]$.

Figura 3.3: Soluciones ajustadas del E-GTSP para la demostración del Teorema 3.3.8 ($\sigma(C) = 8$).

en S_1, y j se elige arbitrariamente en $N \setminus (S_2 \cup C_{h(v)} \cup C_{h(i)})$ (la existencia de estos dos nodos deriva de la hipótesis $\eta(S_1) \geq \eta(S_2)$ y $m \geq 5$). Se sigue que $\bar{\alpha}x = 0$ no es una combinación lineal de las ecuaciones (3.2) y (3.7). Además, no es una combinación lineal de (3.2), (3.7) y $\alpha x = \beta y + \gamma$ puesto que la v-restricción de $\bar{\alpha}x \leq 0$ coincide con $0x \leq 0$, mientras que la v-restricción de $\alpha x \leq \beta y + \gamma$ se ha asumido que define una cara propia de $P_v^=$.

Consideremos ahora la desigualdad elevada

$$(\alpha + \epsilon\bar{\alpha})x \leq \beta y + \gamma, \tag{3.22}$$

donde $\epsilon \geq 0$ es un parámetro real. Nótese que $\epsilon = 0$ lleva a la desigualdad válida $\alpha x \leq \beta y + \gamma$, mientras que cuando ϵ tiende a $+\infty$ la desigualdad (3.22) tiende a la desigualdad inválida $\bar{\alpha}x \leq 0$. Sea ϵ^* el máximo valor de ϵ tal que (3.22) es válida. Este valor puede calcularse como

$$\epsilon^* = \min\left\{ \frac{-\alpha x + \beta y + \gamma}{\bar{\alpha}x} \ : \ (x,y) \in P^=,\ \bar{\alpha}x > 0 \right\},$$

es decir

$$\epsilon^* = \min\left\{\epsilon^1, \epsilon^2\right\},$$

donde

$$\begin{aligned} \epsilon^1 &:= \frac{1}{2}\min\left\{\Delta_e \ : \ e \in E(S_1)\right\}, \\ \epsilon^2 &:= \min\left\{\Delta_e \ : \ e \in \delta(S_1) \setminus \delta(C_{h(v)} \cup S_2)\right\}, \end{aligned}$$

y para todo $[i,j] \in E \setminus \delta(C_{h(v)})$ con $i \in S_1$ y $j \notin S_2$,

$$\Delta_{[i,j]} := \min\{-\alpha x + \beta y + \gamma \ : \ (x,y) \in P^=,\ x_{[i,v]} = x_{[j,v]} = 1\} \geq 0.$$

Para cualquier $\epsilon \in [0, \epsilon^*]$, $\mathcal{H}(\alpha + \epsilon\bar{\alpha}, \beta, \gamma)$ es una cara propia de $P^=$ (puesto que (3.2), (3.7), $\alpha x = \beta y + \gamma$, y $\bar{\alpha}x = 0$ son ecuaciones linealmente independientes) que contiene a $\mathcal{H}(\alpha, \beta, \gamma)$. Además, $\mathcal{H}(\alpha + \epsilon^*\bar{\alpha}, \beta, \gamma)$ contiene también un punto $(x^*, y^*) \in P^=$ tal que $\bar{\alpha}x^* > 0$ (es decir, el punto que determina el valor ϵ^*), lo que prueba que $\dim(\mathcal{H}(\alpha + \epsilon^*\bar{\alpha}, \beta, \gamma)) \geq \dim(\mathcal{H}(\alpha, \beta, \gamma)) + 1$. (Nótese que esto implica que G_v^* es de rango lleno siempre que tanto $\alpha x \leq \beta y + \gamma$ como su v-restricción definan facetas para $P^=$ y $P_v^=$, respectivamente.)

Iterando el procedimiento de elevación anterior, cuya geometría se ilustra en la Figura 3.4, obtenemos una desigualdad que define una faceta de $P^=$.

Ejemplo. Con la intensión de ilustrar la forma de trabajar de este procedimiento, consideremos el siguiente ejemplo simple. Sea $\alpha x \leq \beta y + \gamma$ la desigualdad abanico (3.19),

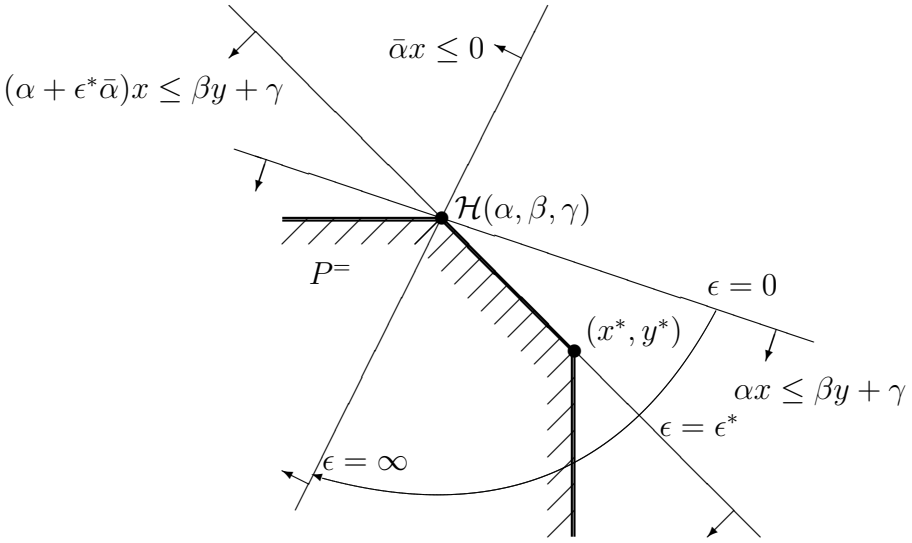


Figura 3.4: Geometría del procedimiento de elevación.

donde $w \in W$ y $C_l = \{u, v\}$. Esta desigualdad se lee como $x_{[u,w]} + x_{[v,w]} \leq y_w (= 1)$. Según el Corolario 3.3.3 y Teorema 3.3.4, $\alpha x \leq \beta y + \gamma$ no define una faceta, mientras que su v-restricción si lo hace. Por tanto el grafo de v-compatibilidad G_v^* asociado con $\alpha x \leq \beta y + \gamma$ no es de rango lleno. En realidad, este grafo sólo contiene los arcos $[w, j]$ para $j \in N \setminus (C_l \cup \{w\})$, por lo que es conexo y bipartito con nodos divididos entre $S_1 := N \setminus (C_l \cup \{w\})$ y $S_2 := \{w\}$. La ecuación $\bar{\alpha}x = 0$ entonces se lee $\sum_{j \in S_1} x_{[v,j]} - x_{[v,w]} = 0$, con lo que $\bar{\alpha}x \leq 0$ no es válida, como se desea (nótese que, en este particular ejemplo, $\bar{\alpha}x \geq 0$ si es válida). Combinamos $\alpha x \leq \beta y + \gamma$ con $\bar{\alpha}x \leq 0$, y obtenemos la siguiente forma para (3.22):

$$x_{[u,w]} + (1 - \epsilon)\, x_{[v,w]} + \epsilon \sum_{j \in S_1} x_{[v,j]} \leq y_w.$$

Fácilmente se determina $\epsilon^1 = 1/2$ ya que $\Delta_e = 1$ para todo $e \in E(S_1)$, y $\epsilon^2 = \infty$ debido a que $\delta(S_1) \setminus \delta(C_{h(v)} \cup S_2) = \emptyset$. Por tanto obtenemos $\epsilon^* = 1/2$, y un punto (x^*, y^*) asociado con cualquier solución del E-GTSP que use dos arcos en $\delta(S_1) \cap \delta(v)$. Obsérvese que la v-restricción de la desigualdad anterior escrita para $\epsilon = \epsilon^*$ es la misma que la de antes de la elevación, mientras que el grafo v-compatible contiene todos los arcos que tenía antes de la elevación, más un nuevo arco (asociado con el punto (x^*, y^*)) conectando dos nodos en S_1. Así G_v^* viene de rango lleno, y la desigualdad elevada define ahora una faceta. Es más, esta desigualdad puede equivalentemente escribirse como $x_{[u,w]} + y_v \leq y_w$, donde $y_w = 1$ y $y_v = 1 - y_u$, es decir hemos obtenido una desigualdad abanico que define una faceta. □

La anterior técnica puede usarse para elevar cualquier desigualdad que defina una faceta $\sum_{e \in E \setminus \delta(v)} \alpha_e x_e \leq \sum_{j \in N \setminus \{v\}} \beta_j y_j + \gamma$ de $P_v^=$, donde $v \in N \setminus W$, hacia una desigualdad que defina una faceta $\sum_{e \in E} \alpha_e x_e \leq \sum_{j \in N} \beta_j y_j + \gamma$ de $P^=$. Para ello primero definimos

(arbitrariamente) los coeficientes α_e para $e \in \delta(v)$, y tomamos β_v lo suficientemente grande como para asegurar su validez para $P^=$. Luego usamos el procedimiento de elevación para "corregir" los valores α_e con $e \in \delta(v)$. Esta construcción muestra que toda desigualdad que defina una faceta de $P_v^=$ tiene una desigualdad que define una faceta en $P^=$ estrechamente relacionada con ella.

Como una inmediata consecuencia, toda faceta del polítopo del TSP (asociado con un grafo con m nodos) puede elevarse para ser una faceta del polítopo del E-GTSP (asociado con un ejemplo con m clusters y $n \geq m$ nodos). Para ello, comenzamos con cualquier desigualdad dada que defina una faceta del polítopo del E-GTSP asociado con un ejemplo con m nodos y m clusters (es decir, para el polítopo del TSP). Luego, iterativamente, añadimos un nodo a cada cluster, y elevamos la desigualdad actual a través del procedimiento anterior.

Capítulo 4

Problema del Viajante de Comercio Generalizado: Un Algoritmo

Consideramos una variante del Problema del Viajante de Comercio simétrico clásico en el que los nodos están divididos en grupos (clusters) y el vendedor tiene que visitar al menos un nodo de cada cluster. El problema es conocido como *Problema del Viajante de Comercio Generalizado* (GTSP), y remitimos al Capítulo 3 para una introducción y motivación al problema. En ese mismo capítulo además hemos modelado el GTSP como un problema de programación lineal entera, y estudiado la estructura facial de dos polítopos asociados con el problema. Aquí proponemos procedimientos de separación exáctos y heurísticos para algunas clases de desigualdades que definen facetas, las cuales serán usadas en un algoritmo de ramificación y corte para determinar la solución exacta del GTSP. También se describen procedimientos heurísticos, y se muestran extensos resultados computacionales tomados de la literatura, llegando a considerar problemas con hasta 442 nodos.

4.1 Introducción

Se propone en este capítulo un algoritmo de ramificación y corte para el *Problema del Viajante de Comercio Generalizado* (GTSP). Remitimos al lector a la introducción del Capítulo 3 para una descripción y motivación al problema. En dicha introducción se establece una notación y un modelo matemático que mantendremos durante este mismo capítulo. Por ello, repetimos brevemente aquí dichos elementos:

Para cada $S \subseteq N$, sea

$$\begin{aligned} E(S) &:= \{\,[i,j] \in E \,:\, i \in S\,,\, j \in S\}, \\ \delta(S) &:= \{\,[i,j] \in E \,:\, i \in S\,,\, j \notin S\}, \\ \mu(S) &:= |\{h \,:\, C_h \subseteq S\}|, \\ \eta(S) &:= |\{h \,:\, C_h \cap S \neq \emptyset\}|. \end{aligned}$$

Para $v \in N$ escribimos $\delta(v)$ en lugar de $\delta(\{v\})$, y denotamos por $C_{h(v)}$ el cluster conteniendo v. También definimos

$$W := \{v \in N \,:\, |C_{h(v)}| = 1\}.$$

Un modelo de programación lineal entera para GTSP es el siguiente. Sea $x_e = 1$ si el arco $e \in E$ aparece en la solución óptima, $x_e = 0$ en otro caso. Además, sea $y_v = 1$ si el nodo $v \in N$ es visitado, $y_v = 0$ en otro caso. GTSP entonces equivale a

$$v(GTSP) := \min \sum_{e \in E} c_e\, x_e \tag{4.1}$$

sujeto a

$$\sum_{e \in \delta(v)} x_e = 2\, y_v \qquad \text{para } v \in N \tag{4.2}$$

$$\sum_{v \in C_h} y_v \geq 1 \qquad \text{para } h \,=\, 1, \ldots, m \tag{4.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\,(y_i + y_j - 1) \qquad \text{para} \quad \begin{array}{l} S \subset N,\, 2 \leq |S| \leq n-2, \\ i \in S,\, j \in N \setminus S \end{array} \tag{4.4}$$

$$x_e \in \{0,1\} \qquad \text{para } e \in E \tag{4.5}$$

$$y_v \in \{0,1\} \qquad \text{para } v \in N. \tag{4.6}$$

Obsérvese que el modelo (4.1)–(4.6) depende fuertemente de las restricciones de integrabilidad sobre las variables y. Si se prescinde de estas restricciones, aparecen como factibles soluciones como la que muestra la Figura 4.1. Por tanto, la relajación LP de este modelo es bastante pobre. En las Sección 3.2 describimos desigualdades válidas adicionales cuya inclusión en el modelo lleva a un considerable fortalecimiento de su LP relajación.

En este capítulo exponemos un algoritmo exacto para el GTSP, basado en una técnica de hiperplanos de corte / ramificación y acotación. En cada nodo del árbol decisional se obtiene una cota inferior sobre el valor de una solución óptima tras resolver una relajación LP del GTSP. Esta relajación LP se fortalece iterativamente añadiendo desigualdades

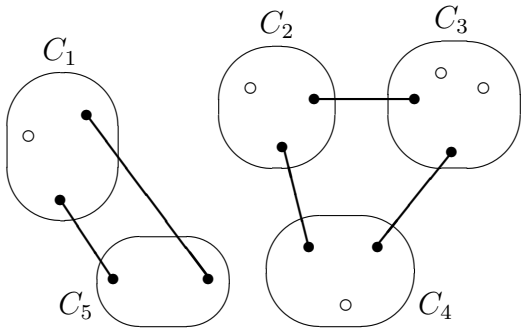


Figura 4.1: Solución GTSP no factible verificando (4.2)–(4.5). Los arcos dibujados se corresponden con $x_e = 1$, los nodos vacíos con $y_v = 0$, y los nodos negros con $y_v = 1/2$.

válidas que son violadas por la solución óptima del modelo lineal del momento; estas desigualdades se identifican a través de procedimientos de separación exactos o heurísticos.

En la Sección 4.2 se proponen algoritmos de separación exactos y heurísticos para encontrar desigualdades violadas entre las desigualdades presentadas en el capítulo anterior. Diversos algoritmos heurísticos para encontrar soluciones aproximadas del GTSP se presentan en la Sección 4.3, donde también se demuestra que el E-GTSP es resoluble en tiempo polinomial cuando se conoce la secuencia de los m clusters. La Sección 4.4 da la descripción global del algoritmo enumerativo para la solución exacta del GTSP. Un extenso análisis computacional sobre diversas clases de problemas test aparecen al final en la Sección 4.5.

4.2 Algoritmos de separación

En esta sección abordaremos el siguiente *problema de separación* (o *identificación*): Dado un punto (fraccionario) $(x^*, y^*) \in [0,1]^{E \cup N}$, encontrar un miembro $\alpha x + \beta y \geq \gamma$ de una familia dada $\mathcal{F}$ de desigualdades válidas para el GTSP, tales que $\alpha x^* + \beta y^* < \gamma$. Un resolución exacta o heurística de este problema es de fundamental importancia para poder utilizar las desigualdades de $\mathcal{F}$ en un algoritmo de hiperplanos de corte.

4.2.1 Un algoritmo de separación exacto para las Restricciones de Eliminación de Subtours Generalizadas

Consideremos la familia $\mathcal{F}$ de las restricciones de eliminación de subtour generalizadas, en la forma de corte (3.9)–(3.11).

Comencemos con las restricciones (3.11):

$$\sum_{e\in\delta(S)} x_e \geq 2\,(y_i + y_j - 1) \qquad \text{si } \mu(S) = \mu(N \setminus S) = 0, i \in S, j \in N \setminus S.$$

Supongamos que los nodos i y j han sido fijados. Entonces, buscar la desigualdad más violada de tipo (3.11) equivale a buscar el corte de capacidad mínima $(S, N \setminus S)$ con $i \in S$ y $j \in N \setminus S$ red no dirigida G^* obtenida de G más la imposición de una capacidad x_e^* a cada $e \in E$. Esto puede hacerse en tiempo $O(n^3)$, puesto que se resuelve con el cálculo del flujo máximo desde i a j (véase, por ejemplo, Ahuja, Magnanti y Orlin [AMO89]). Si el valor del flujo máximo no es menor que $2(y_i^* + y_j^* - 1)$, entonces todas las desigualdades (3.11) para el par dado (i, j) están satisfechas; en otro caso, la capacidad del corte de capacidad mínima separando i de j es estrictamente menor que $2(y_i^*+y_j^*-1)$ y una desigualdad (3.11) de máxima violación se ha detectado entre aquéllas para el par dado (i, j). Intentado ésto con todas las posibles parejas (i, j) obtenemos un algoritmo de separación de complejidad $O(n^5)$. En realidad, se puede obtener un mejor algoritmo de complejidad $O(n^4)$, siguiendo la analogía con el caso del TSP (véase Padberg y Grötschel [PG85]), y usando el esquema de Gomory-Hu [GH61] para problemas de flujo multi-terminales. Aún un más simple algoritmo con la misma complejidad se basa en la trivial observación de que, para cada S, la desigualdad más violada de tipo (3.11) aparece cuando los nodos i y j elegidos son tales que $y_i^* = \max\{y_v^* : v \in S\}$ y $y_j^* = \max\{y_v^* : v \in N \setminus S\}$. Por lo tanto, cualquier nodo s con $y_s^* = \max\{y_v^* : v \in N\}$ puede siempre fijarse para jugar el papel de, por ejemplo, i. De este modo, uno tiene que resolver (como mucho) $n - 1$ problemas de flujo-máximo en el intento de enviar $2(y_s^* + y_j^* - 1)$ unidades de flujo desde s a cualquier $j \in N \setminus \{s\}$. Claramente, los nodos j con $y_s^* + y_j^* - 1 \leq 0$ no necesitan ser considerados.

Veamos ahora el problema de identificar las desigualdades (3.10):

$$\sum_{e\in\delta(S)} x_e \geq 2\,y_i \qquad \text{si } \mu(S) = 0, \mu(N \setminus S) \neq 0, i \in S.$$

Como antes, asumimos que el cluster C_h y el nodo $i \notin C_h$ están fijos. En tal caso una restricción (3.10) con máxima violación se corresponde con un corte de capacidad mínima $(S, N \setminus S)$ con $i \in S$ y $C_h \subseteq N \setminus S$ en la red G^*. Así, ello puede ser detectado encontrando el flujo máximo desde i a t, donde t es un nodo adicional conectado con cada $j \in C_h$ a través de un arco con muy alta capacidad (esto se corresponde a comprimir el cluster C_h en un simple nodo). Tratando todas las parejas $i, C_h)$ se tiene un algoritmo de orden $O(m\,n^4)$. Claramente, los nodos i con $y_i^* = 0$ no necesitan ser considerados.

En cuanto a las restricciones (3.9)

$$\sum_{e\in\delta(S)} x_e \geq 2 \qquad \text{si } \mu(S) \neq 0, \mu(N \setminus S) \neq 0$$

para todas las parejas (C_h, C_k) de clusters distintos, una desigualdad (3.9) de máxima violación se encuentra buscando el flujo máximo desde s a t, donde s (resp. t) es un nodo

adicional conectado con cada $j \in C_h$ (resp. $j \in C_k$) mediante un arco con capacidad muy grande. La complejidad computacional de esta fase es $O(m^2\, n^3)$.

Nótese que una desigualdad violada de tipo (3.10) o (3.11) determinada por los algoritmos de separación antes descritos, no define necesariamente una faceta. Para (3.11) esto ocurre cuando existe un cluster C_h contenido en S o $N \setminus S$; para (3.10), esto ocurre cuando existe un cluster contenido en la parte del corte que contiene al nodo i. En estos casos uno debe obviamente rechazar la desigualdad en favor de la más fuerte (3.9) o (3.10), que además si define una faceta.

Según el esquema anterior el algoritmo de separación para la familia $\mathcal{F}$ conteniendo las desigualdades (3.9)–(3.11) precisa un tiempo $O(m\, n^4)$ en el peor de los casos. En la práctica, el tiempo de cálculo requerido es típicamente mucho menor dado que la red G^* es poco densa, y tiene bastantes nodos aislados. Además, como se explicó previamente, varios cálculos del flujos máximos pueden evitarse dado que algunos valores y^* son pequeños. Por otra parte, consideraciones paramétricas sobre la estructura de los cortes pueden producir posteriores reducciones en el número de llamadas a cálculos del flujo máximo. Por ejemplo, supongamos que en el intento de encontrar una desigualdad violada de tipo (3.11), uno encuentra un flujo máximo desde el nodo s al nodo j de valor, al menos, $2y^*_j$. Entonces el cálculo del flujo máximo desde el nodo j al $C_{h(s)}$ puede evitarse cuando se consideran las desigualdades (3.10). Si el flujo anterior tiene valor 2, podemos igualmente ahorrarnos el cálculo del flujo máximo desde s a $C_{h(j)}$, desde j a $C_{h(s)}$, y desde $C_{h(s)}$ a $C_{h(j)}$. De manera similar, el cálculo del flujo máximo entre dos clusters dados C_h y C_k puede evitarse si el flujo máximo desde s a tanto C_h como a C_k tiene valor 2; en efecto, cualquier corte separando C_h y C_k también separa s de C_h o C_k, con lo que cualquiera de tales cortes tiene capacidad al menos 2.

Consideremos ahora el importante caso en el que $y^*_s := \max\{y^*_v : v \in N\} = 1$, lo que sucede muy frecuentemente durante el algoritmo de hiperplanos de corte. En este caso uno puede encontrar una desigualdad de eliminación de subtour generalizada de máxima violación calculando no más de $n+m-2$ flujos máximos, con complejidad computacional de $O(n^4)$. En efecto, el grado de violación de cualquier desigualdad (3.9) con, por ejemplo, $C_h \subseteq S$ y $C_k \subseteq N \setminus S$ es el mismo que aquél asociado con las desigualdades (3.10) escritas para el mismo S y para $i = s$. Así, las desigualdades (3.9) no necesitan ser consideradas. Consideremos ahora las desigualdades (3.10) con $i \neq s$. Para fijar ideas, sea $i \in S$ y $C_h \subseteq N \setminus S$. Si $s \in S$, entonces el grado de violación de las desigualdades no decrece reemplazando i con s. En otro caso, el grado de violación es el mismo que el de las desigualdades (3.11) escritas para $j = s$. Se sigue entonces que las desigualdades (3.10) con $i \neq s$ no necesitan ser consideradas. Como un resultado, uno tiene que considerar explícitamente sólo las desigualdades (3.10) con $i = s$, y las desigualdades (3.11) (para las cuales puede volverse a asumir que $i = s$).

Un comentario final en este sentido. Consideremos un flujo desde s a un nodo dado j,

y sea f_v la cantidad de flujo atravesando cualquier nodo $v \in N$. Es fácil ver que para todos los $v \in N$ existe un flujo factible desde s a v de valor $\lambda_v = f_v + \min\{f_v, f_j - f_v\}$. En efecto el flujo desde s a j pasa a través de una familia de caminos dirigidos (no necesariamente disjuntos en arcos). Algunos de estos caminos pasan por v, con lo que invirtiendo la dirección del flujo de la porción de caminos desde v hasta j produce un flujo factible desde s hasta v de valor λ_v.

Damos ahora una descripción del algoritmo de separación en Pseudo-Pascal. El algoritmo mantiene una etiqueta α_v representando una cota inferior sobre el valor del flujo máximo desde s a cualquier $v \in N \setminus \{s\}$. Además, tenemos una etiqueta β_h dando una cota inferior del valor del flujo máximo desde s a cada cluster $C_h \neq C_{h(s)}$.

procedure GSEC_SEP;
input: el punto $(x^*, y^*) \in [0,1]^{E \cup N}$;
output: una secuencia $S_1, \ldots, S_r$ de subconjuntos de nodos (no necesariamente distintos) correspondientes a desigualdades violadas GSEC's;
begin
1. renombrar los nodos de manera que $y_1^* \geq y_2^* \geq \ldots \geq y_n^*$;
2. **para todo** $v \in N$ **hacer** $\alpha_v := 0$;
3. $r := 0$; $s := 1$;
4. **desde** $j := 2$ **hasta** n **tal que** $\alpha_j < 2(y_s^* + y_j^* - 1)$ **hacer**
5. **empezar** (**comentario**: desigualdades (3.11))
6. calcular el flujo máximo desde s hasta j, y sea f_v el flujo entrando en $v \in N$, y $(S, N \setminus S)$ sea un corte de capacidad mínima desde s a j;
7. **para todo** $v \in N$ **hacer** $\alpha_v := \max\{\alpha_v, \lambda_v\}$, donde $\lambda_v := f_v + \min\{f_v, f_j - f_v\}$;
8. **si** $f_j < 2(y_s^* + y_j^* - 1)$ **entonces** $r := r + 1,\ S_r := S$
fin;
9. **para todo** $h \in \{1, \ldots, m\}$ **do** $\beta_h := \max\{\alpha_v : v \in C_h\}$;
10. **para todo** $h \in \{1, \ldots, m\} \setminus \{h(s)\}$ **tal que** $\beta_h < 2y_s^*$ y $y_j^* < 1$ para todo $j \in C_h$ **hacer**
11. **empezar** (**comentario**: desigualdades (3.10))
12. calcular el flujo máximo desde s a C_h, y sea φ su valor, y $(S, N \setminus S)$ el correspondiente corte de capacidad mínima;
13. **si** $\varphi < 2y_s^*$ **entonces** $r := r + 1,\ S_r := S$
fin
fin.

Nótese que, en el paso 10, los clusters C_h conteniendo un nodo j con $y_j^* = 1$ no se consideran ya que el grado de violación de las correspondientes desigualdades (3.10) no puede exceder aquél de las desigualdades (3.11) escritas para el mismo conjunto S y para $i = s$ (comparar los pasos 8 y 13, y observar que $\varphi \geq f_j$).

El algoritmo de separación anterior es exacto cuando $\max\{y_v^* : v \in N\} = 1$; en

este caso, podemos devolver los conjuntos $S_1, \ldots, S_r$ correspondientes a desigualdades de máxima violación. Cuando $\max\{y_v^* : v \in N\} < 1$, sin embargo, el algoritmo es de naturaleza heurística. En cualquier caso, cada subconjunto devuelto $S \in \{S_1, \ldots, S_r\}$ puede llevar a una desigualdad violada definiendo un faceta, según describimos a continuación.

procedure GSEC_BUILD;
input: un subconjunto de nodos S, y el punto (x^*, y^*);
output: una desigualdad GSEC asociada a S, de máxima violación;
comienzo
1. $S := S \setminus \{i : y_i^* = 0\}$;
   para cada i tal que $y_i^* = 0$ **hacer**
   sea $j \in C_{h(i)}$ el nodo con $y_j^* \neq 0$ más próximo a i (en el sentido del costo original c_e), y hacer $S := S \cup \{i\}$ cuando $j \in S$;
2. **si** existe $C_h \subseteq S$ y $C_k \subseteq N \setminus S$ **entonces devolver** la desigualdad (3.9);
3. **si** existe $C_h \subseteq S$ **entonces**
   devolver la desigualdad (3.10) con $i := \arg\max\{y_v^* : v \in N \setminus S\}$;
4. **si** existe $C_h \subseteq N \setminus S$ **entonces**
   devolver la desigualdad (3.10) con $i := \arg\max\{y_v^* : v \in S\}$;
5. **devolver** la desigualdad (3.11) con
   $i := \arg\max\{y_v^* : v \in S\}$, y $j := \arg\max\{y_v^* : v \in N \setminus S\}$
fin.

En el Paso 1, el conjunto S se modifica tratando de tener clusters completos dentro de S y fuera de S.

Para problemas del E-GTSP, a fin de obtener desigualdades más fuertes, los Pasos 2 y 5 pueden sustituirse por:

2a. **si** no existe $C_h \subseteq S$ **entonces** $S := S \cup C_{h(i)}$,
   donde $i := \arg\max\{y_v^* : v \in S\}$;
3a. **si** no existe $C_h \subseteq N \setminus S$ **entonces** $S := S \setminus C_{h(j)}$,
   donde $j := \arg\max\{y_v^* : v \in N \setminus S\}$;
4a. **devolver** la desigualdad (3.9)

El procedimiento anterior tiene una complejidad de orden $O(n)$.

4.2.2 Un algoritmo de separación heurístico para las Restricciones de Eliminación de Subtour Generalizadas

El algoritmo de separación GSEC_SEP de la subsección anterior puede consumir excesivo tiempo de cálculo. Describimos ahora dos procedimientos heurísticos más rápidos.

El primer procedimiento, GSEC_H1, considera el subconjunto de las desigualdades (3.9):

$$\sum_{e\in\delta(S)} x_e \geq 2 \quad \text{si } \mu(S)\neq 0, \mu(N\setminus S)\neq 0,$$

con S conteniendo uno de los clusters más pequeños C_l. Para cada $h \in \{1,\ldots,m\}\setminus\{l\}$, el procedimiento calcula una desigualdad de máxima violación y de tipo (3.9) con $C_l \subseteq S$ y $C_h \subseteq N\setminus S$ mediante la búsqueda del flujo máximo desde C_l a C_h. Este procedimiento tiene complejidad $O(m\,n^3)$, y típicamente se ejecuta mucho más rápido que GSEC_SEP.

Tanto GSEC_SEP como GSEC_H1 producen una lista de desigualdades violadas elegidas atendiendo a sus individuales grados de violación, más que atendiendo a sus efectos combinados. Para acelerar la convergencia de la fase de hiperplanos de corte, sin embargo, en cada separación es posible obtener una familia de desigualdades violadas "llenando" todo el grafo. Para ello, consideremos el problema más simple que contempla sólo a las restricciones de eliminación de subtours, conocido como el problema del *árbol generador mínimo* (*Shortest Spanning Tree*) (SST). Es sabido de la teoría de matroides que los subconjuntos de nodos cuyas restricciones de eliminación de subtour son activas en un punto óptimo, definen una familia anidada cubriendo todos los nodos del grafo. Por tanto, un algoritmo de hiperplanos de corte para el SST que añada cortes violados a partir sólo de su grado de violación, precisará un mayor número de iteraciones antes de generar la familia de cortes que determinan un óptimo. En tal sentido, la técnica de compresión (*shrinking*) usada en Padberg y Rinaldi [PR87, PR90] para el TSP, además de reducir el esfuerzo computacional empleado en cada separación, tiene la ventaja de producir rápidamente una familia anidada de restricciones que llenan el grafo.

Describimos a continuación un algoritmo de separación heurístico para las GSEC's, basado en las consideraciones previas. Para ilustrar mejor la idea básica que subyace en el algoritmo, restrinjamos nuestra atención al TSP estándar. Dado el punto fraccionario x^*, buscamos una familia de restricciones de eliminación de subtour violadas. Para ello consideremos el polítopo

$$Q^{SEC} := \left\{ x \geq 0 \,:\, \sum_{e\in E(S)} x_e \leq |S|-1, \text{ para } S\subseteq N, |S|\geq 2 \right\},$$

cuyos vértices son los vectores incidentes de los bosques llenando el grafo. Encontramos un vértice de Q^{SEC} "próximo" a x^*, digamos $\tilde{x}$, y contrastamos la violación de (algunas de) las restricciones de eliminación de subtour definiendo facetas de Q^{SEC} pasando a través de $\tilde{x}$. Para ser más precisos, buscamos $\tilde{x}$ resolviendo el problema $\max\{x^*x : x \in Q^{SEC}\}$, es decir, buscamos un árbol generador de peso máximo en G con pesos en los arcos dados por $x^*_e \geq 0$, $e \in E$. Usamos el clásico algoritmo greedy de Kruskal [K56] y comprobamos la violación de las $n-1$ SEC's asociadas con los subconjuntos $S_i \subset N$, $i = 1,\ldots,n-1$, correspondientes a las componentes conexas encontradas iterativamente.

De la teoría de matroides (véase, por ejemplo, Nemhauser y Wolsey [NW88, página 669]), las SEC's asociadas con estos subconjuntos S_i son las únicas necesarias para demostrar la optimalidad de $\tilde{x}$ (puesto que todas las otras SEC's pueden ser olvidadas sin afectar a la optimalidad de $\tilde{x}$), con lo que probablemente ellas serán violadas por x^*. Observe que (alguno de) los subconjuntos S_i encontrados con el procedimiento anterior pueden equivalentemente ser encontrados determinando las componentes conexas de los subgrafos inducidos por $E(\vartheta) := \{e \in E : x_e^* \geq \vartheta\}$ para todos los valores posibles $\vartheta \in \{x_e^* > 0 : e \in E\}$. De este modo, nuestro heurístico es una versión mejorada de aquél usado en Grötschel y Holland [GH91], que estudia las componentes conexas del subgrafo $G' = (N, E(\vartheta))$ para $\vartheta = \min\{x_e^* > 0 : e \in E\}$.

El esquema anterior puede fácilmente adaptarse para tratar con SEC's generalizados, tal y como indicamos en el siguiente procedimiento de tipo Pseudo-Pascal.

procedure GSEC_H2;
input: el punto $(x^*, y^*) \in [0,1]^{E \cup N}$;
output: una secuencia $S_1, \ldots, S_r$ de subconjuntos anidados de nodos correspondientes a GSEC's violadas;
comienzo
1. ordenar las aristas de manera que $x_{e_1}^* \geq x_{e_2}^* \geq \ldots \geq x_{e_t}^* > 0$, donde $t := |\{e \in E : x_e^* > 0\}|$;
2. $r := 0$; $\tilde{T} := \emptyset$;
3. **para cada** $i \in N$ **hacer** $V_i := \{i\}$; $\mu := n$;
4. **comment:** $\tilde{T}$ es el conjunto de los arcos en el bosque actual, mientras $V_1, \ldots, V_\mu$ son las componentes conexas de $\tilde{G} := (N, \tilde{T})$;
5. **para** $j := 1$ **hasta** t **hacer**
6. **si** e_j conecta dos componentes distintas V_a y V_b (digamos) **entonces**
empezar
7. $\tilde{T} := \tilde{T} \cup \{e_j\}$;
8. **si** la GSEC más violada asociada con $S := V_a \cup V_b$ (tal y como viene del procedimiento GSEC_BUILD de la Sección 4.2.1) es violada por (x^*, y^*) **entonces** $r := r + 1$, $S_r := S$;
9. sustituir V_a y V_b con $V_a \cup V_b$ y decrementar μ
fin
fin.

La complejidad computacional de GSEC_H2 es $O(t \log t + n^2)$, es decir, $O(n^2 \log n)$ en el peor de los casos. El árbol final $\tilde{G} := (N, \tilde{T})$ calculado por el procedimiento se utiliza heurísticamente para producir GSEC's adicionales violadas, asociadas con las componentes conexas de los subgrafos inducidos por $\tilde{T} \setminus \{e\}$ para $e \in \tilde{T}$.

Una aproximación heurística distinta para detectar GSEC's violadas puede obtenerse

cambiando la rega usada en GSEC_H2 para seleccionar las dos componentes V_a y V_b que serán unidas. Así, V_a y V_b podría ser elegida, de una forma greedy, como las dos componentes cuya unión $S := V_a \cup V_b$ produce la mayor violación en la correspondiente GSEC. Usando técnicas paramétricas, el algoritmo modificado puede implementarse para ejecutarse en tiempo $O(n^3)$. Según nuestra experiencia computacional, sin embargo, esta aproximación es mejorada por GSEC_H2.

4.2.3 Algoritmos de separación heurísticos para las Desigualdades Peine Generalizadas

A continuación describimos dos simples procedimientos heurísticos de separación para las desigualdades peine generalizadas.

Primero consideramos las desigualdades 2-emparejamiento generalizadas. De forma similar a aquél propuesto por Padberg y Rao [PR82] para el problema del b-emparejamiento, podemos transformar el problema de separar las desigualdades de 2-emparejamiento en un problema de corte impar de capacidad mínima; así este problema de separación es resoluble exactamente en tiempo polinomial. Sin embargo, tal tarea consume bastante tiempo computacional, y por ello en nuestro código de ramificación y corte usamos el siguiente sencillo heurístico, derivado de procedimientos similares que se han propuesto para el TSP (véase, por ejemplo, Padberg y Grötschel [PG85]). Dado el punto fraccionario (x^*, y^*), definimos el subgrafo $\tilde{G} = (\tilde{N}, \tilde{E})$ inducido por $\tilde{E} := \{e \in E : 0 < x_e^* < 1\}$. Consideramos ahora, una a la vez, cada una de las componentes conexas H de $\tilde{G}$ como el mango de una posible desigualdad 2-emparejamiento generalizada violada, cuyos dientes de 2 nodos se corresponden con los arcos $e \in \delta(H)$ con $x_e^* = 1$ (si el número de estos arcos es par, la desigualdad es rechazada). El procedimiento precisa un tiempo de $O(n + |\tilde{E}|)$, si es implementado propiamente.

Nuestro segundo procedimiento de separación consiste en aplicar la anterior heurística para las desigualdades 2-emparejamiento generalizadas después de haber comprimido cada cluster en un super-nodo, de manera análoga a como se describe en Padberg y Rinaldi [PR90].

4.3 Algoritmos heurísticos

Tanto para el GTSP como para el E-GTSP, se pueden adaptar un gran número de conocidos algoritmos heurísticos para la construcción y mejora de tours para el TSP (véase, por ejemplo, Golden y Stewart [GS85]) Nos centraremos a continuación en heurísticas para producir soluciones factibles para E-GTSP (y por tanto también para GTSP).

Como procedimientos de construcción de tours, describimos una posible adaptación del bien conocido procedimiento de *inserción del más lejano* para el TSP; las variantes *inserción del más cercano* y *inserción de mínimo costo* se pueden adaptar también de forma similar. Para cada par de clusters C_h y C_k, sea d_{hk} la correspondiente *distancia* definida como $d_{hk} := \min\{c_{ij} : i \in C_h, j \in C_k\}$. Comenzamos eligiendo los dos clusters, digamos C_a y C_b, que están lo más alejado posible el uno del otro, y definimos un tour parcial T entre los dos nodos más próximos $i \in C_a$ y $j \in C_b$. En cada iteración, T viene aumentado determinando primero el cluster C_h no visitado más alejado desde los clusters actualmente si visitados por T, y luego insertando un nodo v de C_h entre dos nodos consecutivos i y j de T de manera que se minimice $c_{iv} + c_{vj} - c_{ij}$. El procedimiento para cuando T cubre todos los clusters. Como en el caso del TSP, el procedimiento tiende a producir mejores soluciones cuando los costos satisfacen la desigualdad triangular.

A continuación describimos dos procedimientos para mejorar soluciones.

El primer procedimiento, RP1, se basa en intercambios 2-opt y 3-opt. Sea T la solución E-GTSP actual, visitando exactamente un nodo para cada cluster, y sea $S \subseteq N$ el conjunto de nodos visitados. Claramente cualquier solución próxima a la óptima del TSP sobre el subgrafo inducido por S, puede dar una solución del GTSP mejorada. Una tal solución TSP puede obtenerse heurísticamente aplicando cualquier procedimiento clásico de intercambio 2-opt o 3-opt sobre T. Esta idea nunca cambia el conjunto S de los nodos visitados. Para eliminar tal exigencia, proponemos el siguiente esquema generalizado de 2-opt. Sea $(\ldots, C_\alpha, C_\beta, \ldots, C_\gamma, C_\delta, \ldots)$ la secuencia de clusters correspondiente al tour actual T. Obsérvese la Figura 4.2, donde T se corresponde con la línea continua. Todos los arcos de T no incidentes con los nodos en $C_\alpha \cup C_\beta \cup C_\delta \cup C_\gamma$ (dibujados con trazos gruesos en la figura) son fijos. Tratamos de intercambiar la secuencia de clusters actual en $(\ldots, C_\alpha, C_\gamma, \ldots, C_\beta, C_\delta, \ldots)$. Para ello determinamos los dos pares de nodos (u^*, w^*) y (v^*, z^*) tales que

$$\begin{aligned} c_{iu^*} + c_{u^*w^*} + c_{w^*h} &= \min\{c_{ia} + c_{ab} + c_{bh} : a \in C_\alpha, b \in C_\gamma\}, \\ c_{jv^*} + c_{v^*z^*} + c_{z^*k} &= \min\{c_{ja} + c_{ab} + c_{bk} : a \in C_\beta, b \in C_\delta\}, \end{aligned}$$

donde los nodos i, j, h y k (véase la Figura 4.2) son los nodos visitados por T pertenecientes a los clusters precediendo C_α, siguiendo C_β, precediendo C_γ y siguiendo C_δ, respectivamente.

Este cálculo requiere $|C_\alpha|\,|C_\gamma| + |C_\beta|\,|C_\delta|$ comparaciones. En su globalidad, intentar con todos los posibles pares (C_α, C_β) y (C_γ, C_δ) conlleva a una complejidad computacional $O(n^2)$, puesto que cada arco de G precisa ser considerado sólo dos veces.

Además, RP1 considera el intercambio 3-opt de la Figura 4.3, en el que estudiamos la modificación de la secuencia de clusters

$$(\ldots, C_\alpha, C_\beta, C_\gamma, \ldots, C_\delta, C_\varepsilon, \ldots)$$

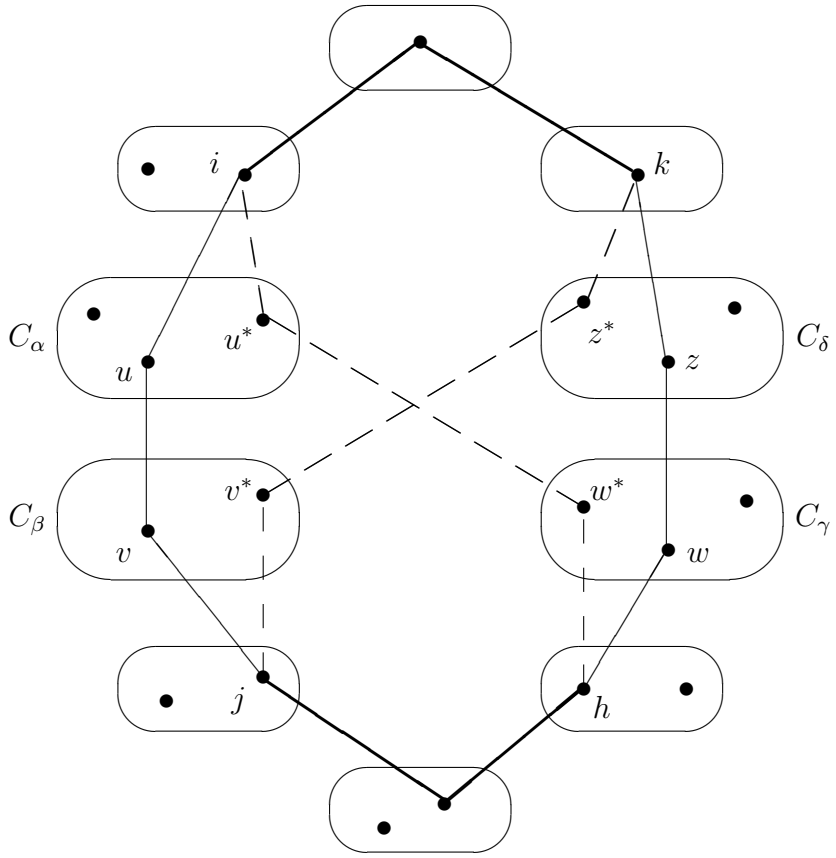


Figura 4.2: El intercambio 2-opt generalizado.

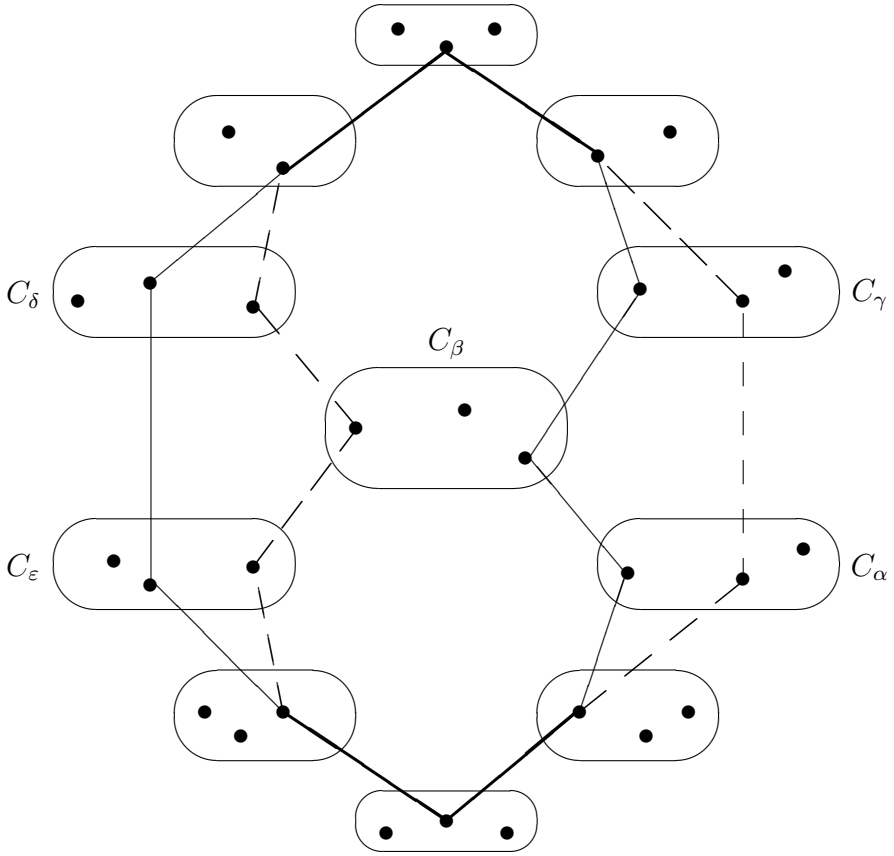


Figura 4.3: Cambiando la posición del cluster C_β.

por

$$(\ldots, C_\alpha, C_\gamma, \ldots, C_\delta, C_\beta, C_\varepsilon, \ldots).$$

Ahora proponemos un segundo procedimiento de mejora, llamado RP2 en lo que sigue, que se ha mostrado más eficiente en nuestras pruebas computacionales. Sea T la solución E-GTSP actual y $(C_{h_1}, \ldots, C_{h_m})$ la secuencia en la que T visita los clusters. Nuestra mejora consiste en encontrar el mejor tour factible, T^*, visitando los clusters según la secuencia dada. Esto puede hacerse, en tiempo polinomial, resolviendo $|C_{h_1}|$ problemas de camino mínimo, como se describe a continuación.

Construimos una red a niveles, LN, teniendo $m+1$ niveles correspondiendo a clusters $C_{h_1}, \ldots, C_{h_m}, C_{h_1}$; véase la Figura 4.4. LN contiene todos los nodos de G, más un nodo extra j' para cada $j \in C_{h_1}$. Hay un arco (i, j) para cada $i \in C_{h_t}$ y $j \in C_{h_{t+1}}$ ($t = 1, \ldots, m-1$), teniendo costo c_{ij}. Además, hay un arco (i, j') para cada $i \in C_{h_m}$ y $j \in C_{h_1}$, teniendo costo c_{ij} (estos arcos conectan los dos últimos niveles de la red). Para un $w \in C_{h_1}$ dado, cualquier camino en LN desde w a w' visita exactamente un nodo de cada nivel (cluster), con lo que produce un tour factible para el E-GTSP. Recíprocamente, cada E-

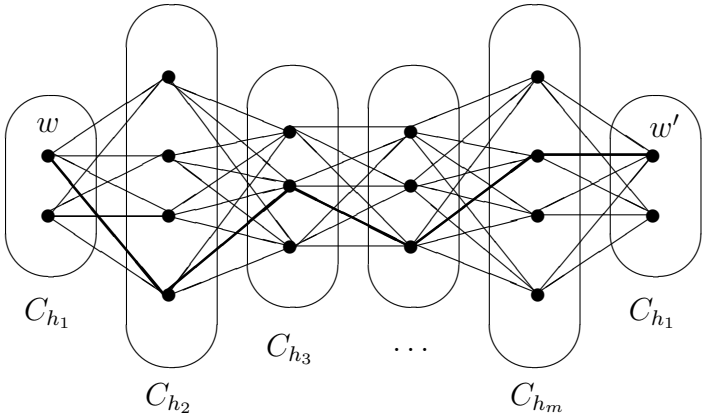


Figura 4.4: La red en niveles *LN*.

GTSP tour visitando los clusters según la secuencia $(C_{h_1}, \ldots, C_{h_m})$ se corresponde con un camino en LN desde un cierto $w \in C_{h_1}$ a w'. Se sigue por tanto que el mejor tour T^* para el E-GTSP visitando los clusters en la secuencia dada, puede encontrarse determinando el camino más corto desde cada $w \in C_{h_1}$ al correspondiente w'. La complejidad global es entonces $|C_{h_1}|\, O(n^2)$, esto es, $O(n^3)$ en el caso peor. En la práctica, el tiempo típicamente empleado se reduce significativamente eligiendo C_{h_1} como el cluster con menor cardinal, y usando un algoritmo para calcular los caminos más cortos especial para grafos acíclicos.

Nótese que el procedimiento de mejora descrito conlleva a un algoritmo exacto para el E-GTSP de complejidad $O((m-1)!\, n^3)$, obtenido tras intentar todas las posibles $(m-1)!$ secuencias de clusters. Por tanto E-GTSP es polinomialmente resoluble para m fijo (independientemente de n).

4.4 El algoritmo enumerativo

En esta sección proponemos un algoritmo enumerativo para la solución exacta del problema. Puesto que todos los ejemplos que consideramos en nuestros resultados computacionales tienen costos que verifican la propiedad triangular, daremos una descripción más detallada de la implementación del algoritmo para el E-GTSP. El algoritmo puede fácilmente se adaptado para el GTSP. Asumimos que todos los costos c_e son enteros.

El algoritmo sigue un esquema de ramificación y acotación, donde la cota inferior se calculan resolviendo una relajación lineal del problema. La relajación es iterativamente fortalecida añadiendo desigualdades válidas al LP del momento, según la llamada aproximación de *hiperplanos de corte*; referimos a los trabajos Padberg y Rinaldi [PR91] y Jünger, Reinelt y Rinaldi [JRR92] para una descripción profunda de la técnica. Ahora describiremos algunos puntos relevantes de nuestra implementación, incluyendo los mejores

valores a los que hemos asignados los parámetros según nuestras experiencias computacionales.

4.4.1 Cálculo de la Cota Inferior

En cada nodo del árbol decisional se calcula la cota inferior resolviendo el problema lineal definido por (4.1), (4.2), (3.7), las restricciones de cotas sobre las variables, las restricciones procedentes de la ramificación, más un subconjunto de GSEC's y desigualdades peine generalizadas. Este subconjunto inicialmente coincide con aquél del nodo padre (para el nodo raíz se describirá más adelante un procedimiento de inicialización 'ad hoc', basado en la optimización Lagrangiana). Nótese que las variables y no son eliminadas utilizando las ecuaciones (4.2), puesto que ello conlleva a matrices de coeficientes en el LP más densas. Comenzamos introduciendo la base óptima del LP del nodo padre. (Para ahorra memoria, esta información es almacenada en un fichero 'scratch'). Luego resolvemos iterativamente el LP, y le añadimos algunas desigualdades violadas por la solución óptima actual. Todas las restricciones violadas encontradas durante la fase de separación (salvo las desigualdades abanico) son almacenadas permanentemente de manera compacta en una estructura global de datos llamada *piscina* (*pool*). Siempre que una desigualdad introducida en el nodo actual del árbol de ramificación tiene una variable dual nula en 5 iteraciones consecutivas de la fase de separación, la eliminamos del LP (por no de la piscina). Además, siempre que el LP-solver requiere más del 5 minutos para resolver el LP del momento, eliminamos todas las desigualdades con variable dual nula introducidas en los nodos anteriores. De este modo tratamos de mantener el LP tan pequeño como en cada momento sea posible, sin perder la posibilidad de recuperar cortes eliminados en momentos posteriores, como se explico arriba.

Identificación de desigualdades violadas

Las desigualdades violadas que deben añadirse en el LP del momento se identifican mediante el siguiente procedimiento de separación.

procedure SEPARACION;
input: la solución óptima (x^*, y^*) del LP;
output: un conjunto $\mathcal{J}$ de desigualdades violadas para ser añadidas al LP;
begin

1. evaluar el grado de violación de las desigualdades en la piscina (exceptuando aquéllas presentes en el LP);

2. evaluar el grado de violación de todas las desigualdades abanico;

3. sea $\mathcal{J}$ el conjunto de desigualdades con grado de violación mayor que 0.1;

   if $|\mathcal{J}| \geq 1$ **then return**;

4. aplicar la heurística GSEC_H2 de la Sección 4.2.2, y evaluar el grado de violación de las GSEC's encontradas;

5. sea $\mathcal{J}$ el conjunto de desigualdades con grado de violación mayor que 0.1;

   if $|\mathcal{J}| \geq 1$ **then return**;

6. aplicar la heurística GSEC_H1 de la Sección 4.2.2, y evaluar el grado de violación de las GSEC's encontradas;

7. **for** $\varepsilon \in \{0.1, 0.01, 0.001\}$, según valores decrecientes, **do**

   begin

8. sea $\mathcal{J}$ el conjunto de desigualdades (distintas) encontradas en los Pasos 1 y 6, y teniendo grados de violación mayor que ε;

   if $|\mathcal{J}| \geq 1$ **then return**

   end

9. aplicar el algoritmo de separación GSEC_SEP de la Sección 4.2.1, y los algoritmos heurísticos de separación para las desigualdades peine generalizadas de la Sección 4.2.3;

10. sea $\mathcal{J}$ conteniendo las GSEC's y las desigualdades peine generalizadas encontradas en el paso 9, y teniendo un grado de violación mayor que 0.001

end.

El tiempo empleado en el paso 1 depende del modo en el que se almacenen las desigualdades, en forma compacta, dentro de la piscina. En nuestra implementación, para cada una de tales desigualdades almacenamos su *tipo*, su valor lado-derecho, un puntero a la fila en el LP del momento (=0 si no está presente), más información adicional relativa al tipo de la desigualdad. Para *type*="desigualdad peine generalizada" (véase (3.17)), almacenamos un vector 0-1 n-dimensional para representar el mango, más un entero n-dimensional dando para cada nodo el índice del diente que lo contiene (=0 si no lo contiene ningún diente). Además, almacenamos para cada diente j los dos nodos a_j y b_j. Usando esta estructura de datos, uno puede evaluar eficientemente los grados de violación de una desigualdad en la piscina observando cada elemento no-nulo de (x^*, y^*) y recuperando (en tiempo constante) los coeficientes asociados.

En el paso 2, el grado de violación de las desigualdades abanico se evalúan como sigue. Para $h = 1, \ldots, m$ y $j \in N \setminus C_h$, inicializamos $degree(h, j) := -y^*_j$. Entonces para cada $[i, j] \in E^* := \{e \in E : x^*_e > 0\}$, consideramos:

$$\begin{aligned} degree(h(i), j) &:= degree(h(i), j) + x^*_{[i,j]}; \\ degree(h(j), i) &:= degree(h(j), i) + x^*_{[i,j]}. \end{aligned}$$

De este modo el paso 2 requiere un tiempo $O(mn + |E^*|)$.

Para el paso 6, su ejecución se evita cuando el procedimiento heurístico GSEC_H2 tiene éxito encontrando desigualdades GSEC's significativamente violadas. Sin embargo, hemos obtenido mejores resultados forzando la ejecución del paso 6 cada 10 llamadas a SEPARACION dentro del mismo nodo del árbol decisional. Esto permite detectar pronto GSEC's que son raramente detectadas mediante GSEC_H2.

El procedimiento SEPARACION precisa contrastar que las desigualdades son distintas entre sí. Esto se hace comparando sus representaciones en forma compacta, y conlleva un tiempo $O(n)$ para cada par de desigualdades. Una considerable aceleración se alcanza asociando a cada desigualdad un valor "hash" de 4 bytes, esto es, un valor obtenido a través de una transformación de amplia imagen desde todas las posibles formas compactas a los enteros de 4 bytes. Ello asegura que diferentes valores hash se corresponden con diferentes desigualdades.

Relajación Lagrangiana

Al comienzo del nodo raíz, aplicamos una optimización Lagrangiana con la intensión de determinar una buena familia de restricciones para el LP inicial, así como una solución heurística próxima a la óptima. Consideramos el siguiente modelo (simplificado) para el E-GTSP, en el que las variables y han sido eliminadas usando las ecuaciones (4.2).

$$\min \sum_{e \in E} c_e x_e \tag{4.7}$$

sujeto a

$$\sum_{e \in E} x_e = m \tag{4.8}$$

$$\sum_{e \in \delta(C_h)} x_e = 2 \qquad \text{para } h = 1, \ldots, m \tag{4.9}$$

$$\sum_{e \in \delta(C_h) \cap \delta(v)} x_e \leq \sum_{e \in \delta(v) \setminus \delta(C_h)} x_e \qquad \text{pra } h = 1, \ldots, m;\ v \in N \setminus C_h \tag{4.10}$$

$$\sum_{e \in E(S)} x_e \leq r - 1 \qquad \text{para } S = \cup_{i=1}^{r} C_{l_i},\ C_1 \subset N \setminus S,\ 2 \leq r \leq m - 2 \tag{4.11}$$

$$x_e \in \{0, 1\} \qquad \text{para } e \in E. \tag{4.12}$$

La ecuación (4.8) es redundante en esta formulación. Las desigualdades (4.10) y (4.11) son las desigualdades abanico y las GSEC's básicas, respectivamente (nótese sin embargo que no todas las GSEC's están incluidas en el modelo).

Dualizamos, en una manera Lagrangiana, las desigualdades abanico (4.10), más las restricciones de grado (4.9) para $h \neq 1$. El problema relajado Lagrangiano busca $m-2$ arcos (cada uno conectando dos clusters distintos) en $E \setminus \delta(C_1)$ no induciendo ciclos entre los clusters, más dos arcos incidentes con C_1. Por tanto, puede ser eficientemente resuelto como sigue:

i) *comprimir* G con respecto a los m clusters, es decir, reemplazar cada cluster C_h con un simple con respecto a los m clusters, es decir, reemplazar *super-nodo* h, y definir para cada par de super-nodos h, k un *super-arco* $[h,k]$ con costo

$$\tilde{c}_{hk} := \min\{c'_{ij} \,:\, i \in C_h \,,\, j \in C_k\}, \tag{4.13}$$

donde c'_{ij} es el costo Lagrangiano del arco $[i,j] \in E$;

ii) calcular el *1-árbol* de costo mínimo (Held y Karp [HK71]) sobre el grafo comprimido;

iii) obtener una solución óptima del problema relajado Lagrangiano reemplazando cada super-arco $[h,k]$ en el 1-árbol detectado en el Paso ii) con su correspondiente arco $[i,j] \in E$ (aquél produciendo el mínimo en (4.13)).

El cálculo de multiplicadores de Lagrange próximos al óptimo se realiza a través de la clásica técnica del subgradiente. En nuestra implementación, iterativamente adaptamos los multiplicadores a través de dos bucles anidados. En el bucle externo adaptamos los multiplicadores para las desigualdades abanico (4.10). Con estos multiplicadores fijos, entramos en el bucle interno en el que los multiplicadores para las restricciones de grado (4.9) se ajustan de manera que se produzca un tour en el grafo comprimido. Esto se realiza inspirados por el éxito de la técnica de Help-Karp para el TSP estándar. Al final del bucle interno, si el 1-árbol final sobre el grafo comprimido es un tour, determinamos una solución del E-GTSP heurísticamente a través del procedimiento de mejora RP2 del la Sección 4.3, donde la secuencia de clusters $C_{h_1}, \ldots, C_{h_m}$ es la inducida por el tour en el grafo comprimido. Esta aproximación computacionalmente se demostró bastante eficiente para determinar soluciones próximas a la óptima al inicio de los cálculos del nodo raíz. En nuestra implementación permitimos, como mucho, 1000 y 50 iteraciones del subgradiente en el bucle externo e interno, respectivamente.

Inicialización del nodo raíz

Sean λ^*_h y μ^*_{hj} los mejores multiplicadores de Lagrange para las restricciones (4.9) y (4.10), respectivamente. El LP inicial al nodo raíz contiene las restricciones (4.2), (3.18),

las restricciones de cota sobre las variables, más el subconjunto de las desigualdades abanico (3.20) con $\mu^*_{hj} > 0$. Además, el LP contiene las GSEC's Básicas (4.11) que resultaron activas en el cálculo del 1-árbol sobre el grafo comprimido con respecto a (λ^*, μ^*). Para ser más precisos, incluimos en el LP todas las restricciones (4.11) cuyo subconjunto S se corresponde con una componente conexa detectada por el algoritmo de Kruskal [K56] para determinar el mejor 1-árbol sobre el grafo comprimido. Con esta inicialización, el valor óptimo del primer LP está garantizado ser al menos tan bueno como aquél propio de la relajación Lagrangiana.

4.4.2 "Pricing" de las variables

Los problemas lineales LP que deben resolverse podrían en principio contener un enorme número de x-variables. En la práctica, sólo un pequeño subconjunto de estas variables, digamos $\{x_e : e \in J\}$, necesitan ser consideradas explícitamente. J se inicializa heuristicamente, al nodo raíz, para contener los arcos que tienen costo reducido nulo al final de la optimización Lagrangiana. Estos costos reducidos se obtiene, en tiempo $O(n^2)$, a partir de los costos reducidos del 1-árbol calculado con los mejores multiplicadores Lagrangianos (λ^*, μ^*), como se describe en Carpaneto, Fischetti y Toth [CFT89]. Además, incluimos también los 5 arcos incidentes a cada nodo con menor costo.

Tras la resolución de cada problema LP, realizamos la operación de *"pricing"* de las variables x_e con $e \in E \setminus J$, es decir, calculamos el costo reducido en el problema lineal $\bar{c}_e := c_e - \sum_{i=1}^{q} \alpha_{ie} u^*_i$, donde q es el número de restricciones en el LP del momento, u^*_i es valor óptimo de la variable dual para la restricción i-ésima, y α_{ie} es el coeficiente de x_e en la i-ésima restricción. (Estos costos reducidos también se utilizan para fijar las variables.) Obsérvese que para cualquier par dado (i, e), los coeficientes α_{ie} pueden calcularse en tiempo constante a través de la información almacenada en la piscina.

Después de esta operación de "pricing", el conjunto J es actualizado mediante la incorporación de los arcos $e \in E^- := \{f \in E : \bar{c}_f < 0\}$ (si $|E^-| > 100$, sólo las 100 variables con menor costo reducido son incluidas). Siempre que $E^- \neq \emptyset$, resolvemos una vez más el problema LP con el mismo conjunto de restricciones y con el conjunto de variables aumentado. De esta manera, la fase de separación es llamada sólo cuando todas las variables tienen los correctos costos relativos.

Para reducir el número de variables en el LP, cuando $E^- = \emptyset$ eliminamos de J todos los arcos d con $\lceil LB + 4\bar{c}_e \rceil \geq UB$, donde LB es el valor de la relajación LP actual, y UB el valor de la mejor solución conocida.

La aproximación estándar para esta operación de "pricing" considera, una a la vez, cada variable x_e con $e \in E \setminus J$ y calcula su costo reducido asociado $\bar{c}_e$. Esta aproximación "columna-a-columna" resulta consumir en ocasiones bastante tiempo de cálculo, puede

que cada para x_e uno debe considerar explícitamente el coeficiente de x_e en cada restricción del LP (incluso cuando este coeficiente es cero). Se puede obtener algo de aceleración determinando heurísticamente un pequeño conjunto $\tilde{E} \subset E$ conteniendo los arcos que estarán con mayor probabilidad en la solución óptima (*core edge set*), y evaluar el costo reducido sólo de las variables x_e para $e \in \tilde{E} \setminus J$. Los costos relativos de las restantes variables se evalúan sólo al final del nodo actual del árbol decisional, o después de cada k (digamos) resoluciones de la LP relajación. Para otros detalles, véase por ejemplo Jünger, Reinelt y Rinaldi [JRR92].

Nosotros hemos implementado un alternativo esquema para el "pricing" "fila-a-fila" que en algunos caso, como en los considerados en este capítulo, resulta más eficiente que el estándar. Inicializamos $\bar{c}_e := c_e$ para todo $e \in E$. Luego, para cada fila del LP $i \in \{1, \ldots, q\}$ con $u_i^* \neq 0$ determinamos el subconjunto de arcos $J_i := \{e \in E : \alpha_{ie} \neq 0\}$. Usando la información en la piscina, J_i se define fácilmente en tiempo $O(n + |J_i|)$. Luego adaptamos los costos reducidos $\bar{c}_e$ para $e \in J_i$ haciendo $\bar{c}_e := \bar{c}_e - \alpha_{ie} u_i^*$. De este modo el conjunto total de variables es examinado en tiempo $O(|E| + nq + \sum_{i=1}^{q} |J_i|)$, mientras que con la aproximación estándar se tomaría un tiempo $O(|E'|\, q)$ para considerar un conjunto dado E'. Por tanto, la nueva aproximación se ejecuta mejor cuando se debe examinar los costos reducidos de una gran conjunto de variables en restricciones poco densas, como en el caso del E-GTSP. No obstante, observamos que el nuevo esquema para el "pricing" de las variables requiere un almacenamiento de $O(n^2)$ bytes de memoria para el vector de reales $\bar{c}_e$, por lo que para ejemplos de grandes dimensiones el esquema propuesto podría resultar impracticable.

En nuestro algoritmo de hiperplanos de corte, aplicamos el procedimiento de "pricing" fila-a-fila después de la resolución de cada LP.

4.4.3 Fijando variables

Algunas variables de decisión pueden fijarse a 0 o a 1 usando la información relativa a los costos relativos del nodo raíz. Sea UB el valor de la mejor solución E-GTSP disponible, y consideremos cualquier iteración del algoritmo de hiperplanos de corte (al nodo raíz) en el que todas las variables tienen el correcto signo en sus costos reducidos. Entonces el valor LB de la actual solución LP da una cota inferior válida sobre el valor E-GTSP óptimo. Sea $\bar{c}_e$ el costo reducido de $e \in E$, y sea $\bar{\gamma}_j$ el costo reducido de la variable y_j, $j \in N$. Nótese que costos reducidos negativos pueden aparecer para variables que no están en la base y que están en sus cotas superiores en la solución del LP. Entonces $LB1_e := \lceil LB + \bar{c}_e \rceil$ y $LB1'_j := \lceil LB + \bar{\gamma}_j \rceil$ dan una cota inferior sobre el valor óptimo del E-GTSP cuando $x_e = 1$ y $y_j = 1$ se imponen, respectivamente. De modo análogo, $LB0_e := \lceil LB - \bar{c}_e \rceil$ y $LB0'_j := \lceil LB - \bar{\gamma}_j \rceil$ dan una cota inferior sobre el valor óptimo del E-GTSP cuando $x_e = 0$ y $y_j = 0$ se imponen, respectivamente. Puesto que estamos

interesado en soluciones factibles de valor estrictamente menor que UB, podemos fijar

$$\begin{aligned} x_e &\equiv 0 \qquad \text{para } e \in E \text{ tal que } LB1_e \geq UB, \\ x_e &\equiv 1 \qquad \text{para } e \in E \text{ tal que } LB0_e \geq UB, \\ y_j &\equiv 0 \qquad \text{para } j \in N \text{ tal que } LB1'_j \geq UB, \\ y_j &\equiv 1 \qquad \text{para } j \in N \text{ tal que } LB0'_j \geq UB. \end{aligned}$$

Además, fijamos

$$\begin{aligned} y_j &\equiv 1 \qquad \text{si } x_e \equiv 1 \text{ para algún } e \in \delta(j), \\ x_e &\equiv 0 \qquad \text{si } y_j \equiv 0 \text{ y } e \in \delta(j), \\ y_j &\equiv 0 \qquad \text{si } y_v \equiv 1 \text{ y } j \in C_{h(v)}. \end{aligned}$$

Inconsistencias al fijar variables implican la optimalidad de UB en el nodo actual del árbol decisional.

Al contrario que LB, los valores $LB1_e$, $LB0_e$, $LB1'_j$ y $LB0'_j$ pueden decrecer en algunas iteraciones. Por ello, almacenamos estos valores en 4 arrays, y los actualizamos iterativamente (sólo en el nodo raíz) para mantener el máximo valor de cada elemento. Estos tests para fijar variables se realizan siempre que se incrementa alguno de las cotas de estos 4 arrays (sólo al nodo raíz), o cuando el UB disminuye (en cualquier nodo del árbol de ramificación).

4.4.4 Cálculo de la Cota Superior

Al comienzo del nodo raíz aplicamos la inserción del más alejado, la inserción del más cercano, y la mejor inserción, cada uno seguido por los procedimientos de mejora, tal como se describe en la Sección 4.3. Además, como se explicó en la Sección 4.4.1, para cada tour entre clusters que se detecte durante la relajación Lagrangiana, obtenemos una nueva solución factible tras aplicar el procedimiento RP2. Todas las soluciones localizadas son tratadas por los procedimientos de mejora de la Sección 4.3.

En cada nodo del árbol decisional explotamos la información asociada con el punto fraccionario dado tras la resolución de cada LP, en el intento de mejorar el actual UB. Para ello, sea (x^*, y^*) la solución óptima del LP. Inicializamos una solución heurística tomando todas los arcos e con $x^*_e = 1$, y luego la completamos mediante un esquema de inserción del más cercano. Una vez más, la solución factible resultante es tratada por un procedimiento de mejora de tour.

4.4.5 Ramificación

Consideramos dos posibilidades para la ramificación: ramificación sobre variables y ramificación sobre cortes. Sea (x^*, y^*) la solución fraccionaria LP al final del nodo actual.

La ramificación sobre variables (la aproximación estándar en los algoritmos de ramificación y corte) consiste en seleccionar una x_e^* fraccionaria, y generar los dos nodos descendientes fijando el valor de x_e o bien a 0, o bien a 1. En nuestra implementación elegimos x_e^* tan próximo como sea posible a 0.5 (los empates se rompen eligiendo el arco e con máximo costo c_e).

La ramificación sobre cortes consiste en elegir un subconjunto $S \subset N$ tal que $\sum_{e\in\delta(S)} x_e^*$ no es un entero par, e imponiendo las alternativas

$$\sum_{e\in\delta(S)} x_e \leq 2k$$

o

$$\sum_{e\in\delta(S)} x_e \geq 2k+2,$$

donde $k := \lfloor \sum_{e\in\delta(S)} x_e^*/2 \rfloor$. En nuestra implementación S se determina como sigue.

Sea $v_1, \ldots, v_m$ la secuencia de vértices correspondiente a la mejor solución E-GTSP actual, digamos $(\tilde{x}, \tilde{y})$, donde los subíndices de v deben de entenderse en módulo m. Nosotros nos restringimos a conjuntos S obtenidos como la unión de clusters consecutivos en el secuencia (es decir, $S := C_{h(v_a)} \cup C_{h(v_{a+1})} \cup \ldots \cup C_{h(v_b)}$ para algún par (a, b)), y tal que $2 + \varepsilon \leq \sum_{e\in\delta(S)} x_e^* \leq 4 - \varepsilon$ para $\varepsilon = 0.2$. Entre estos conjuntos S, si hay alguno, elegimos aquél que maximice

$$L(S) := \min\left\{d(v_i, v_j) \,:\, i = a, a+1, \ldots, b-1 \text{ y } j = b+1, b+2, \ldots, a-2\right\},$$

donde $d(v_i, v_j) := c_{v_iv_j} + c_{v_{i+1}v_{j+1}} - c_{v_iv_{i+1}} - c_{v_jv_{j+1}}$ es el costo adicional correspondiente a la nueva solución obtenida de $(\tilde{x}, \tilde{y})$ mediante el intercambio del par de arcos $([v_i, v_{i+1}], [v_j, v_{j+1}])$ con $([v_i, v_j], [v_{i+1}, v_{j+1}])$. $L(S)$ es una estimación sobre el incremento del costo de la solución óptima (y por tanto sobre la cota inferior del LP) cuando imponemos $\sum_{e\in\delta(S)} x_e \geq 4$. Eligiendo $L(S)$ tan grande como sea posible se espera producir un incremento significativo en la cota inferior de uno de los dos descendientes del nodo actual.

En nuestro estudio computacional usamos la estrategia de "ramificación sobre cortes" (que se presentó como superior), y acudimos a la "ramificación sobre variables" cuando la estrategia anterior no encuentra el corte $\delta(S)$. Dado que la solución heurística calculada al final del nodo raíz es bastante buena, hemos decidido implementar un esquema de exploración del árbol decisional del tipo "depth-first".

4.5 Resultados computacionales

El algoritmo enumerativo descrito en la Sección 4.4 ha sido implementado en lenguaje C, y ejecutado sobre un Hewlett Packard 9000 Series 700 Apollo y sobre un DEC station 5000/240. Como LP-solver hemos usado el paquete CPLEX 2.1 que contiene tanto el algoritmo Simplex primal como el dual.

Consideramos dos pequeños problemas *geográficos*, 15SPAIN47 y 27EUROPE47, correspondientes a 47 ciudades y 15 clusters (regiones) de España, y a 47 ciudades y 27 clusters (países) de Europa, respectivamente (véanse las Figuras 4.5 y 4.6; los correspondientes datos están en la Tabla 4.5). Se han considerado mayores problemas a partir de los problemas TSP de la librería TSPLIB de Reinelt [R91] con $48 \leq n \leq 442$. La procedimiento para dividir los nodos en clusters se ha realizado tratando de simular regiones geográficas, según el siguiente procedimiento. Para un problema dado, fijamos el número de clusters a $m := \lceil n/5 \rceil$. Luego determinamos m *centros* considerando m nodos tan lejos unos de otros como sea posible. Finalmente se obtienen los clusters asignando cada nodo al centro más próximo. El procedimiento resultante es el siguiente.

procedure CLUSTERING;
input: n, m, c_e para $e \in E$ (con $c_{[v,v]} := -\infty$ para todo v);
output: $h(v)$ para todo $v \in N$;
comment sea $far(S) := \arg\max\{\min\{c_{[v,w]} : w \in S\} : v \in N \setminus S\}$
el nodo m as alejado de $S \subseteq N$;
begin
1. $centro_1 := far(\{1\})$;
2. **for** $i := 2$ **to** m **do** $centro_i := far(\{centro_1, \ldots, centro_{i-1}\})$;
3. **for** $v := 1$ **to** n **do** $h(v) := \arg\min\{c_{[v,centro_i]} : i = 1, \ldots, m\}$
end.
(En arg min$\{\cdot\}$ y arg max$\{\cdot\}$, los empates se rompen eligiendo el argumento menor.)

Adicionalmente, para los problemas geográficos GR96 (Africa), GR137 (America), GR202 (Europa), GR229 (Australia-Asia) y GR431 (Australia-Asia-Europa), descritos en Grötschel y Holland [GH91], hemos considerado también un natural proceso de creación de clusters, en la que éstos se corresponden con naciones. Los problemas resultantes son 50GR96, 35GR137, 31GR202, 61GR229, y 92GR431.

Las Tablas 4.3, 4.4 y 4.5 dan resultados computacionales para los problemas test anteriores. Los tiempos son relativos a segundos CPU del un ordenador HP 9000/700. Para cada problema, la Tabla 4.3 da la siguiente información al nodo raíz:

- problem name : en la forma $mXXXXn$, donde m es el número de clusters, y

Tabla 4.1: Costos de los arcos en SPAIN47 y EUROP47

$XXXXn$ es el nombre del problema en la TSPLIB (n representa el número de nodos);

- **Lagr-LB** : porcentaje razón LB/(valor solución óptima), donde LB es la cota inferior calculada a través de la relajación Lagrangiana de la Sección 4.4.1;

- **Lagr-UB** : porcentaje razón UB/(valor solución óptima), conde UB es la cota superior al final de la relajación Lagrangiana (véase la Sección 4.4.4);

- **Lagr-t** : segundos de CPU para la relajación Lagrangiana;

- **basic-LB** : porcentaje razón LB/(valor solución óptima), donde LB es el valor óptimo de la relajación LP lineal del modelo simplificado (4.7)–(4.12);

- **root-LB** : porcentaje razón LB/(valor solución óptimo), donde LB es la cota inferior final al nodo raíz;

- **root-UB** : porcentaje razón UB/(valor solución óptimo), donde UB es la cota superior final al nodo raíz;

- **root-t** : segundos de CPU para el nodo raíz (incluyendo *Lagr-t*).

Según la tabla, la cota superior calculada a través de la relajación Lagrangiana es bastante buena. Coincide con el valor óptimo en 30 ejemplos, es decir, en el 56.6% de los casos, y en media es un 0.5% superior al óptimo. Por otra parte, la bondad de la cota inferior Lagrangiana es bastante pobre, con un error medio del 11.7%. Esto se debe tanto a la solución heurística del problema dual Lagrangiano (resuelto a través del método del subgradiente), como al hecho de que éste esta derivado del modelo simplificado (4.7)–(4.12). Obsérvese que la mejor cota inferior teórica para la relajación Lagrangiana iguala el valor óptimo de la relajación lineal del modelo (4.7)–(4.12). El último valor fue calculado a través de una versión simplificada de nuestro algoritmo de hiperplanos de corte, y se expone en la tabla (columna *basic-LB*). Puede verse que la mejora con respecto a la cota Lagrangiana es considerable, así el gran error observado deriva de la debilidad del modelo simplificado.

La Tabla 4.4 demuestra la ejecución global del algoritmo enumerativo. Para cada problema la tabla da:

- **problem name** ;

- **optval** : valor de la solución óptima;

- **total-t** : segundos CPU para la total ejecución;

- **LP-t** : segundos CPU consumidos por el LP-solver;

- **SEP-t** : segundos CPU consumidos por la separación;

- **nodes** : número de nodos explorados del árbol decisional (=0 si no se ha requerido ninguna ramificación);

- **separ.** : número total de llamadas al procedimiento SEPARACION;

- **cuts** : número total de cortes distintos producidos por SEPARACION;

- **row** : número máximo de restricciones en el LP;

- **col** : número máximo de variables no-slack en el LP.

La tabla muestra que todos los ejemplos considerados pueden resolverse hasta la optimalidad en un tiempo de cálculo aceptable. Además, una significante parte del tiempo total de cálculo es consumido por el LP solver. En un 70% de los casos no fue necesario el branching. Los resultados también muestran que el proceso de creación de clusters en modo natural produce ejemplos más fáciles que aquéllos obtenidos a través de nuestro procedimiento CLUSTERING.

La Tabla 4.5 muestra estadísticas adicionales sobre el tipo de cortes generados, y sus columnas representan:

- **problem name** ;

- **cuts** : número total de cortes generados, incluyendo aquellos encontrados por la inicialización Lagrangiana (Sección 4.4.1) y aquellos recuperados desde la piscina;

- **fan** : número total de desigualdades abanico generadas (en paréntesis aquéllos encontrados por la inicialización lagrangiana);

- **GSEC_H2** : número total de GSEC's encontradas por el procedimiento heurístico GSEC_H2 de la Sección 4.2.2;

- **GSEC_H1** : número total de GSEC's encontradas por el procedimiento heurístico GSEC_H1 de la Sección 4.2.2;

- **Gcomb** : número total de desigualdades peine generalizadas generadas;

- **pool** : número total de desigualdades violadas recuperadas desde la piscina;

En cuanto al procedimiento GSEC_SEP, nunca encontró desigualdades violadas, con la excepción del ejemplo 45TS225 donde detectó 9 cortes. Esto prueba la efectividad de nuestra separación heurística para GSEC's. Las desigualdades recuperadas de la piscina con mayor frecuencia son las GSEC's (3.9).

Para evaluar el efecto del procedimiento de creación de los clusters frente a la variación del número de nodos, hemos también considerado un segundo procedimiento que simula regiones geográficas. Dado un ejemplo del TSP, sean (x_i, y_i) las coordenadas geográficas del i-ésimo nodo $(i = 1, \ldots, n)$. Esta información está incluida en la librería TSPLIB para todos los ejemplos que consideramos en la Tabla 4.6. Sean $xmin$, $xmax$, $ymin$ y $ymax$ las mínima y máxima x- y y-coordenadas, respectivamente. Consideremos el rectángulo cuyos vértices tienen coordenadas $(xmin, ymin)$, $(xmin, ymax)$, $(xmax, ymax)$, y $(xmax, ymin)$, y subdividido para tener una malla $NG \times NG$ en la que cada celda tiene lados de longitud $(xmax - xmin)/NG$ y $(ymax - ymin)/NG$. Cada celda de la malla conteniendo al menos un nodo, se corresponde con un cluster. NG es determinado de manera que se tenga un número prefijado medio μ (parámetro de entrada) de nodos en cada cluster. Para ello, sea *CLUSTER(H)* el número de clusters no vacíos correspondientes a la malla $H \times H$, y definimos NG como el menor entero tal que *CLUSTER(NG)*$\geq n/\mu$.

La Tabla 4.6 da, para cada ejemplo y valor de $\mu = 3, 5, 7, 10$, el tiempo (segundos CPU de un HP 9000/700), el número de nodos del árbol decisional, y el número m de clusters.

Comparando la Tabla 4.6 (para $\mu = 5$) y la Tabla 4.4 se observa que el procedimento de creación de clusters en malla produce ejemplos más difíciles de resolver por nuestro algoritmo. No hay, sin embargo, correlación entre la dificultad del problema y el número medio de nodos en cada cluster.

En su globalidad, nuestras resultado computacionales con el algoritmo de ramificación y corte que se propone son bastante satisfactorias. Todos los problemas test considerados fueron resueltos en tiempos de cálculo aceptables, con la única excepción del problema TS225 y el procedimiento de creación de clusters en malla (caso $\mu = 5$ de la Tabla 4.6). Además, los algoritmo heurísticos propuestos permiten calcular muy buenas soluciones en cortos tiempos de cálculo. Como se muestra en la Tabla 4.3, después de la fase Lagrangiana, el porcentaje de error medio con respecto al valor óptimo es 0.5% (véase la columna *Lagr-UB*), y 0.1% al final del nodo raíz (véase la columna *root-UB*).

Para resumir los resultados empíricos del algoritmo de ramificación y corte, realizamos una estimación de mínimos cuadrados sobre los diversos tiempos de ejecución mostrados en la Tabla 4.4 tomando n como variable independiente, según el modelo $time = \alpha\, n^{\beta}$ y con un nivel de confianza del 95%. Las estimaciones se dan en la Tabla 4.2.

	α		β	
time	value	p-value	value	p-value
total-t	2.4E-7	6.0E-19	3.99	5.7E-24
LP-t	3.9E-8	1.3E-18	4.23	1.3E-22
SEP-t	4.5E-10	2.2E-21	4.70	4.7E-23

Tabla 4.2: Regresión mínimos-cuadrados.

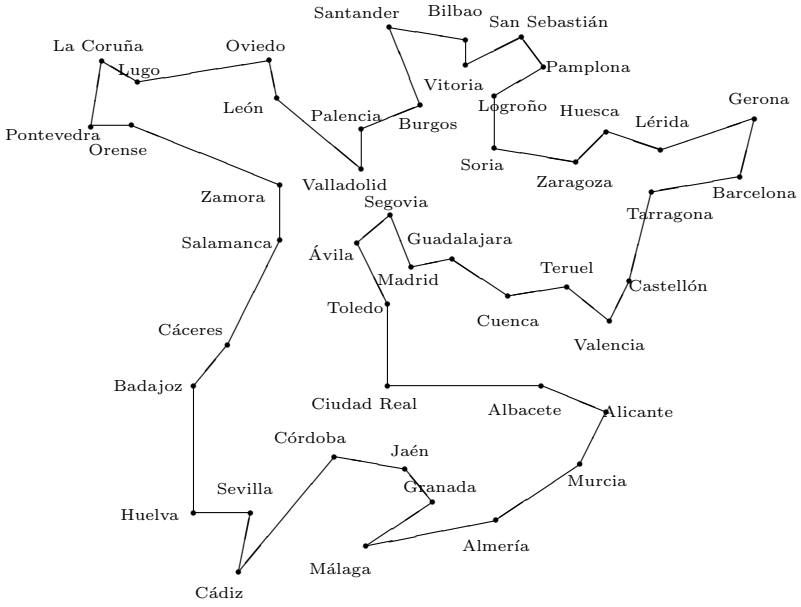


Figura 4.5: Solución óptima de 15SPAIN47.

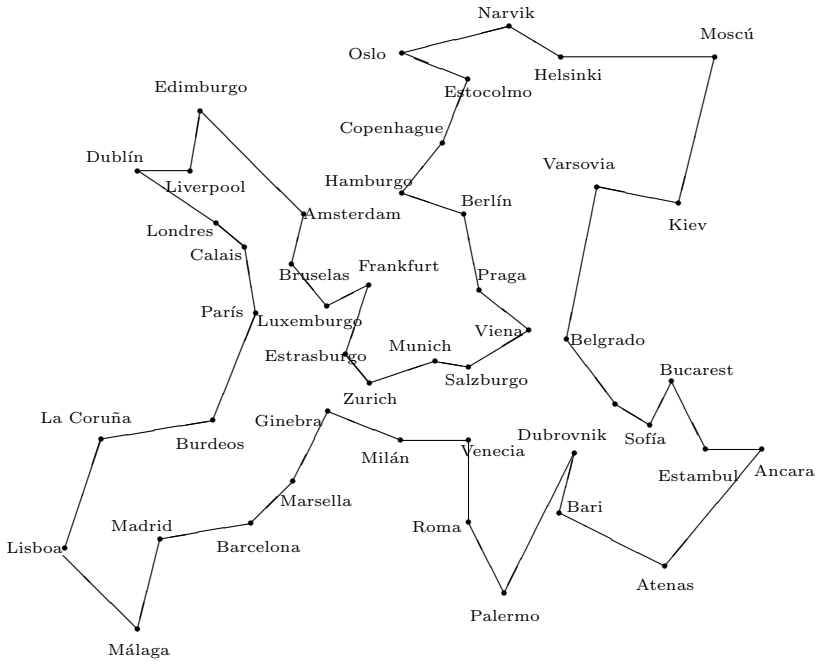


Figura 4.6: Solución óptima de 27EUROPE47.

problem name	Lagr-LB	Lagr-UB	Lagr-t	basic-LB	root-LB	root-UB	root-t
15SPAIN47	91.50	100.00	0.4	91.76	100.00	100.00	1.5
27EUROPE47	93.70	100.01	2.6	93.71	100.00	100.00	3.8
50GR96	94.94	100.37	9.4	95.01	99.81	100.37	16.2
35GR137	84.82	100.00	8.0	86.81	99.44	100.00	31.9
31GR202	85.19	100.21	13.8	85.35	99.85	100.05	464.8
61GR229	85.26	102.39	24.8	85.42	99.81	100.87	156.8
92GR431	86.36	103.55	83.8	86.49	99.84	100.05	2256.5
10ATT48	88.00	100.00	0.9	88.60	100.00	100.00	2.1
10GR48	89.75	100.00	0.5	90.53	100.00	100.00	1.9
10HK48	84.40	100.00	1.1	84.58	100.00	100.00	3.8
11EIL51	87.35	100.00	0.4	87.59	100.00	100.00	2.9
12BRAZIL58	93.11	100.00	1.4	93.13	100.00	100.00	3.0
14ST70	80.38	100.00	1.2	80.60	100.00	100.00	7.3
16EIL76	79.90	100.00	1.4	79.92	100.00	100.00	9.4
16PR76	84.26	100.00	0.6	87.05	100.00	100.00	12.9
20GR96	90.56	100.00	4.3	90.85	99.95	100.00	18.4
20RAT99	78.67	100.00	3.1	78.60	100.00	100.00	51.4
20KROA100	85.90	100.00	2.4	86.24	100.00	100.00	18.3
20KROB100	90.52	100.00	3.1	91.09	100.00	100.00	22.1
20KROC100	86.90	100.00	2.2	87.61	100.00	100.00	14.3
20KROD100	84.09	100.00	2.5	84.28	100.00	100.00	14.2
20KROE100	83.81	100.00	0.9	87.35	100.00	100.00	12.9
20RD100	87.29	100.08	2.6	87.56	100.00	100.00	16.5
21EIL101	81.52	100.00	1.7	81.54	100.00	100.00	25.5
21LIN105	90.38	100.00	2.0	90.62	100.00	100.00	16.2
22PR107	99.11	100.00	2.1	99.25	100.00	100.00	7.3
24GR120	84.00	101.99	4.9	84.45	100.00	100.00	41.8
25PR124	91.29	100.00	3.7	91.40	100.00	100.00	25.7
26BIER127	97.09	100.00	11.2	97.29	100.00	100.00	23.3
28PR136	93.04	100.82	7.2	93.28	100.00	100.00	42.8
28GR137	86.53	101.02	4.6	86.64	100.00	100.00	96.7
29PR144	99.64	100.00	2.3	99.82	100.00	100.00	8.0
30KROA150	84.13	100.00	7.6	84.24	100.00	100.00	100.0
30KROB150	88.15	100.00	9.9	88.35	100.00	100.00	60.3
31PR152	94.91	100.00	9.6	95.14	98.45	100.00	51.4
32U159	86.84	100.00	10.9	86.97	99.96	100.00	139.6
39RAT195	81.50	101.87	8.2	81.71	100.00	100.00	245.5
40D198	93.85	100.48	12.0	93.90	100.00	100.00	762.5
40KROA200	82.64	100.00	15.3	82.88	99.99	100.00	183.3
40KROB200	83.29	100.05	19.1	83.47	100.00	100.00	268.0
41GR202	92.30	100.05	20.9	92.44	100.00	100.00	1021.3
45TS225	83.54	100.09	19.4	83.62	99.11	100.09	1298.4
46PR226	96.70	100.00	14.6	97.21	100.00	100.00	106.2
46GR229	89.83	100.37	49.6	89.92	99.58	100.00	995.2
53GIL262	85.29	103.75	15.8	85.44	99.80	100.89	1443.5
53PR264	91.90	100.33	24.3	91.98	100.00	100.00	336.0
60PR299	84.48	100.00	33.2	84.67	100.00	100.00	811.4
64LIN318	92.48	100.36	52.5	92.64	99.79	100.36	847.8
80RD400	85.28	103.16	59.8	85.52	99.94	102.97	5031.5
84FL417	93.61	100.13	77.2	93.67	100.00	100.00	16714.4
87GR431	93.46	101.18	408.3	93.49	99.94	100.54	26774.0
88PR439	92.26	101.42	146.6	92.39	100.00	100.00	5418.9
89PCB442	82.74	104.22	78.8	83.03	99.49	100.29	5353.9

Tabla 4.3: Estadísticas al nodo raíz.

problem name	optval	total-t	LP-t	SEP-t	nodes	separ.	cuts	row	col
15SPAIN47	3271	1.5	0.7	0.1	0	10	39	135	186
27EUROPE47	18246	3.8	0.7	0.1	0	7	42	162	214
50GR96	36292	18.1	4.2	1.0	2	11	84	338	412
35GR137	28709	41.5	13.0	6.2	6	59	162	293	432
31GR202	14416	504.7	366.7	46.6	2	134	865	547	780
61GR229	65508	215.6	109.3	28.5	4	94	487	627	989
92GR431	75616	3365.2	2342.8	278.2	4	231	1659	1210	1942
10ATT48	5394	2.1	0.7	0.1	0	9	19	139	259
10GR48	1834	1.9	1.0	0.1	0	10	34	142	220
10HK48	6386	3.8	1.9	0.1	0	14	69	164	265
11EIL51	174	2.9	1.8	0.1	0	13	56	175	180
12BRAZIL58	15332	3.0	1.0	0.1	0	10	23	156	297
14ST70	316	7.3	4.6	0.3	0	12	100	242	317
16EIL76	209	9.4	5.9	0.3	0	20	117	253	321
16PR76	64925	12.9	8.9	0.5	0	33	144	236	396
20GR96	29072	19.4	10.2	1.1	2	27	153	321	391
20RAT99	497	51.5	36.8	3.4	0	50	344	323	404
20KROA100	9711	18.4	11.1	0.8	0	31	123	290	388
20KROB100	10328	22.2	13.5	0.6	0	36	123	309	515
20KROC100	9554	14.4	8.7	0.4	0	19	143	310	411
20KROD100	9450	14.3	7.9	0.5	0	21	107	314	512
20KROE100	9523	13.0	8.3	0.5	0	22	128	283	397
20RD100	3650	16.6	9.1	1.1	0	26	118	315	465
21EIL101	249	25.6	16.7	1.6	0	46	197	321	411
21LIN105	8213	16.4	8.8	0.7	0	24	113	310	404
22PR107	27898	7.4	2.9	0.2	0	4	14	340	701
24GR120	2769	41.9	24.9	3.2	0	55	237	364	468
25PR124	36605	25.9	14.4	0.9	0	20	117	371	578
26BIER127	72418	23.6	7.3	0.5	0	18	109	356	462
28PR136	42570	43.0	21.1	3.8	0	50	222	416	543
28GR137	35957	96.9	65.0	6.6	0	75	354	437	617
29PR144	45886	8.2	2.5	0.1	0	2	7	382	776
30KROA150	11018	100.3	63.3	8.0	0	73	368	476	701
30KROB150	12196	60.6	33.0	5.2	0	37	284	453	641
31PR152	51576	94.8	54.0	6.9	2	84	350	424	705
32U159	22664	146.4	98.0	11.2	2	65	354	486	775
39RAT195	854	245.9	167.7	26.3	0	101	776	634	861
40D198	10557	763.1	571.8	56.9	0	146	901	582	940
40KROA200	13406	187.4	108.2	27.4	2	70	420	598	943
40KROB200	13111	268.5	182.7	23.5	0	90	623	685	962
41GR202	23239	1022.2	764.1	119.3	0	176	1331	574	852
45TS225	68340	37875.9	34071.0	2481.6	190	1418	8252	1010	1273
46PR226	64007	106.9	45.4	5.0	0	48	215	570	1141
46GR229	71641	1187.5	870.3	132.2	2	172	1129	658	1042
53GIL262	1013	6624.1	5342.7	943.4	16	322	2250	911	1374
53PR264	29549	337.0	204.6	34.4	0	73	608	779	1170
60PR299	22615	812.8	583.9	43.3	0	122	929	869	1400
64LIN318	20765	1671.9	1038.8	292.6	10	214	1194	952	1259
80RD400	6361	7021.4	4721.7	1658.3	2	317	2425	1260	2165
84FL417	9651	16719.4	12232.3	2964.2	0	627	4506	1139	3395
87GR431	101523	31544.6	24873.2	4540.0	2	454	3833	1174	2739
88PR439	60099	5422.8	3636.5	896.9	0	296	2330	1340	2386
89PCB442	21657	58770.5	43753.5	12712.1	46	894	8735	1520	2326

Tabla 4.4: Estadísticas globales.

problem name	cuts	fan		GSEC_H2	GSEC_H1	Gcomb	pool
15SPAIN47	96	52	(44)	31	0	0	0
27EUROPE47	109	49	(42)	35	0	0	0
50GR96	259	133	(127)	77	0	0	1
35GR137	296	150	(101)	63	22	8	20
31GR202	1112	325	(218)	580	129	0	49
61GR229	793	355	(247)	303	30	2	44
92GR431	2219	713	(460)	1026	146	2	242
10ATT48	88	66	(61)	14	0	0	0
10GR48	98	62	(56)	28	0	0	0
10HK48	140	78	(62)	53	0	0	1
11EIL51	131	82	(65)	39	1	0	0
12BRAZIL58	95	70	(62)	15	0	0	0
14ST70	208	115	(96)	80	1	0	0
16EIL76	220	112	(89)	94	0	0	0
16PR76	241	121	(83)	83	11	0	12
20GR96	302	157	(131)	124	1	0	2
20RAT99	492	167	(130)	277	19	0	11
20KROA100	269	162	(128)	77	5	0	7
20KROB100	284	176	(143)	74	8	0	8
20KROC100	275	150	(114)	99	0	0	8
20KROD100	251	149	(126)	76	2	0	6
20KROE100	260	139	(114)	103	0	0	0
20RD100	266	159	(130)	83	4	0	2
21EIL101	337	175	(121)	115	20	0	8
21LIN105	253	157	(121)	76	0	0	1
22PR107	211	178	(177)	13	0	0	0
24GR120	410	208	(151)	168	0	0	12
25PR124	302	188	(162)	83	6	0	2
26BIER127	272	167	(139)	76	0	0	5
28PR136	431	220	(183)	155	18	0	12
28GR137	549	237	(169)	231	26	0	29
29PR144	209	175	(175)	7	0	0	0
30KROA150	594	254	(198)	264	6	0	42
30KROB150	511	243	(199)	193	41	0	6
31PR152	574	246	(195)	236	14	5	44
32U159	599	260	(215)	232	56	0	21
39RAT195	1104	395	(291)	536	98	0	38
40D198	1189	334	(250)	544	206	0	67
40KROA200	710	339	(252)	262	46	4	21
40KROB200	947	358	(286)	440	79	0	32
41GR202	1597	335	(227)	715	439	0	69
45TS225	8590	499	(295)	789	2300	165	4785
46PR226	513	314	(254)	129	10	0	16
46GR229	1428	393	(255)	573	300	14	104
53GIL262	2676	516	(375)	567	860	27	655
53PR264	1016	479	(357)	340	107	0	39
60PR299	1358	536	(371)	579	106	0	79
64LIN318	1680	585	(424)	562	305	1	165
80RD400	3092	762	(589)	1093	759	0	400
84FL417	5102	962	(514)	2182	1624	0	378
87GR431	4354	672	(436)	1466	1471	5	655
88PR439	2979	778	(563)	1641	277	0	197
89PCB442	9427	949	(605)	1824	2767	38	3762

Tabla 4.5: Estadísticas sobre el procedimiento de separación

problem name	$\mu = 3$			$\mu = 5$			$\mu = 7$			$\mu = 10$		
	time	nodes	m	time	nodes	m	time	nodes	m	time	nodes	m
ATT48	3.6	0	18	3.3	0	13	1.8	0	7	1.8	0	7
EIL51	1.9	0	25	1.4	0	16	1.6	0	9	1.6	0	9
ST70	5.5	0	24	3.9	0	16	3.9	0	16	5.2	0	9
EIL76	42.0	10	34	5.5	0	16	5.4	0	16	5.8	0	9
PR76	17.2	2	31	7.6	0	22	7.4	0	15	10.5	0	9
GR96	21.7	4	34	26.3	0	22	28.4	0	15	28.4	0	15
RAT99	25.6	0	36	26.4	0	25	56.5	0	16	56.5	0	16
KROA100	14.9	0	43	19.5	0	23	14.4	0	16	14.4	0	16
KROB100	39.2	4	44	11.5	0	25	20.6	0	16	20.6	0	16
KROC100	51.0	18	42	38.3	0	25	14.3	0	16	14.3	0	16
KROD100	24.1	2	42	19.8	0	24	25.0	0	16	25.0	0	16
KROE100	15.5	0	42	11.8	0	25	8.5	0	16	8.5	0	16
RD100	84.5	16	36	20.1	4	24	11.0	0	16	11.0	0	16
EIL101	139.2	16	36	18.2	0	25	20.3	0	16	20.3	0	16
LIN105	14.3	0	45	10.0	0	30	15.2	0	16	15.2	0	16
PR107	5.0	0	42	4.2	0	22	9.2	0	16	16.0	0	12
GR120	24.8	0	46	52.4	0	28	56.4	0	21	46.9	0	15
PR124	30.0	6	45	15.2	0	25	13.9	0	19	26.7	0	14
BIER127	234.6	16	50	210.7	6	26	69.4	0	19	25.7	0	14
PR136	181.4	28	60	15.1	0	34	22.3	0	20	13.6	0	16
GR137	81.7	0	46	94.7	0	28	284.5	0	23	972.0	0	14
PR144	28.8	0	48	25.3	0	30	9.9	0	21	21.8	0	16
KROA150	56.3	2	57	72.5	0	36	82.3	0	25	111.2	0	16
KROB150	40.5	0	56	100.0	2	36	117.4	0	25	79.2	0	16
PR152	68.7	4	54	455.2	42	33	67.9	2	24	35.5	0	16
U159	36.5	0	58	154.3	0	38	107.4	0	23	107.4	0	23
RAT195	84.4	2	81	1409.5	18	49	902.1	1	36	423.5	0	25
D198	539.5	0	67	591.2	0	40	1986.9	0	32	2849.9	0	25
KROA200	207.2	2	72	1696.3	30	47	512.2	0	35	339.5	0	25
KROB200	2031.9	54	76	119.1	0	48	132.6	0	36	422.2	0	25
GR202	2644.3	34	73	727.4	2	43	731.2	0	31	450.1	0	21
TS225	453.4	14	75	–	–	45	5427.0	12	35	10601.5	0	25
PR226	130.7	0	78	65.6	0	50	61.6	0	33	105.7	0	24
GR229	312.0	0	80	1895.1	8	46	2524.0	0	34	9391.9	2	23
GIL262	5674.6	104	96	10763.1	42	63	8160.7	49	49	1141.0	0	36
PR264	747.1	16	101	109.6	0	55	352.8	2	42	376.8	0	27
PR299	761.3	2	102	4629.9	12	69	5296.9	6	47	2730.6	0	35
LIN318	37117.4	220	108	16784.8	40	64	1355.9	0	49	71010.7	26	36
RD400	21764.6	118	135	87308.1	198	81	8947.7	6	64	21156.8	2	49
FL417	7687.9	0	142	10373.7	0	93	5364.4	0	61	919.2	0	43
PR439	1905.7	6	163	18876.5	14	96	5189.9	0	74	35652.6	8	48
PCB442	23226.1	86	155	39155.3	24	96	21268.5	0	64	15266.6	0	48

El problema TS225 con $\mu = 5$ requiere más de 100,000 segundos.

Tabla 4.6: Resultados computacionales con otra creación de clusters.

Capítulo 5

Problema de la Orientación

Consideremos un grafo no-dirigido donde cada arco tiene asociado un costo, y cada nodo un premio. El *Problema de Orientación* (que abreviaremos como *OP* al venir del inglés *Orienteering Problem*), consiste en buscar un circuito simple cuyos arcos conlleven un costo total no superior a una cantidad prefijada de antemano, al tiempo que visite los nodos que le produzcan un máximo premio total. Se trata de un problema $\mathcal{NP}$-difícil que aparece en numerosas aplicaciones referentes a problemas de rutas y secuenciaciones óptimas. En este capítulo se presenta un algoritmo de tipo *Ramificación y Corte* (*Branch and Cut*) para encontrar una solución óptima a tal problema. El algoritmo se basa en varias familias de desigualdades válidas para el modelo entero considerado, así como en los denominados *cortes condicionales*, y para ellas se describen algoritmos exactos y heurísticos de separación. También se presentan procedimientos heurísticos para producir soluciones aproximadas del OP. Al final se describe un extenso análisis computacional sobre varias clases de ejemplos procedentes de trabajos de otros autores. Con ello se observa que el algoritmo propuesto llega a resolver ejemplos con hasta 500 nodos, en tiempos de ordenador aceptables, lo que muestra favorablemente nuestro algoritmo en relación con los otros métodos hasta el momento publicados.

5.1 Introducción

Consideremos el siguiente problema de rutas, estrechamente relacionado con el *Problema del Viajante de Comercio* (TSP, *"Travelling Salesman Problem")*. Sea un conjunto de n ciudades, cada una asociada con un *premio* no negativo, y un vehículo estacionado en un depósito localizado por ejemplo en la ciudad 1. Sea $t_{ij} = t_{ji}$ el *tiempo* que dicho vehículo necesitaría para pasar directamente de una ciudad i a otra j (obsérvese que asumimos simetría en la red de carreteras). El *Problema de Orientación* (OP, *"Ori-*

enteering Problem") es el problema de encontrar una ruta para el vehículo de manera que visite cada ciudad no más de una vez, que el tiempo total empleado no exceda un tiempo máximo t_0, y que recoja un premio total máximo a su paso por las ciudades. Este problema es $\mathcal{NP}$-difícil puesto que generaliza el clásico *problema de la mochila* (KP, *"knapsack problem")*, y aparece en diversas aplicaciones de problemas de rutas y secuenciación (véase, por ejemplo, Golden, Levy y Vohra [GLV87]). Prueba de ello es que el *American Institute for Aeronautical and Astronautical Engineers* propuso este problema como un *Artificial Intelligence Design Challenge Problem* en su conferencia anual sobre "Guidance, Navigation and Control" en 1987.

En Tsiligirides [T89], Golden, Levy y Vohra [GLV87], y Golden, Wang y Liu [GWL88] se pueden encontrar algorítmos heurísticos para OP y algunas generalizaciones. En Laporte y Martello [LM90], y en Ramesh, Yoon y Karwan [RYK92] se proponen algoritmos exactos de tipo ramificación-y-acotación. Leifer y Rosenwien [LR94] presentan una forma de obtener una cota superior mediante la resolución de problemas lineales.

Hay varios problemas relacionados con el OP, tales como el *Problema de la Colección de Premios ("Prize-Collecting TSP")*, introducido por Balas y Martin [BM85] como un modelo para la secuenciación de operaciones diarias de corte de láminas de acero. En este problema se debe encontrar una ruta para un vehículo, de manera que una ciudad nunca sea visitada por él más de una vez, coleccione en su visita por las ciudades al menos una cantidad predefinida, y que el tiempo total invertido en el recorrido sea mínimo. Un detallado estudio poliédrico sobre este problema puede encontrarse en Balas [B89, B93a]. Fischetti y Toth [FT88] presentan un algoritmo exacto de tipo ramificación-y-acotación para el mismo.

Otro problema bastante relacionado con OP es el *Problema del Ciclo ("Cycle Problem")*, el cual plantea la búsqueda de una ruta para el vehículo, de manera que cada ciudad sea visitada no más de una vez, y que el costo total del recorrido sea el mínimo posible. Cuando los costos de los arcos pueden ser negativos, este problema es $\mathcal{NP}$-difícil. Balas [B93b], entre otros, han estudiado poliédricamente este problema.

En este artículo proponemos un algoritmo de ramificación-y-corte para OP. En la Sección 5.2 se presenta un modelo básico de programación matemática entera. En la Sección 5.3 se discute un número de restricciones adicionales, que incorporadas a la relajación lineal de modelo básico producen una buena cota superior. Los procedimientos de separación se describen en la Sección 5.4, mientras que en la Sección 5.5 se expone un algoritmo heurístico para encontrar soluciones aproximadas al OP. El algoritmo de tipo ramificación-y-acotación es descrito en la Sección 5.6. La Sección 5.7 muestra los resultados computacionales obtenidos al aplicar dicho algoritmos a ejemplos pertenecientes a varias familias, mostrando que en general el algoritmo es capaz de resolver difíciles ejemplos de OP con hasta 400 ciudades, y en tiempo razonables de cálculo. Ello da una comparación favorable de nuestra propuesta en relación a otras propuestas previas.

5.2 Modelo Matemático

Consideremos un grafo (no-dirigido) completo $G = (V, E)$ con $n := |V|$ nodos. El nodo 1 representa el depósito. Sea p_v el *premio* no-negativo asociado con cada $v \in V$ (con $p_1 = 0$), sea t_e el *tiempo de viaje* no-negativo asociado con cada $e \in E$, y sea t_0 el tiempo de viaje total máximo permitido para el vehículo.

Asumimos a lo largo de este capítulo que todos los valores p_v, t_e, y t_0 son enteros. Además, asumimos sin pérdida de generalidad que el OP tiene al menos una solución factible (esto puede ser controlado en tiempo polinomial, sin más que calcular el ciclo más corto que pasa a través del nodo 1).

Para cada $S \subset V$ usamos la notación

$$E(S) := \{[u, v] \in E : u \in S, v \in S\},$$

$$\delta(S) := \{[u, v] \in E : u \in S, v \notin S\},$$

y para cada $v \in V$ escribimos $\delta(v)$ en lugar de $\delta(\{v\})$. Además, para cualquier $T \subseteq E$ definimos

$$V(T) := \{v \in V : T \cap \delta(v) \neq \emptyset\}$$

como el conjunto de los nodos alcanzables con T. Dada una función de valor real f sobre un dominio finito W, y dado $S \subseteq W$, escribimos $f(S)$ en lugar de $\sum_{w \in S} f(w)$.

Usamos dos tipos de variables de decisión, x_e e y_v, asociadas con los arcos y nodos de G, respectivamente, con el siguiente significado:

$$x_e := \begin{cases} 1 & \text{si el arco } e \text{ es usado,} \\ 0 & \text{en otro caso,} \end{cases} \quad e \in E;$$

$$y_v := \begin{cases} 1 & \text{si el vértice } v \text{ es visitado,} \\ 0 & \text{en otro caso,} \end{cases} \quad v \in V.$$

Entonces OP puede ser formulado como el Problema de Programación Entera 0-1:

$$v(\text{OP}) := \max \sum_{v \in V} p_v y_v \tag{5.1}$$

sujeto a

$$\sum_{e\in E} t_e x_e \le t_0, \tag{5.2}$$
$$x(\delta(v)) = 2\,y_v \quad \text{para } v \in V, \tag{5.3}$$
$$x(\delta(S)) \ge 2\,y_v \quad \text{para } S \subset V, 1 \in S, v \in V \setminus S, \tag{5.4}$$
$$y_1 = 1, \tag{5.5}$$
$$0 \le x_e \le 1 \quad \text{para } e \in E, \tag{5.6}$$
$$0 \le y_v \le 1 \quad \text{para } v \in V \setminus \{1\}, \tag{5.7}$$
$$x_e \text{ entera} \quad \text{para } e \in E, \tag{5.8}$$
$$y_v \text{ entera} \quad \text{para } v \in V \setminus \{1\}. \tag{5.9}$$

La restricción (5.2) impone la limitación máxima t_0 sobre el tiempo total de viaje. Las *ecuaciones de grado* (5.3) obligan a que toda solución factible deberá visitar exactamente una vez cada nodo que visite. La *Restricciones de Eliminación de Ciclos Generalizadas* (GSEC's) (5.4) fuerza a que todo nodo visitado $v \in V \setminus \{1\}$ deberá ser alcanzable desde el nodo 1 mediante dos caminos disjuntos.

Debido a las ecuaciones de grado (5.3), las GSEC's pueden ser equivalentemente escritas como

$$x(E(S)) \le y(S) - y_v \quad \text{para } S \subset V, 1 \in S, v \in V \setminus S \tag{5.10}$$

y

$$x(E(\bar{S})) \le y(\bar{S}) - y_v \quad \text{para } \bar{S} \subset V, 1 \in V \setminus \bar{S}, v \in \bar{S}. \tag{5.11}$$

Observar que las desigualdades

$$x(\delta(S)) \ge 2(y_i + y_j - 1) \quad \text{para } S \subset V, 1 \in S, i \in S, j \in V \setminus S$$

aunque válidas, están dominadas por (5.4) ya que $y_i - 1 \le 0$ para todo $i \in V$. Finalmente (5.5) impone que el nodo 1 deba ser visitado, y (5.6)–(5.9) exige que todas las variables tomen valores 0 ó 1.

Nótese que los ciclos con sólo dos nodos no son posibles según este modelo, ya que el mejor de tales ciclos puede encontrarse muy fácilmente, por separado, mediante simple enumeración en tiempo $O(n)$.

5.3 Desigualdades adicionales

En esta sección describimos las cinco clases de desigualdades adicionales para el OP, que son usadas por el algoritmo de Ramificación y Corte que se propone. Estas desigualdades son irrelevantes en el modelo entero, pero son capaces de fortalecer mucho la relajación

lineal (5.1)–(5.7), acercando su valor óptimo al de aquél. Las dos primeras familias no aprovechan la restricción (5.2), y derivan de la llamada *relajación a ciclo* del OP, tratada en Balas [B93b]. Las restantes clases, sin embargo, si explotan la restricción sobre el tiempo total.

Restricciones lógicas

Claramente, $x_e = 1$ para $e \in \delta(j)$ implica $y_j = 1$. Luego las *restricciones lógicas*

$$x_e \leq y_j \text{ para todo } e \in \delta(j), j \in V \setminus \{1\} \tag{5.12}$$

son válidas para OP. Para cualquier $e = [v, j] \notin \delta(1)$, tenemos que (5.12) es un caso particular de (5.11) donde $\bar{S} = \{v, j\}$. Por otra parte, para $e \in \delta(1)$ también estas desigualdades mejoran la relajación lineal del modelo (5.1)–(5.7), y no están contempladas en (5.11). Para ver esto, consideremos el punto fraccionario (x^*, y^*) con $x^*_{12} = x^*_{13} = 1$, $y^*_1 = 1$, $y^*_2 = y^*_3 = 1/2$ (las restantes componentes son 0). Asumiendo $t_{12} + t_{13} \leq t_0$, este punto satisface todas las restricciones de la relajación, pero no las restricciones (5.12) asociadas con $e = [1, 2]$ y $j = 2$, y con $e = [1, 3]$ y $j = 3$.

Observamos además que añadiendo (5.12) al modelo (5.1)–(5.9) provoca que las exigencias de integrabilidad sobre las variables y sean ahora redundantes. En efecto, sea (x^*, y^*) un punto satisfaciendo (5.2)–(5.8), y definamos $T^* := \{e \in E : x^*_e = 1\}$. Entonces de (5.3) tenemos $y_v = |T^* \cap \delta(v)|/2$ para todo $v \in V$, i.e., $y_v \in \{0, 1/2, 1\}$. Pero $y_v = 1/2$ implicaría $T^* \cap \delta(v) = \{e\}$ para algún $e \in \delta(v)$, lo cual es imposible puesto que en tal caso las restricciones lógicas (5.12) podría ser violadas.

Desigualdades 2-emparejamiento

Las conocidas *restricciones 2-emparejamiento* para el TSP tienen la siguiente extensión en la relajación a ciclo del OP:

$$x(E(H)) + x(T) \leq y(H) + \frac{|T| - 1}{2}, \tag{5.13}$$

donde $H \subset V$ se llama *mango*, y $T \subset \delta(H)$ es un conjunto de $|T| \geq 3$, $|T|$ impar, *dientes* disjuntos dos a dos. Esta desigualdad se obtiene sumando las ecuaciones de grado para todo $v \in H$ junto con las restricciones de cota $x_e \leq 1$ para todo $e \in T$, dividiendo por 2, y luego redondeando por abajo todos los coeficientes al entero más cercano.

Desigualdades cubrimiento

La restricción sobre el tiempo total (5.2), junto con las exigencias $x_e \in \{0,1\}$ para $e \in E$, definen un ejemplo del *Problema de la Mochila 0-1* (KP), en el cual los elementos se corresponden con los arcos. Por ello, toda desigualdad válida para KP podría resultar muy útil para mejorar la relajación lineal del modelo entero para OP. Entre las diversas clases de desigualdades que se conocen para el KP, nosotros consideramos la *desigualdad cubrimiento* (véase, por ejemplo, Nemhauser y Wolsey [NW88] para más detalles):

$$x(T) \leq |T| - 1, \tag{5.14}$$

donde $T \subseteq E$ es un subconjunto de arcos minimal respecto a la inclusión, con $\sum_{e \in T} t_e > t_0$. Esta restricción impone que no todos los arcos de T puedan ser seleccionados en una solución factible del OP.

Una desigualdad cubrimiento puede en algunos casos ser fortalecida. En particular, podemos fácilmente obtener la *desigualdad cubrimiento extendida* válida siguiente:

$$x(T \cup Q) \leq |T| - 1, \tag{5.15}$$

donde $Q := \{e \in E \setminus T : t_e \geq \max_{f \in T} t_f\}$.

Otra forma diferente de mejorar la desigualdad es la siguiente, que explota el hecho que los arcos seleccionados deben definir un ciclo. Esta mejora puede sólo ser aplicada en el caso en que T defina un ciclo no-factible pasando a través del nodo 1, y lleva a la *desigualdad cubrimiento-ciclo*:

$$x(T) \leq y(V(T)) - 1. \tag{5.16}$$

La validez de (5.16) se sigue de la trivial observación de que $x(T) \geq y(V(T))$ implica $x_e = 1$ para todo $e \in T$. La Figura 5.1 muestra un punto fraccionario que no satisface una desigualdad cubrimiento-ciclo, pero si las otras restricciones anteriores.

Desigualdades camino

Las clases previas de desigualdades adicionales (salvo la desigualdad cubrimiento-ciclo) están basadas o bien en la relajación del OP al ciclo o bien al problema de la mochila. A continuación se introduce una nueva familia de restricciones que explotan ambas relajaciones al mismo tiempo.

Sea $P = \{[i_1, i_2], [i_2, i_3], \ldots, [i_{k-1}, i_k]\}$ un camino simple a través de los nodos $V(P) = \{i_1, \ldots, i_k\} \subseteq V \setminus \{1\}$, y definamos el conjunto de nodos

$$W(P) := \{v \in V \setminus V(P) : P \cup \{[i_k, v]\} \text{ puede ser parte de una solución del OP}\}.$$

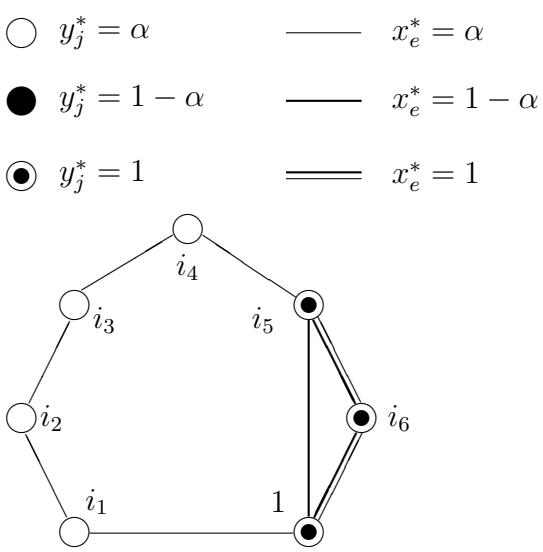


Figura 5.1: Punto fraccionario que no satisface la desigualdad cubrimiento-ciclo para el ejemplo OP con t_0=6 y t_e=1 para todo $e \in E$ $(0 < \alpha \leq 1/2)$. Aquí $T = \{[1, i_1], [i_1, i_2], \ldots, [i_6, 1]\}$, $x(T) = 2 + 5\alpha$, y $y(V(T)) = 3 + 4\alpha$.

Se permite que P represente un camino no factible, en cuyo caso $W(P) = \emptyset$. Entonces la siguiente *desigualdad camino*

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} - \sum_{j=2}^{k-1} y_{i_j} - \sum_{v \in W(P)} x_{i_k v} \leq 0 \tag{5.17}$$

es válida para OP. En efecto, supongamos que hay una solución factible (x^*, y^*) para OP violando (5.17). Entonces

$$x^*_{i_1 i_2} + (x^*_{i_2 i_3} - y^*_{i_2}) + \ldots + (x^*_{i_{k-1} i_k} - y^*_{i_{k-1}}) - \sum_{v \in W(P)} x^*_{i_k v} \geq 1,$$

donde $x^*_{i_j i_{j+1}} - y^*_{i_j} \leq 0$ para todo $j = 2, \ldots, k-1$. Se sigue entonces que $x^*_{i_1 i_2} = 1$ (luego $y^*_{i_2} = 1$), $x^*_{i_2 i_3} - y^*_{i_2} = 0$ (luego $x^*_{i_2 i_3} = 1$ e $y^*_{i_3} = 1$), …, $x^*_{i_{k-1} i_k} - y^*_{i_{k-1}} = 0$ (luego $x^*_{i_{k-1} i_k} = 1$), y $x^*_{i_k v} = 0$ para todo $v \in W(P)$. Pero entonces la solución (x^*, y^*) no podría ser factible, ya que entonces contendría todos los arcos de P, más un arco $[i_k, w]$ con $w \notin W(P)$.

La Figura 5.2 demuestra un punto fraccionario típico que es eliminable mediante una desigualdad camino. Este punto puede ser visto como la combinación convexa de dos ciclos, uno de los cuales es no factible debido a la limitación máxima sobre el tiempo total de viaje. No es difícil comprobar que sin embargo el punto satisface todas las anteriores desigualdades.

La definición de $W(P)$ implica comprobar si para cada $v \in V \setminus V(P)$ existe un ciclo de la forma $C = P_1 \cup (P \cup \{i_k, v\}) \cup P_2$, donde P_1 y P_2 son caminos de nodos disjuntos desde

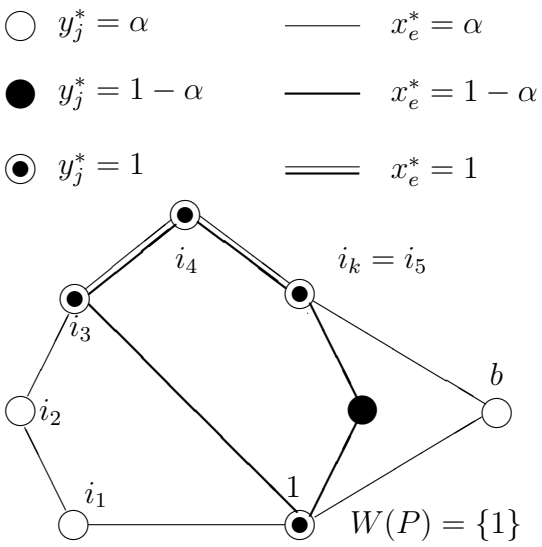


Figura 5.2: Un punto fraccionario no satisfaciendo una desigualdad camino para el ejemplo del OP con t_0=6 y t_e=1 para todo $e \in E$ ($0 < \alpha \leq 1/2$).

1 hasta i_1 y v, respectivamente, tales que $t(P_1) + t(P) + t_{i_k v} + t(P_2) \leq t_0$. Una condición mucho más sencilla (produciendo posiblemente conjuntos $W(P)$ mayores, y por lo tanto debilitando la desigualdad (5.17)) se obtiene al eliminar la exigencia de que P_1 y P_2 no puedan compartir nodos (salvo el nodo 1). Esto lleva a la definición alternativa de $W(P)$ como

$$W(P) := \{v \in V \setminus V(P) : d(1, i_1) + t(P) + t_{i_k v} + d(1, v) \leq t_0\}, \tag{5.18}$$

donde para cada $j \in V \setminus \{1\}$, $d(1, j)$ da el tiempo total asociado con el camino más corto desde el nodo 1 al nodo j.

Cortes condicionales

A continuación mostramos desigualdades que no tienen porque ser necesariamente válidas para todo el problema, pero que sin embargo pueden ser usadas en un algoritmo de hiperplanos de corte.

Supongamos disponer de una solución heurística del OP de valor LB. En el proceso de encontrar una solución óptima del OP estamos claramente interesados en encontrar, si existe, una solución factible de valor estrictamente mayor que LB. Por tanto, cualquier desigualdad puede ser usada como un hiperplano de corte supuesto que sea verificada por toda solución factible del OP con valor estrictamente mayor que LB. A tales desigualdades les damos el nombre de *desigualdades condicionales*.

Consideremos una familia general de desigualdades del tipo

$$x(T) \leq y(V(T)) - 1, \tag{5.19}$$

donde $T \subset E$ se debe elegir en la forma apropiada. Se puede ver fácilmente que $x(T) \leq y(V(T))$ se verifica para toda solución factible, sin importar cómo se elige T. Además, $x(T) = y(V(T))$ implica que la solución del OP consiste en un ciclo completamente contenido en T. Se sigue entonces que (5.19) puede ser usado como un corte condicional, supuesto que no hay contenida en T ninguna solución factible con valor estrictamente mayor que LB. Esto ocurre, en particular, cuando

$$T = E(S) \text{ para algún } S \subset V \text{ tal que } 1 \in S \text{ y } \sum_{v \in S} p_v \leq \text{LB}. \tag{5.20}$$

Una forma distinta de definir también cortes condicionales, basada en técnicas de enumeración, será descrita en la sección siguiente.

5.4 Algoritmos de separación

En esta sección señalamos algoritmos exactos y/o heurísticos para el siguiente *problema de separación*: Sea $\mathcal{F}$ una de las familias de las desigualdades descritas en la Sección 5.3 para el OP; dado un punto $(x^*, y^*) \in [0,1]^{E \times V}$ que satisface (5.2)–(5.3), encontrar un miembro $\alpha x + \beta y \leq \gamma$ de $\mathcal{F}$ que sea (el más) violado por (x^*, y^*), si existe.

Denotamos por $G^* = (V^*, E^*)$ el *grafo soporte* asociado con el punto dado (x^*, y^*), donde $V^* := \{v \in V : y^*_v > 0\}$ y $E^* := \{e \in E : x^*_e > 0\}$.

GSEC's (5.4)

Consideremos x^*_e como la capacidad asociada con cada arco $e \in E^*$. Para cada $v \in V^* \setminus \{1\}$ fijado, determinamos la desigualdad más violada de tipo GSEC (entre aquéllas para el nodo v dado) mediante un $(1, v)$-corte, que denotaremos como $(S_v, V^* \setminus S_v)$, sobre G^*. Ello requiere una complejidad $O(|V^*|^3)$, en el peor de los casos, usando un algoritmos de flujo máximo. Intentando con todos los posibles $v \in V^* \setminus \{1\}$ obtenemos un algoritmo de separación que requiere una complejidad máxima de $O(|V^*|^4)$.

En nuestra implementación consideramos los nodos v en orden decreciente de su valor y^*_v. Siempre que se determina una desigualdad violada de tipo GSEC para (por ejemplo) el par S_v y v, incrementamos la capacidad x^*_{1v} en la cantidad $2 - x^*(\delta(S_v))$. Esto previene reobtener el corte $(S_v, V^* \setminus S_v)$ en iteraciones posteriores. Además, para incrementar el número de desigualdades violadas detectadas en una simple aplicación del algoritmo del

flujo máximo, consideremos dos $(1, v)$-cortes de capacidad mínima para cada v, denotados como $(S'_v, V^* \setminus S'_v)$ y $(V^* \setminus S_v, S_v)$, siendo S_v (respectivamente, S'_v) el conjunto que contiene los nodos alcanzables desde v (respectivamente, nodo 1) en el grafo incremental correspondiente al vector solución del flujo máximo. El conjunto S_v da una posible desigualdad violada de tipo GSEC, mientras que S'_v es usado, como se comenta posteriormente, para producir un corte condicional.

Restricciones lógicas (5.12)

Esta familia puede ser tratada mediante una simple enumeración total, con la complejidad computacional de $O(|E^*|)$.

Restricciones 2-emparejamiento (5.13)

Estas desigualdades pueden ser separadas en tiempo polinomial a través de una simple modificación del esquema de separación de los cortes impares de Padberg y Rao [PR82]. Para reducir el esfuerzo computacional empleado en la separación, sin embargo, hemos implementado la siguiente simple heurística. Los valores x^*_e se consideran como pesos asociados a los arcos. Aplicamos el algoritmo voraz de Kruskal para encontrar un árbol generador de peso mínimo sobre G^*. En cada iteración en la que este algoritmo selecciona un nuevo arco e, determinamos (en el subgrafo G^* inducido por todos los arcos seleccionados hasta el momento) la componente conexa que contiene a e, digamos H. El conjunto de nodos H se considera entonces como el mango de una posible restricción 2-emparejamiento violada, cuyos dientes se determinan (de una manera óptima) a través del siguiente procedimiento voraz. Sea $\delta(H) = \{e_1, \ldots, e_p\}$ con $x^*_{e_1} \geq x^*_{e_2} \geq \ldots \geq x^*_{e_p}$. Inicialmente relajamos el requerimiento de que los dientes deban ser disjuntos a pares. Para cada $|T| \geq 3$ e impar, la mejor elección para T consiste en los arcos $e_1, \ldots, e_{|T|}$. Por lo tanto, una desigualdad con máxima violación se corresponde con la elección de un número impar $|T| \geq 3$ que maximiza:

$$x^*_{e_1} + (x^*_{e_2} + x^*_{e_3} - 1) + \ldots + (x^*_{e_{|T|-1}} + x^*_{e_{|T|}} - 1).$$

Si no se obtiene ninguna desigualdad violada con esta idea, entonces claramente no existe ninguna desigualdad 2-emparejamiento con el mango H. En otro caso, tenemos una desigualdad 2-emparejamiento violada en la que dos arcos dientes, digamos e y f, pueden coincidir en un nodo, digamos v. En tal caso, *simplificamos* la desigualdad definiendo una nueva pareja mango-dientes (H', T') con $T' := T \setminus \{e, f\}$, y $H' := H \setminus \{v\}$ (si $v \in H$) o $H' := H \cup \{v\}$ (si $v \notin H$). Es entonces fácil ver que la desigualdad (5.13) asociada con este nuevo par (H', T') está al menos tan violada como aquélla asociada con la pareja original (H, T). En efecto, reemplazando (H, T) por (H', T') se incrementa la violación

por, al menos, $1 + y_v - x(\delta(v)) \geq 2y_v - x(\delta(v)) = 0$ (si $v \in H$), o $1 - y_v \geq 0$ (si $v \notin H$). Iterando este sencillo paso, podemos detectar siempre una desigualdad 2-emparejamiento violada sin dientes sobrepuestos. (En algunos casos este procedimiento incluso podría llevar a una restricción de 2-emparejamiento con $|T| = 1$; si esto ocurre, rechazamos la restricción en favor de una GSEC asociada al mango.)

Desigualdades camino (5.17)

Asumanos que el punto fraccionario (x^*, y^*) satisface todas las desigualdades lógicas (5.12) como sucede en el caso de nuestra implementación. Observamos que la desigualdad camino asociada con un camino dado P no puede ser violada por (x^*, y^*) si $x^*_{i_h i_{h+1}} = 0$ para algún $[i_h, i_{h+1}] \in P$. Esto se sigue del hecho que (5.17) se puede re-escribir como

$$\sum_{j=1}^{h-1}(x_{i_j i_{j+1}} - y_{i_{j+1}}) + x_{i_h i_{h+1}} + \sum_{j=h+1}^{k-1}(x_{i_j i_{j+1}} - y_{i_j}) - \sum_{v \in W(P)} x_{i_k v} \leq 0$$

donde todos los términos que aparecen en los dos primeros sumatorios son no-positivos por hipótesis. Así cada desigualdad camino violada está asociada con un camino P contenido en el grafo soporte G^*. Puesto que este grafo es normalmente muy poco denso, nosotros hemos implementado un sencillo procedimiento de enumeración para detectar los caminos P que produzcan desigualdades camino de máxima violación. Comenzamos con una secuencia vacía de nodos P. Luego, iterativamente, extendemos el actual P hacia cualquiera de sus posibilidades, comprobando que la desigualdad camino asociada se mantenga violada. Siempre que el camino actual $P = \{[i_1, i_2], \ldots, [i_{k-1}, i_k]\}$ verifique

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} - \sum_{j=2}^{k-1} y_{i_j} \leq 0,$$

retrocedemos en la búsqueda ya que sabemos que ninguna otra extensión de P puede llevar a una desigualdad camino violada.

Desigualdades cubrimiento (5.14)

Primero afrontamos el problema de la separación de las desigualdades cubrimiento (5.14), lo que supone buscar un subconjunto de arcos T con $\sum_{e \in T} t_e > t_0$ maximizando $x^*(T) - |T| + 1$. Es bien sabido que este problema puede formularse como

$$\sigma^* := \min \sum_{e \in E}(1 - x^*_e) z_e \tag{5.21}$$

sujeto a

$$\sum_{e\in E} t_e z_e \geq t_0 + 1, \tag{5.22}$$

$$z_e \in \{0,1\} \qquad \text{para todo } e \in E. \tag{5.23}$$

Si $\sigma^* \geq 1$ entonces no existe ninguna desigualdad cubrimiento violada. En otro caso, $T := \{e \in E : z_e = 1\}$ induce una de tales desigualdades con máxima violación.

Si bien es $\mathcal{NP}$-difícil, el problema de la mochila (5.21)–(5.23) puede resolverse normalmente en muy poco tiempo de cálculo mediante algoritmos especializados (véase Martello y Toth [MT90]). Además, todas las variables z_e con $x_e^* = 0$ pueden fijarse a 0, ya que $z_e = 1$ implicaría $\sigma^* \geq 1$. Análogamente, se puede fijar $z_e = 1$ cuando $x_e^* = 1$, ya que en tal caso su peso en (5.21) no cuenta.

Dado que algunos pesos en (5.21) pueden ser cero, no está garantizado que el conjunto de arcos T que configura un óptimo en (5.21) sea minimal respecto a la propiedad (5.22). Para disponer de desigualdades más fuertes, primero hacemos T minimal (según una idea voraz), y luego generamos la desigualdad extendida (5.15), que será la considerada.

Veamos un algoritmo heurístico de separación para las desigualdades cubrimiento-ciclo (5.16) asociadas con un ciclo no factible T. Como en la separación de las desigualdades 2-emparejamiento, interpretamos los valores x_e^* como pesos asociados con los arcos, y calculamos un árbol generador de peso mínimo sobre G. Luego consideramos, uno a uno, los arcos $e \in E^*$ que no están en el árbol: si añadir e al árbol conlleva un ciclo T que pasa a través del nodo 1 y tal que $\sum_{e\in T} t_e > t_0$, entonces tenemos una desigualdad válida (5.16) cuya violación es examinada.

Cortes condicionales (5.19)

Hemos implementado dos procedimientos heurísticos de separación para los cortes condicionales. Sea LB el mejor valor solución obtenido hasta el momento.

El primer procedimiento está basado en la condición (5.20), y es tratado inmerso dentro con el algoritmo de separación para las restricciones GSEC's antes descritas. Para cada conjunto S'_v, con $1 \in S'_v$, allí descrito y tal que $\sum_{v\in S'_v} p_v \leq LB$, fijamos $T = E(S'_v)$ y estudiamos la violación de la correspondiente (5.19).

Nuestro segundo procedimiento se basa en la observación de que (5.19) puede *siempre* usarse como un corte condicional, supuesto que la cota inferior LB sea previamente adaptada teniendo presente todas las soluciones factibles del OP completamente contenidas en T. Esto supone calcular

$$\text{LB} := \min\{\text{LB}, v(\text{OP}_T)\},$$

donde $v(\mathrm{OP}_T)$ es el valor óptimo del OP cuando $x_e = 0$ viene impuesto para todo arco $e \in E \setminus T$. Aunque el cálculo de $v(\mathrm{OP}_T)$ precisa un tiempo computacional exponencial en el caso peor, para un conjunto de arcos T suficientemente disperso, incluso un simple esquema de enumeración exhaustiva puede resultar satisfactorio en el cálculo práctico de $v(\mathrm{OP}_T)$, con poco tiempo de cálculo. En nuestra implementación, definimos $T := E^*$, asegurando así que el correspondiente corte condicional (5.19) será violado ya que $x^*(T) = x^*(E)$ e $y^*(V(T)) = y^*(V)$, donde $x^*(E) = y^*(V)$ debido a las ecuaciones de grado (5.3). Luego aplicamos un simple algoritmo para resolver el OP sobre el grafo soporte G^*, basado en una enumeración exhaustiva. Si la enumeración acaba en un tiempo máximo prefijado TL, entonces (tras adaptar el valor LB) se tiene garantía de poder añadir (5.19) al LP actual.

5.5 Algoritmos heurísticos

La ejecución de procedimiento de ramificación y corte en su globalidad mejora si uno es capaz de detectar pronto "buenas" soluciones factibles para el OP. A tal fin proponemos el siguiente procedimiento heurístico, trabajando en dos fases. En la primera fase se determina un ciclo factible C, tratándose de que contenga el mayor número posible de arcos en una solución óptima. En la segunda fase se aplican procedimientos de mejora para producir a partir de C un ciclo factible próximo a ser óptimo.

Como argumentos a la primera fase, el heurístico recibe una estimación w_e, $0 \leq w_e \leq 1$, de la probabilidad de que el arco e esté en una solución óptima. El cálculo de tales valores w_e se describe en la Sección 5.6.2. Ordenamos luego los arcos en orden decreciente de w_e, deshaciendo los empates de manera que se consideren primero los arcos con menor duración t_e. Luego detectamos heurísticamente, según una idea voraz, un subconjunto de arcos T conteniendo una familia de caminos disjuntos en arcos, y con la mayor probabilidad de formar parte de una solución óptima. Más concretamente, inicializamos $T := \emptyset$ y luego consideramos, uno a la vez, cada arco e según el orden anterior: Si $T \cup \{e\}$ contiene un nodo con grado mayor que 2, rechazamos el arco; en otro caso, adaptamos $T := T \cup \{e\}$ si $T \cup \{e\}$ no contiene ciclos, o bien paramos el algoritmo si lo contiene.

Comenzando con T, obtenemos el requerido ciclo factible C mediante los siguientes pasos. Primero eliminamos del grafo todos los nodos en $V \setminus \{1\}$ que no son cubiertos por T. Luego, unimos los caminos de T en un ciclo simple, C. Para esto aplicamos un esquema simple de tipo "el vecino más próximo", comenzando desde el nodo 1 y moviéndonos iterativamente hacia el nodo no cubierto más próximo, extremo de un camino en T. Al final de este proceso, controlamos si $\sum_{e \in C} t_e \leq t_0$. Cuando esto no sucede, aplicamos el siguiente procedimiento para convertir a C en un ciclo factible. Para cada nodo v

cubierto por C, sean i_v y j_v los dos vecinos de v en C. El procedimiento iterativamente elimina un nodo v de C, es decir, reemplaza los arcos $[i_v, v]$ y $[j_v, v]$ por $[i_v, j_v]$. En cada iteración elegimos (si es posible) un nodo v con el menor premio y cuya eliminación convertiría a C en factible, o si no existe ninguno entonces un nodo v que minimiza el valor $p_v/(t_{i_v v} + t_{j_v v} - t_{i_v j_v})$.

La segunda fase de nuestro heurístico recibe como entrada el ciclo factible C calculado en la fase primera, e iterativamente trata de mejorarlo. En cada iteración, primero ejecuta una 2-optimalidad intercambiando arcos dentro de C esperando reducir su duración total. Luego tratamos de añadir a C un nodo con premio máximo perteneciente al conjunto $Q(C)$ conteniendo los nodos v no cubiertos por C, y tal que $\min_{[i,j]\in C}\{t_{iv} + t_{jv} - t_{ij}\} \leq t_0 - \sum_{e\in C} t_e$. Si $Q(C) \neq \emptyset$, ejecutamos la inserción de un nodo y repetimos. En otro caso, reaplicamos el procedimiento en su globalidad sobre el ciclo obtenido desde C eliminando, uno a la vez, uno de sus nodos.

5.6 Algoritmo de Ramificación y Corte

A continuación describimos cada uno de los componentes de nuestro algoritmo de ramificación y corte para la solución óptima del OP. Asumimos que el lector tiene conocimientos básicos de las técnicas de ramificación y corte al nivel descrito, por ejemplo, en Padberg y Rinaldi [PR91] o en Reinelt, Jünger y Rinaldi [JRR92].

5.6.1 Inicialización

Al inicio del nodo raíz del árbol decisional, calculamos una cota inferior sobre el valor óptimo del OP a través del algoritmo heurístico descrito en la Sección 5.5, con pesos $w_e = 0$ para todo $e \in E$. Adicionalmente, fijamos el primer Programa Lineal (LP) a ser considerado, formado por:

1. todas las variables y_v, $v \in V$;
2. las variables x_e asociadas con arcos pertenecientes a la solución heurística inicial;
3. para todo $v \in V$, las variables x_e asociadas con los 5 arcos de menor costo $e \in \delta(v)$;
4. la restricción de tiempo total (5.2);
5. las n ecuaciones de grado (5.3);
6. las cotas inferiores y superiores sobre las variables.

Finalmente, inicializamos con vacío una estructura de datos llamada *piscina*, que contendrá las restricciones del OP que no están incluidas en el LP actual, pero que lo estuvieron en alguna iteración precedente.

5.6.2 La fase de hiperplanos de corte

En cada nodo del árbol decisional, determinamos una solución primal y otra dual del LP del momento, digamos (x^*, y^*) y u^* –cuando el LP del momento resulte ser no-factible, introducimos una variable artificial con costo negativo grande. Obsérvese que el valor de la solución primal, calculado como $\sum_{v \in V} p_v y_v^*$, no está garantizado que produzca una cota superior del valor óptimo del OP, ya que el LP del momento contiene sólo un subconjunto de las x-variables. Pasamos luego a la llamada *fase de costos ("pricing phase")*, en la que usamos la solución dual u^* para calcular los costos reducidos $\bar{c}_e$ de las variables x_e que no forman parte del LP del momento, y que por defecto están fijas a valor 0. Si algunas resultan tener costo reducido con signo contrario (es decir, $\bar{c}_e > 0$), las añadimos al LP, y re-optimizamos con el algoritmo del simplex primal. A fin de mantener el tamaño del LP tan pequeño como sea posible, nunca añadimos más de 100 variables en cada iteración de la fase de costos (elegidas entre aquéllas con mayor costo reducido). Iteramos la fase de costos hasta que se logre un signo correcto en el costo reducido de todas las variables, y por tanto, hasta que el valor del LP del momento, digamos UB, sea una verdadera cota superior del valor de la solución óptima del OP. En tal caso si el nodo actual no resultase eliminado, pasamos a la fase de limpieza del LP. Sea LB el valor de la mejor solución del OP conocida hasta el momento. Eliminamos del LP del momento:

1. todas las variables x_e con $\lfloor \mathrm{UB} + 4\bar{c}_e \rfloor \leq \mathrm{LB}$;
2. todas las restricciones que han sido slack en las últimas 5 iteraciones, o aquéllas cuya variable de holgura exceda 0.01.

Además, en el nodo raíz del árbol decisional fijamos a 0 todas las variables x_e con $\lfloor \mathrm{UB} + \bar{c}_e \rfloor \leq \mathrm{LB}$, y a 1 todas las variables con $\lfloor \mathrm{UB} - \bar{c}_e \rfloor \leq \mathrm{LB}$ (esta última condición puede aplicarse sólo a variables del LP fijas a su cota superior).

Luego pasamos a la *fase de separación*, en la que las restricciones violadas por (x^*, y^*) son identificadas y añadidas al LP del momento. Aplicamos los algoritmos del separación descritos en la Sección 5.4. Primero buscamos en la estructura piscina. Luego contrastamos en secuencia: las restricciones lógicas (5.12), las GSEC's (5.4), las restricciones 2-emparejamiento (5.13), las desigualdades de cubrimiento (5.14), las desigualdades camino (5.17), y los cortes condicionales (5.19). Como tiempo límite TL para la enumeración usada en la separación de los cortes condicionales, hemos considerado TL:=5·TS, donde TS es el tiempo de cálculo consumido en la última fase de separación. Siempre que un

subproceso de la fase de separación logra encontrar restricciones violadas, salimos de tal secuencia de procesos y añadimos todos los cortes encontrados al LP.

Para reducir el fenómeno de poco avance en la disminución del valor del LP ("tailing off"), saltamos a la fase de ramificación siempre que la cota superior no haya mejorado en al menos 0.001 durante las últimas 10 iteraciones del algoritmo de hiperplanos de corte correspondiente al nodo del momento.

Cada cinco aplicaciones de la fase de separación, tratamos de mejorar la mejor solución disponible para el OP, a través del algoritmo heurístico descrito en la Sección 5.5. Como valores w_e de entrada para tal algoritmo, consideramos $w_e := x^*_e$ para todo $e \in E$. Esta elección se mostró computacionalmente muy eficiente y típicamente produce soluciones bastante aproximadas a las óptimas. Una heurística adicional aparece implícita en nuestro segundo proceso de separación para los cortes condicionales (5.19), como se describe en la Sección 5.4. En efecto, la enumeración de las soluciones OP contenidas en el grafo soporte de x^*, allí requerida, puede en bastantes casos mejorar el actual LB.

5.6.3 El algoritmo enumerativo

En el nodo raíz del árbol decisional, inicializamos el LP actual y la solución heurística (según se describe en la Sección 5.6.1) y entramos en la fase de hiperplanos de corte detallada en la Sección 5.6.2. Cuando todos los algoritmos de separación fallan, si el nodo actual no resulta eliminado se pasa a la fase de ramificación. No obstante, hemos implementado (sólo para el nodo raíz) el siguiente esquema alternativo.

Según nuestra experiencia, el corte condicional asociado con el grafo soporte $G^* = (V(E^*), E^*)$ de la solución (x^*, y^*) del LP del momento, es decir

$$x(E^*) \leq y(V(E^*)) - 1, \tag{5.24}$$

es bastante efectivo para acercar el valor LB al valor óptimo del OP. Desgraciadamente, para grafos G^* más bien densos nuestro sencillo esquema de enumeración no completa la enumeración de todas las posibles soluciones del OP contenidas en G^*, en el corto límite de tiempo que le damos. Decidimos añadir el corte (5.24) al LP incluso cuando esta enumeración no tiene éxito, y lo llamamos *corte de ramificación cubrimiento*. Esta elección puede sin embargo eliminar como solución factible soluciones óptimas para el OP, siempre que éstas estén contenidas en G^*. Tomamos en consideración estas posibilidades almacenando el grafo G^*, con el objetivo de tratarlo en otro momento. Con el corte de cubrimiento-ramificación añadido al LP, retomamos la fase de hiperplanos de corte, hasta que otra vez vuelva a fallar la separación en su búsqueda de cortes violados por la solución fraccionaria del momento. En tal caso, si se precisa, el esquema anterior es repetido iterativamente: añadimos un nuevo corte de cubrimiento-ramificación almacenamos el grafo soporte actual G^*, y re-entramos en la fase de hiperplanos de corte.

De esta forma, producimos y almacenamos una secuencia de grafos soportes, digamos $G_i^* = (V(E_i^*), E_i^*)$ para $i = 1, \ldots, k$, hasta que el nodo raíz sea eliminado. En tal punto, la resolución del OP no ha terminado puesto que ahora debemos considerar la determinación de la mejor solución del OP en cada uno de los grafos $G_1^*, \ldots, G_k^*$ o, alternativamente, en la "unión" de los mismos, definida como $\tilde{G} = (V(\tilde{E}), \tilde{E} := \cup_{i=1}^{k} E_i^*)$. Para ello eliminamos de la estructura piscina todos los cortes de ramificación cubrimiento, y reaplicamos nuestro algoritmo de ramificación y corte sobre el nuevo OP asociado con $\tilde{G}$. Para poder garantizar la convergencia del algoritmo, evitamos la generación de cortes de ramificación cubrimiento en esta segunda aplicación del algoritmo de ramificación y corte. En su lugar, siempre que un nodo del árbol decisional no puede ser eliminado pasamos a la fase de ramificación, en la forma tradicional, fijando $x_f = 0$ o $x_f = 1$ para una variable x_f elegida como sigue. Seleccionamos las 15 variables fraccionarias x_e con x_e^* lo más cerca posible a 0.5. Para cada una de tales variables x_e candidatas a la ramificación, calculamos dos valores, digamos UB_e^0 y UB_e^1, resolviendo el LP del momento aumentado con la restricción adicional $x_e = 0$ y $x_e = 1$, respectivamente. Luego, elegimos como variable para la ramificación aquélla que ha producido el mayor valor $0.75 \cdot \mathrm{UB}_e^0 + 0.25 \cdot \mathrm{UB}_e^1$.

Como se ha explicado, nuestro esquema de ramificación y corte trabaja en dos fases. En la primera fase, evitamos la ramificación introduciendo los cortes de ramificación-cubrimiento. En la segunda fase, trabajamos sobre el grafo no-denso $\tilde{G}$ (resultante de los cortes de ramificación cubrimiento producidos en la primera fase), y usamos a clásica estrategia de ramificación para alcanzar la optimalidad. Nuestra experiencia computacional muestra que nuestro esquema en su totalidad se presenta típicamente mejor (aunque no lo domina) que el esquema clásico. En efecto, la segunda fase toma ventaja clara de un enorme número de cortes relevantes (producidos en la primera fase y almacenados en la piscina), así como de una muy buena solución aproximada del OP. Por otra parte, para algunos problemas la primera fase presenta una lenta convergencia en sus últimas iteraciones, debido al fenómeno de poco avance en la disminución del valor del LP. Para contrarrestar este comportamiento, aplicamos también la fase de ramificación clásica usando variables en al primera fase. En concreto, aplicamos la fase de ramificación cada vez que se intenta añadir el sexto corte de ramificación cubrimiento en un nodo del árbol decisional.

5.7 Resultados computacionales

El algoritmo de ramificación y corte descrito en las secciones previas ha sido implementado en Lenguaje C, y ejecutado sobre un ordenador Digital DECstation 5000/240 y sobre un Hewlett Packard Apollo 9000/720. Utilizamos CPLEX 3.0 como algoritmo para abordar los problemas de Programación Lineal. Consideramos tres clases diferentes de problemas test.

Los problemas de la primera clase (Clase I) incluyen 15 ejemplos tomados de bibliografía propiamente del OP y del Problema de Rutas de Vehículos (VRP). Los problemas OP21, OP32, y OP33 son ejemplos del OP introducidos por Tsiligirides [T89] (con tiempos de viaje multiplicados por 100 y luego redondeados al entero más próximo). Los problemas ATT48, EIL30, EIL31, EIL33, EIL51, EIL76, EIL101, y GIL262 son ejemplos del VRP tomados de la librería de problemas TSPLIB 2.1 de Reinelt [R91]. Los problemas CMT101, CMT121, CMT151, y CMT200 son ejemplos del VRP tomados de Christofides, Mingozzi y Toth [CMT79]. En todos los ejemplos del VRP se interpretó la demanda de los clientes como el premio a recoger por visitar el correspondiente nodo.

La segunda clase de problemas (Clase II) incluye todos los ejemplos del TSP descritos en TSPLIB 2.1 con hasta 400 nodos (problemas desde ATT48 a RD400). Para esos problemas, los premios en los nodos p_j para $j \in V \setminus \{1\}$ han sido generado de tres formas distintas:

- Generación 1: $p_j := 1$;
- Generación 2: $p_j := 1 + (7141 \cdot j + 73) \bmod (100)$;
- Generación 3: $p_j := 1 + \lfloor 99 \cdot t_{1j}/\theta \rfloor$, donde $\theta := \max_{i \in V \setminus \{1\}} t_{1i}$.

La Generación 1 produce ejemplos del OP en los que el objetivo es cubrir tantos nodos como sea posible, como sucede en algunas aplicaciones. La Generación 2 intenta producir premios pseudo-aleatorios en el rango [1,100], mientras que la Generación 3 lleva a ejemplos difíciles en los que los nodos más alejados del depósito son los que ofrecen premios mayores.

En la tercera clase de problemas (Clase III) consideramos los ejemplos aleatorios usando el código FORTRAN original de Laporte y Martello [LM90]. En esta clase tanto los premios como los tiempos de viaje son generados como números aleatorios en el rango [1,100], con tiempos de viaje triangularizados a través del cálculo de caminos mínimos.

Para todas las clases de problemas, definimos el tiempo de viaje total como $t_0 := \lceil \alpha \cdot v(\text{TSP}) \rceil$, donde $v(\text{TSP})$ es la longitud de un circuito Hamiltoniano mínimo, y α es un parámetro de entrada. En todos los ejemplos tomados del TSPLIB, este valor $v(\text{TSP})$ aparece en la propia librería. Para los ejemplos OP21, OP32, OP33, CMT101, CMT121, CMT151, y CMT200, hemos considerado, respectivamente, los siguientes valores para $v(\text{TSP})$: 4598, 8254, 9755, 505, 545, 699, y 764. Para los problemas aleatorios de la Clase III, usamos el valor $v(\text{TSP})$ calculado por el código FORTRAN de Laporte y Martello.

La Tablas 5.1 – 5.6 presentan el comportamiento de nuestro programa de ramificación y corte en la resolución de las distintas clases de problemas. Cada tabla (salvo la Tabla 5.2) da:

Name : el nombre del problema;

r-time : el tiempo total empleado al final del nodo raíz;

%-LB : la razón porcentaje (optimum–LB)/optimum, donde LB es el valor de la mejor solución heurística calculada durante el nodo raíz;

%-UB : la razón porcentaje (UB–optimum)/optimum, donde UB es la cota superior calculada durante el nodo raíz;

nodes : el número total de nodos del árbol decisional examinados (1 significa que el problema no ha precisado la fase de ramificación);

cuts : el número total de cortes generados;

opt-val : el valor de la solución óptima (sólo para las Clases I y II);

%-visited : el porcentaje del número de nodos visitados por la solución óptima;

time : el tiempo de cálculo total empleado por el código de ramificación y corte.

Los tiempos de cálculo se expresan en segundos, y se refieren al tiempo CPU sobre un ordenador HP Apollo 9000/720 computer funcionando a 80 MHz, con las siguientes características: 59 SPEC's, 58 MIPS, y 18 MFlops. Impusimos un tiempo límite de 18,000 segundos (5 horas) para cada ejecución. Los ejemplos que excedieron tal tiempo límite aparecen señalados con la clave 't.l.' en la columna *time*, y los otros detalles aportados han sido calculados tratando la mejor solución factible del OP disponible como solución óptima (de esta manera, la columna *%-UB* da una cota superior del porcentaje de error aproximado).

La Tabla 5.1 se refiere a los problemas de la Clase I. Consideramos 3 valores para el parámetro α, que fueron 0.25, 0.50, y 0.75. Para esta clase de problemas también mostramos, en la Tabla 5.2, información adicional sobre el tiempo total empleado por el LP solver (*t-LP*) y los procedimientos de separación (*t-sep*), y sobre el número de restricciones generadas de tipo lógica (*log*), GSEC (*gsec*), 2-emparejamiento (*2-mat*), camino (*path*), condicional (*cond*), y ramificación cubrimiento (*b-cover*).

Las tablas 5.3 – 5.5 se refieren a los ejemplos de la Clase II, con los premios calculados según la Generación 1, 2, y 3, respectivamente. El parámetro α ha sido fijado a 0.50 (también consideramos los casos $\alpha = 0.25$ y $\alpha = 0.75$, con comparables resultados).

La Tabla 5.6 muestra los resultados medios (máximos) sobre 10 problemas random pertenecientes a la Clase III, con $\alpha = 0.2$, 0.4, 0.6, y 0.8, y $n = 25$, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500. En contraste, el algoritmo de ramificación y acotación de Laporte y Martello [LM90] cae en dificultades cuando resuelve problemas con $n = 25$ para $\alpha \geq 0.6$,

y con $n = 50$ para $\alpha \leq 0.4$. Por ejemplo, la ejecución del programa de Laporte y Martello sobre los ejemplos con $n = 25$ nodos requiere en media 0.1 segundos para $\alpha = 0.2$, 77.2 segundos para $\alpha = 0.4$, más de 2 horas para $\alpha = 0.6$; mientras que cuando $\alpha = 0.8$ no se logra resolver ningún problema en un tiempo límite de 5 horas.

Por otra parte, las ejecuciones de nuestro programa de ramificación y corte son bastante satisfactorias. En la mayor parte de los casos, las cotas superior e inferior calculadas al final del nodo raíz son bastante ajustadas, y sólo se precisan pocas llamadas a la fase de ramificación. En concreto, fuimos capaces de demostrar la optimalidad en todos los problemas aleatorios tratados de la Clase III, y en la mayor parte de los "problemas reales" de las clases I y II. Para los problemas sobrepasando el tiempo límite, la solución alcanzada está muy cerca de ser óptima, con error aproximativo nunca superior a un 0.5%.

Según la Tabla 5.2, la mayor parte de las restricciones generadas son GSEC's, 2-emparejamiento, lógicas y cortes condicionales. Para algunos problemas "difíciles", también se genera un número considerable de restricciones cubrimiento y camino.

$\alpha = 0.25$									
Name	t_0	r-time	%-LB	%-UB	nodes	cuts	opt-val	%-visited	time
op21	1150	0.6	0.0	0.0	1	64	90	33.3	0.6
op32	2064	1.2	0.0	0.0	1	80	70	25.0	1.2
op33	2439	0.8	0.0	0.0	1	72	250	36.4	0.8
att48	2657	2.6	0.0	0.0	1	129	17	37.5	2.6
eil30	96	2.8	0.0	0.0	1	140	2650	20.0	2.8
eil31	52	0.3	0.0	0.0	1	21	535	38.7	0.3
eil33	111	1.1	0.0	0.0	1	63	800	9.1	1.1
eil51	107	2.2	0.0	0.0	1	141	264	27.5	2.2
eil76	135	48.7	0.0	0.0	1	542	490	30.3	48.7
eil101	158	177.3	0.0	0.5	3	900	572	31.7	189.7
cmt101	127	57.8	0.0	0.0	1	586	530	28.7	57.8
cmt121	137	347.9	1.5	1.0	9	1315	412	33.9	612.5
cmt151	175	128.4	0.0	0.0	1	1066	824	29.1	128.4
cmt200	191	130.7	0.1	0.0	2	1161	1205	33.0	299.1
gil262	595	2541.6	1.4	1.9	18	4800	4466	29.6	t.l.

$\alpha = 0.50$									
Name	t_0	r-time	%-LB	%-UB	nodes	cuts	opt-val	%-visited	time
op21	2299	0.7	0.0	0.0	1	58	205	61.9	0.7
op32	4127	1.3	0.0	0.0	1	91	160	56.2	1.3
op33	4878	1.8	0.0	0.0	1	109	500	63.6	1.8
att48	5314	0.8	0.0	0.0	1	55	30	64.6	0.8
eil30	191	5.2	0.0	0.0	1	170	7600	26.7	5.2
eil31	103	0.5	0.0	0.0	1	32	747	58.1	0.5
eil33	221	8.5	0.0	0.0	1	215	16220	48.5	8.5
eil51	213	2.4	0.0	0.0	1	150	508	54.9	2.4
eil76	269	3.1	0.0	0.0	1	118	907	59.2	3.1
eil101	315	5.6	0.0	0.0	1	159	1049	57.4	5.6
cmt101	253	36.9	1.0	0.0	2	514	1030	56.4	55.2
cmt121	273	411.1	0.0	0.8	39	1350	715	52.9	1525.6
cmt151	350	131.8	0.1	0.1	5	451	1537	55.6	167.3
cmt200	382	147.4	0.0	0.0	17	859	2198	62.0	596.3
gil262	1189	365.8	0.2	0.2	35	2370	8456	54.8	3252.7

$\alpha = 0.75$									
Name	t_0	r-time	%-LB	%-UB	nodes	cuts	opt-val	%-visited	time
op21	3449	1.7	0.0	0.0	1	82	315	71.4	1.7
op32	6191	0.8	0.0	0.0	1	73	230	71.9	0.8
op33	7317	1.1	0.0	0.0	1	70	660	84.8	1.1
att48	7971	1.1	0.0	0.0	1	66	39	83.3	1.1
eil30	286	0.9	0.0	0.0	1	63	11550	73.3	0.9
eil31	155	1.6	0.0	0.0	1	32	865	87.1	1.6
eil33	331	1.7	0.0	0.0	1	85	26380	81.8	1.7
eil51	320	60.0	0.0	0.3	5	143	690	76.5	77.6
eil76	404	56.6	0.2	0.4	71	479	1186	84.2	457.6
eil101	472	4.7	0.0	0.0	1	98	1336	81.2	4.7
cmt101	379	130.7	0.0	0.0	1	676	1480	75.2	130.7
cmt121	409	50.9	0.0	0.0	1	495	1134	76.9	50.9
cmt151	525	114.6	0.2	0.0	7	368	2003	79.5	380.3
cmt200	573	127.9	8.2	0.2	337	2668	2881	84.0	17389.9
gil262	1784	436.5	8.1	0.2	253	2379	11195	77.2	17512.0

Tabla 5.1: Resultados para problemas de la Clase I (ejemplos OP y VRP)

$\alpha = 0.25$										
Name	t-LP	t-sep	cuts	log	gsec	2-mat	cover	path	cond	b-cov
op21	0.4	0.1	64	21	31	0	0	0	11	0
op32	0.8	0.1	80	32	37	1	0	0	9	0
op33	0.4	0.2	72	26	27	3	0	0	15	0
att48	1.5	0.5	129	44	65	8	0	0	11	0
eil30	2.2	0.2	140	54	68	0	0	0	17	0
eil31	0.2	0.0	21	11	3	1	0	0	5	0
eil33	0.8	0.0	63	32	29	0	0	0	1	0
eil51	1.4	0.3	141	48	77	1	0	0	14	0
eil76	27.7	15.4	542	82	271	5	16	11	156	0
eil101	88.3	80.8	900	129	539	41	5	25	156	4
cmt101	41.3	8.1	586	157	354	6	2	0	66	0
cmt121	271.4	239.2	1315	223	687	59	22	249	64	10
cmt151	79.4	29.7	1066	172	739	33	5	0	116	0
cmt200	148.7	101.7	1161	211	634	42	8	0	263	2
gil262	9573.5	1581.3	4800	476	3422	132	10	205	488	66

$\alpha = 0.50$										
Name	t-LP	t-sep	cuts	log	gsec	2-mat	cover	path	cond	b-cov
op21	0.4	0.1	58	27	14	0	0	0	16	0
op32	0.9	0.2	91	31	46	2	1	0	10	0
op33	1.0	0.5	109	31	47	2	0	0	28	0
att48	0.4	0.2	55	23	17	11	0	0	3	0
eil30	3.6	0.8	170	43	84	0	4	0	38	0
eil31	0.2	0.2	32	15	5	1	1	0	9	0
eil33	5.4	2.0	215	42	86	2	14	20	50	0
eil51	1.6	0.4	150	48	54	5	0	0	42	0
eil76	1.4	1.1	118	50	49	7	0	0	11	0
eil101	2.7	1.5	159	61	59	19	1	0	18	0
cmt101	23.9	16.0	514	114	362	14	8	6	8	1
cmt121	503.5	652.9	1350	189	719	86	82	63	172	38
cmt151	27.8	121.0	451	127	210	61	7	0	39	6
cmt200	84.8	422.6	859	177	449	82	38	0	94	18
gil262	740.0	1873.6	2370	262	1131	154	83	49	644	46

$\alpha = 0.75$										
Name	t-LP	t-sep	cuts	log	gsec	2-mat	cover	path	cond	b-cov
op21	0.8	0.6	82	22	17	2	2	0	38	0
op32	0.5	0.1	73	29	16	0	0	0	27	0
op33	0.5	0.4	70	15	16	4	0	0	34	0
att48	0.5	0.4	66	24	26	6	0	0	9	0
eil30	0.6	0.1	63	14	19	2	0	0	27	0
eil31	0.5	0.9	32	14	0	0	5	0	12	0
eil33	0.8	0.6	85	20	16	4	1	0	43	0
eil51	4.4	69.9	143	20	24	29	16	0	48	5
eil76	50.7	328.2	479	59	111	99	18	0	138	53
eil101	1.7	2.1	98	26	30	9	1	0	31	0
cmt101	36.2	86.5	676	88	355	47	14	0	169	2
cmt121	24.9	19.2	495	88	277	63	2	0	64	0
cmt151	26.8	332.5	368	69	125	39	25	0	97	12
cmt200	1934.7	13054.5	2668	120	560	1129	17	0	515	326
gil262	2020.3	12874.4	2379	165	638	789	13	1	547	225

Tabla 5.2: Resultados adicionales para problemas de la Clase I

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	opt-val	%-visited	time
spain47	3101	0.7	0.0	0.0	1	74	28	59.6	0.7
europ47	13353	21.6	0.0	3.1	3	114	32	68.1	22.7
att48	5314	0.7	0.0	0.0	1	61	31	64.6	0.7
gr48	2523	1.1	0.0	0.0	1	96	31	64.6	1.1
hk48	5731	1.7	0.0	0.0	1	105	30	62.5	1.7
eil51	213	1.2	0.0	0.0	1	114	29	56.9	1.2
brazil58	12698	3.2	0.0	0.0	1	120	46	79.3	3.2
st70	338	5.3	0.0	0.0	1	193	43	61.4	5.3
eil76	269	5.7	0.0	0.0	1	216	47	61.8	5.7
pr76	54080	50.9	0.0	0.0	1	322	49	64.5	50.9
gr96	27605	113.8	0.0	1.6	3	518	64	66.7	121.1
rat99	606	53.5	0.0	0.0	1	517	52	52.5	53.5
kroa100	10641	27.1	0.0	0.0	1	440	56	56.0	27.1
krob100	11071	156.7	1.7	0.0	3	616	58	58.0	326.9
kroc100	10375	50.7	0.0	0.0	1	518	56	56.0	50.7
krod100	10647	32.0	0.0	0.0	1	434	59	59.0	32.0
kroe100	11034	82.9	0.0	1.8	67	863	57	57.0	776.0
rd100	3955	30.2	0.0	0.0	1	425	61	61.0	30.2
eil101	315	7.1	0.0	0.0	1	216	64	63.4	7.1
lin105	7190	78.8	0.0	1.5	3	540	66	62.9	83.4
pr107	22152	86.3	0.0	0.0	1	630	54	50.5	86.3
gr120	3471	17.5	0.0	0.0	1	320	75	62.5	17.5
pr124	29515	41.2	0.0	0.0	1	480	75	60.5	41.2
bier127	59141	73.7	0.0	0.0	1	379	103	81.1	73.7
pr136	48386	214.2	0.0	0.0	1	990	71	52.2	214.2
gr137	34927	178.6	0.0	0.0	1	553	81	59.1	178.6
pr144	29269	240.3	0.0	0.0	1	1170	77	53.5	240.3
kroa150	13262	582.2	0.0	1.2	85	2594	86	57.3	4669.0
krob150	13065	145.6	0.0	0.0	1	729	87	58.0	145.6
pr152	36841	204.6	0.0	0.0	1	917	77	50.7	204.6
u159	21040	497.6	0.0	0.0	1	1912	93	58.5	497.6
rat195	1162	331.9	0.0	0.0	1	1323	102	52.3	331.9
d198	7890	716.3	0.0	0.0	1	1431	123	62.1	716.3
kroa200	14684	395.0	0.0	0.0	1	1254	117	58.5	395.0
krob200	14719	683.6	0.0	0.0	1	1635	119	59.5	683.6
gr202	20080	150.6	0.0	0.0	1	603	147	72.8	150.6
ts225	63322	9.7	0.0	0.8	107	6040	125	55.5	t.l.
pr226	40185	1955.3	0.0	5.2	25	1019	134	59.3	t.l.
gr229	1765	75.0	0.0	0.0	1	431	176	76.9	75.0
gil262	1189	120.6	0.0	0.0	1	789	158	60.3	120.6
pr264	24568	2860.2	0.0	0.0	1	1694	132	50.0	2860.2
pr299	24096	5726.3	1.2	1.2	8	4524	162	54.2	14244.0
lin318	21045	2558.0	0.0	0.5	5	2653	205	64.5	3169.9
rd400	7641	874.0	1.7	0.4	29	1821	239	59.8	4272.5

Tabla 5.3: Resultados para problemas de la Clase II (problemas de la TSPLIB) y Generación 1

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	opt-val	%-visited	time
spain47	3101	12.3	0.0	0.0	1	202	1645	53.2	12.3
europ47	13353	48.3	0.0	0.0	1	125	1865	61.7	48.3
att48	5314	3.9	0.0	0.0	1	133	1717	64.6	3.9
gr48	2523	18.0	0.0	0.0	1	196	1761	56.2	18.0
hk48	5731	7.1	0.0	0.0	1	183	1614	56.2	7.1
eil51	213	30.7	0.0	0.0	1	185	1674	52.9	30.7
brazil58	12698	7.8	0.0	0.0	1	230	2220	72.4	7.8
st70	338	147.2	0.0	0.4	7	593	2286	55.7	181.0
eil76	269	7.2	0.0	0.0	1	208	2550	53.9	7.2
pr76	54080	58.6	0.0	0.2	3	271	2708	57.9	62.0
gr96	27605	399.3	0.0	0.1	5	1185	3425	63.5	453.1
rat99	606	125.4	0.0	0.0	1	897	2944	46.5	125.4
kroa100	10641	67.8	0.0	0.0	1	558	3212	55.0	67.8
krob100	11071	259.0	0.0	0.3	7	1414	3241	49.0	481.4
kroc100	10375	207.6	0.1	0.4	7	815	2947	48.0	316.2
krod100	10647	237.2	0.0	0.3	9	757	3307	54.0	334.2
kroe100	11034	285.5	1.4	1.9	43	1518	3090	54.0	1433.9
rd100	3955	27.8	0.0	0.0	1	387	3359	57.0	27.8
eil101	315	51.5	0.0	0.4	17	711	3655	57.4	296.5
lin105	7190	151.9	0.0	0.1	3	556	3544	58.1	163.6
pr107	22152	99.4	0.0	0.0	1	678	2667	50.5	99.4
gr120	3471	303.4	0.0	0.2	21	1022	4371	57.5	650.0
pr124	29515	79.4	0.0	0.0	1	796	3917	59.7	79.4
bier127	59141	73.2	0.0	0.1	7	439	5383	77.2	245.8
pr136	48386	194.3	0.0	0.0	1	1011	4309	47.8	194.3
gr137	34927	797.9	0.0	0.3	81	1929	4294	57.7	3193.0
pr144	29269	668.0	0.0	0.7	11	1579	4003	51.4	1409.0
kroa150	13262	460.1	0.6	0.9	47	2876	4918	54.0	3950.6
krob150	13065	735.7	0.0	0.1	5	1795	4869	52.7	1018.1
pr152	36841	188.6	0.0	0.0	1	836	4279	48.0	188.6
u159	21040	518.0	0.4	0.2	21	1853	4960	54.1	1772.4
rat195	1162	1750.1	0.0	0.2	13	2691	5791	48.2	2498.6
d198	7890	1337.8	0.1	0.1	27	1999	6670	56.1	2517.1
kroa200	14684	515.2	0.0	0.1	15	1147	6547	54.5	805.1
krob200	14719	1240.7	0.1	0.1	17	2563	6419	50.5	3522.8
gr202	20080	441.7	0.8	0.0	31	1945	7848	65.8	3847.6
ts225	63322	763.3	0.0	0.1	5	1627	6834	54.2	1195.5
pr226	40185	3973.9	0.1	0.3	27	1842	6615	46.6	t.l.
gr229	1765	329.5	0.5	0.1	47	1415	9187	72.1	4261.4
gil262	1189	2783.0	0.1	0.2	23	2080	8321	50.8	5574.6
pr264	24568	4253.3	0.0	0.0	1	2211	6654	50.0	4253.3
pr299	24096	10803.8	0.0	0.0	5	1900	9161	49.8	t.l.
lin318	21045	1370.0	0.0	0.0	41	1392	10900	60.7	t.l.
rd400	7641	837.6	0.1	0.2	76	3721	13648	54.5	t.l.

Tabla 5.4: Resultados para problemas de la Clase II (problemas de la TSPLIB) y Generación 2

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	opt-val	% visited	time
spain47	3101	16.9	0.0	0.0	1	207	1443	55.3	16.9
europ47	13353	34.6	0.0	0.0	1	244	1282	59.6	34.6
att48	5314	180.2	0.0	0.8	7	532	1049	60.4	251.8
gr48	2523	27.5	0.0	0.0	1	316	1480	64.6	27.5
hk48	5731	3.5	0.0	0.0	1	101	1764	58.3	3.5
eil51	213	19.5	0.0	0.0	1	214	1399	52.9	19.5
brazil58	12698	2.8	0.0	0.0	1	112	1702	70.7	2.8
st70	338	70.6	0.0	0.0	1	623	2108	51.4	70.6
eil76	269	49.6	0.0	0.0	1	383	2467	57.9	49.6
pr76	54080	34.0	0.0	0.0	1	286	2430	61.8	34.0
gr96	27605	189.1	0.1	0.8	9	804	3182	65.6	416.6
rat99	606	481.1	0.0	0.1	3	993	2908	45.5	487.4
kroa100	10641	144.8	0.0	0.4	13	536	3211	51.0	248.8
krob100	11071	134.1	0.0	0.1	3	564	2804	49.0	138.9
kroc100	10375	228.1	0.0	0.0	1	938	3155	52.0	228.1
krod100	10647	167.8	0.0	0.3	9	559	3167	56.0	230.3
kroe100	11034	184.4	0.0	0.0	1	941	3049	45.0	184.4
rd100	3955	644.9	0.0	0.3	11	1423	2926	55.0	1032.3
eil101	315	120.0	0.0	0.1	7	474	3345	59.4	186.7
lin105	7190	325.9	1.4	1.1	19	1407	2986	55.2	1121.1
pr107	22152	632.4	0.1	3.2	1885	9926	1877	26.2	17609.0
gr120	3471	145.5	0.0	0.0	1	811	3779	57.5	145.5
pr124	29515	1377.8	6.2	5.6	77	4540	3557	57.3	11487.2
bier127	59141	139.3	0.6	0.5	73	1022	2365	68.5	2001.2
pr136	48386	861.6	0.0	0.2	5	1146	4390	51.5	958.5
gr137	34927	2401.9	45.5	0.1	5	5386	3979	51.8	4958.7
pr144	29269	1573.8	0.0	0.1	65	2632	3809	43.1	t.l.
kroa150	13262	269.7	0.1	0.6	181	1646	5039	52.7	3828.9
krob150	13065	1112.5	0.0	0.1	13	1472	5314	56.7	1363.9
pr152	36841	1099.0	0.7	1.3	391	3159	3905	48.7	13736.7
u159	21040	1308.4	0.0	0.2	5	2171	5272	52.8	1447.2
rat195	1162	3672.2	0.1	0.1	9	1528	6195	47.7	3975.4
d198	7890	1810.0	0.0	0.2	197	2361	6320	61.6	8635.7
kroa200	14684	3116.5	0.0	0.2	41	2161	6123	51.5	6548.9
krob200	14719	642.6	0.0	0.1	9	905	6266	51.0	783.7
gr202	20080	654.2	0.5	0.3	298	1947	8632	71.3	11113.5
ts225	63322	3437.6	0.0	0.4	27	1598	7575	55.1	5821.8
pr226	40185	3379.5	16.0	0.1	51	5310	6993	52.2	7923.2
gr229	1765	1667.1	0.0	0.0	11	1613	6347	67.7	1891.5
gil262	1189	6177.4	0.2	0.0	27	2386	9246	56.5	9574.0
pr264	24568	4011.3	0.0	0.0	1	2625	8137	39.8	4011.3
pr299	24096	14699.4	0.0	0.0	2	1787	10358	49.8	t.l.
lin318	21045	8597.5	0.0	0.0	12	1308	10382	60.7	t.l.
rd400	7641	14257.1	0.0	0.0	3	1418	13229	55.8	t.l.

Tabla 5.5: Resultados para problemas de la Clase II (problemas de la TSPLIB) y Generación 3

n	α	r-time		%-LB		%-UB		nodes		cuts		%-visited		time	
25	0.2	1.1	(3.4)	0.0	(0.0)	0.0	(0.0)	1.1	(2.0)	63.4	(92.0)	28.8	(40.0)	1.2	(3.4)
25	0.4	38.8	(290.4)	0.0	(0.0)	0.9	(5.9)	3.8	(15.0)	138.2	(301.0)	56.8	(64.0)	42.7	(296.6)
25	0.6	46.6	(310.2)	0.0	(0.0)	0.3	(2.3)	2.6	(7.0)	101.0	(272.0)	74.0	(80.0)	63.4	(447.8)
25	0.8	42.1	(160.0)	0.0	(0.0)	0.4	(1.5)	4.0	(17.0)	91.5	(184.0)	86.0	(92.0)	51.5	(191.6)
50	0.2	41.5	(374.9)	0.0	(0.0)	0.4	(3.2)	1.6	(5.0)	169.6	(365.0)	32.8	(40.0)	53.8	(484.4)
50	0.4	32.1	(169.4)	0.0	(0.2)	0.5	(1.1)	10.0	(27.0)	220.2	(354.0)	59.8	(66.0)	99.6	(377.8)
50	0.6	30.8	(78.4)	0.3	(1.1)	0.3	(0.5)	21.8	(88.0)	263.1	(829.0)	76.4	(80.0)	147.0	(461.3)
50	0.8	10.5	(25.5)	0.2	(1.1)	0.1	(0.6)	10.4	(67.0)	128.1	(421.0)	90.2	(94.0)	58.9	(279.6)
100	0.2	61.3	(201.5)	0.0	(0.3)	0.3	(1.1)	8.2	(31.0)	359.6	(686.0)	35.3	(39.0)	149.6	(706.2)
100	0.4	35.9	(66.4)	0.2	(0.7)	0.2	(0.4)	28.4	(81.0)	403.5	(802.0)	60.9	(66.0)	340.3	(785.0)
100	0.6	33.7	(63.5)	0.4	(1.2)	0.1	(0.2)	34.6	(77.0)	421.4	(552.0)	80.8	(84.0)	445.2	(685.1)
100	0.8	41.9	(101.3)	0.3	(1.0)	0.0	(0.1)	35.4	(237.0)	273.9	(940.0)	93.2	(95.0)	403.8	(1825.9)
150	0.2	63.5	(212.7)	0.0	(0.0)	0.1	(0.8)	3.8	(9.0)	353.5	(684.0)	33.7	(37.3)	112.6	(343.9)
150	0.4	64.6	(145.8)	0.4	(1.3)	0.1	(0.2)	26.3	(74.0)	448.4	(1336.0)	59.5	(64.0)	493.1	(1460.5)
150	0.6	54.7	(95.9)	0.2	(0.6)	0.1	(0.2)	25.3	(89.0)	335.7	(765.0)	79.2	(80.7)	525.0	(1491.1)
150	0.8	67.8	(123.8)	0.3	(0.5)	0.0	(0.1)	21.9	(127.0)	258.3	(802.0)	95.5	(97.3)	549.9	(2491.6)
200	0.2	85.9	(175.9)	0.0	(0.2)	0.1	(0.3)	9.0	(27.0)	394.9	(642.0)	32.9	(36.5)	175.4	(473.0)
200	0.4	95.8	(200.6)	0.2	(0.9)	0.1	(0.1)	39.6	(155.0)	560.2	(1229.0)	59.2	(63.0)	986.2	(3138.5)
200	0.6	115.7	(235.1)	0.5	(2.9)	0.0	(0.1)	71.0	(226.0)	612.2	(1216.0)	80.4	(83.5)	2062.2	(5142.3)
200	0.8	81.0	(158.9)	0.2	(0.6)	0.0	(0.0)	7.3	(13.0)	186.3	(341.0)	97.3	(100.0)	636.8	(1100.2)
250	0.2	139.9	(338.5)	0.0	(0.1)	0.1	(0.3)	14.4	(51.0)	460.9	(737.0)	31.3	(32.8)	308.8	(761.9)
250	0.4	113.3	(236.4)	0.2	(0.7)	0.1	(0.1)	52.7	(141.0)	583.1	(1133.0)	57.8	(60.4)	1386.4	(3504.6)
250	0.6	154.2	(485.5)	0.7	(5.9)	0.0	(0.1)	49.3	(337.0)	408.9	(1295.0)	79.7	(82.4)	1898.4	(8932.8)
250	0.8	185.2	(409.1)	1.4	(5.5)	0.0	(0.0)	22.7	(153.0)	364.4	(1924.0)	98.4	(100.0)	2850.9	(1 t.l.)
300	0.2	102.7	(177.2)	0.1	(0.4)	0.1	(0.2)	15.0	(47.0)	484.8	(750.0)	30.1	(31.0)	363.5	(946.5)
300	0.4	139.1	(208.5)	0.2	(0.7)	0.0	(0.1)	43.6	(107.0)	652.4	(1202.0)	57.5	(60.7)	1816.5	(3908.2)
300	0.6	203.9	(293.2)	0.2	(0.4)	0.0	(0.0)	26.2	(57.0)	355.3	(605.0)	80.4	(82.3)	1949.5	(3895.6)
300	0.8	237.2	(846.8)	1.5	(9.2)	0.0	(0.0)	4.3	(13.0)	186.6	(628.0)	99.8	(100.0)	2038.5	(10230.2)
350	0.2	144.4	(359.4)	0.0	(0.1)	0.1	(0.1)	8.5	(23.0)	393.9	(591.0)	29.1	(30.0)	257.2	(635.1)
350	0.4	312.1	(402.0)	0.1	(0.4)	0.0	(0.0)	49.2	(129.0)	621.9	(973.0)	56.6	(58.0)	2516.6	(4837.7)
350	0.6	238.2	(477.5)	0.2	(0.4)	0.0	(0.0)	13.2	(48.0)	256.8	(568.0)	80.1	(82.0)	1493.8	(4674.1)
350	0.8	309.1	(477.4)	0.8	(5.0)	0.0	(0.0)	12.4	(48.0)	240.4	(568.0)	89.8	(100.0)	1612.1	(4669.8)
400	0.2	153.2	(299.8)	0.0	(0.1)	0.0	(0.1)	15.4	(67.0)	395.2	(664.0)	28.2	(29.0)	439.1	(1392.5)
400	0.4	181.5	(314.4)	0.2	(0.7)	0.0	(0.0)	41.8	(214.0)	469.6	(1021.0)	55.4	(57.2)	2437.3	(9675.7)
400	0.6	294.2	(550.3)	3.0	(22.0)	0.0	(0.0)	7.3	(20.0)	216.5	(510.0)	79.7	(81.8)	1844.3	(6206.2)
400	0.8	369.4	(624.0)	1.2	(6.1)	0.0	(0.0)	1.5	(4.0)	101.4	(267.0)	100.0	(100.0)	763.9	(3883.8)
450	0.2	195.5	(290.2)	0.1	(0.2)	0.0	(0.1)	32.7	(142.0)	528.7	(961.0)	27.7	(28.2)	1008.7	(3257.9)
450	0.4	265.9	(359.6)	0.6	(4.0)	0.0	(0.0)	9.3	(35.0)	289.2	(388.0)	54.7	(55.3)	837.0	(1909.9)
450	0.6	360.2	(716.8)	1.9	(9.5)	0.0	(0.1)	10.9	(35.0)	382.2	(1188.0)	79.5	(81.6)	6050.5	(3 t.l.)
450	0.8	550.1	(970.8)	1.0	(2.7)	0.0	(0.0)	2.1	(6.0)	125.1	(337.0)	100.0	(100.0)	1954.4	(8008.6)
500	0.2	189.6	(726.8)	0.0	(0.0)	0.0	(0.0)	5.4	(23.0)	300.5	(493.0)	27.0	(27.2)	325.4	(1422.9)
500	0.4	317.4	(501.2)	0.3	(0.8)	0.0	(0.0)	9.9	(22.0)	322.9	(482.0)	53.6	(54.0)	1418.0	(3034.8)
500	0.6	408.4	(639.3)	1.1	(4.5)	0.0	(0.1)	9.7	(30.0)	284.2	(811.0)	78.8	(79.6)	5327.9	(1 t.l.)
500	0.8	650.2	(1206.3)	1.4	(3.3)	0.0	(0.0)	2.4	(8.0)	116.2	(399.0)	100.0	(100.0)	2454.0	(11860.4)

Tabla 5.6: Resultados medios (máximos) sobre 10 problemas aleatorios de la Clase III

Capítulo 6

Problema de Rutas de Vehículos con Capacidades

El *Problema de Rutas de Vehículos con Capacidades* (CVRP) que consideramos en este capítulo aparece cuando una flota de vehículos debe distribuir una mercancía a un conjunto de clientes con el mínimo costo posible. Asumimos que todos los clientes tienen una demanda específica, y que ésta debe ser satisfecha por un único vehículo de la flota. Asumimos también que todos los vehículos tienen una misma capacidad máxima, y que están en un único depósito. En nuestra versión no se exige que se utilicen todos los vehículos de la flota, y los costos aparecen sólo como consecuencia del movimiento de los vehículos por la red.

Nosotros modelizamos el CVRP en una formulación matemática con más variables que los modelos clásicos, pero con la ventaja de permitir separar las soluciones del CVRP en soluciones de problemas de ciclos. Siguiendo tal consideración consideramos nuevas desigualdades que añadidas al modelo conducen a relajaciones lineales mejores que las conocidas, al menos sobre varios problemas de la literatura. Estas experiencias motivan el desarrollo de un algoritmo de ramificación y corte en base a esta formulación matemática.

6.1 Introducción

Los Problemas de Rutas de Vehículos afrontan la utilización óptima de una flota de vehículos para transportar (entregando y/o recogiendo) productos entre depósitos centrales y clientes. Se pueden encontrar interesantes aplicaciones en la secuenciación de los autobuses escolares, la recolección del correo depositado en los buzones, servicios de lavandería, recogidas de basura, ... Dada el enorme número de aplicación prácticas en

el que comparece (cada uno caracterizado por una función objetivo y restricciones específicas), en la literatura se han tratado diversas versiones. Como introducción al tema, véanse por ejemplo Christofides, Mingozzi y Toth [CMT79], Christofides [C85], Laporte y Nobert [LN87], Golden y Assad [GA88], y Laporte [L92].

Existen muchos problemas combinatorios bajo el nombre común de *problema de rutas de vehículos*. Aparecen variantes distintas atendiendo a criterios como, por ejemplo:

1. tamaño de la flota
   (a) un vehículo,
   (b) varios vehículos;
2. tipo de los vehículos;
   (a) todos iguales,
   (b) todos distintos,
   (c) mixto;
3. capacidad de los vehículos
   (a) capacidad ilimitada,
   (b) capacidad limitada;
4. costo máximo de recorrido de los vehículos
   (a) ilimitado,
   (b) limitado;
5. depósitos
   (a) uno,
   (b) varios;
6. uso de vehículos
   (a) se deben usar todos los vehículos disponibles,
   (b) se pueden usar menos de los vehículos disponibles;
7. demanda de los clientes
   (a) todos iguales,
   (b) todos distinta,
   (c) estocástica;

8. servicio a los clientes
   (a) servibles exactamente por un vehículo,
   (b) servibles por al menos un vehículo,
   (c) se pueden servir o no;
9. operaciones en los clientes
   (a) sólo descarga,
   (b) recogida y entrega;
10. costos de transporte
   (a) simétricos,
   (b) asimétricos,
   (c) euclídeos,
   (d) variables según el momento;
11. objetivo
   (a) minimizar el coste total de la ruta,
   (b) minimizar el número de vehículos requerido,
   (c) minimizar una penalización por clientes no servidos.

En este capítulo nosotros consideramos la versión particular que a continuación se describe, si bien mucho de lo que se expone es también extensible a otras variantes más generales. Sea una red de carreteras conectando entre sí un conjunto de clientes $N = \{1, \ldots, n\}$ y un depósito 0 (*puntos*). Consideremos inicialmente en el depósito un conjunto de vehículos idénticos $M = \{1, \ldots, m\}$. Sea c_{ij} es costo de ir directamente desde un punto i a un punto j en la red (es decir, el costo del arco ij), q_i la demanda de un cliente i, y Q la capacidad de un vehículo. Una *ruta factible* se define como un subconjunto de arcos representando un ciclo simple que visita al depósito y a un conjunto de clientes con demanda total no mayor que Q. Entonces, el *Problema de Rutas de Vehículos con Capacidades* (CVRP) es el problema de encontrar una familia de no más de m rutas factibles tales que cada cliente esté en exactamente una ruta factible y el costo total de los arcos considerados resulte lo menor posible.

Existen dos versiones de este problema: el CVRP *simétrico* si $c_{ij} = c_{ji}$ para cada par de clientes i, j; y el CVRP *asimétrico* en otro caso. Es bien-conocido que ambas versiones son problemas $\mathcal{NP}$-difíciles (en el sentido fuerte) ya que coinciden con el problema del Viajante de Comercio (TSP) cuando $m = 1$, y han sido extensamente estudiadas. Para el caso asimétrico, Fischetti, Toth y Vigo [FTV94] proponen un algoritmo de ramificación

y acotación basado en su llamada *Aproximación Aditiva*, llegando a demostrar la optimalidad de problemas aleatoriamente generados con hasta 300 clientes. Para el caso simétrico, hasta nuestro conocimiento, los mejores resultados hasta el momento aparecen en el trabajo de Fisher [F94], donde se logra la prueba de optimalidad para un problema test con hasta 100 clientes mediante el uso de un algoritmo de ramificación y acotación basado en relajación Lagrangiana. Se han estudiado otras técnicas basadas en la aproximación de hiperplanos de corte en los trabajos de Laporte, Nobert y Desrochers [LND85] y Cornuejols y Harche [CH93]. Araque, Kudva, Morin y Pekny [AKMP94] han propuesto recientemente un algoritmo de ramificación y corte para el CVRP en el caso particular de clientes con idénticas demandas, logrando la prueba de optimalidad para ejemplos con hasta 60 clientes. Todas estas aproximaciones se basan en modelos matemáticos con hasta $O(n^2)$ variables de doble-índice.

En este capítulo, mostramos nuestra experiencia en el tratamiento del CVRP simétrico con un *algoritmo de hiperplanos de corte* basado en una formulación de flujos de vehículos con tres índices (es decir, variables con tres índices), que estamos embebiendo en un *algoritmo de ramificación y acotación*. Por una parte, este modelo presenta más variables que los modelos anteriores($O(n^2m)$ en lugar de $O(n^2)$), y ello puede parecer una desventaja del modelo. Pero por otra parte tiene la singular ventaja de permitirnos descomponer una solución (fraccionaria) del CVRP en soluciones (fraccionarias) de particulares problemas de ciclo, y por tanto considerar desigualdades adicionales (nuevas) para fortalecer la relajación lineal del modelo anterior. Experiencias computacionales preliminares con ejemplos del trabajo de Araque, Kudva, Morin y Pekny [AKMP94], nos han adelantado que, en algunos caso, se alcanza ahora la solución óptima (entera) durante la resolución del nodo raíz.

6.2 Modelo Matemático

Representaremos por $V := \{0\} \cup N$ al conjunto de puntos de la red de carreteras dada, y por E al conjunto de sus carreteras. Diremos que $G = (V, E)$ es el *grafo* del CVRP, donde V es el *conjunto de nodos*, y E el *conjunto de arcos*. Cada nodo tiene un peso asociado q_i (la demanda si i representa un cliente, y 0 si i representa el depósito), y cada arco e tiene un costo asociado c_e (el costo c_{ij} si e representa la carretera que une los clientes i y j). Para cada $S \subseteq V$, consideramos la notación:

$$\begin{aligned} E(S) &:= \{e \in E : e \text{ tiene ambos nodos incidentes en } S\}, \\ \delta(S) &:= \{e \in E : e \text{ tiene exactamente un nodo incidente en } S\}. \end{aligned}$$

Dada una cualquier función f de variable real definida sobre un dominio finito D, y dado cualquier subconjunto $W \subseteq D$, usaremos la notación $f(W)$ en lugar de $\sum_{d \in W} f_d$.

Para cada arco $e \in E$ y cada $k \in M$, consideramos una variable de decisión x_e^k que asume el valor 2 si el arco e une el depósito con el único cliente servido por el kj-ésimo

vehículo, 1 si e es usado sólo una vez por el k-ésimo vehículo, y 0 en otro caso. Para cada nodo $i \in V$ y cada $k \in M$, consideremos una variable decisional y_i^k que asume el valor 1 si el vértice i es visitado por el k-ésimo vehículo, y 0 en otro caso.

Entonces el CVRP puede formularse matemáticamente como sigue:

$$\min \sum_{e \in E} c_e \sum_{k \in M} x_e^k$$

sujeto a

$$x^k(\delta(\{i\})) = 2y_i^k \qquad \text{para todo } k \in M \text{ y } i \in V; \tag{6.1}$$

$$x^k(\delta(S)) \geq 2y_i^k \qquad \text{para todo } k \in M \text{ y } S \subset V, i \in S, 0 \notin S; \tag{6.2}$$

$$\sum_{i \in V} q_i y_i^k \leq Q \qquad \text{para todo } k \in M; \tag{6.3}$$

$$\sum_{k \in M} y_i^k = 1 \qquad \text{para todo } i \in V \setminus \{0\}; \tag{6.4}$$

$$x_e^k, y_i^k \in \{0,1\} \qquad \text{para todo } k \in M, e \in E \setminus \delta(\{0\}), i \in V; \tag{6.5}$$

$$x_e^k \in \{0,1,2\} \qquad \text{para todo } k \in M, e \in \delta(\{0\}). \tag{6.6}$$

Las ecuaciones (6.1) se llaman *restricciones de grado*, y establecen que cada vértice visitado por un vehículo debe tener dos de sus arcos incidentes en la correspondiente ruta factible. Las desigualdades (6.2) se llaman *restricciones de eliminación de subtour* (GSEC's, para abreviar), e imponen las condiciones de conexidad y el uso del depósito por toda ruta factible. (6.3) recibe el nombre de *restricción mochila*, y limita la máxima carga de los vehículos. Finalmente, las igualdades (6.4) aseguran que cada cliente está visitado por exactamente un vehículo. Por supuesto, usando las igualdades (6.1) sería posible eliminar las variables y_i^k y las restricciones de grado del modelo.

6.3 Fortaleciendo la Relajación Lineal

Con la intensión de disponen de una buena relajación lineal del modelo anterior, se precisan nuevas restricciones adicionales. Ante todo observemos que todas las desigualdades válidas propuestas en las aproximaciones de hiperplanos de cortes previamente propuestas in la literatura son también válidas aquí, tal y como sucede con las conocidas *restricciones de capacidad*:

$$\sum_{k \in M} x^k(\delta(S)) \geq 2 \; bpp(S), \tag{6.7}$$

donde $\emptyset \neq S \subseteq V \setminus \{0\}$ y $bpp(S)$ es el mínimo número de vehículos que se precisan para servir los clientes del subconjunto S; o las *desigualdades peine*:

$$\sum_{l \in \{0,1,\ldots,s\}} \sum_{k \in M} x^k(\delta(S_l)) \geq 3s + 1, \tag{6.8}$$

con $S_0, S_1, \ldots, S_s \subseteq V \setminus \{0\}$ verificando:

(i) $|S_l \setminus S_0| \geq 1$, $l = 1, \ldots, s$,

(ii) $|S_l \cap S_0| \geq 1$, $l = 1, \ldots, s$,

(iii) $|S_l \cap S_t| = 0$, $1 \leq l < t \leq s$,

(iv) s impar y mayor o igual que 3.

(Véase Laporte y Nobert [LN84] para más detalles, y Cornuejols y Harche [CH93] para otras desigualdades de este tipo.)

Pero utilizando la descomposición que nuestro modelo (y más concretamente las variables de tres índices) nos produce de una solución del CVRP en soluciones (x^k, y^k) $(k = 1, \ldots, m)$ de problemas de ciclos, podemos considerar las siguientes desigualdades:

$$x^k(\delta(S)) \geq 2\ f(S, x^k, y^k), \tag{6.9}$$

donde $k \in M$, $\emptyset \neq S \subseteq V \setminus \{0\}$ y f es una función que asume el valor 0 cuando (x^k, y^k) se corresponde con una ruta factible que no visite clientes en S, y un valor en $(0, 1]$ en otro caso. Es fácil ver entonces que las restricciones (6.2) son un caso particular de (6.9) que aparece cuando

$$f(S, x^k, y^k) = y_i^k.$$

Si consideramos

$$f(S, x^k, y^k) = \frac{\sum_{i \in S} q_i y_i^k}{Q},$$

obtenemos otra desigualdad válida. Esta desigualdad puede fortalecerse incluso cuando $q(S) > Q$. En efecto, observemos que si (x^k, y^k) es una ruta factible que verifica con igualdad la restricción (solución ajustada para la desigualdad), entonces o bien $\sum_{i \in S} q_i y_i^k = 0$, o bien $\sum_{i \in S} q_i y_i^k = Q$. En ambos casos se tiene que $x_e^k = 0$ para todo $e \in \delta(S) \setminus \delta(0)$, por lo que dicho punto también verifica con igualdad la desigualdad no válida $x^k(\delta(S) \setminus \delta(0)) \leq 0$. Esto da también una idea de como obtener una nueva desigualdad que domine estrictamente a la anterior: basta encontrar el máximo valor ϵ^* de ϵ en $[0, +\infty)$ que mantenga factible la desigualdad combinada

$$x^k(\delta(S)) - \epsilon^*\ x^k(\delta(S) \setminus \delta(0)) \geq 2\frac{\sum_{i \in S} q_i y_i^k}{Q}.$$

Ahora bien, tal ϵ^* es difícil de calcular. La cuestión se simplifica si notamos que análogamente el punto verifica con igualdad todas las desigualdades no válidas $x^k(\delta(S) \cap \delta(j)) \leq 0$ para $j \notin S$ y $j \neq 0$. De este modo, el razonamiento anterior nos lleva a buscar $\epsilon_j^* \in [0, +\infty)$ grande tales que

$$x^k(\delta(S)) - \sum_{j \notin S, j \neq 0} \epsilon_j^*\ x^k(\delta(S) \cap \delta(j)) \geq 2\frac{\sum_{i \in S} q_i y_i^k}{Q},$$

se mantenga válida. Considerando $\epsilon_j^* = q_j/Q$ se tiene la desigualdad de tipo (6.9) con

$$f(S, x^k, y^k) = \frac{\sum_{i\in S} q_i y_i^k + \sum_{j\notin S} q_j \sum_{i\in S} x_{ij}^k}{Q}.$$

También esta nueva desigualdad es separable en tiempo polinomial mediante la resolución de un problema de flujos en redes sobre una particular red (de modo análogo a como veremos para la próxima familia de desigualdades).

Otra desigualdad de tipo (6.9) resulta de considerar

$$f(S, x^k, y^k) = \frac{\sum_{i\in S} q_i (y_i^k - x_{[0,i]}^k)}{Q - q_{(1)} - q_{(2)}}, \tag{6.10}$$

donde $q_{(1)}$ y $q_{(2)}$ son el primer y segundo mínimo en $\{q_1, \ldots, q_n\}$. Para concluir la validez de esta desigualdad obsérvese que $Q - q_{(1)} - q_{(2)}$ es la capacidad máxima de cualquier vehículo para los clientes que no serán servidos directamente desde el depósito (es decir, para clientes intermedios en su ruta), mientras que $y_i^k - x_{[0,i]}^k$ asume el valor 1 si i es un cliente intermedio en la ruta de k y 0 en otro caso, por lo que $\sum_{i\in S} q_i (y_i^k - x_{[0,i]}^k)$ es la capacidad real necesaria por el vehículo k-ésimo para sus clientes intermedios. Además, cuando las demandas de los clientes son todas idénticas, entonces esta particular familia de restricciones domina las *restricciones multi-estrella grandes* propuestas por Araque [A89] y Hall [G89]. Finalmente notemos que esta nueva familia de desigualdades puede separarse en tiempo polinomial mediante la resolución de un problema de flujo máximo en un grafo con $n+2$ nodos. En efecto, para verlo reescribamos la desigualdad propuesta como

$$x^k(\delta(S)) \geq 2\frac{\sum_{i\in N} q_i}{Q - q_{(1)} - q_{(2)}} + 2\frac{\sum_{i\in S} q_i (y_i^k - x_{[0,i]}^k - 1)}{Q - q_{(1)} - q_{(2)}} - 2\frac{\sum_{j\notin S} q_j}{Q - q_{(1)} - q_{(2)}},$$

o mejor como

$$x^k(\delta(S) \setminus \delta(0)) + \sum_{i\in S} \left(\frac{2q_i(1 - y_i^k + x_{[0,i]}^k)}{Q - q_{(1)} - q_{(2)}} + x_{[0,i]}^k \right) + \frac{\sum_{j\notin S} 2q_j}{Q - q_{(1)} - q_{(2)}} \geq \frac{\sum_{i\in N} 2q_i}{Q - q_{(1)} - q_{(2)}}.$$

De este modo, si consideramos un grafo completo sobre el conjunto de nodos $\{0, 1, \ldots, n, n+1\}$, y le asociamos el peso

1. $x_{[j,i]}^k$ a los arcos $[j, i]$ con $j \neq 0$ e $i \neq n+1$,
2. $\frac{2q_i(1-y_i^k+x_{[0,i]}^k)}{Q-q_{(1)}-q_{(2)}} + x_{[0,i]}^k$ a los arcos $[0, i]$ con $j \neq n+1$,
3. $\frac{\sum_{j\notin S} 2q_j}{Q-q_{(1)}-q_{(2)}}$ a los arcos $[j, n+1]$ con $j \neq 0$,

4. 0 al arco $[0, n+1]$,

entonces existe una desigualdad violada por la solución actual si y sólo si el flujo máximo entre 0 y $n+1$ es inferior a $\frac{\sum_{i \in N} 2q_i}{Q - q_{(1)} - q_{(2)}}$ (y en tal caso, está definido por el conjunto S que se corresponde con el lado del corte mínimo que contiene a 0).

Adicionalmente, cuando $q(S) > (m-1)Q$ podemos considerar la desigualdad (6.9) con $f(S, x^k, y^k) = 1$. Estas desigualdades pueden encontrarse heurísticamente, y resultan de gran utilidad para los ejemplos donde $q(N)$ es muy próximo a mQ (tal y como sucede frecuentemente en los problemas test de la literatura). Por ejemplo, si $q(N \setminus \{i\}) > (m-1)Q$, entonces el cliente i no puede ser servido aisladamente por ningún vehículo, y la desigualdad previa se convierte en $x^k_{[0,i]} \leq y^k_i$.

Utilizando una vez más la descomposición de la solución de CVRP en soluciones de problemas de ciclos, es también posible considerar nuevas desigualdades. Sea P cualquier camino simple visitando $V(P) := \{i_1, i_2, \ldots, i_{h-1}, i_h\} \subseteq V \setminus \{0\}$. Definamos

$$W(P) := \{i_{h+1} \in V \setminus V(P) : q(\{i_1, \ldots, i_h, i_{h+1}\}) \leq Q\}.$$

Entonces la siguiente *desigualdad camino* es válida para cada $k \in M$:

$$x^k(P) \leq y^k(V(P) \setminus \{i_1, i_h\}) + x^k(\delta(\{i_h\}) \cap \delta(W(P))). \tag{6.11}$$

En efecto, si por reducción al absurdo hubiese una solución del CVRP que no satisface dicha restricción, entonces

$$\sum_{l=1}^{h-1} x^k_{[i_l, i_{l+1}]} \geq 1 + \sum_{l=2}^{l-1} y^k_{i_l} + \sum_{j \in W(P)} x^k_{[i_h, j]},$$

o equivalentemente

$$x^k_{[i_1, i_2]} + (x^k_{[i_2, i_3]} - y^k_{i_2}) + \ldots + (x^k_{[i_{h-1}, i_h]} - y^k_{i_{h-1}}) - \sum_{j \in W(P)} x^k_{[i_h, j]} \geq 1,$$

y consecuentemente $x^k_{[i_1, i_2]} = \ldots = x^k_{[i_{h-1}, i_h]} = 1$ y $x^k_{[i_h, j]} = 0$ para todo $j \in W(P)$, lo que contradice que (x^k, y^k) sea una ruta factible que contiene a P y a un arco $[i_h, j]$ con j no en $W(P)$. En cuanto a la separación de estas desigualdades, resulta clave observar que cuando se verifica que $x^k_{[i,j]} \leq y^k_i$ para todos los pares de clientes i, j, entonces no hay desigualdades violadas de tipo 6.11 asociadas con caminos P no completamente contenidos en el grafo soporte de la solución fraccionaria (esto es, con $x^k_e = 0$ para algún $e \in P$). De este modo, y dado que los grafos soportes no suelen ser extremadamente densos, una simple enumeración de sus caminos P suele ser bastante eficiente.

De la restricción mochila (6.3) también se pueden considerar otras familias de desigualdades, como por ejemplo las conocidas *restricciones de cubrimiento*:

$$y^k(S \cup S') \leq |S| - 1, \tag{6.12}$$

donde $S \subseteq V$ es un conjunto de vértices minimal tal que $q(S) > Q$, y $S' := \{i \in V \setminus S : q_i \geq \max_{j \in S} q_j\}$. Cuando las demandas son idénticas, las desigualdades (6.12) están dominadas por (6.3).

Para evitar soluciones equivalentes obtenidas intercambiando el tercer índice, resulta muy útil introducir una renumeración conveniente de los clientes, de manera que los vehículos con menor índice visiten clientes con menor índice. Con más precisión, para cualquier permutación dada σ del conjunto de clientes N, podemos asumir que

$$i(1) \leq i(2) \leq \ldots \leq i(m),$$

donde $i(k) := \min\{i \in N : y^k_{\sigma(i)} = 1\}$ es el menor número de cliente (con respecto a la permutación σ) de los clientes visitados por el vehículo k ($i(k) := \infty$ si no se utiliza el vehículo k). Entonces se verifica el siguiente sistema de restricciones:

$$y^1_{\sigma(1)} = 1, \tag{6.13}$$

$$y^k_{\sigma(i)} \leq \sum_{j \in \{1,\ldots,i-1\}} y^{k-1}_{\sigma(j)} \quad \text{para todo } k \geq 3\,(k \in M) \text{ y } i \geq 2\,(i \in N). \tag{6.14}$$

Al inicio se calcula el menor número de vehículos para servir a todos los clientes $\underline{m} := bpp(N)$, y se compara con m para determinar la factibilidad del CVRP. Luego se fija a 1 todas las y^k_0 con $k = 1, \ldots, \underline{m}$.

6.4 Resultados computacionales

En el marco de un algoritmo de hiperplanos de cortes, hemos resuelto la relajación lineal del modelo entero presentado en la Sección 6.2, fortalecido con las consideraciones detalladas en la Sección 6.3. La Tabla 6.1 muestra nuestros resultados preliminares a mayo de 1995 sobre ejemplos 1, 13, 15, 16, 17, 18, 19 y 20 de Araque, Kudva, Morin y Pekny [AKMP94], y algunas variantes de los ejemplos EIL23.VRP y EIL51.VRP de la librería TSPLIB de Reinelt [R91], todos ellos con demanda idéntica 1 para todos los clientes. El significado de las columnas es:

Name : AKMP# significa que es el ejemplo # de [AKMP94]; EIL#.VRP significa que se trata del correspondiente ejemplo de la TSPLIB;

n : número de clientes;

m : número de vehículos disponible;

Q : capacidad máxima de cada vehículo;

z_{opt} : valor objetivo óptimo del modelo entero;

Name	n	m	Q	z_{opt}	z_{LP} nuestro	z_{LP} en [AKMP94]
AKMP15	21	7	3	530	530	530
AKMP16	21	3	7	341	341	341
AKMP17	29	8	4	832	832	830.80
AKMP18	29	5	6	639	639	639
AKMP19	32	7	5	627	626.4626	625.27
AKMP20	32	4	8	497	497	497
EIL23.VRP	22	2	11	527	527	???
EIL23.VRP	22	5	5	719	719	???
EIL51.VRP	50	5	10	???	515.9041	???
EIL51.VRP	50	10	5	???	707.1405	???
AKMP1	40	4	10	647	639.6419	638.00
AKMP13	60	2	30	695	688.7019	684.25

Tabla 6.1: Comparando la cota inferior de nuestro modelo con la de Araque, Kudva, Morin y Pekny [AKMP94].

z_{LP} **nuestro** : valor objetivo óptimo de nuestra relajación lineal;

z_{LP} **en [AKMP94]** : valor objetivo óptimo de la relajación lineal considerada en Araque, Kudva, Morin y Pekny [AKMP94].

No se dispone de la solución óptima de todos los problemas considerados, y cuando esto ocurre se indica en la tabla con el símbolo "???". No hemos incorporado una columna con el tiempo precisado para la resolución de las correspondientes relajaciones lineales porque nuestro objetivo no era tanto dar un eficiente algoritmo, como demostrar que con el modelo a tres índices se podían mejorar las cotas propuestas por otros autores recientes. Conviene observar que en esta sección sólo hemos considerado una simple técnica de hiperplanos de corte, no beneficiada por heurísticos que fijen o eliminen variables, ni otras herramientas propias de procedimientos de ramificación y corte. En cualquier caso, nuestros tiempos son perfectamente comparables con los de otros autores sobre los ejemplos considerados. Claramente se precisa más trabajo sobre esta nueva aproximación a la resolución del CVRP, tanto estudiándola sobre ejemplos mayores, como sobre la mejora de nuestro código de resolución (los problemas lineales son particularmente difíciles, no tanto por su dimensión como por su estructura combinatoria del tipo *multi-flujos*). Además, pretendemos incorporar esto en un procedimiento de ramificación y corte para la resolución exacta de CVRP.

Capítulo 7

Problema de Garantizar la Privacidad en Tablas Estadísticas Públicas

En este capítulo se estudia el problema de proteger datos particulares (*datos sensibles*) en tablas estadísticas, cuando los restantes datos deben hacerse públicos junto con los totales marginales. Para ello, se permite la supresión adicional de datos no sensibles, si bien al tiempo se pretende que la información no desvelada resulte lo menor posible. Aquí se presenta un algoritmo exacto de tipo Ramificación y Corte (*Branch-and-Cut*) para el caso de tablas bi-dimensionales (que puede usarse también como algoritmo heurístico). Se analizan resultados computacionales sobre problemas de aplicaciones reales y aleatoriamente generados, de los que se concluye que tal algoritmo es claramente mejor que los procedimientos propuestos en los trabajos precedentes. Finalmente se extienden las ideas usadas al caso de tablas estadísticas multi-dimensionales.

7.1 Introducción

La materia prima de los censos e informes es la información obtenida como respuestas individuales. Normalmente, estos datos se obtienen bajo un compromiso de *confidencialidad*: el organismo recolector/difusor se compromete a no publicar cualquier dato o resumen de datos en los cuales la información de respuestas individuales pueda ser revelada (*datos sensibles*). En este capítulo se estudia el problema de proteger datos sensibles en una tabla estadística, cuando se pretenden publicar los datos no sensibles junto con los totales marginales. Cada celda contiene inicialmente un valor dado (al que llamare-

Edad	A	B	C	Total
bajo 45	7	10	8	25
45–64	12	9	5	26
65–74	7	3	3	13
sobre 74	5	0	**2**	7
Total	31	22	18	71

Tabla 7.1: Millonarios por edades y lugar de residencia.

mos *valor nominal*), y conlleva una pérdida de información en caso de resultar suprimida (valor denominado *peso de supresión* de la celda). En algunos casos prácticos el peso de supresión viene dado coincidiendo con el valor nominal, en otros como un valor constante, y en otros es más complejo. Una celda se considera *protegida* si el máximo (resp. mínimo) posible valor que puede asumir en la tabla estadística es mayor (resp. menor) que el valor nominal original más (resp. menos) una nivel de protección superior (resp. inferior). Si existe una celda no protegida en la tabla, entonces aparece el problema de elegir celdas adicionales (*celdas secundarias*) que serán también suprimidas, pero de manera que se minimice la pérdida global de *información* en la tabla final propuesta. Cuando el peso de supresión viene dado como una constante, –es decir, igual para todas las celdas de la tabla–, entonces el objetivo del problema es minimizar el número de celdas secundarias.

Las faltas de protección en celdas sensibles surgen por las relaciones creadas con los marginales totales, y por el conocimiento de que los datos ocultos son cuanto menos no negativos. En realidad, motivados por las aplicaciones prácticas reales de este problema, asumimos *conocido* que un organismo no está interesado en ocultar nunca celdas con valores nominales cero, y por tanto, se supone globalmente asumido que el menor valor posible para una celda suprimida es 1 (y no 0).

Por ejemplo, la Tabla 7.1 muestra una tabla bi-dimensional correspondiente al número de millonarios de una ciudad, clasificados según edad y lugar de residencia. Asumiendo que en la pequeña zona C todos los habitantes conocen que sólo hay dos residentes con más de 74 años de edad, entonces cualquier habitante podría concluir que éstas dos personas son millonarias. Para proteger la tabla frente a tal revelación estadística de datos individuales, la Celda (4,3) viene considerada como una celda sensible, y su valor nominal 2 se sustituye por el símbolo *. Sin embargo, esto no es suficiente: apoyándose en los marginales totales, sería posible calcular el valor 2 de la Celda (4,3). Por tanto, se precisa suprimir algún otro dato. Si los nivel de protección inferior y superior para la Celda (4,3) son 1 y 3, respectivamente, entonces una solución óptima para el problema será suprimir las celdas (1,1), (1,3), (4,1) y (4,3) en la tabla final. (En este ejemplo estamos asumiendo que el peso de supresión de una celda coincide con su valor nominal.)

Este problema se conoce con el nombre de *Problema de la Supresión de Celdas* (abreviadamente CSP por venir de *Cell Suppression Problem*), y las primeras referencias bibliográficas que lo abordan específicamente se deben a Lawrence H. Cox en las memorias de

los congresos de la Asociación Estadística Americana, durante los últimos años de los 70. El CSP aparece entonces como parte del *Control de Revelaciones Estadísticas* (*Statistical Disclosure Control*), está motivado por el deseo de preservar la privacidad y seguridad en la publicación de datos no sensibles, y es por ello un problema de considerable importancia para cualquier Instituto Nacional de Estadística (y así lo han manifestado a través de trabajos publicados los correspondientes institutos de Estados Unidos, Canadá, Holanda, ...). Hasta la actualidad, todos los intentos para abordar el problema han sido heurísticas o muy sencillas técnicas de ramificación y acotación basadas en enormes modelos matemáticos de programación entera mixta (ver [CCE85], [C87], [G88], [WW94]). Para una introducción más detallada sobre el tema y su relevancia dentro del Control de Revelaciones Estadísticas, se aconseja el trabajo de Geurts [G92] (desarrollado en el *Netherlands Central Bureau of Statistics*) o el trabajo de Cox [C92] (en el *U.S. Bureau of the Census*). Este segundo es básicamente una argumentación práctica, mientras que el primero contiene además los mejores resultados computacionales del momento: mediante un nuevo algoritmo de ramificación y acotación, resuelven exactamente el CSP sobre tablas de dimensión desde 4×4 hasta 14×5 en tiempos próximos a 1 hora de CPU sobre un potente ordenador "SUN SPARC station +1"; además, los tiempos del heurístico que proponen precisan tiempos no pequeños con errores de hasta el 45.5 % sobre la solución óptima.

Este capítulo presenta una alternativa más simple y eficiente de tratar el CSP. En la Sección 7.2 se demuestra que el CSP es un problema de Optimización Combinatoria de tipo $\mathcal{NP}$-difícil, en base a un análisis (teórico) del caso particular de tablas unidimensional (de poco interés práctico). El caso bi-dimensional se aborda con detalle en la Sección 7.3, introduciendo en la Sección 7.3.1 un nuevo modelo matemático de Programación Lineal Entera, ideal para ser tratado por el algoritmo de *Ramificación y Corte* (*Branch-and-Cut*) que se propone en la Sección 7.3.4. En la Sección 7.3.2 se dan diversas desigualdades para fortalecer la relajación lineal del modelo entero, y en la Sección 7.3.3 se describe un algoritmo heurístico que usa la información de soluciones de relajaciones lineales. La Sección 7.3.5 analiza resultados computacionales del algoritmo propuesto sobre problemas reales y generados aleatoriamente, mostrando que ahora se resuelven problemas más difíciles que los considerados por Geurts [G92], con muchísimo menos esfuerzo (segundos sobre un PC 486 frente a minutos sobre una SUN SPARC). Finalmente, la Sección 7.4 extiende algunas de las anteriores ideas al CSP en tablas multi-dimensionales basándonos en una interesante extensión de la conocida *Técnica de Descomposición de Benders*.

Aunque hemos tratado de que este capítulo pueda ser también leído por un estadístico no especializado en técnicas de *Programación Matemática*, en algunos casos se utilizan conceptos básicos de la *Teoría Poliédrica* y de los *Algoritmos de Ramificación y Corte* que no se introducen. Indicamos el libro de Nemhauser y Wolsey [NW88] y el artículo de Padberg y Rinaldi [PR91] como referencia para los mismos.

7.2 Complejidad Algorítmica

En esta sección demostraremos que el CSP para tablas k-dimensionales es $\mathcal{NP}$-difícil. Para ello veremos primero que está en la clase $\mathcal{NP}$, es decir, que se puede demostrar en tiempo polinomial que una solución dada es factible y que su peso total de supresión es menor que un valor dado, cuando efectivamente lo sea. En efecto, dada una propuesta de solución, la factibilidad se determina resolviendo un problema lineal de "máximo" y un problema lineal de "mínimo" para cada celda sensible, donde las variables son los posibles valores de las celdas que la solución propone ocultar, la función objetivo es el valor de la propia variable sensible, y las restricciones son que los valores de tales variables junto con los valores constantes de las celdas no suprimidas, deben de mantener congruentes los totales marginales, así como las cotas sobre sus valores mínimos posibles de dichas variables (véanse las secciones 7.3 y 7.4 para más detalle). Dado que los problemas de Programación Lineal se resuelven en tiempo polinomial, el CSP está en la clase $\mathcal{NP}$.

Para demostrar que es además $\mathcal{NP}$-duro observaremos que el clásico *problema de la mochila* (abreviadamente KP por *Knapsack Problem*) es reducible al CSP con una única supresión. Dado que cualquier tabla uni-dimensional puede considerarse como una particular tabla k-dimensional, para cualquier natural k, entonces el CSP es también un problema $\mathcal{NP}$-difícil.

Consideremos el siguiente KP:

$$\max\{\sum_{j=1}^{n} p_j\, z_j \;:\; \sum_{j=1}^{n} w_j\, z_j \leq c \;\text{ y }\; z_j \in \{0,1\}\,(j=1,\ldots,n)\},$$

donde p_j y w_j son, respectivamente, el beneficio y el peso asociados al objeto j, y c es la capacidad de la mochila (todos número enteros positivos). Con el cambio de notación $x_j := 1 - z_j$ y $c' := \sum_{j=1}^{n} w_j - c$, el problema anterior equivale a resolver

$$\min\{\sum_{j=1}^{n} p_j\, x_j \;:\; \sum_{j=1}^{n} w_j\, x_j \geq c' \;\text{ y }\; x_j \in \{0,1\}\,(j=1,\ldots,n)\}.$$

Creamos la siguiente tabla uni-dimensional con n+2 celdas. A cada celda j $(j = 1,\ldots,n)$ le asociamos un valor nominal $w_j + 1$ y un peso de supresión p_j. La celda $(n+1)$-ésima viene considerada como celda sensible, y le asociamos valor nominal 1, peso de supresión 0, nivel de protección inferior 0, y nivel de protección superior c'. En la celda $n+2$ colocamos el valor nominal total de los anteriores, es decir, $\sum_{j=1}^{n} w_j + n + 1$, con un muy alto peso de supresión. Entonces el CSP sobre esta tabla plantea la búsqueda de un conjunto de celdas $S \subseteq \{1,\ldots,n\}$ (si existe) de manera que suprimiendo tales celdas, el mayor valor posible para la celda $(n+1)$-ésima, es decir $1 + \sum_{j\in S} w_j$, no sea inferior a $1 + c'$, y que su peso de supresión total $\sum_{j\in S} p_j$ resulte lo menor posible. Es decir, busca una solución óptima para el KP. Dado que éste es un problema $\mathcal{NP}$-difícil, el CSP uni-dimensional lo es también, y por tanto el CSP.

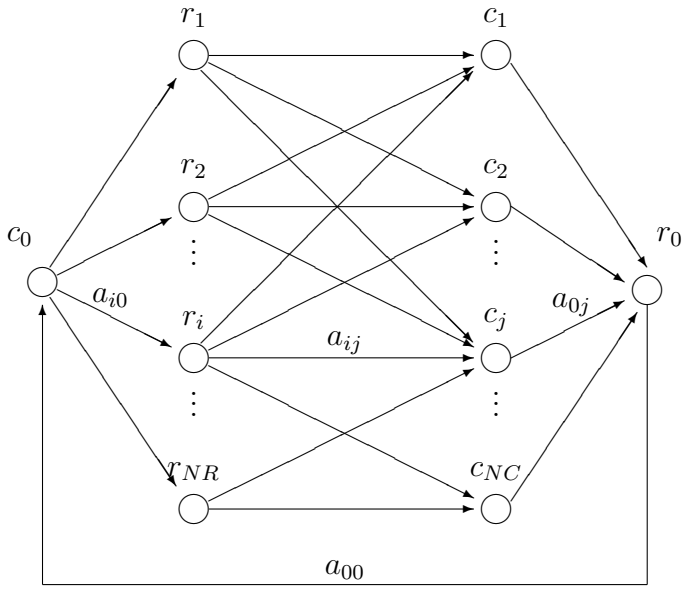


Figura 7.1: Red representando una tabla bi-dimensional

7.3 Tablas bi-dimensionales

Una tabla estadística bi-dimensional puede considerarse como una matriz $[a_{ij} : i = 0, 1, \ldots, NR; j = 0, 1, \ldots, NC]$ (donde el índice 0 se corresponde con los *totales marginales*) que satisface:

$$\begin{cases} a_{i0} = \sum_{j=1}^{NC} a_{ij} & \text{para } i = 1, \ldots, NR, \\ a_{0j} = \sum_{i=1}^{NR} a_{ij} & \text{para } j = 1, \ldots, NC, \\ a_{00} = \sum_{i=1}^{NR} a_{i0} & (= \sum_{j=1}^{NC} a_{0j}). \end{cases} \tag{7.1}$$

El Sistema Lineal (7.1) puede representarse de forma natural mediante la *red* $G = (V, A)$ de la Figura 7.1. Esta red tiene un nodo r_i para cada fila $i = 1, \ldots, NR$; un nodo c_j para cada columna $j = 1, \ldots, NC$; más dos nodos extra c_0 y r_0. Existe una correspondencia entre los arcos y los elementos de la matriz de manera que aparece

- un arco (r_i, c_j) asociado con cada elemento a_{ij} con $i \neq 0$ y $j \neq 0$,
- un arco (c_0, r_i) para cada elemento marginal a_{i0} con $i \neq 0$,
- un arco (c_j, r_0) para cada elemento marginal a_{0j} con $j \neq 0$,
- un arco (r_0, c_0) correspondiente al elemento suma total a_{00}.

A partir de ahora no distinguiremos entre un elemento (i, j) y su correspondiente arco $(u, v) \in A$. Con otro poco de abuso de notación, también escribiremos, por ejemplo, a_{uv} para indicar el valor a_{ij} del elemento (i, j) asociado con el arco (u, v) de G.

Sobre la representación en forma de red, las ecuaciones (7.1) dicen que el flujo total que entra en un nodo debe ser igual al flujo total que sale del mismo. De esta manera, los valores a_{ij} se corresponden con un *flujo circulante* sobre dicha red.

Claramente, hay una infinidad de valores a_{ij} que satisfacen (7.1). Si asumimos para generalizar que cada elemento a_{ij} tiene un rango fijo de valores posibles, digamos $[a_{ij} - lb_{ij}, a_{ij} + ub_{ij}]$, se dirá que un conjunto de valores $\tilde{a}_{ij}$ es *factible* si y sólo si

- $a_{ij} - lb_{ij} \leq \tilde{a}_{ij} \leq a_{ij} + ub_{ij}$,
- $[\tilde{a}_{ij}]$ satisface (7.1).

En términos de la red, $[\tilde{a}_{ij}]$ define entonces un *flujo circulante factible* en $G = (V, A)$, donde cada arco $(u, v) \in A$ tiene ahora dos cotas superior/inferior dadas, digamos $a_{uv} - lb_{uv}$ y $a_{uv} + ub_{uv}$, sobre el flujo de este arco (típicamente, $1/+\infty$). Asumiremos siempre que los valores dados $[a_{ij}]$ definen un flujo circulante factible sobre G, al que llamaremos *flujo circulante nominal*.

A continuación caracterizamos los flujos circulantes factibles sobre G que pueden obtenerse cambiando (con respecto a a_{ij}) el flujo sobre un subconjunto de arcos *dados*. Para ser más precisos, sea SUP un subconjunto de arcos de G (elementos SUPrimidos de la matriz). Obsérvese que SUP puede contener arcos correspondientes a totales marginales. Todos los arcos $(u, v) \notin SUP$ deberán mantener sus valores *nominales*, digamos a_{uv}. El flujo sobre los arcos $(u, v) \in SUP$, sin embargo, puede cambiar respecto a su valor nominal, supuesto que su nuevo valor se mantiene dentro del rango factible marcado por sus cotas inferior y superior.

Podemos ahora definir una *red incremental* $G(SUP) = (V, A(SUP))$ a partir de G como sigue:

- eliminar todos los arcos $(u, v) \in A \setminus SUP$,
- reemplazar cada arco $(u, v) \in SUP$ por dos arcos, que son:
  - un *arco adelante* (u, v) con capacidad $k_{uv} := ub_{uv}$,
  - un *arco atrás* (v, u) con capacidad $k_{vu} := lb_{uv}$.

Cada arco adelante de $G(SUP)$ se corresponde con un *incremento* en el flujo sobre su arco de G, mientras que cada arco atrás con un *decremento* del valor del flujo por su arco. Con más precisión, el flujo circulante $[\tilde{a}_{uv}]$ sobre G asociado con cada flujo circulante y sobre la red incremental se calcula como:

$$\begin{array}{ll} \tilde{a}_{uv} := a_{uv} & \text{para cada } (u, v) \in A \setminus SUP, \\ \tilde{a}_{uv} := a_{uv} + (y_{uv} - y_{vu}) & \text{para cada } (u, v) \in SUP. \end{array}$$

Supongamos ahora que nos preguntamos por el máximo *incremento* que el flujo puede tomar sobre un arco *específico* (u_0, v_0) en una solución factible. Esto se corresponde con encontrar el máximo flujo circulante y sobre $G(SUP)$ que maximiza $y_{u_0v_0} - y_{v_0u_0}$, es decir, el incremento del flujo sobre (u_0, v_0). Así podemos calcular tal incremento máximo simplemente eliminando de $G(SUP)$ los dos arcos asociados con (u_0, v_0), obteniendo ahora $G(SUP \setminus \{(u_0, v_0)\})$, y encontrando el *flujo máximo* desde v_0 a u_0 en $G(SUP \setminus \{(u_0, v_0)\})$. Análogamente, el máximo *decremento* asociado con (u_0, v_0) se corresponde con el valor del flujo máximo en $G(SUP \setminus \{(u_0, v_0)\})$ desde u_0 a v_0.

Las anteriores consideraciones nos permiten calcular eficientemente, el máximo incremento (digamos $\Delta^+_{u_0v_0} \geq 0$) y decremento (digamos $\Delta^-_{u_0v_0} \geq 0$) posible sobre cada celda $(u_0, v_0) \in SUP$, cualquiera que sea el conjunto SUP previamente dado. Además, por el clásico *Teorema de máximo-flujo/mínimo-corte* (es decir, la teoría de la dualidad aplicada a redes) obtenemos la siguiente útil caracterización:

Teorema 7.3.1 $\Delta^+_{u_0v_0} \geq 0$ *es igual a la capacidad del mínimo* $\{v_0, u_0\}$*-corte en* $G(SUP \setminus \{(u_0, v_0)\})$*, mientras que* $\Delta^-_{u_0v_0} \geq 0$ *es igual a la capacidad del mínimo* $\{u_0, v_0\}$*-corte en* $G(SUP \setminus \{(u_0, v_0)\})$*.*

7.3.1 Modelo Matemático

Sean una tabla *nominal* $[a_{ij}]$, donde cada elemento (i, j) tiene asociado un valor inferior $a_{ij} - lb_{ij}$ (típicamente 1), un valor superior $a_{ij} + ub_{ij}$ (típicamente $+\infty$), y un *peso de supresión* w_{ij}. Además, consideremos un conjunto $PS \subset A$ de *supresiones primarias*; para cada $(u_0, v_0) \in PS$ tenemos dos *nivel de protección* asociados, definidos por los dos valores $a_{u_0v_0} - LPL_{u_0v_0}$, y $a_{u_0v_0} + ULP_{u_0v_0}$ (por *L*ower *P*rotection *L*evel y *U*pper *P*rotection *L*evel).

Queremos encontrar un conjunto $SUP \subseteq A$ de peso mínimo tal que

1. $PS \subseteq SUP$,
2. para cada $(u_0, v_0) \in PS$ se verifique $\Delta^+_{u_0v_0} \geq ULP_{u_0v_0}$ y $\Delta^-_{u_0v_0} \geq LPL_{u_0v_0}$ (donde $\Delta^+_{u_0v_0}$ y $\Delta^-_{u_0v_0}$ son calculados con respecto a SUP).

Debido a la discusión anterior, establecemos el siguiente modelo de Programación 0-1. Sea

$$x_{uv} = \begin{cases} 1 & \text{si } (u, v) \in SUP \\ 0 & \text{en otro caso} \end{cases}, \text{ para todo } (u, v) \in A.$$

Recordemos que $G(A)$ es la red incremental asociada con $SUP = A$ (todas las celdas están suprimidas). Para cada (u, v) de $G(A)$, sea $\phi(u, v)$ el correspondiente arco de G (luego $\phi(u, v) \in \{(u, v), (v, u)\}$). Entonces CSP se puede formular como:

$$\min \sum_{(u,v)\in A} w_{uv}x_{uv} \tag{7.2}$$

sujeto a

$$0 \le x_{u,v} \le 1 \text{ entero} \qquad \text{para todo } (u,v) \in A, \tag{7.3}$$
$$x_{uv} = 1 \qquad \text{para todo } (u,v) \in PS, \tag{7.4}$$

y para cada $(u_0, v_0) \in PS$:

$$\sum_{(u,v)\in\delta^+(S)\setminus\{(v_0,u_0)\}} k_{uv}\, x_{\phi(u,v)} \ge ULP_{u_0v_0} \qquad \text{para todo } S \subset V : v_0 \in S, u_0 \notin S, \tag{7.5}$$
$$\sum_{(u,v)\in\delta^-(S)\setminus\{(u_0,v_0)\}} k_{uv}\, x_{\phi(u,v)} \ge LPL_{u_0v_0} \qquad \text{para todo } S \subset V : v_0 \in S, u_0 \notin S, \tag{7.6}$$

donde $\delta^+(S)$ y $\delta^-(S)$ son el conjunto de arcos de $G(A)$ saliendo y entrando en un $S \subset V$ dado, respectivamente, y las capacidades k_{uv} de los arcos son las definidas anteriormente.

7.3.2 Una cota inferior: la fase de corte

En el intento de "podar" al máximo el árbol decisional que deberá examinar el algoritmo de ramificación y acotación, resulta muy conveniente disponer de potentes procedimientos que proporcionen ajustadas cotas inferior y superior. En la siguiente sección abordaremos cómo se pueden encontrar soluciones heurísticas cuyo valor sea una buena cota superior. Veremos en ésta cómo la relajación lineal del modelo entero anterior puede ser fortalecida para dar una buena cota inferior. Describiremos 4 nuevas familias de desigualdades válidas, así como procedimientos para identificar (si existen) sus elementos más violados por una solución de la relajación, para resolver los correspondientes *problemas de separación*.

Desigualdad capacidad

Cada desigualdad (7.5) puede mejorarse reemplazando, para cada (u_0, v_0) arbitrario pero fijo, cada coeficiente k_{uv} por $k'_{uv} := \min\{k_{uv}, ULP_{u_0v_0}\}$ (dado que $x \ge 0$ y entero). Análogamente, cada miembro de (7.6) puede mejorarse reemplazando, para cada (u_0, v_0) arbitrario pero fijo, cada coeficiente k_{uv} por $k''_{uv} := \min\{k_{uv}, LPL_{u_0v_0}\}$. Esto es, (7.5) y (7.6) están dominadas por:

$$\sum_{(u,v)\in\delta^+(S)\setminus\{(v_0,u_0)\}} k'_{uv}\, x_{\phi(u,v)} \ge ULP_{u_0v_0} \qquad \text{para todo } S \subset V : v_0 \in S, u_0 \notin S, \tag{7.7}$$
$$\sum_{(u,v)\in\delta^-(S)\setminus\{(u_0,v_0)\}} k''_{uv}\, x_{\phi(u,v)} \ge LPL_{u_0v_0} \qquad \text{para todo } S \subset V : v_0 \in S, u_0 \notin S, \tag{7.8}$$

a las que llamaremos *desigualdad capacidad* (*capacity constraint*).

Aunque el modelo contiene un número exponencial de restricciones, su *relajación lineal* puede resolverse eficientemente dado que el problema de separación para (7.7) y (7.8) se resuelve abordando problemas de flujo máximo sobre $G(A)$ en el que los dos arcos adelante/atrás asociados con cualquier $(u, v) \in A$ tienen capacidad $k'_{uv}\ x^*_{uv}$ y $k''_{uv}\ x^*_{uv}$, respectivamente, siendo x^* la solución dada.

A continuación se presenta un esquema del procedimiento de separación.

Procedimiento CUT_SEP;
entrada el punto (fraccionario) $x^* \in [0,1]^A$
salida cortes más violados de tipo (7.7)–(7.8)
comienzo
1. **para cada** $(u_0, v_0) \in PS$ **hacer**
2. **desde** dirección:=1 **hasta** 2 **hacer comienzo**
3. **si** dirección=1 **entonces comienzo** /* cortes (7.7) */
$\mu := ULP_{u_0v_0}$; $s := v_0$; $t := u_0$ **fin**
en otro caso comienzo /* cortes (7.8) */
$\mu := LPL_{u_0v_0}$; $s := u_0$; $t := v_0$ **fin**;
6. definir una red incremental vacía;
5. **para cada** $(u, v) \in A \setminus \{(u_0, v_0)\} : x^*_{uv} > 0$ **hacer**
6. añadir dos arcos a la red incremental, digamos
un arco adelante (u, v) con capacidad $x^*_{uv}\ \min\{ub_{uv}, \mu\}$ y
un arco atrás (v, u) con capacidad $x^*_{uv}\ \min\{lb_{uv}, \mu\}$;
7. intentar enviar μ unidades de flujo desde s a t;
8. si esto no es posible, entonces almacenar la restricción (7.7) o (7.8) asociada a un $\{s, t\}$-corte de mínima capacidad en la red incremental
fin
fin.

En realidad, tras obtener un flujo máximo se examina su grafo incremental, almacenando una restricción por cada corte mínimo que se detecta (=número de componentes conexas menos 1).

Nuestras experiencias computacionales nos han mostrado que la solución de este modelo es, en muchos casos, ya una solución entera (y por tanto óptima) para el CSP. No obstante, aumentando la dimensión de los problemas tratados se observa que no siempre ocurre así, y que tal relajación debe ser fortalecida con nuevos cortes para aproximar su valor lineal al valor óptimo del modelo entero.

Desigualdad puente

Observemos que podemos exigir que las celdas suprimidas no puedan determinarse de modo exácto. Esto se obtiene considerando otro proceso adicional de separación similar al anterior, pero en el que se fija $ULP_{uv} = LPL_{uv} = x^*_{uv}$ para *todo* $(u, v) \in A$, $k'_{uv} = k''_{uv} = 1$, y se reemplaza el Paso 1. por

1'. **para cada** $(u_0, v_0) \in A : x^*_{u_0v_0} > 0$ **hacer**

Las nuevas restricciones (7.7) y (7.8) ahora coinciden —el grafo incremental es ahora simétrico debido a que $k'_{uv} = k''_{uv}$—, y se escriben como

$$\sum_{e \in \delta(S) \setminus \{e_0\}} x_e \geq x_{e_0} \quad \text{para todo } S \subset V \text{ y } e_0 \in \delta(S), \tag{7.9}$$

donde $\delta(S)$ representa el conjunto de arcos (en el grafo incremental simétrico) con un nodo incidente en S o el otro fuera de S.

Estas restricciones reciben el nombre de *desigualdad puente* (*bridgeless inequality*), y requieren que cada celda e con $x^*_e = 1$ —es decir, cada celda suprimida— deba estar contenida en un "circuito". Esta condición generaliza la propuesta por Geurts (véanse las restricciones 6a–6b en la página 52 de [G92]), donde esto mismo aparece sólo para los cortes triviales asociados a conjuntos S de un sólo elemento. Aunque las restricciones (7.9) son redundantes en un modelo entero que contenga (7.7) y (7.8), experiencias computacionales nos han demostrado que su consideración ayuda notablemente a fortalecer la relajación lineal original.

Desigualdad peine

El resultado del punto anterior nos ha estimulado al estudio de un problema que subyace bajo el CSP en tablas bi-dimensionales, y que es el *problema del subgrafo sin puentes* (BSP por *bridgeless subgraph problem*): Un arco e se llama *puente* en un subgrafo H de G cuando su eliminación conlleva a un subgrafo $H \setminus \{e\}$ con una componente conexa más que H. Asumiendo un costo w_e asociado con cada arco e de G, y un subconjunto PS de arcos de G, el BSP busca un subgrafo H sin puentes, que contenga los arcos de PS, y con el menor costo $\sum_{e \in H} w_e$. (Obsérvese que en una solución factible H del BSP, dos vértices diferentes de una misma componente conexa deberán estar conectados por al menos dos caminos, esto es, las componentes conexas de H deberán ser *2-conexas*.) Relajando las imposiciones de los niveles de protección (esto es, manteniendo que lo único importante es que no se conozca el valor exacto de las celdas que se decidan ocultar), y dado que los pesos de supresión se asumen positivos, es claro que las soluciones que nos interesan para el CSP deberán ser también factibles para el BSP. Ello implica que cortes válidos para el

BSP sean válidos también para el CSP, y motiva por tanto el estudio de este problema. La demostración de que BSP pertenece a la clase de los problemas $\mathcal{NP}$-difíciles está en el trabajo de Frederickson y Ja'Ja' [FJ81]. Sustituyendo cada arco $[u, v]$ de un grafo G por un nodo w y los dos arcos $[u, w]$ y $[w, v]$, se concluye que el BSP es $\mathcal{NP}$-difícil incluso cuando G es bipartito, por lo que no se espera poder disponer de una descripción lineal completa para él. Sin embargo, sí de una descripción lineal parcial, y una primera familia de cortes válidos está formada por los anteriores cortes puente; otra familia es la que detallamos a continuación.

Consideremos un subconjunto $H \subset V$ particionado en $S_1, \ldots, S_k$, donde k es un entero impar, y un arco diferente $e_i \in \delta(S_i) \cap \delta(H)$ para cada $i = 1, \ldots, k$. Sumando para todo $i = 1, \ldots, k$ las restricciones puente $x_{e_i} - \sum_{e \in \delta(S_i) \setminus \{e_i\}} x_e \leq 0$, y cota superior $x_{e_i} \leq 1$ se obtiene la desigualdad:

$$\sum_{i=1}^{k} \left[2\, x_{e_i} - \sum_{e \in \delta(S_i) \setminus \{e_i\}} x_e \right] \leq k.$$

Dividimos todo por 2, y teniendo presente que el lado izquierdo será un valor entero para toda solución factible, entonces redondeamos por defecto la constante del lado derecho de tal desigualdad. Obtenemos así la desigualdad

$$\sum_{i=1}^{k} x_{e_i} - \sum_{e \in Q} x_e \leq \frac{k-1}{2}, \tag{7.10}$$

siendo $Q := [\cup_{v \in H} \delta(v)] \setminus \{e_1, \ldots, e_k\}$. Se trata por tanto de una desigualdad válida para el CSP, a la que llamaremos *desigualdad peine* (*comb inequality*), donde H es el *mango* de la desigualdad, y $e_1, \ldots, e_k$ los *dientes*.

La Figura 7.2 representa el *grafo soporte* de una solución fraccionaria que no satisface la desigualdad peine (7.10) para $k = 3$, $S_1 = \{r_2, r_7, c_3, c_6\}$, $S_2 = \{r_6\}$, $S_3 = \{c_7\}$, $e_1 = (c_3, r_3)$, $e_2 = (r_6, c_2)$, y $e_3 = (c_7, r_3)$. Las líneas continuas representan variables a valor 1, las discontinuas variables a valor 1/2, y las ausentes variables a valor 0. Las 5 líneas continuas de trazo más grueso representan las 5 celdas sensibles de la tabla estadística considerada, y cuyos datos nominales se muestran en la Tabla 7.3 (los pesos de supresión coinciden con los nominales, y los niveles de protección son iguales a 1).

Para resolver el correspondiente problema de separación, una primera idea heurística se basa en la compresión (*shrinking*) de ciertos subconjuntos candidatos, para luego proceder al cálculo del *corte impar* de mínimo peso, inspirado ello por el trabajo de Padberg y Grötschel [PG85] sobre el *Problema del Viajante de Comercio* (*Travelling Salesman Problem*). Otra alternativa se basa en observar que las desigualdades anteriores son cortes procedentes de una *derivación 0-1/2 de Chvátal-Gomory*, y por tanto son separables buscando *ciclos impares* en un grafo particularmente diseñado según indican Caprara y Fischetti [CF96]. No describimos los detalles teóricos porque su implementación práctica no

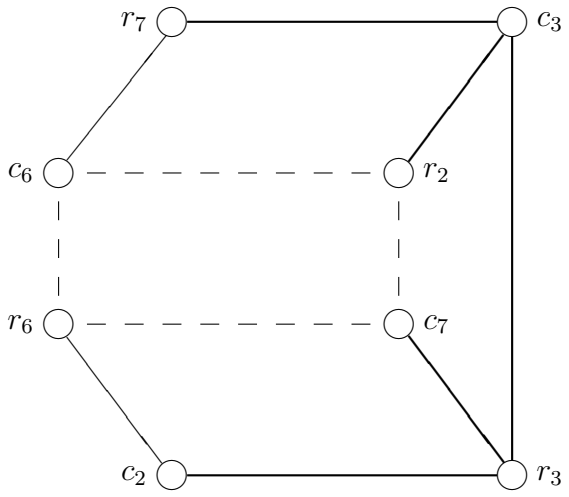


Figura 7.2: Solución fraccionaria que no satisface un desigualdad peine.

está aún al nivel deseado, y consecuentemente tampoco será contemplada en las experiencias computacionales de la Sección 7.3.5.

Actualmente estamos trabajando en estos procesos de separación.

Desigualdades de cubrimiento

Hasta el momento se ha fortalecido la relajación lineal olvidando los niveles de protección y estudiando el problema resultante; también el proceso contrario es posible, tratando de fortalecer la relajación lineal sobre las restricciones que vienen de imponer los niveles de protección. Así, los cortes (7.7)-(7.8) han sido impuestos olvidando el hecho de que las x-variables deben de ser enteras 0–1. Cada una de estas restricciones definen un *problema de la mochila* (KP) en forma de "$\geq$", que puede expresarse en forma de "$\leq$" introduciendo las variables $x_{ij} = 1 - z_{ij}$ (de forma análoga a como hicimos en la Sección 7.2). De este modo todas las restricciones adicionales implicadas por la limitación de la capacidad de la mochila y por la exigencia de que las variables deban de ser enteras, son también cortes válidos para nuestro problema. Conviene observar que gran parte de la dificultad del CSP se debe a que el KP está bajo él. La Figura 7.3 muestra una solución fraccionaria procedente de una relajación lineal conteniendo todas las desigualdades precedentes.

Inspirados por el trabajo de Crowder, Johnson y Padberg [CJP83] para problemas generales de Programación Entera 0-1, para cada desigualdad capacidad del tipo (7.7) y (7.8), y que representaremos genéricamente mediante

$$\sum_{e \in D} k'_e x_e \geq p, \tag{7.11}$$

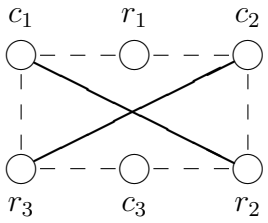


Figura 7.3: Solución fraccionaria combinación convexa de dos subgrafos sin puentes.

hemos considerado las desigualdades

$$\sum_{e \in Q} x_e \geq 1, \tag{7.12}$$

siendo $Q \subseteq D$ un subconjunto minimal respecto a la inclusión de conjuntos, y tal que tal que $\sum_{e \in D \setminus Q} k'_e < p$. Tales desigualdades válidas para CSP reciben el nombre de *desigualdades de cubrimiento* (*cover inequalities*), y su proceso de separación se describe a continuación.

Supongamos tener una solución fraccionaria x^* de la relajación lineal, y una desigualdad capacidad del tipo genérico (7.11) obtenida en alguna salida precedente del procedimiento CUT_SEP. Buscamos entonces un subconjunto de arcos $Q \subseteq D$ tal que $\sum_{e \in D \setminus Q} k'_e < p$ y minimizando $\sum_{e \in Q} x^*_e$. Para ello introducimos una variable decisional z_e para cada arco $e \in D$ asumiendo valor 1 si y sólo si $e \notin Q$, y resolvemos el siguiente KP:

$$\max \left\{ \sum_{e \in D} x^*_e \, z_e \ : \ \sum_{e \in D} k'_e \, z_e \leq p - 1 \ \text{ y } \ z_e \in \{0,1\} (e \in D) \right\}.$$

Si el valor óptimo de dicho problema es menor que $\sum x^*_e - 1$, entonces concluimos que no existe ninguna desigualdad de cubrimiento a partir de (7.11) violada por el punto actual. En otro caso, basta considerar la desigualdad (7.12) para el conjunto Q inducido por la correspondiente solución óptima del KP, hecho minimal.

Para la resolución cada KP usamos el programa desarrollado por Martello y Toth [MT90]. Aunque obviamente tal algoritmo no tiene una complejidad teórica polinomial, resulta bastante efectivo en la práctica. Téngase presente que cuando $x^*_e = 1$ entonces $e \notin Q$ y cuando $x^*_e = 0$ entonces $e \in Q$. Esto, unido con que el número de variables x_e con $0 < x^*_e < 1$ no suele ser grande, produce KP relativamente pequeños.

7.3.3 Una cota superior: el heurístico

La efectividad de un algoritmo de ramificación y acotación también aumenta fuertemente con la rapidez en el generar "buenas" soluciones del problema. En la bibliografía se

han considerado varios heurísticos para el CSP, siendo el propuesto por Kelly, Golden y Assad [KGA91] el que presenta mejores resultados. Su heurística trata no con todas las supresiones primarias simultáneamente, sino con cada una por separado. Dividen el proceso en dos fases. En la primera consideran una red donde las capacidades de los arcos son los máximos incrementos/decrementos respecto a los valores nominales, y los costos son los pesos de supresión. Luego resuelven sobre ella dos *problemas del flujo a costo mínimo* para enviar $ULP_{u_0v_0}$ unidades de flujo desde v_0 a u_0, y $LPL_{u_0v_0}$ desde u_0 a v_0, para cada celda primaria (u_0, v_0), proponiendo suprimir todas las celdas asociadas a arcos que resultaron útiles para algún flujo. En la segunda fase tratan de ver si alguna una de las celdas secundarias propuestas por la fase anterior, puede mantenerse sin suprimir, sin que por ello se infrinjan los niveles de protección de alguna celda primaria.

Nuestras experiencias computacionales nos muestran que esta heurística consume bastante tiempo de cálculo especialmente en la fase primera, debido a las resoluciones de los problemas de flujo a costo mínimo. Por otra parte, este proceso penaliza las celdas que propone de suprimir de manera proporcional al flujo que pasa por ellas, mientras que el CSP las penaliza sólo si son usadas (independientemente de cuánto se usen), lo que puede justificar los malos resultados que nos produjo esta heurística en nuestras experiencias.

Buscando un procedimiento heurístico mucho más veloz, capaz de ser ejecutado en varias ocasiones (a ser posible tras cada iteración de nuestra fase de hiperplanos de corte), e incorporando la información que nos dan las soluciones fraccionarias sobre la solución óptima del CSP, hemos propuesto un heurístico del estilo anterior, pero sustituyendo la fase primera por:

Procedimiento HEURISTICO;
entrada el punto (fraccionario) $x^* \in [0,1]^A$
salida una solución heurística en SUP
comienzo
1. Asociarle a cada arco e el costo $c_e := x^*_e\, w_e$;
   $SUP := \emptyset$;
   $L := \emptyset$;
2. **para cada** $(u_0, v_0) \in PS$ **hacer comienzo**
   desde dirección:=1 **hasta** 2 **hacer**
   si dirección=1 **entonces comienzo** $\mu := ULP_{u_0v_0}$; $s := v_0$; $t := u_0$ **fin**
   en otro caso comienzo $\mu := LPL_{u_0v_0}$; $s := u_0$; $t := v_0$ **fin**;
   $L := L \cup \{(\mu, s, t)\}$;
   end;
3. **para cada** $(\mu, s, t) \in L$ **hacer comienzo**
   construir la red incremental $G(A)$;
4. calcular el camino de costo mínimo para ir de s a t en $G(A)$
   y el incremento máximo f de flujo que se puede enviar por él;
   para cada e en el camino **hacer comienzo**

```
                    c_e := 0;
                    SUP := SUP ∪ {e}
                fin
5.              si f < μ entonces comienzo
                    disminuir la capacidad de cada arco del camino en f unidades;
                        (eliminando obviamente los arcos de capacidad nula)
                    disminuir μ en f unidades;
                    volver al Paso 4.
                fin
            fin
        fin
    fin.
```

donde los (μ, s, t) son extraídos de L en el Paso 3. en orden decreciente de su valor asociado μ.

7.3.4 El algoritmo de ramificación y corte

Al igual que cualquier técnica de ramificación y acotación, nuestro algoritmo examina nodos del árbol decisional, insertándolos y eliminándolos dinámicamente en una *lista de nodos* $\mathcal{L}$ hasta que ésta queda vacía. Cada nodo en la lista contendrá información sobre variables fijadas y restricciones que lo determinan, así como el valor lineal óptimo (*cota inferior*) y la solución x^* básica de la relajación lineal última considerada en dicho nodo durante la fase de acotación. Paralelamente se trabaja con una *cota superior UB* procedente de una solución factible para el CSP.

La lista $\mathcal{L}$ es inicializada con un único nodo (llamado *nodo raíz*), conteniendo sólo como fijas las variables asociadas a celdas sensibles (fijas a valor 1) y las variables asociadas con celdas de valor nominal nulo (fijas a valor 0), y como restricciones sólo las *desigualdades triviales* $0 \leq x_e \leq 1$ para las restantes variables. Este nodo tiene asociada por defecto la solución (entera probablemente no factible) con $SUP = PS$ y de valor 0.

Iterativamente se extrae un nodo de $\mathcal{L}$ para ser estudiado por la fase de acotación. Usamos el criterio de elegir el nodo con la menor cota inferior de la lista. Es decir, realizamos un recorrido "*best-first*" (o "*lowest-first*") por el árbol decisional, y no el clásico "*depth-first*" mucho más fácil de implementar pero seguramente más ineficiente cuando el algoritmo heurístico no produce un óptimo.

Cuando la fase de *acotación* estudia un nodo, trata –iterativamente– de mejorar su cota inferior. En cada iteración dispone de una solución x^* con la que inicia el proceso de separación, siempre que ésta no sea ya una solución factible del CSP, o si su cota inferior (redondeada por exceso) no es inferior a *UB*. El proceso de separación tratará, mediante

los correspondientes algoritmos de separación, de determinar desigualdades violadas de las descritas en la Sección 7.3.2 (de ahí el nombre alternativo de *fase de corte*). Estas familias de desigualdades son examinadas en el orden

1. desigualdades capacidad (7.7)–(7.8);
2. desigualdades puente (7.9);
3. desigualdades cubrimiento (7.12).

Las desigualdades determinadas (nunca más de 30) se introducen en el Programa Lineal (LP) del momento, y la fase de acotación prosigue resolviendo tal LP, obteniendo una nueva solución x^* (quizás fraccionaria), eliminando del LP ciertas restricciones y variables ahora inútiles (atendiendo a los valores de sus variables de holgura y costos reducidos, respectivamente), para volver luego a entrar en la fase de separación. Al tiempo que se introduce una restricción en el LP, también se almacena una copia (en forma comprimida) dentro de una estructura conocida como *piscina (pool)*, de la que nunca viene eliminada (aunque lo sea su copia del LP). Esto permite disponer de forma rápida de una amplia gama de restricciones que han sido útiles para el LP, que en un momento dado lo dejaron de ser, pero que pueden volver a resultar útiles. Por ello, antes de aplicar los procedimientos de separación anteriores, se buscan desigualdades violadas en la piscina.

Si para un determinado punto x^* la separación no detecta ninguna desigualdades, y tampoco se puede concluir el examinar dicho nodo del árbol decisional por cuanto ni tal punto es entero, ni su valor (redondeado por exceso) excede o iguala el valor UB, entonces se pasa a la fase de *ramificación*. Alternativamente, también se pasa directamente a esta fase cuando en las últimas 10 iteraciones, la cota inferior del nodo no ha mejorado en al menos 0.001 (*tailing off*). En cuanto a la actualización de UB, ejecutamos el heurístico descrito en la Sección 7.3.3 al inicio del nodo raíz, y cada 10 iteraciones del proceso de separación en cada nodo del árbol decisional,

En la fase de ramificación elegimos 10 variables x_e con $0.05 \leq x_e^* \leq 0.95$ lo más próximo posible a 0.5 (o todas las fraccionarias si no hay 10 con tales condiciones). Para cada variable x_e considerada resolvemos dos problema de programación lineal creados considerando el PL último, más la restricción $x_e = 0$ en uno, y $x_e = 1$ en otro, dando los valores LP_e^0 y LP_e^1, respectivamente. Si durante la resolución de tales problemas lineales sucede que alguno de dichos valores redondeado por exceso excede o iguala el valor UB, entonces paramos la fase de ramificación fijando la correspondiente variable a su otro valor, y continuando la fase de acotación con el nuevo punto fraccionario. En otro caso, elegimos la variable fraccionaria x_e con mayor valor $0.5\,LP_e^0 + 0.5\,LP_e^1$ para crear dos nodos nuevos (*hijos*) del árbol decisional a partir del nodo actual (*padre*), uno con x_e fijo a valor 0, otro con x_e fijo a 1. El nodo actual deja de ser estudiado, y los dos nodos nuevos son almacenados en la lista $\mathcal{L}$ con sus correspondientes cotas inferiores y soluciones básicas lineales óptimas.

10593	268	26	714	2890	102	422	15015
1069	**2**	**1**	38	340	11	23	1484
4436	9	0	216	752	39	47	5499
6	0	0	**1**	**1**	0	0	8
14163	255	0	925	3837	126	305	19611
138	274	5	6	122	**3**	5	553
16	72	5	7	480	**3**	93	676
30421	880	37	1907	8422	284	895	42846

Tabla 7.2: Tabla estadística real ISTAT.A

64153	666	38	5042	11726	333	1019	82977
680	21	**2**	72	173	10	11	969
34	**2**	**3**	6	6	0	**2**	53
4853	83	6	1275	612	65	82	6976
24251	234	8	1591	1959	168	377	28588
865	7	0	84	151	10	8	1125
5702	76	**3**	519	781	33	166	7280
100538	1089	60	8589	15408	619	1665	127988

Tabla 7.3: Tabla estadística real ISTAT.B

7.3.5 Resultados computacionales

Todo ha sido programado en lenguaje ANSI C. Se ha usado la librería CPLEX 3.0 para abordar los problemas lineales. Las experiencias computacionales se han llevado a cabo sobre un PC 486 a 50 Mhz. y con 8 Mb. de memoria RAM, funcionando bajo MS-DOS.

Inicialmente hemos generado ejemplos bi-dimensionales del CSP de forma completamente aleatoria, en el sentido de que los valores a_{ij} son los número al azar en el rango $[0, 1000]$, los w_{ij} son iguales a los a_{ij}, las celdas sensibles son elegidas al azar en toda la tabla, los niveles de protección superior se fijan a 100, y los niveles de protección inferior se fijan a $\min\{100, a_{ij} - 1\}$. Con el límite de 8 "Megabytes" de memoria RAM de nuestro PC 486, hemos logrado resolver ejemplos de éstos con hasta 300 filas, 100 columnas y 100 celdas sensibles, en tiempos nunca superior a la media hora de cálculo, y siempre al nodo raíz del árbol decisional. No obstante, obviamente, ésta podría no ser la generación ideal de ejemplos *reales* para el CSP, y por ello eludimos los resultados.

Siguiendo sugerencias de Roberto Benedetti, del Instituto Nacional de Estadística italiano (quien además nos proporcionó los datos reales que mostramos en las tablas 7.2 y 7.3, respectivamente), y buscando poner en dificultades nuestro código, hemos considerado los dos siguientes generadores de problemas:

Generador A: Con probabilidad 5/100 una celda toma un valor nominal aleatorio en

[0,10[, y con 95/100 en [10,100[. Para cada celda, el peso de supresión coincide con su valores nominal, salvo cuando ésta sea 0, en cuyo caso el peso de supresión se pone a un valor grande porque sabemos que la correspondiente celda no deberá estar entre las suprimidas. Clasificamos como sensibles las celdas con valor nominal en [1,3] (a las que les anulamos sus pesos de supresión). Los niveles de protección superiores coinciden con el valor nominal, y los inferiores con el valor nominal menos 1.

Generador B: Con probabilidad 5/100 una celda toma un valor nominal aleatorio en [0,5[, y con 95/100 en [5,500[. Si celdas con valor nulo no se suprimen, y las celdas con valores en]0,5[se consideran *sensibles*, con nivel de protección inferior igual a su valor nominal menos 1, y con nivel de protección superior igual a su valor nominal. Los pesos de supresión son iguales a los correspondientes valores nominales.

Los parámetros de entrada de ambos generadores son la semilla *seed* para inicializar un generador pseudo-aleatorio, el número de filas NR y el número de columnas NC (sin contar los totales marginales).

Las columnas que aparecen en las tablas de resultados indican:

nombre: semilla para la pseudo-aleatorización, o nombre para los procedentes de una librería;

n**:** número de filas (sin contar la marginal);

m**:** número de columnas (sin contar la marginal);

p**:** número de celdas sensibles (supresiones primarias);

$\mathbf{UB}_0$**:** razón porcentaje de la cota superior al final del nodo raíz, sobre el valor óptimo;

$\mathbf{LB}_0$**:** razón porcentaje de la cota inferior al final del nodo raíz, sobre el valor óptimo;

t_0**:** tiempo al final del nodo raíz;

z^***:** valor óptimo del problema;

s**:** número de supresiones secundarias en el óptimo;

nod: número de nodos examinados en el árbol decisional;

t_{total}**:** tiempo total de ejecución;

t_{heur}**:** tiempo total consumido por HEURISTICO;

cap: desigualdades capacidad (7.7)–(7.8);

pue: desigualdades puente (7.9);

cub: desigualdades cubrimiento (7.12);

donde los tiempos están dados en segundos sobre un PC 486 a 50 Hhz.

La Tabla 7.4 muestra los resultados de utilizar nuestro programa para resolver los dos problemas reales, más 35 problemas producidos por el Generador A al combinar los parámetros *seed* $\in \{111, 222, 333, 444, 555\}$, y $NC = NR \in \{10, 15, 20, 25, 30, 35, 40\}$. Analizando dicha tabla se observa que los dos problemas reales fueron trivialmente resueltos. En cuanto a los problemas generados aleatoriamente, un análisis estadístico nos muestra que la variación de *time* en función de n se ajusta mucho a *time*=$e^{-15}n^6$, lo que parecería desconcertante si se olvida que la generación ha sido elegida pensando también en poner en dificultades al algoritmo en cuestión, y que produce ejemplos del CSP mucho más difíciles que los considerados por otros autores (por ejemplo, la proporción de supresiones secundarias sobre las supresiones primarias en la solución óptima de los ejemplos aleatorios de la Tabla 7.4 es mayor que en los ejemplos de los autores precedentes). El problema generado con *seed*=333 y $NC = NR = 40$ no se resolvió en el tiempo límite de 2 horas sobre nuestro PC (los datos que figuran en la Tabla 7.4 se han calculado asumiendo que la última solución heurística es la óptima).

Las Tablas 7.5 y 7.6 muestran los resultados sobre los problemas procedentes del Generador B. Cada fila representa la media de los datos tras resolver los 5 problemas obtenidos con *seed*$\in \{111, 222, 333, 444, 555\}$. Como se puede apreciar, estos problemas resultaron más fáciles que los anteriores, y los límites en la dimensión de los problemas tratados se ha debido sólo a limitaciones en la memoria RAM del ordenador usado.

También hemos probado el algoritmo que aquí se propone sobre una librería de ejemplos proporcionada por C.A.J. Hurkens, y usada en Geurts [G92]. Tales ejemplos son tablas estadísticas bidimensionales, algunas generadas artificialmente y otras procedentes de casos reales. Cada uno, y para cada celda, muestra el valor nominal, el peso de supresión, y una etiqueta indicando si dicha celda debe ser necesariamente suprimida en la tabla final, puede ser considerada como supresión secundaria, o no debe ser suprimida. Los datos valores nominales y pesos de supresión de los problemas VB19, VB20 y VB21 fueron multiplicados por 10 para tratarlos como números enteros. Los niveles de protección inferior y superior fueron sugeridos por Hurkers como 0 y 1, respectivamente, si el valor nominal es 0, o ambos iguales a 0.1 veces el valor nominal, en otro caso. Los resultados obtenidos por la aplicación del algoritmo que proponemos se muestran en la Tabla 7.7, y como se deduce de ella, todos los ejemplos fueron satisfactoria y trivialmente resueltos sobre nuestro PC 486/50. (Es de notar que en [G92] se propone un método exacto que requiere de 2445, 4244, y 3072, segundos sobre un ordenador SUN SPARC 1+ para la resolución exacta del problema VB20, VB26, y VB31, respectivamente, mientras que nuestra propuesta precisa sólo de 2 segundos sobre un PC 486/50 para cualquiera de estos mismos problemas.)

nombre	n	p	UB_0	LB_0	t_0	z^*	s	nod	t_{total}	t_{heur}	cap	pue	cub
A	8	6	100.00	100.00	0.3	197	9	1	0.55	0.1	29	24	0
B	8	5	100.00	100.00	0.3	136	6	1	0.66	0.1	31	14	0
111	10	4	100.00	100.00	0.7	140	6	1	0.77	0.0	47	45	2
222	10	1	100.00	100.00	0.2	66	5	1	0.27	0.0	8	26	0
333	10	1	100.00	100.00	0.2	71	3	1	0.28	0.1	6	18	0
444	10	1	100.00	100.00	0.2	57	3	1	0.27	0.0	6	18	0
555	10	2	100.00	100.00	0.3	111	4	1	0.33	0.1	12	24	0
111	15	7	113.30	92.55	2.3	188	10	8	13.73	0.1	115	76	20
222	15	3	100.00	85.71	1.5	98	10	4	4.78	0.1	59	107	2
333	15	2	100.00	100.00	0.3	68	4	1	0.33	0.0	8	16	0
444	15	2	100.00	100.00	0.3	84	6	1	0.39	0.0	14	36	0
555	15	3	113.45	85.71	1.2	119	9	4	5.93	0.1	68	155	14
111	20	8	104.35	87.89	4.0	161	14	22	46.74	0.7	167	82	32
222	20	5	116.00	92.00	2.9	125	9	3	6.37	0.2	77	89	10
333	20	5	112.42	87.27	3.3	161	12	23	77.17	0.4	222	278	10
444	20	7	115.38	91.42	2.1	169	12	12	34.43	0.5	150	135	10
555	20	5	124.73	86.81	2.4	182	8	7	17.63	0.3	123	140	11
111	25	11	120.77	87.68	4.2	207	14	44	140.50	2.2	222	204	14
222	25	10	128.65	87.08	4.4	178	14	40	139.84	1.9	281	171	10
333	25	6	110.61	88.55	4.6	179	16	36	128.08	1.1	281	317	23
444	25	11	115.83	86.67	5.2	240	21	115	659.16	11.2	363	230	67
555	25	13	102.31	97.92	4.3	216	18	3	12.69	0.9	123	62	29
111	30	11	100.00	90.00	4.8	120	10	2	12.64	0.7	156	71	11
222	30	14	107.11	88.83	11.9	197	18	69	456.71	7.6	345	179	77
333	30	13	110.48	84.76	6.2	210	21	210	2097.28	16.0	577	514	104
444	30	16	120.55	75.11	6.1	219	21	152	761.76	20.5	385	157	71
555	30	16	110.51	84.44	7.1	257	23	333	3792.55	45.1	486	380	81
111	35	15	126.69	86.77	10.8	251	24	623	6085.04	69.8	537	485	95
222	35	16	103.64	87.95	10.4	220	17	95	997.20	17.8	447	264	91
333	35	18	110.79	92.95	9.5	241	27	57	241.97	8.3	258	105	41
444	35	22	109.47	91.86	14.6	264	26	76	560.64	18.7	337	113	50
555	35	22	110.89	93.58	12.6	257	22	21	158.79	7.4	274	109	19
111	40	18	109.57	86.30	16.6	230	23	264	4425.43	66.5	588	363	132
222	40	21	108.37	90.09	21.2	227	21	174	1531.74	39.6	405	170	65
333	40	27	110.56	82.30	31.6	322	32	495	7200.0*	155.4	695	406	134
444	40	26	109.63	79.30	14.1	301	29	275	4348.19	113.3	505	223	110
555	40	28	117.21	91.54	24.1	337	27	432	3671.30	134.6	503	144	87

(* : La resolución de este problema fue detenida tras 2 horas de cálculo.)

Tabla 7.4: Problemas reales y del Generador A.

n	m	p	UB_0	LB_0	t_0	s	nod	t_{total}	t_{heur}	cap	pue	cub
81	11	30.6	100.0	99.3	3.9	27.0	1.2	5.9	0.8	78.4	8.2	8.6
81	21	56.8	100.5	99.3	11.4	43.2	2.2	22.3	5.6	146.0	11.0	17.4
81	31	89.8	100.4	99.4	27.6	51.6	2.8	56.6	23.2	179.8	3.2	32.4
81	41	117.4	101.2	99.3	49.1	60.2	11.2	378.6	185.1	240.0	2.4	48.8
81	51	146.0	102.3	98.8	82.8	61.4	6.8	247.6	134.6	268.6	5.6	48.2
81	61	173.4	102.3	99.3	125.4	63.4	4.4	249.4	159.3	256.2	0.6	52.0
81	71	205.8	101.2	99.2	168.6	65.8	3.0	257.6	169.1	255.8	1.6	63.0
81	81	234.2	101.5	99.0	195.8	66.2	6.6	467.3	302.7	259.6	1.6	58.6
91	11	33.4	100.0	100.0	3.8	28.8	1.0	4.5	0.9	76.8	3.8	6.2
91	21	66.0	100.6	99.3	15.3	49.2	2.4	26.0	9.0	153.0	10.6	17.4
91	31	101.2	100.3	99.6	32.8	56.6	1.8	51.4	23.6	186.2	3.2	31.6
91	41	132.8	100.1	99.8	71.8	65.0	3.6	154.2	77.6	230.6	6.0	39.8
91	51	162.4	100.8	99.3	96.3	67.0	2.8	168.5	102.9	257.6	1.6	49.0
91	61	198.6	100.8	99.8	151.8	68.8	2.4	188.0	130.4	241.6	0.4	55.0
91	71	230.4	101.7	98.8	184.3	69.0	4.6	394.8	262.6	256.6	1.2	60.0
91	81	268.8	101.2	99.4	263.4	69.8	5.6	605.8	396.5	250.0	0.0	62.6
91	91	308.4	101.3	99.1	342.2	69.2	4.4	590.1	431.1	246.4	0.8	75.0
101	11	37.4	100.0	100.0	4.0	32.4	1.0	5.0	1.0	85.2	2.6	5.4
101	21	74.0	100.1	99.6	18.8	53.4	1.4	23.6	8.8	157.2	6.4	16.2
101	31	112.0	100.0	100.0	34.3	62.6	1.0	35.3	19.4	182.6	0.4	28.0
101	41	146.0	100.0	100.0	72.6	72.6	1.0	73.6	45.5	221.8	1.6	31.0
101	51	181.8	100.4	99.8	131.1	75.8	2.0	160.3	99.0	249.4	0.6	44.6
101	61	220.2	100.7	99.7	177.3	76.4	2.8	231.2	156.8	245.4	0.4	49.0
101	71	259.6	100.4	99.4	223.3	77.4	6.4	454.3	309.0	252.8	0.4	62.8
101	81	304.4	100.3	99.7	298.6	72.6	1.4	366.8	268.1	235.8	0.0	59.0
111	11	40.4	100.0	100.0	4.5	35.0	1.0	5.4	1.0	89.6	0.0	5.0
111	21	82.2	100.4	100.0	20.8	58.6	1.2	22.8	10.0	161.2	6.8	16.0
111	31	121.2	100.0	100.0	39.2	69.0	1.0	40.2	23.1	187.2	0.4	27.2
111	41	157.6	100.0	100.0	82.1	78.0	1.0	83.1	50.5	229.4	2.8	36.2
111	51	202.0	100.1	99.9	130.1	82.6	1.2	144.3	93.1	239.0	1.6	40.0
111	61	241.4	100.3	100.0	196.2	83.2	1.2	206.6	145.0	251.8	0.0	56.2
111	71	291.8	100.5	99.7	269.6	81.8	1.8	329.6	236.1	237.0	1.2	62.0
121	11	43.4	100.0	100.0	5.0	37.6	1.0	6.0	1.3	92.6	1.2	4.4
121	21	89.8	100.0	99.9	21.1	63.6	1.2	23.8	10.1	161.0	3.2	13.8
121	31	132.8	100.0	100.0	40.6	74.2	1.0	41.6	23.2	184.8	0.0	24.2
121	41	173.4	100.0	100.0	94.0	87.8	1.0	94.9	61.9	232.2	0.4	35.0
121	51	220.2	100.1	99.9	148.8	89.8	1.2	156.7	104.3	242.8	0.0	41.4
121	61	268.8	100.0	100.0	202.2	90.0	1.0	203.2	143.0	242.8	0.0	53.2
121	71	317.8	100.0	100.0	310.0	85.6	1.0	311.1	232.1	237.8	0.4	65.4

Tabla 7.5: Problemas del Generador B.

n	m	p	UB_0	LB_0	t_0	s	nod	t_{total}	t_{heur}	cap	pue	cub
131	11	46.6	100.0	100.0	5.2	40.0	1.0	6.2	1.5	94.2	2.0	4.0
131	21	97.4	100.0	100.0	24.0	68.8	1.0	25.0	10.7	165.6	4.6	14.6
131	31	143.2	100.0	100.0	45.7	80.8	1.0	46.7	28.6	196.4	0.2	24.4
131	41	191.6	100.0	100.0	108.9	94.2	1.0	109.9	70.9	235.6	0.4	37.0
131	51	238.2	100.1	99.9	167.4	95.8	1.2	176.4	125.5	250.8	0.0	45.0
131	61	295.4	100.0	100.0	267.4	96.6	1.0	268.4	191.0	255.4	0.0	58.0
141	11	49.6	100.0	100.0	5.8	43.2	1.0	6.7	1.6	99.4	4.8	4.0
141	21	104.4	100.0	100.0	25.8	73.4	1.0	26.8	11.7	168.4	6.6	15.2
141	31	152.0	100.0	100.0	52.8	86.2	1.0	53.8	34.7	204.2	0.2	25.4
141	41	205.8	100.0	100.0	124.5	101.0	1.0	125.5	82.7	246.8	0.0	37.6
141	51	259.6	100.0	100.0	184.0	102.8	1.0	185.1	132.5	251.0	0.0	44.0
151	11	52.6	100.0	100.0	6.8	46.0	1.0	7.8	1.8	106.4	4.2	4.0
151	21	112.0	100.0	100.0	31.9	78.6	1.0	32.9	14.1	178.0	5.6	15.6
151	31	162.4	100.0	100.0	66.2	94.0	1.0	67.2	41.1	219.4	0.0	26.6
151	41	220.2	100.0	100.0	133.7	106.8	1.0	134.8	90.2	248.4	0.0	37.0
151	51	283.4	100.0	100.0	218.4	109.4	1.0	219.4	160.7	256.2	0.0	47.8
161	11	56.8	100.0	100.0	7.3	49.2	1.0	8.3	2.4	112.6	1.4	4.6
161	21	117.4	100.0	99.9	32.8	83.4	1.2	36.7	16.2	183.8	8.2	16.0
161	31	173.4	100.0	100.0	73.0	100.8	1.0	74.0	46.3	224.0	0.0	26.8
161	41	234.2	100.0	100.0	156.5	114.0	1.0	157.5	105.0	262.0	0.0	40.4
161	51	304.4	100.0	100.0	261.1	117.4	1.0	262.1	190.7	272.8	0.0	53.4
171	11	60.6	100.0	100.0	8.3	52.4	1.0	9.2	2.6	116.8	2.0	4.2
171	21	124.8	100.0	100.0	33.9	88.4	1.0	34.9	17.8	191.6	1.2	17.2
171	31	186.0	100.0	100.0	87.2	107.8	1.0	88.2	54.1	236.4	0.0	30.0
171	41	252.0	100.0	100.0	166.0	121.0	1.0	167.0	122.3	268.2	0.0	42.0
181	11	66.0	100.0	100.0	8.9	56.4	1.0	9.7	3.5	123.0	2.0	5.2
181	21	132.8	100.0	100.0	37.7	92.2	1.0	38.4	20.9	199.4	0.0	17.6
181	31	198.6	100.0	100.0	99.3	113.8	1.0	100.0	66.0	244.6	0.0	32.2
181	41	268.8	100.0	100.0	196.3	127.8	1.0	197.0	141.3	279.8	0.0	46.0
191	11	70.4	100.0	100.0	10.4	59.8	1.0	11.2	4.0	128.0	2.0	5.4
191	21	140.0	100.0	100.0	44.0	98.0	1.0	44.8	24.4	208.6	1.2	17.8
191	31	209.0	100.0	100.0	111.4	120.0	1.0	112.2	76.0	254.6	0.0	34.0
191	41	287.8	100.0	100.0	231.7	135.8	1.0	232.5	170.2	290.4	0.0	52.0
201	11	74.0	100.0	100.0	11.6	63.0	1.0	12.4	4.6	133.6	2.6	5.4
201	21	146.0	100.0	100.0	48.3	102.6	1.0	49.1	27.1	216.8	1.4	18.6
201	31	220.2	100.0	100.0	124.7	126.4	1.0	125.5	83.7	270.2	0.0	34.6
201	41	304.4	100.0	100.0	267.9	143.2	1.0	268.7	200.7	308.0	0.0	53.4

Tabla 7.6: Problemas del Generador B (continuación).

nombre	m	n	p	UB_0	LB_0	t_0	z^*	s	nod	t_{total}	t_{heur}	cap	pue	cub
DEMO1	3	4	9	100.00	100.00	1.1	5	1	1	1.81	0.1	4	2	2
DEMO2	6	7	11	100.00	100.00	1.4	218	4	1	2.14	0.2	22	11	20
DEMO3	4	4	4	100.00	100.00	2.0	320	5	1	2.69	0.3	58	32	45
DEMO4	6	6	5	100.00	100.00	1.4	88	4	1	2.14	0.2	46	2	26
DEMO5	5	4	4	115.79	94.74	1.6	19	5	2	4.12	0.3	18	8	9
DEMO6	4	4	4	100.00	100.00	1.2	33	4	1	1.92	0.1	16	6	8
DEMO7	8	6	14	100.00	100.00	2.6	27289	8	1	3.35	0.7	43	5	26
DEMO8	19	6	6	100.00	80.00	2.1	3000	5	2	6.32	0.3	50	8	26
DEMO9	3	3	3	100.00	100.00	1.3	161	3	1	1.98	0.1	12	6	4
DEMO10	5	3	4	100.00	100.00	1.2	6	2	1	1.87	0.2	16	6	8
FREQ1	19	6	22	100.00	100.00	1.3	20	6	1	2.08	0.2	30	0	0
FREQ2	19	6	40	100.00	100.00	1.4	23	6	1	2.14	0.2	24	0	6
MOPS2C	6	5	4	100.00	100.00	1.3	644	5	1	2.09	0.2	20	8	10
MOPS70	17	6	11	101.52	98.56	1.8	3749	5	2	6.38	0.3	58	14	29
MOPS80	17	6	32	100.00	100.00	2.0	9435	8	1	2.75	0.7	30	0	6
VB01	5	10	32	100.00	100.00	2.7	208245	6	1	3.40	1.5	13	2	11
VB02	2	10	9	100.00	100.00	1.1	12169	2	1	1.92	0.1	5	4	4
VB03	13	6	24	100.00	100.00	2.1	7334	9	1	2.91	0.4	34	4	9
VB04	20	6	17	100.00	100.00	1.7	32	11	1	2.47	0.3	38	4	4
VB05	17	5	9	100.00	100.00	1.9	1741	9	1	2.58	0.3	46	20	12
VB06	16	6	6	100.00	100.00	1.2	488	1	1	1.92	0.1	4	0	4
VB07	18	5	9	100.00	100.00	2.1	7956	4	1	2.80	0.3	34	15	36
VB09	13	6	14	100.00	100.00	2.5	7829	10	1	3.24	0.5	60	6	30
VB10	6	5	2	100.00	100.00	1.1	10	2	1	1.92	0.1	8	4	2
VB11	3	4	2	100.00	100.00	1.2	32	2	1	1.97	0.2	8	4	4
VB12	9	5	9	100.00	100.00	1.2	10	3	1	1.98	0.1	20	0	10
VB13	3	10	6	100.00	100.00	1.2	19	4	1	1.92	0.1	24	0	6
VB14	6	8	16	100.00	100.00	1.6	61096	5	1	2.36	0.3	20	0	11
VB15	12	5	10	100.00	100.00	1.3	61815	7	1	2.03	0.1	32	0	2
VB16	4	12	12	100.00	100.00	1.4	712879	4	1	2.14	0.2	25	0	7
VB17	12	3	7	100.00	100.00	1.3	63740	5	1	1.97	0.2	26	0	4
VB18	8	4	5	100.00	100.00	1.6	5057	5	1	2.30	0.3	28	15	14
VB19	13	6	24	100.00	100.00	1.3	36	5	1	2.08	0.1	20	1	9
VB20	6	6	1	100.00	100.00	1.2	41	3	1	1.98	0.1	12	14	6
VB21	3	5	4	100.00	100.00	1.1	5	2	1	1.92	0.1	16	1	13
VB22	4	4	4	[255]	[0.0]	0.0	0	0	1	0.22	0.1	0	0	0
VB23	3	3	2	100.00	100.00	1.1	3364	5	1	1.92	0.2	18	14	10
VB24	15	4	6	100.00	100.00	1.5	1247	5	1	2.31	0.2	26	0	9
VB25	2	10	4	100.00	100.00	1.3	8791	2	1	2.04	0.2	15	4	11
VB26	5	4	7	100.00	100.00	1.5	12119	4	1	2.20	0.3	26	12	24
VB27	5	3	1	100.00	100.00	1.3	6105	5	1	2.03	0.2	16	26	8
VB28	5	10	4	100.00	100.00	1.9	818	10	1	2.69	0.3	34	32	45
VB29	14	5	11	100.00	100.00	1.5	18	5	1	2.20	0.2	30	0	0
VB30	5	6	3	100.00	100.00	1.4	9	3	1	2.15	0.2	24	18	12
VB31	7	12	16	100.00	100.00	1.6	14	5	1	2.31	0.3	30	0	0
VB32	11	6	9	100.00	100.00	1.4	11895	8	1	2.20	0.3	44	7	9
VB33	7	6	8	100.00	100.00	1.3	1291	1	1	2.04	0.2	10	4	9
VB34	4	7	6	100.00	100.00	1.6	6903	6	1	2.31	0.3	33	8	17
VB35	4	4	1	100.00	100.00	1.1	11	3	1	1.92	0.2	12	16	6

Tabla 7.7: Problemas de la librería de Hurkens.

7.4 Tablas multi-dimensionales

Extendemos ahora el modelo de Programación Entera anterior al caso de tablas con 3 o más índices. Consideremos una tabla k-dimensional de dimensión $NENT := (N_1 + 1) \times (N_2 + 1) \times \ldots \times (N_k + 1)$, en la que algunas líneas se corresponden con totales marginales. Para simplificar la notación, usaremos un *singular* índice j para referirnos a cada elemento a_j de la tabla, esto es,

$$a = [a_j : j = 1, \ldots, NENT],$$

se presenta ahora como un vector en el espacio real $\mathbb{R}^{NENT}$. El sistema lineal de ecuaciones (7.1) ahora toma la forma compacta $Ma = 0$, es decir,

$$\sum_{i=1}^{NENT} m_{ij} a_j = 0 \quad (i = 1, \ldots, NEQ), \tag{7.13}$$

donde NEQ es el número total de totales marginales especificados, y $m_{ij} \in \{-1, 0, +1\}$ están apropiadamente definidos. Para $k = 2$ tenemos que (7.13) y (7.1) coinciden, luego podemos introducir la representación de red del sistema. Desgraciadamente, esto no se mantiene válido para $k \geq 3$. No obstante, podemos aún obtener un modelo de Programación Entera 0-1 para nuestro problema, según se indica a continuación.

Sea PS el conjunto de elementos j_0 que tienen que ser suprimidos dentro de sus correspondientes niveles de protección, digamos $[a_{j_0} - LPL_{j_0}, a_{j_0} + ULP_{j_0}]$, donde $[a_j]$ es la solución *nominal* para (7.13). Cualquier solución $\tilde{a}$ para el sistema (7.13) se considera *factible* si

$$a_j - LPL_j \leq \tilde{a}_j \leq a_j + ULP_j \quad (i = 1, \ldots, NENT), \tag{7.14}$$

donde lb_j y ub_j son valores dados no-negativos (por ejemplo, 1 y $+\infty$, respectivamente).

Sea $x_j \in \{0, 1\}$, $j = 1, \ldots, NENT$, una variable decisional asumiendo el valor 1 si el elemento j se suprime. Entonces el problema que tratamos puede formularse como sigue:

$$\min \sum_{j=1}^{NENT} w_j x_j \tag{7.15}$$

sujeto a

$$0 \leq x_j \leq 1 \text{ entero} \qquad \text{para todo } j = 1, \ldots, NENT, \tag{7.16}$$

$$x_{j_0} = 1 \qquad \text{para todo } j_0 \in PS \tag{7.17}$$

y, para cada $j_0 \in PS$ tenemos

$$\max \left\{ \begin{array}{ll} \tilde{a}_{j_0}: & (7.13)-(7.14) \\ & \tilde{a}_j = a_j, \text{ para todo } j \text{ tal que } x_j = 0 \end{array} \right\} \geq a_{j_0} + ULP_{j_0} \tag{7.18}$$

$$\min \left\{ \begin{array}{ll} \tilde{a}_{j_0}: & (7.13)-(7.14) \\ & \tilde{a}_j = a_j, \text{ para todo } j \text{ tal que } x_j = 0 \end{array} \right\} \geq a_{j_0} - LPL_{j_0} \tag{7.19}$$

Dada una solución x^* cumpliendo (7.16)–(7.17), entonces las condiciones (7.18)–(7.19) pueden fácilmente ser contrastadas resolviendo 2 problemas de Programación Lineal para cada $j_0 \in PS$. A fin de tener un modelo de Programación Lineal Entera, sin embargo, las condiciones (7.18)–(7.19) deberán ser reemplazadas por inecuaciones y/o ecuaciones lineales. En el caso de tablas bi-dimensionales, obtenemos este resultado usando la interpretación de redes de los problemas (7.18) y (7.19), y explotando la caracterización de flujos máximos en términos de cortes de capacidad mínima. En el caso más general, la misma idea se alcanza mediante la dualidad de la Programación Lineal, aplicada también a (7.18) y (7.19), pero según el esquema conocido como *Descomposición de Benders*. A continuación describimos cómo se realiza sobre el caso concreto de exigir sólo que se verifique (7.18) sobre el elemento j_0. La misma construcción puede aplicarse para considerar (7.19).

A fin de paralelizar la derivación con el caso de tablas bi-dimensionales, expresemos la restricción (7.18) en términos de *variaciones* positivas/negativas con respecto al valor nominal y sea

$$\tilde{a}_j = a_j + \delta_j^+ - \delta_j^- \text{ para todo } j = 1, \ldots, NENT,$$

donde δ_j^+, δ_j^- son ambos no-negativos. Entonces (7.18) se mantiene si y sólo si el siguiente sistema admite una solución (recordemos que $x_j^* \in \{0,1\}$ son valores previamente *dados*):

$$\delta_j^+ - \delta_j^- \geq ULP_{j_0}, \tag{7.20}$$

$$M(\delta^+ - \delta^-) = 0, \tag{7.21}$$

$$0 \leq \delta_j^+ \leq ub_j \; x_j^* \qquad j = 1, \ldots, NENT, \tag{7.22}$$

$$0 \leq \delta_j^- \leq lb_j \; x_j^* \qquad j = 1, \ldots, NENT. \tag{7.23}$$

Las restricciones (7.21) estipulan que $\tilde{a} = a + \delta^+ - \delta^-$ satisface (7.13), mientras (7.22)–(7.23) imponen las restricciones de cotas inferior/superior junto con la condición $\delta_j^+ = \delta_j^- = 0$ para todo j tal que $x_j^* = 0$. Para simplificar la notación, sea $c^T(\delta^+ - \delta^-) \geq ULP_{j_0}$ la representación de (7.20) donde $c_j = 1$ para $j = j_0$, y $c_j = 0$ para $j \neq j_0$. Obsérvese que (7.21)–(7.23) admite la solución factible $\delta^+ = \delta^- = 0$. Por tanto (7.20)–(7.23) es consistente si y sólo si (7.20) es una desigualdad válida para el polítopo definido por (7.21)–(7.23). Mediante la Dualidad en Programación Lineal (en concreto, el Lema de Farkas), esto a su vez es equivalente a la existencia de $u \in I\!R^{NEQ}$ y $\alpha, \beta \in I\!R^{NENT}$ tales que

$$\begin{cases} \alpha, \beta \geq 0 \\ c^T - u^T M - \alpha^T \leq 0^T \\ -c^T + u^T M - \beta^T \leq 0^T \end{cases} \tag{7.24}$$

y

$$\sum_{j=1}^{NENT} (\alpha_j \; ub_j \; x_j^* + \beta_j \; lb_j \; x_j^*) \geq ULP_{j_0}. \tag{7.25}$$

Claramente, uno puede sin pérdida de generalidad considerar sólo las soluciones básicas (*vértices*) de (7.24).

Nótese que, para cada solución (básica) de (7.24), uno tiene una desigualdad válida (7.25), que es una desigualdad lineal en las variables x_j^*. Se sigue que (7.18) puede sustituirse por un número exponencial de restricciones (7.25) asociadas con todas las posibles soluciones básicas (vértices) de (7.24). Para cualquier solución (posiblemente fraccionaria) x^*, el más violado de tales cortes puede obtenerse resolviendo el Problema Lineal

$$\begin{cases} \max & \delta_{j_0}^+ - \delta_{j_0}^- \\ & M(\delta^+ - \delta^-) = 0 \\ & 0 \leq \delta_j^+ \leq ub_j \; x_j^* \quad j = 1, \ldots, NENT \\ & 0 \leq \delta_j^- \leq lb_j \; x_j^* \quad j = 1, \ldots, NENT \end{cases}$$

y tomando la solución óptima dual asociada (u^*, α^*, β^*) en caso de que el máximo sea estrictamente menor que ULP_{j_0}. Obsérvese que este procedimiento de separación de cortes puede aplicarse *para cualquier* $x^* \in [0,1]^{NENT}$ dado, y no sólo para vectores enteros 0-1. Esto resulta una nota fundamental, que extiende el procedimiento general de la Descomposición de Benders.

7.5 Conclusiones

En el presente capítulo consideramos el problema (CSP) de suprimir celdas (secundarias) en una tabla estadística (con totales marginales) de manera que se garantice la seguridad y privacidad de la información individual en ciertas celdas (primarias). Se requiere que, en la tabla final, los valores posibles para las celdas primarias satisfagan ciertos niveles de protección prefijados. Cada posible celda secundaria tiene un peso de supresión, e interesa que, en la tabla final, el peso total de las celdas secundarias resulte mínimo. Hemos estudiado fundamentalmente el caso bi-dimensional, para el que se propone un algoritmo de

ramificación y corte cuyos resultados experimentales superan ampliamente los publicados por otros autores. Este trabajo abre a su vez interesantes cuestiones como el de encontrar nuevas desigualdades, o el de fortalecer los coeficientes (*lifting*) de las desigualdades aquí propuestas —por ejemplo, las desigualdades puentes (7.9), aún induciendo facetas para el polítopo del BSP, podrían en algunos casos ser mejorables para el CSP debido al efecto de las restricciones de capacidad (7.7)-(7.8)—.

Capítulo 8

Problema de la Selección de Indices en Bases de Datos

Introducimos una generalización del estándar Problema de Localización sin Capacidades, en el que los clientes pueden ser servidos no sólo por una única planta sino también por subconjuntos de plantas. El problema se denomina *Problema de Localización sin Capacidades Generalizado* (GUFLP para abreviar), y fue inspirado por el Problema de la Selección de Indices en el diseño físico de bases de datos. Formulamos GUFLP como un problema de empaquetamiento de conjuntos *Set Packing Problem*, demostrando que nuestro modelo contiene todas las desigualdades peña (*clique*), que son un número polinomial. Además, describimos un procedimiento de separación exacto para las desigualdades de ciclo impar, basado en la particular estructura de este problema. Estos resultados se utilizan en un algoritmo de ramificación y corte para la resolución exacta del GUFLP. Se dan finalmente resultados computacionales de éste sobre dos familias distintas de ejemplos test.

8.1 Introducción

El *Problema de Localización sin Capacidades* (UFLP) puede formularse como sigue. Consideremos un conjunto potencial de *plantas* y un conjunto de *clientes*. Cada planta puede ser abierta pagando un *coste fijo*, o no abierta sin coste alguno; cada cliente debe ser servido por una planta abierta, lo que conlleva el pago de un *coste de transporte*. El objetivo del problema es decidir las plantas que se deben abrir de manera que se minimice la suma total de los costos fijos más los costos de transporte. Se conoce que este problema pertenece a la clase de los problemas $\mathcal{NP}$-difícil en el sentido fuerte. Además, tiene muy diversas aplicaciones prácticas, y ha sido ampliamente estudiado en los últimos 30 años; véase Krarup y Pruzan [KP83] y Cornuejols, Nemhauser y Wolsey [CNW91] como fuentes

de trabajos. En una formulación equivalente del UFLP, cada planta puede también ser abierta bajo un coste fijo, pero cada cliente ha de ser servido por *no más de* una planta abierta, conllevando un oportuno *beneficio de transporte*. El objetivo es ahora abrir plantas de manera que se maximice la suma de los beneficios de transporte menos la suma de los costes fijos. Esta formulación alternativa ha generado diversos resultados poliédricos para el UFLP; véase Guignard [G80], Cornuejols y Thizy [CT82a], Cho, Johnson, Padberg y Rao [CJPR83], y Cho, Padberg y Rao [CPR83].

Una interesante aplicación de UFLP aparece cuando se afronta el llamado *Problema de Selección de Indices* (ISP), quien representa una fase del diseño físico de las bases de datos. El ISP se puede describir brevemente como sigue (véase Finkelstein, Schkolnick y Tiberio [FST88] para una descripción más detallada). Consideremos un conjunto de *índices* potenciales y un conjunto de *preguntas* (*queries*). Se puede optar por abrir un índice, y en tal caso se precisa un determinado *tiempo de mantenimiento*. Cada pregunta tiene que ser respondida. El *tiempo de respuesta* para cada pregunta depende del conjunto de índices (abiertos) que se utilice para responderla. El ISP busca la elección de los índices que se abrirán de manera que se minimice el tiempo total de ejecución, dado por la suma de los tiempos de mantenimiento para los índices abiertos y los tiempos de respuesta para todas las preguntas. Para ciertos sistemas de gestión de bases de datos, cada pregunta puede responderse o bien no usando ningún índice, lo que implica un cierto tiempo de respuesta, o bien usando un determinado índice abierto, lo que conlleva un beneficio sobre el tiempo de respuesta anterior. En tal caso, el ISP puede formularse como un UFLP. En efecto, cada índice se corresponde con una planta, y cada pregunta se corresponde con un cliente. El costo fijo de una planta es el tiempo de mantenimiento de su correspondiente índice. El beneficio del transporte de un par cliente-planta es el beneficio del tiempo de respuesta para el correspondiente par pregunta-índice. Para más detalles, véase Caprara, Fischetti y Maio [CFM95]. Sin embargo, en la mayor parte de los sistemas avanzados de gestión de bases de datos, no sólo se pueden responder preguntas utilizando un único índice a la vez, sino que (de un modo más general) se pueden utilizar conjuntos de índices (llamados *configuraciones*). En particular, cada pregunta puede responderse o bien no usando ningún índice –con el correspondiente tiempo de respuesta–, o bien usando un conjunto de índices abiertos –con el correspondiente beneficio en el tiempo de respuesta determinado según dicho conjunto– (típicamente, solo pocos conjuntos de índices suelen en realidad ser usados para responder preguntas). En este otro caso, el ISP ya no puede ser formulado directamente como un UFLP.

Este capítulo estudia el siguiente problema, estrechamente relacionado con el UFLP. Una vez más, consideremos un conjunto de plantas potenciales y un conjunto de clientes. Además, consideremos un conjunto de *configuraciones*, cada una asociada con un subconjunto de plantas. Cada planta puede ser o bien abierta a un *costo fijo*, o bien continuar cerrada. Una configuración se dirá *activa* sólo si *todas* sus plantas asociadas son abiertas. Cada cliente puede ser servido por *no más de* una configuración activa, con un cierto *beneficio de transporte*. El objetivo es decidir las plantas que se deben abrir de manera que se

maximice la suma de los beneficios de transporte menos la suma de los costes fijos. Con más formalidad, un ejemplo del problema puede definirse como sigue. Sea $N := \{1, \ldots, n\}$ el conjunto de plantas, $M := \{1, \ldots, m\}$ el conjunto de clientes, y $P := \{1, \ldots, p\}$ el conjunto de configuraciones. Cada planta $j \in N$ tiene un costo $f_j > 0$, y para cada par cliente-configuración $(i, k) \in M \times P$ existe un beneficio $g_{ik} \geq 0$. Cada configuración $k \in P$ está asociada con un conjunto $N_k \subseteq N$, representando el subconjunto de las plantas que tienen que ser abiertas para que la configuración k sea activa. Denotemos con $P_j := \{k \in P : j \in N_k\}$ el conjunto de las configuraciones asociadas con una planta $j \in N$. Sin pérdida de generalidad, asumimos que P_j no es un conjunto trivial de P para ningún $j \in N$, es decir, que $P_j \neq \emptyset$ y $P_j \neq P$ para todo $j \in N$. En efecto, si $P_j = \emptyset$, entonces la planta j continuará cerrada en cualquier solución óptima, puesto que abrir j sólo supone pagar la cantidad f_j sin obtener ningún tipo de beneficio. Si por otra parte $P_j = P$, entonces o bien la planta j se abre en cualquier solución óptima, o bien el valor óptimo es 0. El problema entonces busca un subconjunto $S \subseteq N$ de plantas a abrir, de manera que se maximice la función objetivo $z(S) := \sum_{i \in M} \max\{0, \max\{g_{ik} : N_k \subseteq S\}\} - \sum_{j \in S} f_j$. Es fácil ver que este problema es una generalización del UFLP, el cual aparece cuando cada configuración está asociada exactamente con una planta. Por ello, el problema es $\mathcal{NP}$-difícil (en el sentido fuerte), y nos referiremos a él con el nombre de *UFLP Generalizado* (GUFLP para abreviar). El ISP es una aplicación del GUFLP, donde cada índice se corresponde con una planta, cada pregunta con un cliente, y cada posible subconjunto de índices con una configuración. Hasta nuestro conocimiento, no existe ningún trabajo precedente en la literatura sobre el GUFLP.

Este capítulo está organizado como sigue. La Sección 8.2 describe una formulación de programación lineal entera para el GUFLP, y que viene usada en la Sección 8.3 para concluir resultados poliédricos y desigualdades válidas mediante la visión del GUFLP como un caso particular del Problema del Empaquetamiento de Conjuntos *Set Packing Problem*. La Sección 8.4 trata el problema de separar las desigualdades presentadas en la Sección 8.3. La Sección 8.5 presenta algunos algoritmos heurísticos , y la Sección 8.6 describe un algoritmo de ramificación y corte. Finalmente, la Sección 8.7 presenta nuestra experiencia computacional con problemas test aleatoriamente generados, y la Sección 8.8 indica algunas conclusiones.

8.2 Modelo Matemático

A fin de proponer un modelo matemático para el GUFLP, introduciremos las siguientes variables decisionales 0-1. Para todo $j \in N$ consideremos una variable decisional y_j que toma el valor 1 si la planta j se abre, y 0 en otro caso. Para todo $i \in M$ y $k \in P$, consideremos una variable decisional x_{ik} que asume el valor 1 si el cliente i es servido por la configuración k, y 0 en otro caso. Una posible formulación como problema de

programación lineal entera (ILP) es entonces

$$\max \sum_{i\in M}\sum_{k\in P} g_{ik}x_{ik} - \sum_{j\in N} f_j y_j \tag{8.1}$$

sujeto a

$$\sum_{k\in P} x_{ik} \le 1 \qquad \text{para } i \in M, \tag{8.2}$$

$$x_{ik} \le y_j \qquad \text{para } i \in M, j \in N, k \in P_j, \tag{8.3}$$

$$0 \le x_{ik}, y_j \le 1 \qquad \text{para } i \in M, j \in N, k \in P, \tag{8.4}$$

$$x_{ik}, y_j \text{ entero} \qquad \text{para } i \in M, j \in N, k \in P. \tag{8.5}$$

Las restricciones (8.2) establecen que cada cliente puede servirse por no más de una configuración, mientras que las restricciones (8.3) aseguran que cada cliente puede servirse sólo usando configuraciones activas.

Como para el UFLP, es fácil ver que las exigencias de integrabilidad sobre las variables x puede relajarse sin afectar a la solución En efecto, supongamos que (x, y) es una solución de (8.2)–(8.4), con y_j entero para todo $j \in N$. Para todo $i \in M$, denotaremos por $k(i)$ la configuración activa (con respecto a las plantas abiertas en y) con máximo beneficio de transporte para el cliente i, esto es, $g_{i,k(i)} = \max\{g_{ik} : y_j = 1 \text{ para todo } j \in N_k\}$ (si no hay ninguna configuración activa, entonces hacemos $k(i) := 0$ para todo $i \in M$). Entonces la solución 0-1 (x', y) definida por $x'_{ik} := 1$ si $k = k(i)$ y por $x'_{ik} := 0$ en otro caso, para todo $i \in M$ y $k \in P$, tiene función objetivo de valor no menor que (x, y). En otras palabras, y_j son las variables estratégicas, y una vez que se tiene el vector entero y, es fácil determinar las variables 0-1 x de manera que se maximice la función objetivo para el y dado.

Cuando cada configuración está asociada con sólo una planta, entonces la formulación anterior se corresponde con la llamada *formulación ILP fuerte* para el UFLP. Sin embargo, la formulación (8.1)–(8.5) puede fácilmente "reforzarse" para el GUFLP como demuestra el siguiente teorema.

Teorema 8.2.1 *La formulación ILP obtenida sustituyendo las restricciones (8.3) con*

$$\sum_{k\in P_j} x_{ik} \le y_j \qquad \textit{para } i \in M, j \in N \tag{8.6}$$

es válida para el GUFLP.

Demostración. Consideremos un par cliente-planta $(i, j) \in M \times N$ dado. Puesto que $P_j \subseteq P$, de la desigualdad (8.2) se sigue $\sum_{k\in P_j} x_{ik} \le 1$. Además, según la restricción (8.3), $x_{ik} > 0$ para algún $k \in P_j$ implica $y_j = 1$, de lo que se sigue la validez de la desigualdad (8.6). □

El número de restricciones (8.6) es mn, y por tanto es menor que el número de restricciones (8.3). Así, para un número dado de plantas y clientes, la nueva formulación tiene $m+mn$ restricciones al igual que la formulación ILP fuerte del UFLP. Además, las restricciones (8.6) son más fuertes que las (8.3), en el sentido de que cada miembro de (8.6) puede obtenerse elevando los coeficientes de algún miembro de (8.3). En realidad, demostraremos en la Sección 8.3 que las restricciones (8.6) (as igual que sucede con las restricciones (8.2)) definen facetas para el polítopo definido por la envolvente convexa de las soluciones factibles del GUFLP. Por este motivo llamaremos *formulación ILP fuerte* para el GUFLP al modelo (8.1),(8.2),(8.6),(8.4),(8.5).

La literatura muestra el gran esfuerzo que se ha hecho para desarrollar heurísticas duales y algoritmos exactos para resolver la relajación lineal de la formulación ILP fuerte del UFLP; véase por ejemplo Bilde y Krarup [BK77], Erlenkotter [E78], Cornuejols y Thizy [CT82b], Conn y Cornuejols [CC90]. Típicamente, esta relajación tiene una solución entera, o al menos lleva a una cota superior bastante próxima al óptimo. El mismo comportamiento no se ha observado, sin embargo, en alguno ejemplos GUFLP de interés, para los cuales el valor óptimo de la relajación lineal de (8.1),(8.2),(8.6),(8.4) está bastante lejos del valor solución óptima (véase la Sección 8.7). Por ello se precisan nuevas desigualdades válidas para fortalecer esta relajación lineal, tan útil para certificar la calidad de un heurístico dado, como para desarrollar algoritmos exactos para el GUFLP.

8.3 El GUFLP como Problema de Empaquetamiento de Conjuntos

Como sucede con el UFLP, el GUFLP puede verse como un caso particular de *Problema de Empaquetamiento de Conjuntos* (*Set Packing Problem*, SPP), el cual se define como

$$\max\left\{cz : Az \leq \mathbf{1}, z \in \{0,1\}^h\right\},$$

donde h y l son enteros positivos, $c \in I\!R^h$, A es una $l \times h$ matriz 0-1, y $\mathbf{1}$ es un l-vector de 1's. Enviamos al lector interesado a Balas y Padberg [BP76] para una fuente de trabajos sobre el SPP. En efecto, introduciendo las variables complementarias $\bar{y}_j := 1 - y_j$ para $j \in N$ ($\bar{y}_j = 0$ si y sólo si se abre la planta j), la formulación ILP fuerte del GUFLP se lee ahora como

$$\max \sum_{i\in M}\sum_{k\in P} g_{ik}x_{ik} + \sum_{j\in N} f_j\bar{y}_j - \sum_{j\in N} f_j \tag{8.7}$$

sujeto a

$$\sum_{k\in P} x_{ik} \leq 1 \qquad \text{para } i \in M, \tag{8.8}$$

$$\sum_{k\in P_j} x_{ik} + \bar{y}_j \leq 1 \qquad \text{para } i \in M, j \in N, \tag{8.9}$$

$$x_{ik}, \bar{y}_j \geq 0 \qquad \text{para } i \in M, j \in N, k \in P, \tag{8.10}$$

$$\bar{y}_j \text{ entero} \qquad \text{para } j \in N. \tag{8.11}$$

En esta sección se demuestra que (8.8) y (8.9) son desigualdades fuertes (definen facetas) para la descripción matemática del GFULP. Además, se mejora la correspondiente relajación lineal (LPR) añadiendo una nueva familia de restricciones. Para ambos propósitos, adaptamos conocidos resultados del SPP al caso particular del GUFLP.

Necesitamos una formulación de teoría de grafos de nuestro problema, por lo que primero introduciremos algunas definiciones básicas. Consideremos un grafo no-dirigido $G = (V, E)$, con $V := \{1, \ldots, v\}$ su conjunto de nodos, y $E \subseteq \{(i, j) : i, j \in V, i \neq j\}$ su conjunto de arcos. El *arco-complemento* de G es el grafo $\bar{G} = (V, \bar{E})$ donde $\bar{E} := \{(i, j) : i, j \in V, (i, j) \notin E\}$. El *subgrafo de G inducido por $W \subseteq V$* es el grafo $H = (W, E(W))$ donde $E(W) := \{(i, j) : i, j \in W, (i, j) \in E\}$. La *longitud* de un subgrafo inducido por W se define como la cardinalidad de su conjunto de nodos $|W|$, que denotaremos por w. En algunos caso, por conveniencia en la notación, asumiremos implícitamente que los nodos de W están renombrados según una secuencia particular $W := \{1, \ldots, w\}$. Entonces dos nodos $i, j \in W$ se llaman *consecutivos* en W si $|j - i| = 1$ o $|j - i| = w - 1$. Cuando denotamos un nodo de W, digamos, por i, implícitamente asumiremos que i se ha tomado módulo w. Una *peña* (*clique*) en G se define como un subgrafo completo H de G; una peña se llamará *peña maximal* si ella no es un subgrafo proprio de otra peña. Un *hueco* (*hole*) en G es un ciclo sin cuerdas de longitud ≥ 5; un hueco se llama *hueco impar* si su longitud es impar. Un *anti-hueco* (*anti-hole*) es el arco-complemento de un hueco. Dados dos números enteros w, k tales que $w \geq 5$ y $2 \leq k \leq \lfloor \frac{w}{2} \rfloor$, el subgrafo de G inducido por $W \subseteq V$ se llama un tejido *web*, y se denota por $B_{w,k}$, si $|W| = w$ y $E(W) = \{(i, j) : i, j \in W, k \leq |j - i| \leq w - k\}$. Nótese que para $k = 2$ un tejido es claramente un anti-hueco. Además, cuando w es impar y $k = \lfloor \frac{w}{2} \rfloor$, un tejido $B_{w,k}$ es también un hueco, ya que es un grafo conexo donde cada nodo $i \in W$ es adyacente exactamente a dos nodos $i + k$ y $i + k + 1$ de W. Un *anti-tejido* (*anti-web*) $\bar{B}_{w,k}$ es el arco-complemento de un tejido. Un tejido o anti-tejido se dice que es *primo* si k y w son números primos entre sí.

Un conjunto de nodos $I \subseteq V$ se dirá que es un *conjunto independiente* si $(i, j) \notin E$ para todo $i, j \in I$, es decir, $E(I) = \emptyset$. Dada una $l \times h$ matrix 0-1 $A = [a_{ij}]$, el *grafo de intersección* de A se define como $G_A = (V_A, E_A)$, $V_A := \{1, \ldots, h\}$, $E_A := \{(j, k) : j, k \in V_A, \sum_{h=1}^{l} a_{hj} a_{hk} > 0\}$. Dado un peso c_j asociado con la j-ésima columna de A para todo $j \in V_A$, entonces resolver el SPP $\max\{cz : Az \leq \mathbf{1}, z \in \{0, 1\}^h\}$ equivale a encontrar un conjunto independiente de máximo peso en G_A, donde cada nodo $j \in V_A$ tiene asociado

el peso c_j. Para cada fila i de A, el conjunto $\{j : a_{ij} = 1\} \subseteq V_A$ induce una peña en G_A. Si esta peña es maximal, la correspondiente desigualdad peña $\sum_{j \in V_A} a_{ij} z_j \leq 1$ define una faceta del polítopo asociado con el SPP, y que se define como conv $\{z \in \{0,1\}^h : Az \leq \mathbf{1}\}$ (véase Padberg [P73]).

A partir de (8.7)–(8.11) y de la anterior discusión, el GUFLP puede verse como el problema de encontrar un conjunto independiente de máximo peso en el grafo intersección $G_I = (V_I, E_I)$ asociado con la matriz 0-1 I de la formulación (8.8)–(8.9). V_I contiene un nodo para cada variable decisional. Dos nodos resultan adyacentes en G_I si y sólo si las dos correspondientes variables aparecen con coeficientes 1 en al menos una misma restricción del sistema (8.8)–(8.9) (recordemos que los arcos representan incompatibilidades entre las variables). Usaremos el término x-nodo y $\bar{y}$-nodo para indicar un nodo asociado con una variable x y con una variable $\bar{y}$, respectivamente. Además, indicamos con $X \subseteq V_I$ el conjunto de todos los x-nodos, y con $\bar{Y} \subseteq V_I$ el conjunto de todos los $\bar{y}$-nodos. Si no se ocasiona confusión, usaremos la notación x_{ik} y $\bar{y}_j$ para denotar los nodos.

Ahora establecemos algunas propiedades de G_I, que son una extensión inmediata de propiedades del grafo de intersección asociado con la formulación SPP del UFLP.

Propiedad 8.3.1 *$(x_{ik}, x_{hl}) \in E_I$ si y sólo si $h = i$, es decir, dos x-nodos son adyacentes si y sólo si las correspondientes variables están asociadas con el mismo cliente.*

Propiedad 8.3.2 *$(x_{ik}, \bar{y}_j) \in E_I$ si y sólo si $j \in N_k$ (o equivalentemente $k \in P_j$), es decir, un x-nodo y un $\bar{y}$-nodo son adyacentes si y sólo si la correspondiente variable x está asociada con una configuración que contiene la planta asociada con la variable $\bar{y}$.*

Propiedad 8.3.3 *$(\bar{y}_j, \bar{y}_k) \notin E_I$ para todo $j, k \in N$, es decir, no hay ningún par de $\bar{y}$-nodos adyacentes.*

Por la Propiedad 8.3.1 tenemos que cualquier subgrafo de G_I inducido por un subconjunto de X está dividido en peñas disjuntas. Por la Propiedad 8.3.3 tenemos que cualquier subgrafo de G_I inducido por un subconjunto de $\bar{Y}$ es una colección de nodos aislados.

Un primer resultado se refiere a la relajación lineal (8.7)–(8.10).

Teorema 8.3.1 *Si P_j no es un subconjunto trivial de P para todo $j \in N$, esto es, $P_j \neq \emptyset$ y $P_j \neq P$, las restricciones (8.8) y (8.9) se corresponden con* todas *las desigualdades peñas maximales de G, y por tanto definen facetas para el GUFLP.*

Demostración. Primero demostramos que cada peña correspondiente a una desigualdad (8.8) y (8.9) es maximal. Consideremos la restricción (8.8) correspondiente al cliente i,

esto es $\sum_{k\in P} x_{ik} \leq 1$, y sea $D := \{x_{ik} : k \in P\}$ el correspondiente conjunto de nodos de la peña. Entonces $V_I \setminus D = \bar{Y} \cup (X \setminus D)$. Por la Propiedad 8.3.1, ningún nodo en $X \setminus D$ es adyacente a un nodo en D. Además, puesto que $P_j \neq P$ para todo $j \in N$, por la Propiedad 8.3.2, ningún nodo en $\bar{Y}$ es adyacente a todos los nodo de D. Entonces D induce una peña maximal en G_I. Consideremos ahora la restricción (8.9) correspondiente al cliente i y a la planta j, esto es $\sum_{k\in P_j} x_{ik} + \bar{y}_j \leq 1$, y sea $F := \{\bar{y}_j\} \cup \{x_{ik} : k \in P_j\}$ el correspondiente conjunto de nodos de la peña. Nótese que $\{x_{ik} : k \in P_j\} \neq \emptyset$, ya que se ha asumido que $P_j \neq \emptyset$. Tenemos $V_I \setminus F = (\bar{Y} \setminus \{\bar{y}_j\}) \cup \{x_{il} : l \in P \setminus P_j\} \cup \{x_{hl} : h \in M \setminus \{i\}, l \in P\}$. Por la Propiedad 8.3.1, ningún nodo en $\{x_{hl} : h \in M \setminus \{i\}, l \in P\}$ es adyacente a ningún x-nodo en F. Por la Propiedad 8.3.3, ningún nodo en $\bar{Y} \setminus \{\bar{y}_j\}$ es adyacente a $\bar{y}_j$. Finalmente, por la Propiedad 8.3.2, ningún nodo en $\{x_{il} : l \in P \setminus P_j\}$ es adyacente a $\bar{y}_j$. Por ello F induce una peña maximal.

Mostramos ahora que cada peña de G_I está contenida en al menos una de las peñas arriba consideradas. Sea una peña de G_I y sea W su conjunto de nodos. Si W sólo contiene x-nodos, entonces ella está contenida en una peña asociada con una asociada a alguna restricción (8.8), por la Propiedad 8.3.1. En otro caso, por la Propiedad 8.3.3, W contiene exactamente un $\bar{y}$-nodo, y por la Propiedad 8.3.2 está contenida en una peña asociada con una restricción (8.9). □

El Teorema 8.3.1 da una descripción completa de las desigualdades que proceden de peñas maximales para G_I. Al igual que para la formulación ILP fuerte del UFLP, la formulación (8.7)–(8.11) contiene todas las desigualdades peña maximales, siendo en número $m+mn$. Para el SPP, además de las peñas (maximales), las siguientes estructuras simples de subgrafos también producen desigualdades válidas: huecos impares, anti-huecos impares, tejidos primos, y anti-tejidos primos (véase Balas y Padberg [BP76]).

Ahora afrontaremos las desigualdades huecos impares para el GUFLP. En el UFLP, todo hueco es tal que cada terna de nodos consecutivos contiene exactamente dos x-nodos y un $\bar{y}$-nodo; el menor de tales posibles huecos impares tiene longitud 9, conteniendo 3 plantas y 3 clientes. Aquí tenemos la siguiente propiedad para los huecos en GUFLP, derivada de las Propiedades 8.3.1 y 8.3.3.

Propiedad 8.3.4 *En todo hueco de G_I, cada terna de nodos consecutivos contiene uno o dos x-nodos, y cada par de nodos consecutivos contiene al menos un x-nodo. Por lo tanto, cada hueco de longitud w contiene como mínimo $\lceil \frac{w}{3} \rceil$ y como máximo $\lfloor \frac{w}{2} \rfloor$ $\bar{y}$-nodos.*

El menor hueco impar posible tiene longitud 5, como sucede en el caso de $m = n = 2, p = 3$, $N_1 = \{1\}, N_2 = \{2\}, N_3 = \{1, 2\}$, donde $W = \{\bar{y}_1, x_{11}, x_{12}, \bar{y}_2, x_{23}\}$, véase la Figura 8.3.

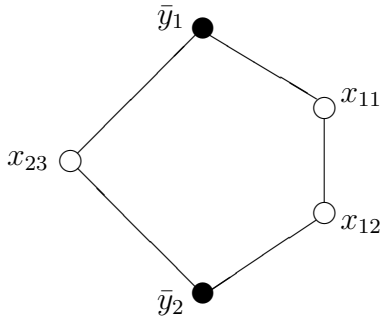


Figura 8.1: Un hueco impar en G_I.

Dado un hueco impar en G_I inducido por W, la asociada desigualdad

$$\sum_{x_{ik}\in W} x_{ik} + \sum_{\bar{y}_j\in W} \bar{y}_j \leq \left\lfloor \frac{w}{2} \right\rfloor \tag{8.12}$$

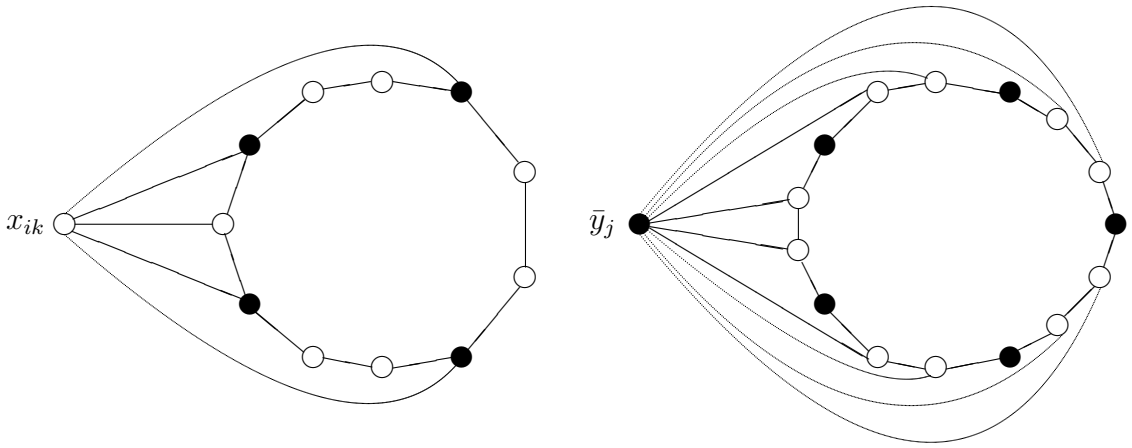
es válida para GUFLP. Además, la desigualdad (8.12) define una faceta para el subproblema teniendo fijas a 0 todas las variables asociadas con nodos en $V_I \setminus W$ fijas a 0 (véase Padberg [P73]). Para el problema original, (8.12) puede elevarse a una desigualdad del tipo

$$\sum_{x_{ik}\in W} x_{ik} + \sum_{\bar{y}_j\in W} \bar{y}_j + \sum_{x_{ik}\in X\setminus W} \alpha_{ik} x_{ik} + \sum_{\bar{y}_j\in \bar{Y}\setminus W} \beta_j \bar{y}_j \leq \left\lfloor \frac{w}{2} \right\rfloor \tag{8.13}$$

encontrando no-negativos α_{ik} y β_j (llamados *coeficientes de elevación*). Para un SPP general, cada coeficiente de elevación de una desigualdad hueco impar está en $[0, \lfloor \frac{w}{2} \rfloor]$ (véase Padberg [P73]), y para el UFLP está en $[0, 1]$ (véase Cornuejols y Thizy [CT82a]).

Teorema 8.3.2 *Dada un hueco impar en G_I inducido por W y la correspondiente desigualdad (8.12), entonces los coeficientes de elevación que mantienen a (8.13) válida para el GUFLP están en $[0, \lfloor \frac{w}{2} \rfloor - \lceil \frac{w}{3} \rceil + 1]$ para las variables x y en $[0, \lfloor \frac{w}{2} \rfloor - \lceil \frac{w}{3} \rceil]$ para las variables $\bar{y}$.*

Demostración. Sea $H = (W, F)$ un hueco impar. Para cada $j \in V_I \setminus W$, el máximo valor del correspondiente coeficiente θ_j en (8.13) está dado por $\lfloor \frac{w}{2} \rfloor - \gamma(H \setminus \delta(j))$, donde $\delta(j) := \{i \in W : (i, j) \in E_I\}$ es el conjunto de vecinos de j en W, $\gamma(G)$ denota el número de estabilidad de un grafo G (es decir, el máximo cardinal de un conjunto independiente en G), y $H \setminus \delta(j)$ denota el subconjunto de H inducido por $W \setminus \delta(j)$. Si j es un x-nodo, por las Propiedades 8.3.1 y 8.3.2, éste puede ser adyacente a todos los $\bar{y}$-nodos en W, más como mucho 2 x-nodos adyacentes. Entonces, por la Propiedad 8.3.4, $\gamma(H \setminus \delta(j)) \geq |W \cap \bar{Y}| - 1 \geq \lceil \frac{w}{3} \rceil - 1$, y $\theta_j \leq \lfloor \frac{w}{2} \rfloor - \lceil \frac{w}{3} \rceil + 1$. Si por otra parte j es un $\bar{y}$-nodo, por la Propiedad 8.3.3 no es adyacente a ningún $\bar{y}$-nodo en W; mientras que por la Propiedad 8.3.2 puede ser adyacente a cada x-nodo en W. Por tanto, por la Propiedad 8.3.4, $\gamma(H \setminus \delta(j)) \geq |W \cap \bar{Y}| \geq \lceil \frac{w}{3} \rceil$, y $\theta_j \leq \lfloor \frac{w}{2} \rfloor - \lceil \frac{w}{3} \rceil$. □



(a) $\alpha_{ik} = 2$ (b) $\beta_j = 2$

Figura 8.2: Coeficientes de elevación para las desigualdades hueco impar (los círculos en negro denotan $\bar{y}$-nodos, los círculos en blanco denotan x-nodos).

Las cotas superiores sobre los coeficientes de elevación dadas por el teorema anterior están ajustadas tanto para las variables x como para las variables y, para todo impar $w \geq 5$. La Figura 8.3.1 muestra dos ejemplos de coeficientes de elevación > 1.

Finalmente demostramos que para el GUFLP (al igual que sucede con el UFLP, véase Cornuejols y Thizy [CT82a]) el grafo de intersección G_I no contiene anti-huecos (impares), ni tejidos (primos), ni anti-tejidos (primos), y por ello las correspondientes desigualdades del SPP no son útiles para un posterior fortalecimiento de la relajación lineal de nuestro modelo.

Teorema 8.3.3 *G_I no contiene ningún anti-hueco de longitud > 5 como un subgrafo inducido.*

Demostración. Supongamos que G_I contiene un anti-hueco $\bar{H} = (W, F)$, con $w > 5$. Entonces W contiene $\bar{y}$-nodos; de no ser así, por la Propiedad 8.3.1, contendría una familia de peñas disjuntas. Además, W contiene como mucho 2 $\bar{y}$-nodos (consecutivos), debido a la Propiedad 8.3.3. Puesto que $w > 5$, existe una terna $\{i, i+1, i+3\}$ de x-nodos en W tal que $(i, i+3), (i+1, i+3) \in F$. Por la Propiedad 8.3.1, esto implica que $(i, i+1) \in F$, y por ello $\bar{H}$ no es un anti-hueco. □

Teorema 8.3.4 *Para cada par de números enteros w, k tales que $w \geq 5$ y $3 \leq k \leq \lfloor \frac{w}{2} \rfloor - 1$, G_I no contiene ningún tejido $B_{w,k}$ o anti-tejido $\bar{B}_{w,k}$ como un subgrafo inducido.*

Demostración. Supongamos que G_I contiene un tejido $B_{w,k} = (W, F)$, con $3 \leq k \leq$

$\lfloor \frac{w}{2} \rfloor - 1$. Por la Propiedad 8.3.1 W contiene $\bar{y}$-nodos. Además, por la Propiedad 8.3.3, W contiene como mucho k $\bar{y}$-nodos, y si i y j son $\bar{y}$-nodos en W, entonces $|j - i| < k$. Así, hay al menos $w - k \geq k + 2$ x-nodos consecutivos en W. Entonces existe una terna $\{i, i+1, i+k+1\}$ de x-nodos en W tales que $(i, i+k+1), (i+1, i+k+1) \in F$. Por la Propiedad 8.3.1, esto implica que $(i, i+1) \in F$, y así $B_{w,k}$ no es un tejido.

Supongamos ahora que G_I contiene un anti-tejido $\bar{B}_{w,k} = (W, F)$, $3 \leq k \leq \lfloor \frac{w}{2} \rfloor - 1$. Por la Propiedad 8.3.3 hay al menos $k-1$ (≥ 2) x-nodos entre cualquier par de $\bar{y}$-nodos en W. Entonces existe una terna $\{i, i+1, i+k\}$ de x-nodos en W tal que $(i, i+1), (i+1, i+k) \in F$. Por la Propiedad 8.3.1, esto implica que $(i, i+k) \in F$, y así $\bar{B}_{w,k}$ no es un anti-tejido. □

8.4 Procedimientos de separación

En el algoritmo de ramificación y corte de la Sección 8.6, usamos la relajación lineal (8.7)–(8.10) aumentada considerando las restricciones (8.12). Como es usual, los problemas lineares contendrán sólo un subconjunto de todas esas desigualdades, y tendremos por tanto que afrontar el siguiente *problema de separación*: dado un punto $z^* = (x^*, \bar{y}^*) \in I\!R^{(M \times P) \cup N}$ y una familia de restricciones $\mathcal{F}$ válidas para el GUFLP, encontrar una restricción en $\mathcal{F}$ violada por z^*, o demostrar que no existe ninguna. El problema de separación para las desigualdades (8.8) y (8.9) se resuelve trivialmente mediante enumeración explícita, lo que supone un tiempo de orden $O(np)$ y $O(mnp)$, respectivamente.

Brevemente describiremos el procedimiento de separación que usamos para identificar las restricciones hueco impar (8.12). Dada una solución (fraccionaria) z^* del problema lineal relajado de un SPP, se conoce que la identificación de una desigualdad hueco impar de máxima violación exige determinar un *ciclo impar de peso mínimo* en el correspondiente grafo intersección, donde cada arco (i, j) tiene el peso $1 - z_i^* - z_j^*$. Si el valor de la solución óptima de este problema es menor que 1, entonces el correspondiente hueco impar define una desigualdad de máxima violación; en otro caso no existe desigualdad hueco impar violada. El problema de determinar un ciclo impar de mínimo peso puede resolverse como un problema de camino mínimo sobre un cierto grafo bipartito particularmente construido (véase, por ejemplo, Grötschel, Lovász y Schrijver [GLS88], Página 235).

Para UFLP, se observa en Caprara y Fischetti [CF96] que la separación de las desigualdades hueco impar se corresponden con el cálculo de un ciclo impar de peso mínimo sobre un grafo teniendo sólo un nodo por cada variable (fraccionaria), obtenido a partir del grafo de intersección original mediante simples operaciones de pre-procesamiento que eliminan los x-nodos. Para el GUFLP también se mantiene la misma propiedad, tal y como mostramos a continuación. Consideremos una solución fraccionaria $(x^*, \bar{y}^*)$ de (8.7)–(8.10), y asignemos a cada arco $(i, j) \in E_I$ el peso $1 - z_i^* - z_j^*$, donde z_i y z_j son las variables asociadas con i y j, respectivamente. Introducimos el multi-grafo $H = (\bar{Y}, F)$,

donde cada arco en F tiene un signo, esto es, puede ser par o impar. Para cada para de $\bar{y}$-nodos $(\bar{y}_j, \bar{y}_l)$, F contiene un arco *impar* $(\bar{y}_j, \bar{y}_l)$ con el peso $3 - \bar{y}_j^* - \bar{y}_l^* - 2\sigma_{jl}^*$, donde $\sigma_{jl}^* := \max_{i \in M}\{x_{ik}^* + x_{ih}^* : k \in P_j, h \in P_l, k \neq h\}$. Además, para cada par de $\bar{y}$-nodos $(\bar{y}_j, \bar{y}_l)$ tales que $P_j \cap P_l \neq \emptyset$, F contiene un arco *par* $[\bar{y}_j, \bar{y}_l]$ con peso $2 - \bar{y}_j^* - \bar{y}_l^* - 2\rho_{jl}^*$, donde $\rho_{jl}^* := \max_{i \in M}\{x_{ik}^* : k \in P_j \cap P_l\}$. Un ciclo en H se llama impar si contiene un número impar de nodos impares. Debido a la Propiedad 8.3.4, cada hueco impar en G_I se corresponde con un ciclo impar en H del mismo peso. Recíprocamente, dado un ciclo impar en H, existe un ciclo impar en G_I de igual peso, y se conoce del SPP como obtener de este ciclo un hueco impar de peso no superior (véase Nemhauser y Sigismondi [NS92]). El punto clave del procedimiento de separación es entonces el cálculo de un ciclo impar de peso mínimo en H. Este problema puede resolverse como un problema de camino mínimo sobre el siguiente grafo $B = (S \cup T, C)$. Cada nodo $\bar{y}_j$ en H se corresponde con un par de nodos $s_j \in S, t_j \in T$ en B, cada arco impar $(\bar{y}_j, \bar{y}_l) \in F$ se corresponde con un par de arcos $(s_j, t_l), (s_l, t_j) \in C$, y cada arco par $[\bar{y}_j, \bar{y}_l] \in F$ se corresponde con el par de arcos $(s_j, s_l), (t_j, t_l) \in C$. Entonces un ciclo impar de peso mínimo pasando a través de $y_j \in V$ se corresponde con un camino mínimo desde s_j a t_j en B. Damos ahora una descripción en Pseudo-Pascal de nuestro procedimiento de separación.

procedure ODD_HOLE_SEP;
input: el punto $(x^*, \bar{y}^*) \in [0,1]^{(M \times P) \cup N}$;
output: una secuencia $L_1, \ldots, L_r$ de huecos impares en G_I correspondiente a una desigualdad hueco impar violada;
begin
1. construir el multi-grafo H a partir de G_I y $(x^*, \bar{y}^*)$;
2. $\bar{Y}^H := \emptyset$; $r := 0$;
3. **para todo** $\bar{y}_j \in \bar{Y}$ **do**
4. calcular un ciclo impar de peso mínimo C pasando a través de y_j y conteniendo sólo los nodos en $\bar{Y} \setminus \bar{Y}^H$, y sea c el correspondiente peso;
5. **si** $c < 1$ **entonces**
6. $\bar{Y}^H := \bar{Y}^H \cup \{\bar{y}_j\}$; $r := r + 1$;
7. construir desde C un hueco impar L_r en G_I correspondiente a una desigualdad (8.12) violada
end si
end para
end.

El Paso 1. puede implementarse para funcionar en tiempo $O(n^2 p + mn^2)$. El Paso 4. requiere un tiempo $O(n^2)$ si se utiliza el algoritmo de Dijkstra. Finalmente, el Paso 7. supone un tiempo $O(n^2)$ en el peor de los caso. Por lo tanto, la complejidad algorítmica global es de orden $O(n^3 + n^2 p + mn^2)$.

El procedimiento previo da (si existe) una desigualdad hueco impar violada de tipo (8.12). Para obtener desigualdades elevadas de tipo (8.13), usamos el siguiente método

(véase Caprara y Fischetti [CF96] para detalles). Cada arco de un hueco impar es o bien un par (x_{ik}, x_{hl}) con $i = h$, o bien un par $(x_{ik}, \bar{y}_j)$ con $k \in P_j$, y está asociado como con la restricción (8.8) para el cliente i en el primer caso, y con la restricción (8.9) para el cliente i y la planta j en el segundo caso. Puede obtenerse a partir de un hueco impar en G_I una desigualdad hueco impar elevada de tipo (8.13) aplicando la derivación de Chvátal-Gomory, combinando con coeficientes $\frac{1}{2}$ las restricciones asociadas con cada arco, y luego redondeando por defecto los coeficientes del lado izquierdo y la constante del lado derecho de la desigualdad resultante.

8.5 Algoritmo heurísticos

EL GUFLP busca un subconjunto $S \subseteq N$ de plantas que deberán abrirse, de manera que se maximice la función objetivo $z(S) := \sum_{i \in M} \max\{0, \max\{g_{ik} : N_k \subseteq S\}\} - \sum_{j \in S} f_j$. Mediante esta formulación combinatoria se pueden adaptar diversos algoritmos heurísticos del UFLP al GUFLP.

El conocido *algoritmo greedy* para el UFLP se puede adaptar como sigue. Inicialmente, sea $S := \emptyset$ y $S' := \emptyset$. En cada iteración, la cantidad $\delta_j := z(S \cup \{j\}) - z(S)$ se calcula para cada planta $j \in N \setminus S$, y $S := S \cup \{j'\}$, donde j' es tal que $\delta_{j'} = \max_{j \in N \setminus S} \delta_j$. Si $z(S) - z(S') > 0$ entonces $S' := S$ y se realiza otra iteración, en otro caso el algoritmo greedy se detiene con la solución S'. Para el UFLP se conoce que la función objetivo $z(S)$ es *submodular*, esto es, $z(S \cup \{j\}) - z(S) \leq z(R \cup \{j\}) - z(R)$ para todo $j \notin S$ y $R \subseteq S \subseteq N \setminus \{j\}$ (véase Cornuejols, Nemhauser y Wolsey [CNW91]). Ello implica que la solución S' dada por el algoritmo greedy no puede mejorarse añadiendo *ningún* subconjunto de plantas. Para el GUFLP, $z(S)$ no mantiene la propiedad de submodularidad ya que es posible tener una solución S que puede mejorarse sólo añadiendo dos o más plantas. Esto permite entonces modificar ligeramente el algoritmo anterior como sigue: si $z(S) - z(S') \leq 0$, dejamos S' sin cambios y ejecutamos otra iteración; el procedimiento se detiene cuando $S = N$, siendo la solución S'. La complejidad algorítmica global es de orden $O(mn^2p)$.

La *heurística de 2-intercambio* comienza con un conjunto arbitrario S, usualmente obtenido mediante un algoritmo heurístico previo. En cada iteración busca

1. una planta $j' \in S$ tal que $z(S \setminus \{j'\}) > z(S)$: si se encuentra tal j' entonces $S := S \setminus \{j'\}$ y se ejecuta otra iteración;

2. una planta $j'' \notin S$ tal que $z(S \cup \{j''\}) > z(S)$: si se encuentra tal j'' entonces $S := S \cup \{j''\}$ y se ejecuta otra iteración;

3. un par de plantas $(j', j''), j' \in S, j'' \notin S$, tal que $z(S \cup \{j''\} \setminus \{j'\}) > z(S)$: si se encuentra tal par (j', j'') entonces $S := S \cup \{j''\} \setminus \{j'\}$ y se ejecuta otra iteración.

El procedimiento se detiene cuando S permanece inalterado durante una iteración. La complejidad global es de orden $O(mn^3p)$. En cada paso del heurístico, el orden en el que las plantas se consideran se basa en el valor q_j ($j \in N$) que representa una estimación de la verosimilitud de tener la planta j abierta en una solución óptima.

Pueden definirse muchas posibles variantes y generalizaciones de estos dos simples algoritmos heurísticos. Nótese que el GUFLP puede re-escribirse como el problema de determinar un subconjunto $T \subseteq P$ de configuraciones que maximice la función $z'(T) := \sum_{i \in M} \max\{0, \max\{g_{ik} : k \in T\}\} - \sum_{j \in \bigcup_{k \in T} N_k} f_j$. Mediante esta nueva formulación se pueden diseñar otros algoritmos heurísticos. En particular, consideramos las variantes de las heurísticas greedy y 2-intercambio, tratando con configuraciones en lugar de con plantas.

8.6 El algoritmo de ramificación y corte

En esta sección describimos nuestro algoritmo enumerativo para la resolución exacta del GUFLP. Usaremos un esquema de ramificación y acotación, en el que las cotas superiores se calculan resolviendo una relajación lineal del problema. La relajación se fortalece iterativamente añadiendo desigualdades válidas al problema lineal del momento, según la técnica conocida como *hiperplanos de corte*. El método en su globalidad se conoce comúnmente como *algoritmo de ramificación y corte* (véase Padberg y Rinaldi [PR91] para detalles). La Figura 8.6.1 muestra un diagrama de flujo muy resumido del algoritmo. La motivación para considerar este tipo de técnica basada en hiperplanos de corte se debe fundamentalmente a la discusión que se presenta al final de la Sección 8.2.

Pre-procesamiento

Antes de comenzar propiamente el algoritmo de ramificación y corte, se aplica la siguiente técnica de reducción:

1. Fijar a 0 toda variable x con un no-positivo beneficio de transporte.
2. Eliminar todo cliente i con x_{ik} fija a 0 para todo $k \in P$.
3. Eliminar toda configuración k con x_{ik} fija a 0 para todo $i \in M$.
4. Eliminar toda configuración k tal que otra configuración $l < k$ exista con $N_k = N_l$, y poner $g_{il} := \max\{g_{ik}, g_{il}\}$ para todo $i \in M$.
5. Eliminar toda planta j con $P_j = \emptyset$.

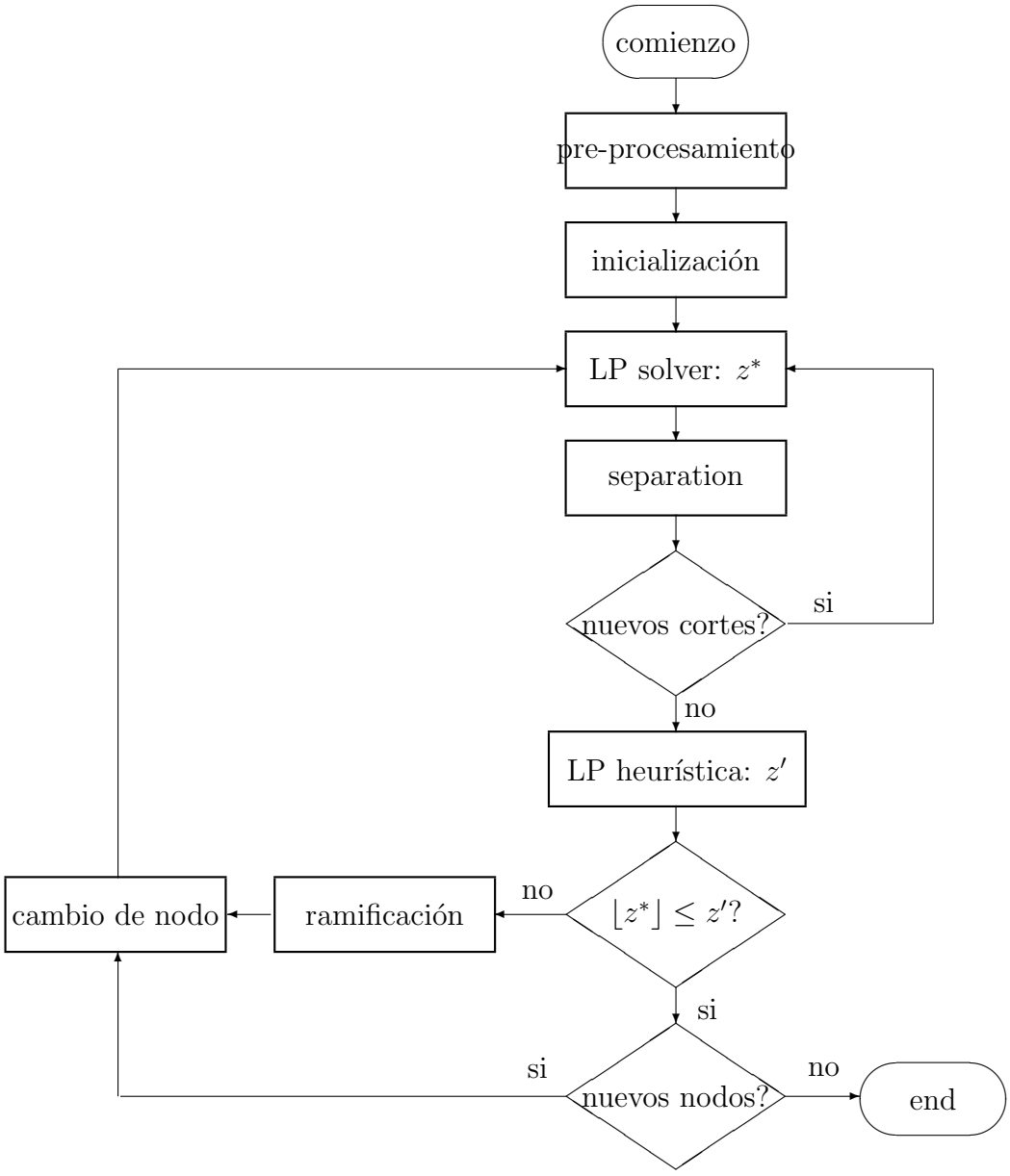


Figura 8.3: Diagrama de flujo del algoritmo de ramificación y corte

6. Eliminar toda planta j con $P_j = P$. (En este caso, se tiene un nuevo ejemplo del GUFLP. Sea S' el conjunto de plantas abiertas en una solución óptima de este nuevo ejemplo, y sea $z(S')$ el correspondiente valor objetivo. Si $z(S') > f_j$ entonces $S \cup \{j\}$ es una solución óptima del ejemplo original, y $z(S') - f_j$ es el valor óptimo. En otro caso, la solución óptima se corresponde con no abrir ninguna planta, y tiene valor 0.)

Inicialización

Aplicando los procedimientos heurísticos de la Sección 8.5 se calcula una solución heurística inicial $(x', \bar{y}')$. El mejor valor z' solución del GUFLP se inicializa con el correspondiente valor heurístico. El problema lineal (LP) inicial se crea considerando todas las variables no fijadas, todas las restricciones (8.8), y una restricción (8.9) para cada planta. Para evitar un problemas lineales no-acotados en las primeras iteraciones, se impone la cota superior 1 sobre todas las variables $\bar{y}$. Finalmente, una *estructura piscina* donde se almacenarán todas las desigualdades que se consideren en el algoritmo, es inicializada con las restricciones (8.8) y (8.9).

LP-solver

Utilizando un resolutor de problemas lineales, se calculan el valor óptimo z^* y la solución óptima $(x^*, \bar{y}^*)$ del problema lineal del momento. Si $(x^*, \bar{y}^*)$ es factible para GUFLP o $\lfloor z^* \rfloor \leq z'$ entonces el nodo del árbol decisional del momento se abandona. En otro caso, el problema lineal se reduce en filas y columnas según las pautas siguientes. Eliminamos del problema lineal todas las variables con costo reducido $\bar{c}$ tal que $\lfloor z^* + \bar{c} \rfloor \leq z'$. Además, eliminamos del problema lineal todas las restricciones con variable de holgura con valor mayor que un parámetro no-negativo *MIN_SLACK* (=0.01 en nuestra implementación). Nunca eliminamos restricciones de la estructura piscina.

Separación

El objetivo de esta fase es fortalecer la relajación lineal del momento identificando desigualdades violadas. En nuestro algoritmo, "desigualdad violada" significa una desigualdad válida $\alpha x + \beta \bar{y} \leq \gamma$ tal que $\alpha x^* + \beta \bar{y}^* - \gamma$ supera un cierto parámetro no-negativo *MIN_VIOLATION* (=0.01).

Comenzamos esta fase buscando desigualdades violadas entre las almacenadas en la estructura piscina. Si se encuentran algunas, la fase de separación finaliza incorporándolas al problema lineal del momento (pero nunca más de *MAX_POOL_CUTS* (=200)). En otro

caso, aplicamos el procedimiento de separación de la Sección 8.4 para identificar desigualdades hueco impar (elevadas) violadas por la solución fraccionaria del momento $(x^*, \bar{y}^*)$. Si se encuentra alguna, la fase de separación termina añadiendo estas restricciones al problema lineal del momento (pero nunca más de *MAX_HOLE_CUTS* (=100)). En otro caso, estimulados por el reciente trabajo de Ceria [C94], buscamos cortes de Gomory violados con un número de coeficientes no nulos y lado derecho no mayor que *MAX_DENSITY* $(=n+m+p)$ y *MAX_RHS* $(=n+m+p)$, respectivamente. Si se encuentran algunos, se añaden al problema lineal del momento (pero nunca más de *MAX_GOMORY_CUTS* (=10)), y se detiene la fase de separación. En otro caso, se ejecuta la fase de ramificación, y no añadimos cortes de Gomory durante los próximos *NO_GOMORY_NODES* (=4) nodos del árbol decisional.

Ramificación

Cuando la relajación lineal del momento no puede mejorarse añadiendo nuevas desigualdades, se sustituye el actual nodo del árbol decisional por otros dos nuevos (nodos hijos). En particular, se elige una planta j con el menor valor de $|\bar{y}_j^* - 0.5|$, y se fija a 0 la variable $\bar{y}_j$ en el primer nuevo nodo decisional, y a 1 en el segundo. Además en este último caso también se fijan a 0 todas las variables x asociadas con las configuraciones en P_j, debido a las restricciones (8.8).

Cambios de nodo decisional

El árbol decisional se explora siguiendo un esquema "depth-first scheme", es decir, usamos una estructura en pila para almacenar los nodos del árbol decisional que se van creando. Cada vez que se ejecuta la fase de ramificación, ponemos en el final de la estructura en pila, primero el nodo originado de la restricción $\bar{y}_j = 1$, y luego el nodo del árbol decisional originado de la restricción $\bar{y}_j = 0$. La fase de cambio de nodo decisional toma el siguiente nodo a considerarse desde el final de la estructura en pila.

Heurística a partir del problema lineal

En un sentido, la solución fraccionaria del problema lineal del momento $(x^*, \bar{y}^*)$ da información útil para guiar la búsqueda de una buena solución del GUFLP $(x', \bar{y}')$. Para explotar esta información, usamos los procedimientos heurísticos de la Sección 8.5. En particular, aplicamos el algoritmo greedy al subproblema en el que se fijan a 0 todas las variables con valor 0 en $(x^*, \bar{y}^*)$. Además, en la heurística de 2-intercambio, las plantas se consideran según un orden basado en los valores de $\bar{y}^*$; dado que este procedimiento

consume bastante tiempo de cálculo, sólo lo aplicamos al final del examen de cada nodo decisional.

8.7 Resultados computacionales

Hemos implementado el algoritmo descrito en la Sección 8.6 en el lenguaje de programación estándar C de Kernighan & Ritchie. Nuestro código funciona sobre un ordenador Digital DECstation 5000/240 y sobre un Hewlett Packard Apollo 9000/720, usando CPLEX 2.1 como resolutor de problemas lineales.

Hemos estudiado el comportamiento de nuestro programa sobre dos familias de problemas del GUFLP generados aleatoriamente. Puesto que el GUFLP fue inspirado por ISP, la generación fue diseñada expresamente para producir ejemplos test de estructura similar a la de los ejemplos del ISP de aplicaciones reales, según ideas de Caprara, Fischetti y Maio [CFM95]. En particular, en ambas familias tenemos muchos elementos $g_{ik} = 0$, es decir, la matriz de beneficios es poco densa, como es característica típica del ISP. El generador aleatorio se ejecuta sobre el Hewlett Packard Apollo 9000/720 y utiliza la función *rand()* del sistema operativo HP-UX 9.0. También utiliza la función *srand()* para inicializar el generador con una determinada semilla s.

Damos ahora descripciones en Pseudo-Pascal de los procedimientos que generan los ejemplos de la Clase A y B. Puesto que los ejemplos de la Clase B resultaron ser más fáciles que los ejemplos de la Clase A, generamos mayores ejemplos de la Clase B que los ejemplos de la Clase A. Dado un conjunto finito S, la notación $r :\in S$ significa que r se elige aleatoriamente entre los elementos en S, según una distribución de probabilidad (pseudo-)uniforme. Los ejemplos de la Clase A fueron generados como sigue

procedure TYPE_A_GEN;
begin
$m := 50$; $n := 50$; $p := 500$; $M := \{1, \ldots, m\}$; $N := \{1, \ldots, n\}$; $P := \{1, \ldots, p\}$;
for $j := 1$ **to** n **do** $f_j := 100$;
for $k := 1$ **to** p **do**
$n_k :\in \{1, \ldots, 5\}$; $N_k := \emptyset$;
for $j := 1$ **to** n_k **do**
$j_k :\in (N \setminus N_k)$; $N_k := N_k \cup \{j_k\}$
end for;
for $i := 1$ **to** m **do** $g_{ik} := 0$;
comment: M_k es el conjunto de clientes servidos por la configuración k con un beneficio de transporte positivo;
$m_k := 5$; $M_k := \emptyset$;
for $i := 1$ **to** m_k **do**
$i_k :\in (M \setminus M_k)$; $M_k := M_k \cup \{i_k\}$;

```
            g_{i_k k} :=∈ {1, ..., tn_k}
        end for
    end for
end.
```

mientras que los ejemplos de la Clase B se generaron como sigue

```
procedure TYPE_B_GEN;
begin
    m := 100; n := 100; p := 0; M := {1, ..., m}; N := {1, ..., n}; P := ∅;
    for j := 1 to n do f_j := 500;
    for i := 1 to m do
        comment: N_i es el conjunto de todas las plantas asociadas con configuraciones
        que sirven al cliente i con un beneficio de transporte positivo;
        n_i := 5; N_i := ∅;
        for j := 1 to n_i do
            j_i :=∈ (N \ N_i); N_i := N_i ∪ {j_i}
        end for;
        comment: p_i es el número de configuraciones
        que sirven al cliente i con un beneficio de transporte positivo;
        p_i := 15;
        for k := 1 to p_i do
            p := p + 1; P := P ∪ {p};
            n_p :=∈ {1, ..., 5}; N_p := ∅;
            for j := 1 to n_p do
                j_p :=∈ (N_i \ N_p); N_p := N_p ∪ {j_p}
            end for;
            for h := 1 to m do g_hp := 0;
            g_ip :=∈ {1, ..., tn_p}
        end for
    end for
end.
```

En ambos casos t representa un parámetro de entrada, y antes de salir limpiamos M, N y P borrando, respectivamente, los clientes que no pueden ser servidos por ninguna configuración, las plantas asociadas a ninguna configuración, y las configuraciones excedentes asociadas con el mismo conjunto de conjunto de plantas.

Combinando $s \in \{111, 222, 333, 444, 555\}$ y $t \in \{200, 175, 150, 125, 100, 75, 50\}$, hemos considerado 35 ejemplos para la Clase A. En cuanto a la Clase B, hemos elegido los valores de T en $\{132, 130, 128, 126, 124, 122, 120, 118\}$, puesto que para mayores valores de t convenía siempre abrir casi todas las plantas en la solución óptima, mientras que para menores valores de t no convenía abrir ninguna planta. Las Tablas 8.1 y 8.2 muestra

los resultados de nuestro algoritmo sobre los distintos ejemplos. Los tiempos se dan en segundos CPU del ordenador HP 9000/720 (80 MHz, 59 Specs, 58 MIPS, 18 Mflops). Las columnas en estas tablas dan la siguiente información:

instance es el nombre del ejemplo en la forma $is.t$, donde s y t se corresponden con los valores de los parámetro de entrada usados para generar los ejemplos;

p es el número de configuraciones;

n es el número de plantas;

m es el número de clientes;

$\mathbf{LB_0}$ es el porcentaje razón LB/(valor solución óptima), donde LB es el inicial valor de una solución del GUFLP; o [LB] si el valor óptimo es 0;

$\mathbf{UB_0}$ es el porcentaje razón UB/(valor solución óptima), donde UB es el valor objetivo de la relajación lineal (8.7)-(8.10); o [UB] si el valor óptimo es 0;

$\mathbf{t_0}$ es el tiempo de ejecución al final de la fase de inicialización;

$\mathbf{UB_1}$ es el porcentaje razón LPR/(valor solución óptima), donde LPR es el valor objetivo de la relajación lineal al final del nodo raíz; o [LPR] si el valor óptimo es 0;

$\mathbf{t_1}$ es el tiempo de ejecución al final del nodo raíz;

opti es el valor solución óptima del GUFLP;

plts es el número de plantas abiertas en la solución óptima;

$\mathbf{t_2}$ es el tiempo de ejecución cuando se alcanza la solución óptima del GUFLP;

basi es el número de desigualdades (8.10) distintas que se han considerado;

hole es el número de desigualdades hueco impar (elevadas) distintas que se han considerado;

gomo es el número de desigualdades de Gomory distintas que se han considerado.

Por construcción, todos los ejemplos producidos por nuestros generadores no vienen alterados durante el pre-procesamiento (el cual se toma sobre unos 0.05 segundos para contrastar cada ejemplo). Los ejemplos de la Clase B se presentan como más fáciles de resolver por nuestro algoritmo —incluso, a pesar de que en algunos casos la diferencia entre el valor óptimo y la cota superior al final del nodo raíz sea bastante grande, se precisan muy pocas ramificaciones para eliminar esta diferencia—, Sin embargo, la clase A resulta más difícil de resolver, y tal dificultad aumenta a medida que se disminuye el valor del parámetro t: no fuimos capaces de resolver 4 sobre 5 ejemplos con $t = 25$ en nuestro

tiempo límite (=50,000 segundos de CPU). La principal causa de este comportamiento se debe a la enorme diferencia entre el valor óptimo y la cota superior que se tiene al final del nodo raíz, y que no se logra disminuir significativamente mediante la ramificación.

Finalmente analizamos la ejecución de nuestro algoritmo cuando no se consideran en la fase de separación, o bien sólo los cortes de Gomory, o bien tanto los cortes de Gomory como las desigualdades hueco impar. Las Tablas 8.3 y 8.4 muestran los resultados Estas tablas contienen la siguiente información:

LPR es el valor objetivo de la relajación lineal al final del nodo raíz;

bran es el número de llamadas a la fase de ramificación;

time es el tiempo total necesitado por el algoritmo.

Para los ejemplos de la Clases A y B, el uso de las desigualdades hueco impar disminuye significativamente el valor de la cota superior durante el nodo raíz. Además, para los ejemplos más duros de la Clase A, la versión en la que no se consideran las desigualdades hueco impar excede el tiempo límite, o bien es en cualquier caso mucho más lenta que la versión original. Para los ejemplos de la Clase B, sin embargo, la versión en la que las desigualdades hueco impar no se consideran, se presenta en muchos caso como la más veloz. Esto se debe al hecho de que a menudo el valor de la cota superior se reduce drásticamente tan pronto como se generan nuevos subproblemas tras ir imponiendo las condiciones de ramificación. Para todos los ejemplos para los que no se demuestra la optimalidad al final del nodo raíz, el uso de cortes de Gomory disminuye el valor de la cota superior en una muy pequeña cantidad, conllevando a menudo unos tiempos de ejecución muy superiores.

8.8 Conclusiones

Los resultados computacionales muestran que para algunos ejemplos, nuestro algoritmo produce una solución óptima en poco tiempo de cálculo, mientras que para otros ejemplos se requiere un mayor esfuerzo. Para estos ejemplos difíciles resultaría muy útil un procedimiento de elevación de desigualdades hueco impar más sofisticado, al tiempo que se deberían considerar otras nuevas familias de desigualdades válidas en este mismo marco de una técnica de hiperplanos de corte. También, para estos ejemplos, nuestro algoritmo heurístico consume bastante tiempo de cálculo, y en mucho casos no tiene éxito en la búsqueda de una solución óptima (o próxima a la óptima); se puede desarrollar nuevos trabajos adicionales en esta otra línea de investigación. Sería por supuesto de gran interés contrastar nuestro código sobre ejemplos del ISP reales, y estamos trabajando en ello.

Instance	p	n	m	LB_0	UB_0	t_0	UB_1	t_1	opti	plts	t_2	basi	hole	gomo
i200.111	445	50	50	100.0	100.0	1.4	100.0	1.4	41755	46	0.5	2391	0	0
i200.222	445	50	50	100.0	100.0	1.3	100.0	1.3	40699	46	0.5	2375	0	0
i200.333	433	50	50	100.0	100.0	1.2	100.0	1.2	41012	45	1.2	2374	0	0
i200.444	431	50	50	100.0	100.0	1.3	100.0	1.3	38976	47	0.4	2357	0	0
i200.555	443	50	50	100.0	100.1	1.5	100.0	1.6	41398	46	0.5	2366	11	0
i175.111	445	50	50	100.0	100.0	1.5	100.0	1.5	35975	46	0.5	2391	0	0
i175.222	445	50	50	100.0	100.0	1.5	100.0	1.7	35039	45	0.6	2375	13	0
i175.333	433	50	50	99.9	100.0	1.5	100.0	1.5	35340	44	1.5	2374	0	0
i175.444	431	50	50	100.0	100.0	1.4	100.0	1.4	33537	46	0.5	2357	0	0
i175.555	443	50	50	100.0	100.4	1.5	100.0	2.1	35670	46	0.5	2366	48	0
i150.111	445	50	50	100.0	100.8	1.7	100.0	2.3	30195	46	0.6	2391	74	0
i150.222	445	50	50	100.0	100.8	2.1	100.0	3.0	29428	43	1.1	2375	86	0
i150.333	433	50	50	100.0	100.0	1.8	100.0	1.8	29688	44	1.0	2374	0	0
i150.444	431	50	50	100.0	100.4	1.6	100.0	2.3	28126	46	0.5	2357	67	0
i150.555	443	50	50	100.0	101.3	1.7	100.0	3.2	29923	46	0.5	2366	90	0
i125.111	445	50	50	100.0	102.1	1.7	100.0	4.7	24415	46	0.5	2391	156	0
i125.222	445	50	50	100.0	102.0	2.0	100.0	4.7	23847	43	0.8	2375	119	0
i125.333	433	50	50	100.0	100.4	1.6	100.0	1.8	24025	44	0.7	2374	45	0
i125.444	431	50	50	100.0	101.7	2.0	100.0	4.8	22680	46	0.5	2357	136	0
i125.555	443	50	50	100.0	102.6	1.9	100.0	9.4	24211	45	0.6	2366	234	0
i100.111	445	50	50	99.6	103.9	2.4	100.2	47.7	18704	41	47.7	2391	559	40
i100.222	445	50	50	100.0	104.0	2.4	100.1	45.4	18217	43	1.1	2375	531	40
i100.333	433	50	50	100.0	102.4	1.9	100.0	5.0	18358	44	0.7	2374	151	0
i100.444	431	50	50	100.0	104.0	2.3	100.4	50.3	17269	45	0.9	2357	625	24
i100.555	443	50	50	99.5	104.6	2.4	100.4	43.8	18536	41	77.4	2366	571	60
i075.111	445	50	50	99.7	107.1	2.5	102.4	85.1	13027	39	101.2	2391	1733	241
i075.222	445	50	50	99.3	107.0	3.0	102.1	86.3	12697	37	567.4	2375	1978	261
i075.333	433	50	50	99.4	105.8	2.2	100.7	60.8	12740	37	91.7	2374	704	40
i075.444	431	50	50	99.1	107.6	2.5	102.9	92.8	11963	37	500.9	2357	2130	308
i075.555	443	50	50	98.9	107.8	2.5	102.3	92.2	12903	40	121.3	2366	1692	330
i050.111	445	50	50	99.6	116.0	4.8	108.7	157.0	7478	35	336.3	2391	8629	1336
i050.222	445	50	50	97.6	116.1	3.4	108.3	142.6	7305	35	1787.2	2375	8621	1236
i050.333	433	50	50	98.2	113.4	4.1	106.3	141.0	7292	37	141.0	2374	3737	384
i050.444	431	50	50	98.4	116.9	5.4	109.7	153.0	6836	34	645.1	2357	9799	1316
i050.555	443	50	50	97.5	116.3	4.2	109.3	150.0	7377	37	1024.3	2366	10512	2179

Tabla 8.1: Resultados computacionales sobre los ejemplos de la Clase A.

Instance	p	n	m	LB_0	UB_0	t_0	UB_1	t_1	opti	plts	t_2	basi	hole	gomo
i132.111	955	100	100	96.0	105.3	7.8	100.0	12.4	3317	83	12.4	500	81	0
i132.222	940	98	100	98.8	100.0	6.4	100.0	6.4	4600	86	6.4	500	0	0
i132.333	928	100	100	95.1	104.6	15.1	100.0	18.1	3455	75	18.1	500	62	0
i132.444	928	99	100	89.6	100.3	7.3	100.0	7.8	4512	78	7.8	500	2	0
i132.555	915	100	100	97.2	104.1	5.5	100.0	8.5	3450	89	8.5	500	70	0
i130.111	955	100	100	93.7	114.2	8.2	100.0	19.2	2651	83	19.2	500	116	0
i130.222	940	98	100	98.2	100.3	7.2	100.0	8.4	3898	86	8.4	500	37	0
i130.333	928	100	100	92.9	111.0	15.3	100.0	22.6	2860	75	22.6	500	113	0
i130.444	928	99	100	86.8	100.7	7.7	100.0	8.1	3878	78	8.1	500	2	0
i130.555	915	100	100	95.0	109.9	6.6	100.0	30.1	2794	80	30.1	500	170	80
i128.111	955	100	100	89.3	130.3	8.9	104.4	60.9	1985	78	72.3	500	281	80
i128.222	940	98	100	97.8	105.1	8.3	100.0	13.3	3158	85	13.3	500	93	0
i128.333	928	100	100	89.8	123.8	15.3	100.0	31.1	2222	75	31.1	500	139	0
i128.444	928	99	100	81.9	102.4	8.1	100.0	9.4	3212	76	9.4	500	38	0
i128.555	915	100	100	90.4	121.2	6.8	100.0	47.3	2141	80	47.3	500	262	80
i126.111	955	100	100	79.5	162.4	9.0	111.9	54.5	1320	78	71.1	500	246	60
i126.222	940	98	100	96.6	113.5	8.6	100.0	18.9	2497	81	18.9	500	118	0
i126.333	928	100	100	92.3	147.6	16.3	100.0	54.2	1606	75	54.2	500	207	40
i126.444	928	99	100	75.3	110.6	8.0	100.0	10.3	2571	76	10.3	500	68	0
i126.555	915	100	100	90.6	144.1	7.5	100.0	50.4	1484	78	50.6	500	226	60
i124.111	955	100	100	51.8	248.9	8.9	151.3	57.5	701	78	80.1	500	314	40
i124.222	940	98	100	94.7	131.0	11.3	100.0	91.4	1815	81	91.4	500	211	100
i124.333	928	100	100	76.4	201.6	11.2	121.2	90.8	997	75	116.9	500	348	60
i124.444	928	99	100	62.6	123.6	7.8	100.0	44.6	1961	72	44.6	500	182	119
i124.555	915	100	100	77.8	201.6	7.6	117.7	67.7	860	76	83.4	500	330	44
i122.111	955	100	100	0.0	1412.7	9.4	648.1	80.0	99	71	155.2	500	327	80
i122.222	940	98	100	89.9	174.5	10.8	115.8	77.6	1121	81	100.6	500	345	81
i122.333	928	100	100	35.7	422.6	11.2	193.6	119.4	392	75	157.9	500	314	180
i122.444	928	99	100	37.6	148.0	8.2	100.0	57.2	1347	72	57.2	500	285	80
i122.555	915	100	100	0.0	584.1	9.7	244.1	60.0	235	76	96.2	500	343	84
i120.111	955	100	100	0.0	[1076.3]	9.5	[241.2]	72.9	0	0	72.9	500	309	60
i120.222	940	98	100	67.4	340.2	10.9	160.2	115.2	466	76	137.2	500	345	90
i120.333	928	100	100	0.0	815.8	11.7	184.5	93.7	159	4	141.5	500	437	60
i120.444	928	99	100	0.0	227.4	8.2	112.3	78.9	709	72	81.9	500	326	100
i120.555	915	100	100	0.0	[1020.1]	9.8	[109.3]	76.8	0	0	76.8	500	330	41
i118.111	955	100	100	0.0	[780.2]	9.8	[0.0]	44.3	0	0	44.3	500	292	0
i118.222	940	98	100	0.0	[1233.1]	11.5	[405.4]	67.5	0	0	67.5	500	356	40
i118.333	928	100	100	0.0	788.5	11.3	100.0	41.8	120	4	41.8	500	317	0
i118.444	928	99	100	0.0	877.4	8.6	177.4	116.2	148	68	132.8	500	439	60
i118.555	915	100	100	0.0	[685.9]	9.8	[0.0]	28.0	0	0	28.0	500	267	0

Tabla 8.2: Resultados computacionales sobre los ejemplos de la Clase B.

holes		yes			yes			no	
gomory		yes			no			no	
instance	LPR	bran	time	LPR	bran	time	LPR	bran	time
i200.111	41755.0	0	1.34	41755.0	0	1.40	41755.0	0	1.36
i200.222	40699.0	0	1.21	40699.0	0	1.21	40699.0	0	1.23
i200.333	41012.0	0	1.06	41012.0	0	1.05	41012.0	0	1.06
i200.444	38976.0	0	1.11	38976.0	0	1.14	38976.0	0	1.14
i200.555	41398.0	0	1.61	41398.0	0	1.59	41439.5	1	1.39
i175.111	35975.0	0	1.46	35975.0	0	1.42	35975.0	0	1.43
i175.222	35039.0	0	1.65	35039.0	0	1.66	35042.0	1	1.30
i175.333	35340.0	0	1.38	35340.0	0	1.38	35340.0	0	1.34
i175.444	33537.0	0	1.22	33537.0	0	1.22	33537.0	0	1.21
i175.555	35670.0	0	2.09	35670.0	0	2.10	35829.0	4	5.05
i150.111	30195.0	0	2.32	30195.0	0	2.33	30422.5	9	5.66
i150.222	29428.0	0	2.96	29428.0	0	3.00	29665.0	8	9.81
i150.333	29688.0	0	1.68	29688.0	0	1.70	29688.0	0	1.68
i150.444	28126.0	0	2.25	28126.0	0	2.28	28246.5	3	3.94
i150.555	29923.0	0	3.22	29923.0	0	3.27	30318.5	11	12.62
i125.111	24415.0	0	4.59	24415.0	0	4.72	24931.0	72	67.65
i125.222	23847.0	0	4.67	23847.0	0	5.03	24314.5	29	40.19
i125.333	24025.0	0	1.73	24025.0	0	1.76	24110.5	3	3.12
i125.444	22680.0	0	4.74	22680.0	0	4.66	23061.0	27	25.77
i125.555	24211.0	0	9.31	24211.0	0	9.66	24846.5	49	56.09
i100.111	18745.3	2	55.55	18747.2	2	50.52	19434.5	423	473.70
i100.222	18233.2	1	45.57	18236.1	1	36.41	18952.0	298	352.28
i100.333	18358.0	0	4.88	18358.0	0	5.02	18790.0	37	39.58
i100.444	17330.3	3	72.36	17332.1	2	58.46	17959.8	339	476.00
i100.555	18613.4	5	86.98	18621.8	5	71.14	19384.0	384	427.19
i075.111	13336.5	28	674.47	13339.3	21	497.25	13953.5	3176	5352.37
i075.222	12963.5	31	763.66	12966.1	20	501.91	13587.5	3286	4624.73
i075.333	12834.2	4	112.98	12837.1	4	100.90	13481.0	721	900.38
i075.444	12304.3	39	967.69	12306.8	41	887.00	12870.5	4471	8324.17
i075.555	13195.1	32	745.16	13198.5	29	601.24	13915.5	4853	7652.03
i050.111	8126.3	266	8276.28	8138.0	288	7528.05	8675.1	24103	50000.00*
i050.222	7912.1	253	8784.04	7914.8	278	7565.24	8481.8	22484	50000.00*
i050.333	7754.9	72	2311.93	7756.5	78	2265.61	8266.7	8849	19236.57
i050.444	7496.3	317	10824.70	7500.0	363	10197.02	8675.1	24103	50000.00*
i050.555	8064.9	406	13004.22	8066.0	472	11045.34	8481.8	22484	50000.00*

(*: se ha excedido el tiempo límite.)

Tabla 8.3: Comparando tres relajaciones lineares sobre los ejemplos de la Clase A.

holes		yes			yes			no	
gomory		yes			no			no	
instance	LPR	bran	time	LPR	bran	time	LPR	bran	time
i132.111	3317.0	0	12.44	3317.0	0	12.48	3491.7	5	27.14
i132.222	4600.0	0	6.46	4600.0	0	6.46	4600.0	0	6.46
i132.333	3455.0	0	18.09	3455.0	0	18.10	3613.4	3	34.53
i132.444	4512.0	0	7.76	4512.0	0	7.77	4523.5	1	9.86
i132.555	3450.0	0	8.48	3450.0	0	8.48	3590.5	5	21.44
i130.111	2651.0	0	19.21	2651.0	0	19.25	3027.4	12	22.40
i130.222	3898.0	0	8.40	3898.0	0	8.42	3909.9	1	10.72
i130.333	2860.0	0	22.59	2860.0	0	22.52	3175.7	5	33.92
i130.444	3878.0	0	8.15	3878.0	0	8.18	3907.0	1	10.24
i130.555	2794.0	0	30.10	2864.6	2	24.87	3072.0	6	34.20
i128.111	2072.7	2	79.56	2086.8	2	66.02	2586.4	13	22.47
i128.222	3158.0	0	13.27	3158.0	0	13.29	3318.2	12	59.70
i128.333	2222.0	0	31.12	2222.0	0	31.19	2750.4	13	31.83
i128.444	3212.0	0	9.37	3212.0	0	9.38	3290.2	1	13.24
i128.555	2141.0	0	47.33	2311.3	4	43.72	2595.7	6	12.08
i126.111	1476.8	3	90.95	1486.7	3	81.48	2143.6	11	17.89
i126.222	2497.0	0	18.94	2497.0	0	19.01	2834.2	8	36.97
i126.333	1606.0	0	54.25	1613.0	1	50.99	2370.0	11	29.83
i126.444	2571.0	0	10.35	2571.0	0	10.34	2843.8	5	42.65
i126.555	1484.2	1	51.41	1495.9	1	32.97	2138.4	8	13.55
i124.111	1060.9	5	116.28	1067.0	3	75.19	1744.9	16	23.69
i124.222	1815.0	0	91.40	1921.2	1	50.93	2377.0	15	34.63
i124.333	1208.1	4	156.33	1232.7	3	89.81	2009.9	13	24.68
i124.444	1961.0	0	44.58	2124.2	2	33.04	2423.9	7	37.37
i124.555	1011.9	3	106.81	1028.4	2	65.30	1734.1	9	13.23
i122.111	641.6	5	159.73	658.3	5	108.15	1398.6	31	37.48
i122.222	1297.6	3	129.89	1335.0	2	64.13	1956.2	20	26.59
i122.333	758.9	6	197.62	837.9	4	134.47	1656.7	14	25.12
i122.444	1347.0	0	57.23	1640.4	4	27.49	1993.3	11	19.24
i122.555	573.6	5	155.90	583.6	4	97.12	1372.6	14	20.06
i120.111	241.2	3	103.70	263.1	3	89.86	1076.3	10	15.63
i120.222	746.7	4	152.50	802.6	6	128.24	1585.4	25	30.42
i120.333	293.4	3	141.53	324.6	2	88.10	1297.1	14	28.76
i120.444	796.3	1	90.29	1116.8	3	37.10	1612.1	15	19.51
i120.555	109.3	2	99.41	129.5	2	75.75	1020.1	16	24.41
i118.111	0.0	0	44.27	0.0	0	44.31	780.2	10	17.66
i118.222	405.4	4	134.58	417.1	4	105.64	1233.1	13	23.60
i118.333	120.0	0	41.83	120.0	0	41.88	946.2	9	21.70
i118.444	262.6	2	148.63	290.3	3	95.46	1298.6	13	18.39
i118.555	0.0	0	28.03	0.0	0	27.98	685.9	7	13.57

Tabla 8.4: Comparando tres relajaciones lineales sobre ejemplos de la Clase B.

Bibliografía

[AMO89] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, "Network Flows", in G.L. Nemhauser, A.H.G. Rinnooy Kan, M.J. Todd (Editors), *Optimization*, vol. I, North-Holland (1989) 211–370.

[A89] J.R. Araque, "Contributions to the polyhedral approach to vehicle routing", Ph.D. Dissertation, State University of New York, 1989.

[AKMP94] J.R. Araque, G. Kudva, T.L. Morin, J.F. Pekny, "A branch-and-cut algorithm for vehicle routing problems", *Annals of Operations Research* 50 (1994) 37–59.

[BG82] A. Bachem, M. Grötschel, "New aspects of polyhedral theory", in B. Korte (Editor), *Modern Applied Mathematics, Optimization and Operations Research*, North-Holland, Amsterdam, 1982.

[B89] E. Balas, "The Prize Collecting Traveling Salesman Problem", *Networks* 19 (1989) 621–636.

[B93a] E. Balas, "The Prize Collecting Traveling Salesman Problem: II. Polyhedral Results", working paper, Carnegie Mellon University, July 1993.

[B93b] E. Balas, "On the Cycle Polytope of a Directed Graph", working paper, Carnegie Mellon University, July 1993.

[BF92] E. Balas, M. Fischetti, "The Fixed-Outdegree 1-Arborescence Polytope", *Mathematics of Operations Research* 17 (1992) 1001–1018.

[BF93] E. Balas, M. Fischetti, "A Lifting Procedure for the Asymmetric Traveling Salesman Polytope and a Large New Class of Facets", *Mathematical Programming* 58 (1993) 325–352.

[BP76] E. Balas, M.W. Padberg, "Set Partitioning: A survey", *SIAM Review* 18 (1976) 710–760.

[B94] P. Bauer, "The Circuit Polytope: Facets", working paper, University of Köln, March 1994.

[BK77] O. Bilde, J. Krarup, "Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem", *Annals of Discrete Mathematics* 1 (1977) 79–97.

[ESI95] M.H. Bjorndal, A. Caprara, P.I. Cowling, F. Della Croce, H. Lourenço, F. Malucelli, A.J. Orman, D. Pisinger, C. Rego, J.J. Salazar, "Some thoughts on combinatorial optimisation", *European Journal of Operational Research* 83 (1995) 253–270.

[CF96] A. Caprara, M. Fischetti, "0-1/2 Chvátal-Gomory Cuts", *Mathematical Programming* (to appear) (1996).

[CFM95] A. Caprara, M. Fischetti, D. Maio, "Exact and Approximate Algorithms for the Index Selection Problem in Physical Database Design", *IEEE Transactions on Knowledge and Data Engineering* 7 (1995) 955–967.

[CS96a] A. Caprara, J.J. Salazar, "A Branch-and-Cut Algorithm for a Generalization of the Uncapacitated Facility Location Problem", *TOP* 4/1 (1996) 135–163.

[CS96b] A. Caprara, J.J. Salazar, "Separating lifted odd-hole inequalities to solve the index selection problem", *Discrete Applied Mathematics* 92 (1999) 111–134.

[CFT89] G. Carpaneto, M. Fischetti, P. Toth, "New Lower Bounds for the Symmetric Travelling Salesman Problem", *Mathematical Programming* 45 (1989) 233–254.

[CCE85] B.D. Causey, L.H. Cox, L.R. Ernst, "Applications of Transportation Theory to Statistical Problem", *Journal of the American Statistical Association*, Vol. 80, No. 392 (1985) 903–909.

[C94] S. Ceria, "Solving Mixed Integer Programs with General Cutting Planes", talk presented at the workshop *Solving Large and Complex Combinatorial Optimization Problems*, Giens, May 1994.

[CJPR83] D.C. Cho, E.L. Johnson, M.W. Padberg, M.R. Rao, "On the Uncapacitated Plant Location Problem. I: Valid Inequalities and Facets", *Mathematics of Operations Research* 8 (1983) 579–589.

[CPR83] D.C. Cho, M.W. Padberg, M.R. Rao, "On the Uncapacitated Plant Location Problem. II: Facets and Lifting Theorems", *Mathematics of Operations Research* 8 (1983) 590–612.

[C85] N. Christofides, "Vehicle routing", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, (1985) 431–448.

[CMT79] N. Christofides, A. Mingozzi, P. Toth, "The vehicle routing problem", in: N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.), *Combinatorial Optimization*, Wiley, Chichester, (1979) 315–338.

[CC90] A.R. Conn, G. Cornuejols, "A Projection Method for the Uncapacitated Facility Location Problem", *Mathematical Programming* 46 (1990) 273–298.

[CCPS95] W. Cook, W.H. Cunningham, W.R. Pulleyblank, A. Schrijver, *Combinatorial Optimization*, working book, 1995.

[CH93] G. Cornuejols, F. Harche, "Polyhedral study of the capacitated vehicle routing problem", *Mathematical Programming* 60 (1993) 21–52.

[CNW91] G. Cornuejols, G.L. Nemhauser, L.A. Wolsey, "The Uncapacitated Facility Location Problem", in P.B. Mirchandani, R.L. Francis, *Discrete Location Theory*, John Wiley, 1991.

[CT82a] G. Cornuejols, J.M. Thizy, "Some Facets of the Simple Plant Location Polytope", *Mathematical Programming* 23 (1982) 50–74.

[CT82b] G. Cornuejols, J.M. Thizy, "A Primal Approach to the Simple Plant Location Problem", *SIAM Journal on Algebraic and Discrete Methods* 3 (1982) 504–510.

[CP89] C.R. Coullard, W.R. Pulleyblank, "On Cycle Cones and Polyhedra", *Linear Algebra and its Applications* 114/115 (1989) 613–640.

[C87] L.H. Cox, "A Constructive Procedure for Unbiased Controlled Rounding", *Journal of the American Statistical Association*, Vol. 82, No. 398 (1987) 520–524.

[C92] L.H. Cox, "Solving Confidentiality Protection Problems in Tabulations using Network Optimization: A Network Model for Cell Suppression in U.S. Economic Censuses", working paper, U.S. Bureau of the Census, Washington, 1992.

[CJP83] H.P. Crowder, E.L. Johnson, M.W. Padberg, "Solving Large-Scale Zero-One Linear Programming Problems", *Operations Research* 31 (1983) 803–834.

[E78] D. Erlenkotter, "A Dual-Based Procedure for Uncapacitated Facility Location", *Operations Research* 26 (1978) 992–1009.

[FJ81] G.N. Frederickson, J. Ja'Ja', "Approximation algorithms for Several Graph Augmentation Problems", *SIAM Journal on Computing* 2 (1981) 270–283.

[FST88] S. Finkelstein, M. Schkolnick, P. Tiberio, "Physical Database Design for Relational Databases", *ACM Transactions on Database Systems* 13 (1988) 91–128.

[FS95] M. Fischetti, J.J. Salazar, "Técnicas Poliédricas para Garantizar la Privacidad en Tablas Estadísticas Públicas", working paper, University of La Laguna, October 1995.

[FST94] M. Fischetti, J.J. Salazar, P. Toth, "On the Cycle Polytope of a Undirected Graph", working paper, University of Bologna, Aprile 1994.

[FST95a] M. Fischetti, J.J. Salazar, P. Toth, "The Symmetric Generalized Travelling Salesman Polytope", *Networks* 26 (1995) 113-123.

[FST95b] M. Fischetti, J.J. Salazar, P. Toth, "Solving the Orienteering Problem through Branch-and-Cut", *INFORMS Journal on Computing* 10/2 (1998) 133–148.

[FST95c] M. Fischetti, J.J. Salazar, P. Toth, "Experiments with a multi-commodity formulation for the Symmetric Capacitated Vehicle Routing Problem", *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*, Barcelona 1995.

[FST96] M. Fischetti, J.J. Salazar, P. Toth, "A Branch-and-Cut Algorithm for the Symmetric Generalized Travelling Salesman Problem", *Operations Research* 45/3 (1997) 378–394.

[FT88] M. Fischetti, P. Toth, "An Additive Approach for the Optimal Solution of the Prize-Collecting Travelling Salesman Problem" in B.L. Golden and A.A. Assad (Editors), *Vehicle Routing: Methods and Studies*, North-Holland (1988) 319–343.

[FT92] M. Fischetti, P. Toth, "A Branch-and-Cut Algorithm for Solving Hard Instances of the Asymmetric Traveling Salesman Problem", working paper, University of Bologna, November 1992.

[FTV94] M. Fischetti, P. Toth, D. Vigo, "A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs", *Operations Research* 42 (1994) 846–859.

[F94] M.L. Fisher, "Optimal Solution of Vehicle Routing Problems using minimum k-Trees", *Operations Research* 42 (1994) 626–642.

[GLS96] M. Gendreau, G. Laporte, F. Semet, "The Covering Tour Problem", to appear in *Operations Research*, 1996.

[G92] J. Geurts, "Heuristics for Cell Suppression in Tables", working paper, Netherlands Central Bureau of Statistics, Voorburg, September 1992.

[GA88] B.L. Golden, A.A. Assad, *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam, 1988.

[GLV87] B.L. Golden, L. Levy, R. Vohra, "The Orienteering Problem", *Naval Research Logistics* 34 (1987) 307–318.

[GS85] B.L. Golden, W.R. Stewart, "Empirical analysis of heuristics", Chapter 7 in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Editors), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[GWL88] B.L. Golden, Q. Wang, L. Liu, "A Multifaceted Heuristic for the Orienteering Problem", *Naval Research Logistics* 35 (1988) 359–366.

[GH61] R.E. Gomory, T.C. Hu, "Multi-Terminal Network Flows", *SIAM Journal* 9 (1961) 551–570.

[G85] M. Grötschel, "Polyhedral Combinatorics", Chapter 1 in M. O'hEigeartaigh, J.K. Lenstra, A.H.G. Rinnooy Kan (Editors), *Combinatorial Optimization - Bibliography*, John Wiley & Sons, 1985.

[GH91] M. Grötschel, O. Holland, "Solution of Large-Scale Symmetric Travelling Salesman Problems", *Mathematical Programming* 51 (1991) 141–202.

[GL93] M. Grötschel, L. Lovász, "Combinatorial Optimization: A Survey", *DIMACS* Technical Report 93-29, May 1993.

[GLS88] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlang, 1988.

[GP85] M. Grötschel, M.W. Padberg, "Polyhedral theory", Chapter 8 in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Editors), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[G80] M. Guignard, "Fractional Vertices, Cuts and Facets of the Simple Plant Location Problem", *Mathematical Programming Study* 12 (1980) 150–162.

[G88] D. Gusfield, "A Graph Theoretic Approach to Statistical Data Security", *SIAM Journal on Computing*, Vol. 17, No. 3 (1988) 552–571.

[G89] L. Hall, "Two topics in discrete optimization: Polyhedral structure of capacitated trees and approximation algorithms for scheduling", Ph.D. Dissertation, M.I.T., 1989.

[HK71] M. Held, R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II", *Mathematical Programming* 1 (1971) 6–25.

[HP93] K.L. Hoffman, M. Padberg, "Solving Airline Crew Scheduling Problems by Branch-and-Cut", *Management Science*, Vol. 39, No. 6 (1993) 657–682.

[JRR92] M. Jünger, G. Reinelt, G. Rinaldi, "The Travelling Salesman Problem", in *Handbook in Operations Research and Management Science: Network Models* (eds. M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser), Elsevier, Amsterdam, 1995.

[KGA91] J.P. Kelly, B.L. Golden, A.A. Assad, "Cell Suppression: Disclosure Protection for Sensitive Tabular Data", working paper, University of Maryland, 1991.

[KP83] J. Krarup, P.M. Pruzan, "The Simple Plant Location Problem: Survey and Synthesis", *European Journal of Operational Research* 12 (1983) 36–81.

[K56] J.B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *Proc. AMS* 7 (1956) 48–50.

[L92] G. Laporte, "The Vehicle Routing Problem: An overview of exact and approximate algorithms", *European Journal of Operational Research* 59 (1992) 345–358.

[LM90] G. Laporte, S. Martello, "The Selective Traveling Salesman Problem", *Discrete Applied Mathematics* 26 (1990) 193–207.

[LMN87] G. Laporte, H. Mercure, Y. Nobert, "Generalized Travelling Salesman Problem through n sets of nodes: the asymmetrical case", *Discrete Applied Mathematics* 18 (1987) 185–197.

[LN83] G. Laporte, Y. Nobert, "Generalized Travelling Salesman through n sets of nodes: an integer programming approach", *INFOR* 21 (1983) 61–75.

[LN84] G. Laporte, Y. Nobert, "Comb inequalities for the vehicle routing problem", *Methods of Operations Research* 51 (1984) 271–276.

[LN87] G. Laporte, Y. Nobert, "Exact algorithms for the vehicle routing problem", in: S. Martello, G. Laporte, M. Minoux, C. Ribeiro (eds.), *Surveys in Combinatorial Optimization*, North-Holland, Amsterdam, (1987) 147–184.

[LND85] G. Laporte, Y. Nobert, M. Desrochers, "Optimal routing under capacity and distance restrictions", *Operations Research* 33 (1985) 1050–1073.

[LLRS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Editors), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[LR94] A.C. Leifer, M.B. Rosenwein, "Strong Linear Programming Relaxations for the Orienteering Problem", *European Journal of Operations Research* 73 (1994) 517–523.

[MT90] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, 1990.

[NR93] D. Naddef, G. Rinaldi, "The graphical relaxation: A new framework for the Symmetric Traveling Salesman Polytope", *Mathematical Programming* 58 (1993) 53–88.

[NS92] G.L. Nemhauser, G. Sigismondi, "A Strong Cutting Plane/Branch-and-Bound Algorithm for Node Packing", *Journal of Operational Research Society* 43 (1992) 443–457.

[NW88] G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, 1988.

[N88] C.E. Noon, "The Generalized Traveling Salesman Problem". Unpublished Dissertation, Department of Industrial and Operations Research, University of Tennessee, 1988.

[NB91] C.E. Noon, J.C. Bean, "A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem", *Operations Research* 39 (1991) 623–632.

[NB89] C.E. Noon, J.C. Bean, "An Efficient Transformation of the Generalized Traveling Salesman Problem", *INFOR* 31 (1993) 39–44.

[P73] M.W. Padberg, "On the Facial Structure of Set Packing Problems", *Mathematical Programming* 5 (1973) 199–215.

[P75] M.W. Padberg, "A note on zero–one programming", *Operations Research* 23 (1975) 833–837.

[PG85] M.W. Padberg, M. Grötschel, "Polyhedral computations", Chapter 9 in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Editors), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[PR82] M.W. Padberg, M.R. Rao, "Odd Minimum Cut-Sets and b-Matchings", *Mathematics of Operations Research* 7/1 (1982) 67–80.

[PR87] M.W. Padberg, G. Rinaldi, "Optimization of a 532-city Symmetric Traveling Salesman Problem", *Operations Research Letters* 6 (1987) 1–7.

[PR90] M.W. Padberg, G. Rinaldi, "Facet Identification for the Symmetric Traveling Salesman Polytope", *Mathematical Programming* 47 (1990) 219–257.

[PR91] M.W. Padberg, G. Rinaldi, "A Branch-And-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", *SIAM Review* 33 (1991) 60-100.

[P82] W.R. Pulleyblank, "Polyhedral Combinatorics", in A. Bachem, M. Grötschel, B. Korte (Editors), *Mathematical Programming. The State of the Art*, Bonn, 1982.

[P91] W.R. Pulleyblank, "Polyhedral Combinatorics", Chapter 5 in G.L. Nemhauser, A.H.G. Rinnooy Kan, M.J. Todd (Editors), *Handbooks in Operations Research and Management Science* Vol.1 *Optimization*, North-Holland, 1991.

[RYK92] R. Ramesh, Yong-Seok Yoon, Mark H. Karwan, "An Optimal Algorithm for the Orienteering Tour Problem", *ORSA Journal on Computing* 4/2 (1992) 155–165.

[R91] G. Reinelt, "TSPLIB - A Traveling Salesman Problem Library", *ORSA Journal on Computing* 3 (1991) 376–384.

[BM85] ROLL-A-ROUND: Software Package for Scheduling the Rounds of a Rolling Mill, Copyright Balas and Martin Associates, Pittsburgh, PA.

[S92] J.J. Salazar, "Algoritmi per il problema del Commesso Viaggiatore Generalizzato", PhD Dissertation, Royal Spanish College in Bologna, Italy, 1992.

[S86] A. Schrijver, "Theory of Linear and Integer Programming", John Wiley & Sons, Chichester, 1986.

[S91] M.M. Sepehri, "The Symmetric Generalized Travelling Salesman Problem", PhD Dissertation, University of Tennessee, Knoxville, 1991.

[S79] P.D. Seymour, "Sums of circuits", *Graph Theory and Related Topics* (J.A. Bondy and U.S.R. Murty, Eds.), Academic, New York (1979) 341–355.

[T89] T. Tsiligirides, "Heuristic Methods Applied to Orienteering", *J. Opl. Res. Soc.* 35/9 (1984) 797–809.

[WW94] A.G. de Waal, L.C.R.J. Willenborg, "Minimizing the Number of Local Suppressions in a microdata Set", working paper, Netherlands Central Bureau of Statistics, 1994.