Trabajo Fin de Máster

Gabriel González Domínguez

Trabajo Fin de Máster

Gabriel González Domínguez

Tutorizada por

Prof. Justo Puerto Albandoz

Índice general

Abstract

It is reviewes the theory involved in multicommodity-flow problems and it is analyzed several aspects concerning the optimal routing of pipelines in naval design motivated by a recent collaboration with a leading Naval Engineering company. We start by considerering branch-and-cut algorithm which is a important tool to implement some desired properties in naval design as the minimum required distance between the pipelines volumes or the minimum allowed distances between pipe elbows. We also study Lagrangian relaxation and columns generation, both imperative for a deeper comprehesion of multicommodity-flow problems. In addition, it is studied the minimum cost flows as the the most basic of all flow problems and base for solving multicommodity-flow problems. Moreover, we apply all these mathematical tools into multicommodity-flow problems. Finally, it is proposed a general methodology for the automatic routing. We construct a network-shape framework for feasible solutions of the problem and a cost-based structure for the network that incorporates different desired physical characteristics for the routings. Then, a multicommodity-flow based approach is proposed and a heuristic algorithm is designed to solve it that take into account some desired properties of the routes.

1 Introducción

El problema de flujo con coste mínimo modela el flujo de un único servicio a través de una red. Los problemas de flujo multiservicio surgen cuando varios servicios utilizan la misma red subyacente. Los servicios pueden diferenciarse por sus características físicas o simplemente por sus pares de origen-destino. Los diferentes servicios tienen diferentes orígenes y destinos, y los servicios tienen restricciones de balance de flujo independientes en cada nodo. Sin embargo, comparten las capacidades de los arcos comunes que llevan varios servicios. El problema de flujo multiservicio se puede ver como un problema de asignación de capacidad en el que los servicios compiten por las capacidades de los arcos de la red.

En este trabajo se realiza un estudio de la teoría necesaria para trabajar con los problemas de flujo multiservicio y se aplica parte de esta en la resolución del problema de trazado de canalizaciones en el interior de un buque. Este problema está motivado por una colaboración con la empresa GHENOVA, por lo que se trabajan con datos reales. Así mismo, las capacidades del problema son fijas pero se introducen dos requerimientos propios del diseño naval, como son la imposición de una distancia mínima entre servicios (test de revestimiento) o una distancia mínima entre los codos de un mismo servicio (test de codos). En este caso, puede verse como un problema en el que los servicios compiten por ocupar el volumen disponible.

Se introduce la técnica de ramificación y corte, herramienta que se pone en práctica para implementar los tests de codos y revestimiento. Así mismo, se estudia y ejemplifica la relajación lagrangiana y la generación de columnas, ambos importantes métodos a la hora de trabajar en problemas de flujo multiservicio.

También se estudian los problemas de flujo de mínimo coste y cómo aprovechar el hecho de que siempre tienen al menos una solución de árbol de expansión. Esto es necesario para seguidamente estudiar cómo trabajar con los problemas de flujo multiservicio.

Finalmente, se describe el problema del trazado de canalizaciones en barcos y se plantea un algoritmo heurísticos para resolverlo. A su vez, se realiza un estudio computacional comparando la resolución exacta y heurística del problema. Por último, se resuelve un escenario real proporcionado por GHENOVA mediante la heurística presentada.

2 Herramientas

En la Sección 2.1 se presenta el algoritmo de ramificación y corte, debido a las enormes dimensiones del problema que se presenta en el Capítulo 5, este algoritmo se usa en la Subsección en 5.3.1. En la seccionnes 2.2 y 2.3 se presentan los algoritmos de relajación lagrangiana y generación de columnas respectivamente; ambos son fundamentales en el Capítulo 4.

2.1 Algoritmos de ramificación y corte

Los métodos de ramificación y corte son algoritmos exactos para problemas de programación entera. Son una combinación de métodos de planos de corte y algoritmos de ramificación y acotación, consisten en resolver una secuencia de relajaciones de programación lineal de un problema de programación entera. Los métodos de planos de corte mejoran la relajación del problema para aproximarse más al problema entero, y los algoritmos de ramificación y acotación siguen un procedimiento sofisticado de divide y vencerás para resolver problemas. Todas las demostraciones de los resultados y proposiciones que aparecen en esta sección se encuentran en el libro publicado por *Floudas* y *Pardalos* [1] en 2009.

Estos métodos son capaces de resolver y probar optimalidad en escenarios mucho mayores que otros métodos. Los planos de corte suelen ser resultado del estudio de la combinatoria poliédrica del correspondiente problema entero. Añaden cortes profundos (desigualdades en general), que permiten reducir considerablemente el tamaño del árbol de ramificación y acotación.

Los métodos de ramificación y corte son en general de gran interés para la programación entera. Normalmente no se puede resolver de forma eficiente un problema de programación entera usando sólo planos de corte, es por tanto necesario ramificar

también. Un enfoque puro de ramificación y poda se puede acelerar considerablemente mediante el empleo de un esquema de planos de corte, ya sea justo en la parte superior del árbol o en cada nodo del árbol.

Ejemplo 2.1. Considérese el problema de programación entera

$$\begin{aligned} \text{mín} \quad & -5x_1 - 6x_2 && (2.1)\\ \text{sujeto a} \quad & x_1 + 2x_2 \leq 7 && (2.2)\\ & 2x_1 - x_2 \leq 3 && (2.3)\\ & x_1, x_2 \geq 0 \text{ y enteros} && (2.4) \end{aligned}$$

Este problema se ilustra en la Figura 2.1, en la que se indican los puntos enteros factibles. La relajación de programación lineal se obtiene ignorando las restricciones de integralidad; esto viene dado por el poliedro contenido en las líneas continuas.

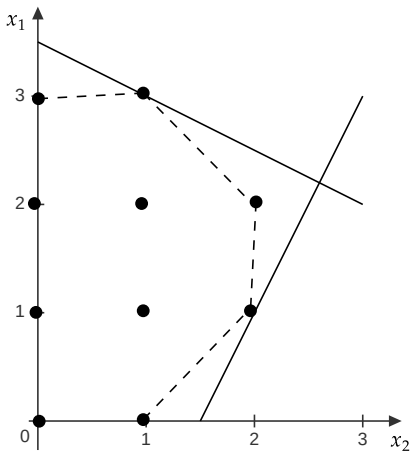

Figura 2.1: Un ejemplo de ramificación y corte.

El primer paso en la ramificación y corte es resolver la relajación de programación lineal, la cual da el punto (2.6, 2.2) con valor −26.2. Ahora se puede elegir: si se debería mejorar la relajación del problema lineal añadiendo un plano de corte, por ejemplo, $x_1 + x_2 \leq 4$, o si se debería dividir el problema en dos limitando el valor de una variable en un caso y otro. Supóngase que el algoritmo hace lo segundo, dividiendo

x_2 y creando dos nuevos problemas:

$$\begin{aligned} \text{mín} \quad & -5x_1 - 6x_2 && (2.5)\\ \text{sujeto a} \quad & x_1 + 2x_2 \leq 7 && (2.6)\\ & 2x_1 - x_2 \leq 3 && (2.7)\\ & \mathbf{x}_2 \geq 3 && (2.8)\\ & x_1, x_2 \geq 0 \text{ y enteros} && (2.9) \end{aligned}$$

y

$$\begin{aligned} \text{mín} \quad & -5x_1 - 6x_2 && (2.10)\\ \text{sujeto a} \quad & x_1 + 2x_2 \leq 7 && (2.11)\\ & 2x_1 - x_2 \leq 3 && (2.12)\\ & \mathbf{x}_2 \leq 2 && (2.13)\\ & x_1, x_2 \geq 0 \text{ y enteros} && (2.14) \end{aligned}$$

La solución óptima del problema original será la mejor de las soluciones de estos dos subproblemas. La solución de la relajación de programación lineal del primer subproblema es $(1, 3)$, con valor 23. Dado que esta solución es entera, resuelve el primer subproblema. Esta solución se convierte en la mejor solución factible conocida. La solución óptima para la relajación de la programación lineal del segundo subproblema es $(2.5, 2)$, con valor -24.5. Dado que este punto no es entero, no resuelve el subproblema. Por tanto, hay que seguir atacando el segundo subproblema.

Es posible ramificar usando x_1 en el segundo subproblema; pero en lugar de esto, supóngase que el algoritmo usa un método de planos de corte y añade la desigualdad $x_1 + 2x_2 \leq 6$. Esta es una desigualdad válida, puesto que la satisface todo punto entero que es factible en el segundo subproblema. Esta desigualdad es un plano de corte y es violada por $(2.5, 2)$. Se suma esta desigualdad a la relajación y al resolver da la solución óptima $(2.4, 1.8)$, con valor 22.6. El subproblema aún no tiene una solución entera. Sin embargo, el valor óptimo para esta relajación modificada es mayor que el valor de la solución actual. El valor de la solución entera óptima del segundo subproblema debe ser al menos igual de grande. Por lo tanto, la solución existente es mejor que cualquier solución entera factible para el segundo subproblema, por lo que en realidad resuelve el problema original.

Por supuesto, hay varios problemas a resolver, incluidas las cuestiones fundamentales de decidir si ramificar o cortar y decidir cómo ramificar y cómo generar planos de corte. Obsérvese que el plano de corte introducido en el segundo subproblema no es

válido para el primer subproblema. Esta desigualdad se puede modificar para que sea válida para el primer subproblema utilizando una técnica de elevación de coeficientes.

2.1.1 Heurística primaria

En el problema de ejemplo 2.1, una vez encontrado un plano de corte apropiado fue posible podar el segundo subproblema por acotaciones. En este caso, la solución de la relajación de la programación lineal del primer subproblema es entera, proporcionando una buena solución. En muchos casos, se necesitan muchas iteracciones hasta que la solución de la relajación es entera. Por lo tanto, a menudo es útil tener buenas heurísticas para convertir la solución fraccional de una relajación en una buena solución entera que se puede utilizar para eliminar otros subproblemas.

2.1.2 Preprocesamiento

Un componente muy importante de un algoritmo práctico de ramificación y corte es el preprocesamiento para eliminar restricciones innecesarias, determinar una variable fija y simplificar el problema de otras formas.

2.1.3 Familia de planos de corte

Algunas familias de planos de corte son los cortes de Chvátal-Gomory o los cortes derivados de problemas de mochila con variables binarias. Cualquier desigualdad en variables binarias se puede representar como una desigualdad de mochila $\sum_{i \in N} a_i x_i \leq b$ con todo $a_i > 0$ para un subconjunto N de variables, donde se eliminan variables o se reemplazan una variable x_j por $1 - x_j$ cuando sea necesario. La estructura facial del politopo de mochila se puede utilizar para derivar desigualdades válidas para el problema. Por ejemplo, si $R \subseteq N$ con $\sum_{i \in R} a_i > b$, entonces $\sum_{i \in R} x_i \leq |R| - 1$ es una desigualdad válida. Además, si R es un conjunto mínimo (eliminar cualquier elemento de R hace que $\sum_{i \in R} a_i < b$), entonces la desigualdad define una cara del politopo de mochila correspondiente. Otras desigualdades pueden derivarse del politopo de mochila. Estas desigualdades se extienden a mochilas con variables enteras generales y una variable continua y a problemas binarios con cotas superiores generalizados.

Otra familia de desigualdades útiles son las de *elevación y proyección* o *desigualdades*

disyuntivas. Dada la región factible para un problema de programación binaria $S := \{x : Ax \leq b, x_i \in \{0,1\}, \forall i\}$, cada variable puede usarse para generar un conjunto de desigualdades disyuntivas. Sea $S_j^0 := \{x : Ax \leq b, 0 \leq x_i \leq 1, \forall i, x_j = 0\}$ y $S_j^1 := \{x : Ax \leq b, 0 \leq x_i \leq 1, \forall i, x_j = 1\}$. Entonces $S \subseteq S_j^0 \cup S_j^1$, por lo que se pueden generar desigualdades válidas para S encontrando desigualdades válidas para la envolvente convexa de $S_j^0 \cup S_j^1$. Estas desigualdades se generan al resolver problemas de programación lineal. Debido al coste, los cortes generalmente sólo se generan en el nodo raíz. No obstante, pueden ser muy eficaces computacionalmente.

2.1.4 Cuándo agregar planos de corte

La sobrecarga computacional de buscar planos de corte puede resultar prohibitiva. Por lo tanto, es común no buscar en todos los nodos del árbol. Una alternativa es buscar en cada octavo nodo o en cada nodo a una profundidad de un múltiplo de ocho en el árbol.

Generalmente, en cada nodo del árbol de ramificación y acotación se resuelve la relajación de programación lineal, se encuentran planos de corte, estos se añade a la relajación y se repite el proceso. Se suele llegar a un punto en el que el proceso decae, es decir, la solución de una relajación no es mucho mejor que las soluciones de las relajaciones anteriores. Es entonces aconsejable dejar de trabajar en este nodo y esta rama. Esta reducción gradual es más por la falta de conocimiento sobre la estructura poliédrica de la relajación, que por debilidad del método de planos de corte. En algunas implementaciones, se realiza un número fijo de rondas de búsqueda de planos de corte en un nodo, con quizás varias rondas realizadas en el nodo raíz y un número menor de rondas más abajo en el árbol.

La variante de *corte y ramificación* agrega planos de corte sólo en el nodo raíz del árbol. Por lo general, la implementación de dicho método requerirá un gran esfuerzo para generar planos de corte, requiriendo un tiempo mucho mayor que simplemente resolver la relajación en la raíz. Los beneficios de cortar y ramificar incluyen:

- Los cortes generados son válidos en la raíz, luego también son válidos en todo el árbol.
- La contabilidad se reduce, ya que las relajaciones son idénticas en todos los nodo.
- No se pierde tiempo generando planos de corte en otros nodos.

Cortar y ramificar es, en general, una técnica excelente para muchos problemas de programación entera, pero carece del poder que tiene ramificar y cortar en el caso de algunos problemas difíciles.

2.1.5 Elevación de cortes

Un corte añadido en un nodo del árbol de ramificación y corte puede no ser válido para otro subproblema. En caso de que el corte no se añada a ningún otro nodo se llama corte local. Este corte sólo afectará al subproblema actual y sus descendientes. El inconveniente de este método es el posible requisito de memoria de tener que almacenar una versión diferente del problema para cada nodo del árbol. Para que un corte sea válido en todo el árbol (corte global), es necesario hacerlo más arriba.

La elevación se puede considerar como un método para ajustar una restricción. Volviendo al problema de ejemplo una vez más, la restricción $x_1 + 2x_2 \leq 6$ es válida si $x_2 \leq 2$. Para extender esta restricción y que sea válida cuando $x_2 \geq 3$, considérese la desigualdad

$$x_1 + 2x_2 + \alpha(x_2 - 2) \leq 6. \tag{2.15}$$

Se desea elegir un α tan grande como sea posible mientras sea seguro que se trata de una desigualdad válida. Si $x_2 = 3$ entonces $x_1 \leq 1$, luego la desigualdad es válida para $x_2 = 3$ siempre que $\alpha \leq -1$. Si $x_2 = 1$, la desigualdad es válida siempre que $\alpha \geq -2$. Luego, la desigualdad es válida cuando $x_2 = 0$ siempre que $\alpha \geq -2.5$. La combinación de estas condiciones da que el rango válido es $-2 \leq \alpha \leq -1$. Las dos opciones extremas $\alpha = -1$ y $\alpha = -2$ dan las desigualdades válidas $x_1 + x_2 \leq 4$ y $x_1 \leq 2$, respectivamente. Otras opciones válidas para α dan desigualdades que son combinaciones convexas de estas dos. De esta manera, las desigualdades válidas para un nodo del árbol se pueden extender a desigualdades válidas en todos los nodos.

A menudo no es posible eliminar desigualdades porque la cota superior e inferior de los coeficientes entran en conflicto. Por supuesto, si una desigualdad es válida en el nodo raíz, entonces es válida en todo el árbol, por lo que no es necesario elevarla.

A continuación se describe el método de cálculo de coeficientes en el caso de problemas binarios. La desigualdad generada en un nodo del árbol generalmente sólo usará las variables que no están fijas en ese nodo. La elevación se puede utilizar para hacer que la desigualdad sea válida en cualquier nodo del árbol. Es necesario aplicar el proceso de elevación para cada variable que se ha fijado en el nodo, examinando el valor

opuesto para esa variable. Por ejemplo en la desigualdad

$$\sum_{j \in J} a_j x_j \leq h \text{ para un subconjunto } J \subseteq \{1, \ldots, n\} \tag{2.16}$$

es válida en un nodo donde x_i se ha fijado a cero, la desigualdad elevada toma la forma

$$\sum_{j \in J} a_j x_j + \alpha_i x_i \leq h \tag{2.17}$$

para unos escalares α_i. Estos escalararares deben maximizarse para que la desigualdad sea lo más fuerte posible. Maximizar los α_i requiere resolver otro problema de programación entera, por lo que puede ser necesario hacer una aproximación. Este proceso debe aplicarse sucesivamente a cada variable que se haya fijado en el nodo. El orden en el que se examinan las variables puede afectar la desigualdad final, y se pueden obtener otras desigualdades válidas elevando más de una variable a la vez.

2.1.6 Detalles de la implementación

Los detalles de la implementación incluye, entre otras cuestiones, la selección de nodos, la selección de variables de ramificación y los requisitos de almacenamiento. Normalmente, un algoritmo de ramificación y acotación almacena la solución en un nodo como una lista de índices de variables básicas. Los algoritmos de ramificación y corte pueden requerir más almacenamiento si los cortes se agregan localmente, porque sería necesario poder recrear la relajación actual en cualquier nodo activo con sólo las restricciones apropiadas. Si los cortes se agregan globalmente, basta con almacenar una única representación del problema.

Es posible corregir variables utilizando información sobre costes reducidos y el valor de la mejor solución entera factible conocida. Una vez que las variables se han corregido de esta manera, a menudo es posible corregir variables adicionales utilizando implicaciones lógicas. Para aprovechar al máximo la fijación de variables, la reconstrucción del nodo padre se realiza de la siguiente manera. Una vez que se ha seleccionado un nodo padre, no se divide inmediatamente en dos hijos, sino que se resuelve de nuevo utilizando el algoritmo de los planos de corte. Cuando finaliza el procedimiento de planos de corte, se ha reconstruido el vector de costes reducidos óptimo y se utiliza para fijar variables.

Muchas implementaciones de ramificación y corte utilizan un *conjunto de cortes*. Por lo general, se trata de un conjunto de restricciones que se han generado anteriormente y que no se incluyeron en la relajación o que se eliminaron posteriormente porque

ya no parecían estar activas. Es fácil verificar estos cortes para ver si hay violaciones y esto generalmente se hace antes de llamar rutinas de separación más complicadas. El conjunto de cortes también permite reconstruir el nodo padre de manera más eficiente, en parte porque disminuyen las dificultades de la reducción.

Una forma de permitir la solución de problemas mucho mayores es utilizar cálculo en paralelo. La naturaleza de los algoritmos de ramificación, y corte y ramificación y acotación hace posible que se realicen en paralelo: típicamente, una relajación de programación lineal se resuelve en un núcleo del ordenador.

2.2 Relajación lagrangiana

La relajación lagrangiana es un método que aproxima la solución de un problema difícil mediante la solución de un problema más sencillo. Este problema más sencillo se obtiene del problema original al eliminar una o varias de sus restricciones e introducirlas en la función objetivo. La restricción aparece en la función objetivo como penalización y su peso viene dado por el valor del multiplicador de Lagrange μ. En la práctica, este problema relajado suele ser más fácil de resolver que el problema original. Se trata de un método ampliamente usado en optimización discreta y, en particular, en problemas de redes. Todas las demostraciones de los resultados y proposiciones que aparecen en esta sección se encuentran en el libro publicado por *Ahuja*, *Magnanti* y *Orlin* [3] en 1993.

2.2.1 Problema de relajación. Ramificación y poda

Considérese el siguiente problema de programación entera:

$$\text{Minimizar } cx \tag{2.18}$$

sujeto a

$$x \in F. \tag{2.19}$$

En esta formulación F representa el conjunto factible de soluciones de un problema de programación entera. Esto es, el conjunto de soluciones $x = (x_1, x_2, \dots, x_J)$ del sistema

$$Ax = b, \tag{2.20}$$

$$x_j = 0 \text{ ó } 1 \text{ para } j = 1, 2, \dots, J. \tag{2.21}$$

Conceptualmente este problema se resuelve de forma trivial: se enumera todas las posibles combinaciones de las variables de decisión. Esto es, de todos los posibles vectores de ceros y unos $(x_1, x_2, \ldots, x_J)$ que satisfacen $\mathcal{A}x = b$ se elige el que devuelve un menor valor de la función objetivo cx. Debido al explosivo crecimiento de la combinatoria, la aplicabilidad de este procedimiento se reduce a problemas pequeños.

¿Cómo se pueden reducir este cálculo? Supóngase $F = F^1 \cup F^2$. Por ejemplo, se puede obtener F^1 añadiendo la restricción $x_1 = 0$ a F y F^2 añadiendo $x_1 = 1$. Véase que la solución óptima del conjunto factible F es la mejor de las soluciones óptimas de F^1 y F^2. Supóngase que ya se ha encontrado una solución óptima $\bar{x}$ que minimiza $\{cx : x \in F^2\}$ y el valor de su función objetivo es $z(\bar{x}) = 100$. El número de soluciones enteras potenciales en F^1 es entonces 2^{J-1}, aún serían prohibitivamente grande todas estas posibilidades, salvo que J sea pequeño.

En lugar de intentar resolver el problema entero en la región factible F^1 supóngase que se resuelve una versión relajada del problema (por ejemplo relajando restricciones de integralidad). En general, se obtiene una relajación eliminando algunas restricciones del modelo: por ejemplo, reemplazando las restricciones $x_j \geq 0$ enteras, por la restricción $x_j \geq 0$, o eliminando una o más restricciones de la forma $\alpha x = \beta$. En la relajación lagrangiana no sólo se eliminan restricciones, también cambia la función objetivo del problema. Sólo se requiere que se relaje algunas de las restricciones del problema y que el valor de la función objetivo de la relajación sea una cota inferior del valor de la función objetivo del problema original.

Sea x' una solución óptima a la relajación, y sea $z(x')$ el valor de la función objetivo de esta solución. Se consideran cuatro posibilidades:

1. La solución x' no existe porque el problema relajado no tiene una solución factible.
2. La solución x' se encuentra en F^1 (aunque se hayan relajado algunas de las restricciones).
3. La solución x' no está en F^1 y el valor de su función objetivo $z(x')$ satisface la desigualdad $z(x') \geq z(\overline{x}) = 100$.
4. La solución x' no se encuentra en F^1 y el valor de su función objetivo $z(x')$ satisface la desigualdad $z(x') < z(\overline{x}) = 100$.

Estas cuatro alternativas engloban todos los resultados posibles y son mutuamente excluyentes. Debe ocurrir uno de ellas.

En los casos 1 a 3, terminan los cálculos: se ha resuelto el problema original sobre el conjunto F, aunque no se ha resuelto explícitamente ningún problema entero (asumiendo que se ha obtenido la solución sobre el conjunto F^2 sin resolver un programa entero). En el caso 1, dado que la relajación del conjunto F^1 está vacío, el conjunto F^1 también está vacío, por lo que la solución $\overline{x}$ resuelve el problema entero original (totalmente). En el caso 2, como se ha encontrado la solución óptima en la relajación (y por lo tanto un superconjunto) del conjunto F^1, y esta solución está en F^1, también se ha encontrado la mejor solución en F^1; por lo tanto, $\overline{x}$ o x' es la solución al problema original (la solución con el menor valor de la función objetivo). Téngase en cuenta que en este caso se ha considerado (enumerado) implícitamente todas las soluciones de F^1 en el sentido de que se sabe que ninguna solución en este conjunto es mejor que $\overline{x}$. En el caso 3, la solución $\overline{x}$ tiene un valor de la función objetivo que es tan bueno como la mejor solución de una relajación de F^1, por lo que tiene un valor de función objetivo que es tan bueno como cualquier solución en F^1. Por tanto, $\overline{x}$ resuelve el problema original. Téngase en cuenta que en el caso 3 se ha utilizado la cota de la función objetivo para eliminar las soluciones en el conjunto F^1 procedentes de otras consideraciones.

En el caso 4, aún no se ha resuelto el problema original. Se puede resolver el problema de minimizar $\{cx : x \in F^1\}$ mediante algún método directo de programación entera o se puede dividir F^1 en dos conjuntos F^3 y F^4. Por ejemplo, se podría obtener F^3 de F imponiendo $x_1 = 0$ y $x_2 = 0$ y obtener F^4 estableciendo $x_1 = 0$ y $x_2 = 1$. Entonces se podría aplicar cualquier relajación o método directo para los problemas definidos en los conjuntos F^3 y F^4.

En un procedimiento general de ramificación y poda, se divide sistemáticamente la región factible F en subregiones $F^1, F^2, F^3, \ldots, F^K$. Sea $\overline{x}$ la mejor solución factible (respecto al valor de la función objetivo) que se ha obtenido en cálculos anteriores. Supóngase que para cada $k = 1, 2, \ldots, K$, F^k está vacío o x^k es una solución de una relajación del conjunto F^k y $c\overline{x} \leq cx^k$. Entonces ningún punto en ninguna de las regiones $F^1, F^2, F^3, \ldots, F^K$ puede mejorar el valor de la función objetivo que $\overline{x}$, por lo que $\overline{x}$ resuelve el problema de optimización original. Sin embargo, si $c\overline{x} > cx^k$ para alguna región F^k, se necesita subdividir esta región "ramificando" en algunas de las variables (es decir, dividiendo una subregión en dos estableciendo $x_j = 0$ o $x_j = 1$ para alguna variable j y definir así dos nuevas subregiones). Siempre que se satisfaga $c\overline{x} \leq cx^k$ para todas las subregiones (o se sabe que están vacías), se ha resuelto el problema original.

La finalidad de usar el método de ramificación y poda es encontrar una solución ópti-

ma resolviendo solo una pequeña cantidad de relajaciones. Para hacerlo, se necesitaría obtener buenas soluciones rápidamente y obtener buenas relajaciones de modo que el valor de la función objetivo $z(x^k)$ de la solución x^k a la relajación del conjunto F^k esté cerca de la solución óptima de F^k.

En la práctica, al implementar el procedimiento de ramificación y poda, es necesario tomar muchas decisiones con respecto al orden para elegir las subregiones, las variables a las que se ramificará para cada subregión y los mecanismos (por ejemplo, procedimientos heurísticos) que se podrían utilizar para encontrar "buenas" soluciones viables. También es necesario desarrollar buenas relajaciones que permitan obtener cotas inferiores efectivas (ajustadas): si las cotas inferiores son débiles, los casos 2 y 3 rara vez ocurrirán y el procedimiento de ramificación y poda degenerará en una enumeración total. Por otro lado, si las cotas son muy ajustadas, las relajaciones permitirán eliminar gran parte de la enumeración y desarrollar procedimientos de solución muy eficaces.

2.2.2 Técnicas de relajación lagrangiana

Considérese el siguiente modelo de optimización donde x es el vector de variables de decisión:

$$z^* = \text{mín}\, cx \tag{2.22}$$

$$\text{s. a} \quad \mathcal{A}x = b, \tag{2.23}$$

$$x \in X. \tag{2.24}$$

En este modelo (2.22)-(2.24) tanto las función objetivo tanto como las restricciones son lineales. Las variables de decisión x están restringidas a pertenecer a un conjunto dado X que, como se verá, a menudo modela redes de flujo. Por ejemplo, el conjunto de restricciones $X = \{x : \mathcal{N}x = q, 0 \leq x \leq u\}$ podrían ser todas las soluciones factibles para un problema de flujo de red con un vector de oferta/demanda q. O bien, el conjunto X puede contener los vectores de incidencia de todos los árboles de expansión de un grafo determinado. A menos que se indique lo contrario, se asume que el conjunto X es finito (por ejemplo, para problemas de flujo de red, es el conjunto finito de soluciones de árbol de expansión).

El procedimiento de relajación lagrangiana utiliza la idea de relajar las restricciones lineales explícitas llevándolas a la función objetivo mediante los multiplicadores de

Lagrange μ. Así el problema queda como:

$$\text{Minimizar} \quad cx + \mu(\mathcal{A}x - b) \tag{2.25}$$

$$s.a \quad x \in X, \tag{2.26}$$

que es una relajación lagrangiana y se refieren a la función

$$L(\mu) = \text{mín}\{cx + \mu(\mathcal{A}x - b) \,:\, x \in X\}, \tag{2.27}$$

como la función lagrangiana. Dado que al hacer la relajación lagrangiana se eliminan las restricciones $\mathcal{A}x - b$, la solución del subproblema lagrangiano no necesita ser factible para el problema original (2.22)-(2.24). ¿Es posible obtener alguna información útil sobre el problema original incluso cuando la solución al subproblema de Lagrange no es factible?

Lema 2.1 (Principio delimitador de la lagrangiana.). Para cualquier vector μ de los multiplicadores lagrangianos, el valor de la función lagrangiana $L(\mu)$ es una cota inferior del valor óptimo de la función objetivo z^* del problema de optimización original (2.22)-(2.24).

Para obtener la mejor cota inferior es necesario resolver el siguiente problema de optimización.

$$L^* = \max_{\mu} L(\mu) \tag{2.28}$$

que es el problema dual lagrangiano asociado al problema de optimización original (2.22)-(2.24).

Propiedad 2.1 ((Dualidad débil).). El valor óptimo de la función objetivo L^* del problema dual lagrangiano es siempre una cota inferior del valor óptimo de la función objetivo del problema (2.22)-(2.24) (es decir, $L^* \leq z^*$).

Esto proporciona cotas válidos para comparar los valores de la función objetivo del problema dual lagrangiano y la optimalidad de (2.22)-(2.24) con cualquier elección de los multiplicadores de Lagrange μ y cualquier solución factible x:

$$L(\mu) \leq L^* \leq z^* \leq cx. \tag{2.29}$$

Estas desigualdades garantizan la optimalidad de la solución del problema dual Lagrangiano o del problema original (2.22)-(2.24).

Propiedad 2.2 (Prueba de optimalidad.). (a) Supóngase que μ es un vector de multiplicadores lagrangianos y x es una solución factible al problema de optimización (2.22)-(2.24) que satisface la condición $L(\mu) = cx$. $L(\mu)$ es entonces una solución óptima del problema dual lagrangiano ($L^* = L(\mu)$) y x es una solución óptima del problema de optimización (2.22)-(2.24).

(b) Si para alguna elección del vector multiplicador lagrangiano μ, la solución x^* de la relajación lagrangiana es factible en el problema de optimización (2.22)-(2.24), entonces x^* es una solución óptima para el problema de optimización (2.22)-(2.24) y μ es una solución óptima para el problema dual lagrangiano.

Obsérvese que por el apartado (b) de la Propiedad 2.2, $L(\mu) = cx^* + \mu(\mathcal{A}x^* - b)$ y $\mathcal{A}x^* = b$. Por lo tanto, $L(\mu) = cx^*$ y el apartado (a) implica que x^* resuelve el problema (2.22)-(2.24) y μ resuelve el problema dual lagrangiano.

Como se indica en la Propiedad 2.2, el principio de delimitación proporciona una garantía ($L(\mu) = cx$) de que la solución factible x al problema de optimización (2.22)-(2.24) es una solución óptima. Incluso si $L(\mu) < cx$, tener la cota inferior permite medir qué tan lejos está una solución de la solución optima. Por ejemplo, si $[cx - L(\mu)]/L(\mu) \leq 0.05$ se sabe que el valor de la función objetivo de la solución factible x es como mucho un 5 % de lo óptima que podría ser. Este tipo de cota es muy útil en la práctica: permite evaluar el grado de subóptimalidad de soluciones dadas y permite terminar la búsqueda de una solución óptima cuando se tiene una solución lo suficientemente cerca de la optimalidad.

2.2.3 Relajación lagrangiana y restricciones de desigualdad

En el modelo de optimización (2.22)-(2.24), las restricciones: $\mathcal{A}x = b$ son todas restricciones de igualdad. En la práctica, a menudo se encuentran modelos, como el problema de la ruta más corta restringida, que se formulan de forma más natural en forma de desigualdad: $\mathcal{A}x \leq b$. El problema dual lagrangiano para estos problemas es una ligera variante del anterior:

$$L^* = \max_{\mu \geq 0} L(\mu) \tag{2.30}$$

El único cambio en el problema dual lagrangiano es que los multiplicadores están restringidos a ser no negativos.

Sin embargo, existe una diferencia sustancial entre relajar las restricciones de igualdad y las restricciones de desigualdad. Cuando se relajan las restricciones de desigualdad

$\mathcal{A}x \leq b$, si la solución x^* del subproblema lagrangiano satisface estas restricciones, no es necesario que sea óptima. Además de ser factible, esta solución debe satisfacer la condición de holgura complementaria $\mu(\mathcal{A}x^* - b) = 0$.

Propiedad 2.3. Supóngase la aplicación de la relajación lagrangiana en el problema de optimización (2.30) definido como minimizar $\{cx : \mathcal{A}x \leq b \text{ y } x \in X\}$ al relajar las desigualdades $\mathcal{A}x \leq b$. Supóngase, además, que para alguna elección del vector multiplicador lagrangiano μ, la solución x^* de la relajación lagrangiana (a) es factible en el problema de optimización (2.30) , y (b) satisface la condición de holgura complementaria $\mu(Ax^* - b) = 0$. Entonces x^* es una solución óptima al problema de optimización (2.30).

¿Son útiles las soluciones del subproblema de Lagrange para resolver el problema original? Las Propiedades 2.2 y 2.4 muestran que ciertas soluciones del subproblema lagrangiano resuelven el problema original. Se podría distinguir otros dos casos:

1. cuando las soluciones obtenidas mediante la relajación de las restricciones de desigualdad son factibles pero probablemente no son óptimas para el problema original (ya que no satisfacen la condición de holgura complementaria), y
2. cuando las soluciones a la relajación lagrangiana no son factibles en el problema original.

En el primer caso, es posible que las soluciones sean óptimas (puede usarse en ellas un procedimiento de ramificación y poda). En el segundo caso se podrían usar las soluciones obtenidas del subproblema lagrangiano como soluciones ”aproximadas” al problema original, incluso cuando no sean probadamente óptimas; en estos casos, se puede usar la relajación lagrangiana como un método heurístico para generar soluciones buenas en la práctica (debido a la información que proporciona la cota inferior lagrangiana). El desarrollo de estos métodos heurísticos depende en gran medida del contexto del problema estudiado.

2.2.4 Resolver el problema dual lagrangiano

Considérese el problema de la ruta más corta restringida en el que dada una red, cada arista (i, j) tiene asociado un coste c_{ij} y un tiempo de recorrido t_{ij}. Se desea encontrar la ruta más corta desde la fuente 1 hasta el sumidero n restringiendo la elección a aquellos caminos que no requieran más de T unidades de tiempo. Se formula como el

siguiente problema de programación entera

$$\text{mín} \sum_{(i,j)\in A} C_{ij} X_{ij} \tag{2.31}$$

$$s.a \quad \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(i,j)\in A\}} x_{ji} = \begin{cases} 1 & \text{para } i = 0 \\ 0 & \text{para } i \in N - \{1, n\}, \\ -1 & \text{para } i = n \end{cases} \tag{2.32}$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leq T, \tag{2.33}$$

$$x_{ij} = 0 \text{ ó } 1 \text{ para todo } (i,j) \in A. \tag{2.34}$$

Cuando se relaja la restricción (2.33), la lagrangiana $L(\mu)$ para el problema de la ruta más corta restringida es:

$$L(\mu) = \text{mín}\{c_P + \mu(t_P - T) \,:\, P \in \mathcal{P}\}. \tag{2.35}$$

Donde (P) es una ruta factible desde la fuente 1 hasta el sumidero n con coste $c_P = \sum_{(i,j)\in P} c_{ij}$ y tiempo de recorrido $t_P = \sum_{(i,j)\in P} t_{ij}$; $\mathcal{P}$ es la colección de todas las rutas factibles. El coste compuesto de la ruta (P) es la cantidad $c_P + \mu(t_P - T)$. Para un valor específico del multiplicador lagrangiano μ, se puede resolver $L(\mu)$ enumerando todas las rutas dirigidas en $\mathcal{P}$ y eligiendo la ruta con el menor coste compuesto. En consecuencia, se puede resolver el problema dual lagrangiano determinando $L(\mu)$ para todos los valores no negativos del multiplicador lagrangiano μ y eligiendo el valor que alcanza $\text{máx}_{\mu\geq 0} L(\mu)$.

Para encontrar el valor óptimo del multiplicador μ^* del problema dual lagrangiano, se necesita encontrar el punto más alto de la lagrangiana $L(\mu)$. Considérese el poliedro definido por aquellos puntos que se encuentran sobre o debajo de la función $L(\mu)$. Luego, geométricamente, se encuentra el punto más alto en un poliedro definido por la función $L(\mu)$, que es un problema lineal.

Esta situación es completamente general. Considérese el modelo de optimización genérico (2.30), definido como $\text{mín}\{cx \,:\, \mathcal{A}x = b, x \in X\}$ y supóngase que el conjunto $X = x^1, x^2, \ldots, x^K$ es finito. Al relajar las restricciones $\mathcal{A}x = b$, se obtiene la lagrangiana $L(\mu) = \text{mín}\{cx + \mu(\mathcal{A}x - b) \,:\, x \in X\}$. Por definición,

$$L(\mu) \leq cx^k + \mu(\mathcal{A}x^k - b) \qquad \text{para todo } k = 1, 2, \ldots, K. \tag{2.36}$$

En el espacio de costes compuestos y multiplicadores de Lagrange μ cada función $cx^k + \mu(\mathcal{A}x^k - b)$ es un hiperplano (si μ es bidimensional, es un plano). La función

multiplicadora lagrangiana $L(\mu)$ es la envolvente inferior de los hiperplanos $cx^k + \mu(\mathcal{A}x^k - b)$ para $k = 1, 2, ..., K$. En el problema dual lagrangiano, se desea determinar el punto más alto en esta curva. Se puede encontrar este punto resolviendo el problema de optimización

$$\text{Maximizar } w \tag{2.37}$$

$$\text{s. a} \quad w \leq cx^k + \mu(\mathcal{A}x^k - b) \text{ para todo } k = 1, 2, \ldots, K, \tag{2.38}$$

$$\mu \text{ no restringido,} \tag{2.39}$$

lo cual es un problema lineal. Véase este resultado como teorema.

Teorema 2.1. *El problema dual lagrangiano* $L^* = \max_\mu L(\mu)$ *con* $L(\mu) = \min\{cx^k + \mu(\mathcal{A}x - b) : x \in X\}$ *es equivalente al problema lineal* $L^* = \max\{w : w \leq cx^k + \mu(\mathcal{A}x^k - b)$ *para* $k = 1, 2, \ldots, K\}$.

Dado que, como se muestra en el teorema anterior, el problema dual lagrangiano es un problema lineal, podría resolverse aplicando la metodología de la programación lineal. Una de las desventajas de este proceder es que requiere la solución de una serie de problemas lineales que son bastante costosos computacionalmente. Una alternativa podría ser aplicar algún tipo de método de gradiente a la lagrangiana $L(\mu)$. La complicación adicional de esta metodología es que la lagrangiana $L(\mu)$ no es diferenciable. Es diferenciable siempre que la solución óptima del subproblema lagrangiano sea única; pero cuando el subproblema tiene dos o más soluciones, la lagrangiana generalmente no es diferenciable.

2.2.5 Ejemplo. Rutas más cortas restringidas

Considérese la red que se muestra en la Figura 2.2. Se desea encontrar la ruta más corta desde la fuente 1 hasta el sumidero 6, pero restringiendo la elección a aquellos caminos que no requieran más de $T = 10$. En este ejemplo se trata un problema de optimización difícil (el problema de la ruta más corta restringida (2.31)-(2.34) es un problema $\mathcal{NP}$ completo). A continuación, se resuelve eliminando una o más restricciones del problema (en este caso la restricción de tiempo) que hacen que el problema sea mucho más difícil de resolver.

Es un problema de ruta más corta con una restricción adicional (2.33). En lugar de resolver este problema directamente, supóngase un método indirecto combinando tiempo y coste en un único coste modificado; es decir, se tasa el tiempo en dinero.

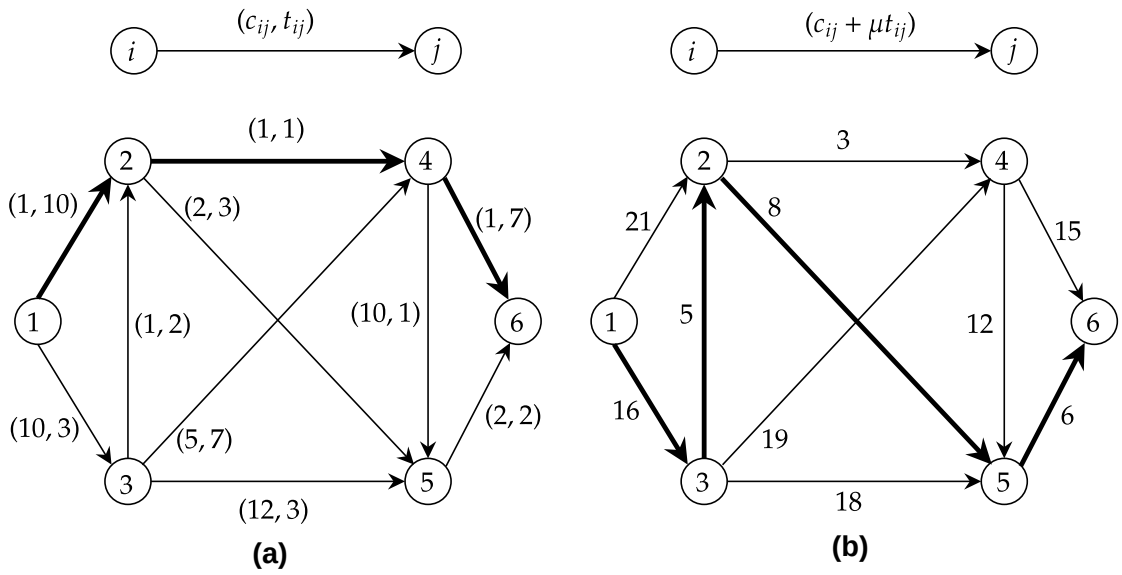

Figura 2.2: Problema del camino más corto con tiempo restringido: (a) problema del camino más corto restringido (las líneas en negrita indican el camino más corto para $\mu = 0$); (b) coste modificado $c + \mu t$ con multiplicador lagrangiano $\mu = 2$ (las líneas en negrita indican el camino más corto).

Entonces, en lugar de un límite de tiempo total en la ruta elegida, se establece un ”peaje” en cada arista proporcional al tiempo que se tarda en recorrer esa arista. Por ejemplo, se añade un coste de 2 por cada hora necesaria para atravesar una arista. Téngase en cuenta que si el peaje es cero, se ignora el tiempo por completo y el problema se convierte en el problema de ruta más corta estándar. Por otro lado, si el valor es muy alto, este cargo temporal se convierten en el coste dominante y simplemente se busca el camino más rápido.

Para un peaje μ, se resuelve el problema de ruta más corta con costes modificados $c_{ij} + \mu t_{ij}$. En la Figura 2.2 (a), si $\mu = 0$, la ruta más corta 1-2-4-6 tiene longitud 3. Este valor es una cota inferior de la longitud de la ruta más corta restringida, ya que ignora la restricción de tiempo. Ahora sea $\mu = 2$, la Figura 2.2 (b) muestra los costes modificados $c_{ij} + 2t_{ij}$. El camino más corto 1-3-2-5-6 tiene una longitud de 35. En este caso, el camino 1-3-2-5-6 que resuelve el problema modificado requiere 10 unidades para atravesarlo, por lo que es una solución factible para la ruta más corta. ¿Es un camino más corto restringido óptimo?

Sea (P) cualquier camino factible del problema del camino más corto restringido con

coste $c_p = \sum_{(i,j)\in P} c_{ij}$ y tiempo de recorrido $t_p = \sum_{(i,j)\in P} t_{ij}$, y sea $l(\mu)$ la longitud óptima del camino más corto con los costes modificados cuando se impone un peaje μ. Dado que el camino (P) es factible, el tiempo t_p que tarda en atravesarlo es como máximo $T = 10$. Respecto a los costes modificados $c_{ij} + \mu t_{ij}$, el coste $c_p + \mu t_p$ de la ruta (P) es el coste real de la ruta c_p más $\mu t_p \leq \mu T$. Por lo tanto, si se resta μT al coste modificado $c_p + \mu t_p$ de esta ruta, se obtiene una cota inferior $c_p + \mu t_p - \mu T = c_p + \mu(t_p - T) \leq c_p$. La longitud óptima es menor o igual al coste modificado de cualquier camino, $l(\mu) \leq c_p + \mu t_p$, luego $l(\mu) - \mu T$ es cota inferior de la longitud de cualquier camino factible (9) y, por tanto, de la longitud del camino más corto restringido. Debido a que este argumento es completamente general y se aplica a cualquier peaje $\mu \geq 0$, si se resta μT a la longitud óptima, se obtiene una cota inferior del coste óptimo del camino más corto restringido.

Para cualquier valor no negativo del peaje μ, la longitud $l(\mu)$ del camino más corto modificado con coste $c_{ij} + \mu t_{ij}$ menos el valor μT es una cota inferior de la longitud del camino más corto restringido.

Aplicado al ejemplo numérico de la Figura 2.2, para $\mu = 2$, el coste de la ruta más corta modificada es 35 unidades y, por lo tanto, $35 - 2(T) = 35 - 2(10) = 15$ es una cota inferior de la longitud del camino más corto restringido óptimo. Pero dado que la ruta $1 - 3 - 2 - 5 - 6$ es una solución viable para el problema de la ruta más corta restringida y su coste es igual a la cota inferior de 15 unidades, se puede estar seguro de que es la ruta más corta restringida óptima.

El ejemplo es afortunado, se ha encontrado un camino más corto restringido al resolver el subproblema lagrangiano para una elección del peaje μ. No siempre es así; sin embargo, el encontrar una cota inferior de la relajación lagrangiana proporciona con frecuencia información valiosa que se puede explotar algorítmicamente.

2.2.6 Técnica de optimización subgradiante

Al resolver problemas de optimización con una función objetivo no lineal $f(x)$ de un vector x n-dimensional, a menudo se usan variaciones de la siguiente idea clásica: se forma el gradiente $\nabla f(x)$ de f definido como un vector fila con componentes $(\partial f(x)/dx_1, \partial f(x)/dx_2, \ldots, \partial f(x)/dx_n)$. La derivada direccional de f en la dirección d satisface la igualdad

$$\lim_{\theta \to 0} \frac{f(x + \theta d) - f(x)}{\theta} = \nabla f(x) d. \tag{2.40}$$

Entonces, si se elige la dirección d de modo que $\nabla f(x)d > 0$ y se mueve en la dirección d con una "longitud de paso" θ lo suficientemente pequeña (cambiar x por $x + \theta d$), la función f crece. Este es el núcleo de los métodos de gradiente.

Supóngase que al resolver el problema dual lagrangiano, se está en un punto donde la función lagrangiana $L(\mu) = \text{mín}\{cx + \mu(\mathcal{A}x - b) : x \in X\}$ tiene una solución única $\overline{x}$, por lo que es diferenciable. Como $L(\mu) = c\overline{x} + \mu(\mathcal{A}\overline{x} - b)$ y la solución $\overline{x}$ sigue siendo óptima cuando se dan pequeños cambios en el valor de μ, el gradiente en este punto es $\mathcal{A}\overline{x} - b$, por lo que un método de gradiente cambiaría el valor de μ así:

$$\mu \leftarrow \mu + \theta(\mathcal{A}\overline{x} - b). \tag{2.41}$$

En esta expresión, θ es un tamaño de paso (un escalar) que especifica qué tan lejos se mueve uno en la dirección del gradiente. Téngase en cuenta que este procedimiento tiene una interpretación intuitiva. Si $(\mathcal{A}\overline{x} - b)_i = 0$, la solución x usa exactamente las unidades requeridas de la i-ésimo iteración, y se mantiene el multiplicador de Lagrange (el peaje) μ_i de esa iteración con su valor actual; si $(\mathcal{A}\overline{x} - b)_i < 0$, la solución x usa menos unidades que las disponibles en la i-ésima iteración y se disminuye el multiplicador de Lagrange μ_i en esa iteración; y si $(\mathcal{A}\overline{x} - b)_i > 0$, la solución x usa más unidades de las que hay disponibles de la i-ésima iteración y se aumenta el multiplicador de Lagrange μ_i en esa iteración.

Para resolver el problema dual de Lagrange, se deja que μ^0 sea cualquier elección inicial del multiplicador de Lagrange; se determinan los valores subsiguientes μ^k para $k = 1, 2, ..., K$ de los multiplicadores de Lagrange de la siguiente manera:

$$\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b). \tag{2.42}$$

En esta expresión, x^k es cualquier solución al subproblema de Lagrange cuando $\mu = \mu^k$ y θ_k es la longitud del paso en la k-ésima iteración.

Para asegurarse de que este método resuelve el problema dual de Lagrange, se debe tener cierto cuidado en la elección de los tamaños de paso. Si se eligen demasiado pequeños, el algoritmo se atascará en el punto actual y no convergerá; si se eligen tamaños de paso demasiado grandes, las iteraciones μ^k podrían sobrepasar la solución óptima y quizás incluso oscilar entre dos soluciones no óptimas. El siguiente compromiso garantiza que el algoritmo logre un equilibrio adecuado entre estos extremos y converja:

$$\theta_k \to 0 \quad \text{y} \quad \sum_{j=1}^{k} \theta_j \to \infty. \tag{2.43}$$

Eligiendo $\theta_k = 1/k$ se satisface estas condiciones que aseguran la convergencia del algoritmo en una solución óptima del problema dual.

Una variante importante del procedimiento de optimización subgradiente es una adaptación del "método de Newton" para resolver sistemas de ecuaciones no lineales. Supóngase, como antes, que $L(\mu^k) = cx^k + \mu^k(\mathcal{A}x^k - b)$; es decir, x^k resuelve el subproblema lagrangiano cuando $\mu = \mu^k$. Supóngase que x^k continúa resolviendo el subproblema lagrangiano a medida que se varía μ; o, dicho de otra manera, se hace una aproximación lineal de $r(\mu) = cx^k + \mu(\mathcal{A}x^k - b)$ a $L(\mu)$. Supóngase, además, que se conoce el valor óptimo L^* del problema dual lagrangiano (el cual no se conoce). Entonces uno podría moverse en la dirección del subgradiente hasta que el valor de la aproximación lineal sea exactamente igual a L^*.

En general, se establece una longitud de paso θ_k para que

$$r(\mu^{k+1}) = cx^k + \mu^{k+1}(\mathcal{A}x^k - b) = L^*, \tag{2.44}$$

o dado que, $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$,

$$r(\mu^{k+1}) = cx^k + [\mu^k + \theta_k(\mathcal{A}x^k - b)](\mathcal{A}x^k - b) = L^*. \tag{2.45}$$

Reordenando términos, recordando que $L(\mu^k) = cx^k + \mu(\mathcal{A}x^k - b)$, y tomando $\|y\| = \left(\sum_j y_j^2\right)^{1/2}$ (norma euclídea del vector y), se puede resolver para hallar la longitud de paso:

$$\theta_k = \frac{L^* - L(\mu^k)}{\|\mathcal{A}x^k - b\|^2}. \tag{2.46}$$

Dado que no se conoce el valor óptimo de la función objetivo L^* del problema dual lagrangiano (después de todo, eso es lo que se trata de encontrar), suele utilizarse la siguiente heurística para seleccionar la longitud de paso:

$$\theta_k = \frac{\lambda_k[UB - L(\mu^k)]}{\|\mathcal{A}x^k - b\|^2}. \tag{2.47}$$

En esta expresión, UB es una cota superior del valor óptimo de la función objetivo z^* del problema (2.30) y, por lo tanto, una cota superior de L^*; y λ_k es un escalar elegido (estrictamente) entre 0 y 2. Inicialmente , la cota superior es el valor de la función objetivo de cualquier solución factible conocida del problema (2.30). A medida que avanza el algoritmo, si se genera una solución factible mejor (es decir, de menor

coste) y se utiliza el valor de la función objetivo de esta solución en lugar de la cota superior UB. En realidad, se eligen los escalares λ_k comenzando con $\lambda_k = 2$ y luego reduciendo λ_k en un factor de 2 siempre que el mejor valor de la lagrangiana encontrado hasta ahora no haya aumentado en un número específico de iteraciones. Dado que esta versión del algoritmo no tiene criterios de detención convenientes, generalmente termina después de haber realizado un número específico de iteraciones.

Si se aplica la relajación lagrangiana a un problema con restricciones $\mathcal{A}x \leq b$ expresadas en forma de desigualdad en lugar de las restricciones de igualdad, los multiplicadores de Lagrange μ están restringidos a ser no negativos. La fórmula actualizadora $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$ puede hacer que uno o más de las componentes μ_i de μ se vuelvan negativas. Para evitar esta posibilidad, se modifica la fórmula actualizadora de la siguiente manera:

$$\mu^{k+1} = \left[\mu^k + \theta_k(\mathcal{A}x^k - b)\right]^+. \tag{2.48}$$

En esta expresión, la notación $[y]^+$ denota la "parte positiva" del vector y; es decir, el i-ésimo componente de $[y]^+$ es igual al máximo entre 0 e y_i. Dicho de otra manera, si la fórmula actualizadora $\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$ causase que el i-ésimo componente de μ_i fuera negativo, entonces simplemente se establecería el valor de este componente como cero. Luego, se implementan todos los demás pasos del procedimiento de subgradiante (es decir, la elección del tamaño de paso θ en cada paso y la solución de los subproblemas de Lagrange) exactamente igual que para los problemas con restricciones de igualdad. Para problemas con restricciones de igualdad y desigualdad, se usa una mezcla sencilla de las versiones de igualdad y desigualdad del algoritmo: siempre que la fórmula actualizadora para los multiplicadores de Lagrange cause que cualquier componente μ_i de μ correspondiente a una restricción de desigualdad sea negativa, se establece el valor de ese multiplicador en cero.

2.2.7 Relajación lagrangiana y programación lineal

Teorema 2.2. *Supóngase que se aplica la técnica de relajación lagrangiana a un problema de programación lineal* (P') *definido como* $\min\{cx : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\}$ *relajando las restricciones* $\mathcal{A}x = b$. *Entonces, el valor óptimo* L^* *del problema dual de Lagrange es igual al valor óptimo de la función objetivo de* (P').

El Teorema 2.2 anterior muestra que la técnica de relajación lagrangiana proporciona un método alternativo para resolver un problema de programación lineal. En lugar

de resolver el problema directamente usando cualquier algoritmo de programación lineal, se puede relajar un subconjunto de las restricciones y resolver el problema dual de Lagrange utilizando la optimización subgradiente y resolver una secuencia de problemas relajados. En algunas situaciones, el problema relajado es fácil de resolver, mientras el problema original no lo es. En estas situaciones, un algoritmo basado en la relajación lagrangiana es un método de solución atractivo.

A continuación, supóngase que se aplica la relajación lagrangiana a un problema de optimización discreto (P) definido como $\text{mín}\{cx : \mathcal{A}x = b, x \in X\}$. Sea el conjunto discreto $X = \{x : \mathcal{D}x \leq q, x \geq 0 \text{ y entero}\}$ para una matriz entera $\mathcal{D}$ y un vector de enteros q. En consecuencia, el problema (P) se convierte en:

$$z^* = \text{mín}\{cx : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0 \text{ y entero}\} \tag{P}$$

No se incurre en pérdida de generalidad al especificar el conjunto X de esta manera porque se puede formular casi todos los problemas de optimización discreta de la vida real como problemas de programación entera. Sea (LP) la relajación de programación lineal del problema (P) y sea z^0 el valor óptimo de su función objetivo. Es decir,

$$z^0 = \text{mín}\{cx : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\} \tag{LP}$$

El conjunto de soluciones factibles de (P) se encuentra dentro del conjunto de soluciones factibles de (LP), luego $z^0 \leq z^*$. Por lo tanto, la relajación del problema lineal proporciona una cota inferior válida del valor óptimo de la función objetivo de (P). En la Propiedad 2.1 se ha mostrado anteriormente que el problema dual de Lagrange también proporciona una cota inferior L^* del valor óptimo de la función objetivo de (P). Ahora se muestra que $z^0 \leq L^*$; es decir, la relajación lagrangiana proporciona una cota inferior que es al menos tan buena como la obtenida de la relajación del problema lineal. Se establece este resultado mostrando que el problema dual de Lagrange también resuelve un problema de programación lineal pero el espacio de solución de este problema está contenido dentro del espacio de solución del problema (LP). El problema de programación lineal que resuelve el problema dual de Lagrange utiliza la "convexificación" del espacio de solución $X = \{x : \mathcal{D}x \leq q, x \geq 0 \text{ y entero}\}$.

Sea $X = \{x^1, x^2, \dots, x^K\}$ un conjunto finito. Una solución x es una combinación convexa de soluciones $x^1, x^2, \dots, x^K$ si $x = \sum_{k=1}^{K} \lambda_k x^k$ para pesos no negativos $\lambda_1, \lambda_2, \dots, \lambda_K$ que satisfacen la condición $\sum_{k=1}^{K} \lambda_k = 1$. Sea $\mathcal{H}(X)$ la envolvente convexa de X (es decir, el conjunto de todas las combinaciones convexas de X). Se tienen las siguientes propiedades de $\mathcal{H}(X)$.

Propiedad 2.4. 1. El conjunto $\mathcal{H}(X)$ es un poliedro: puede expresarse como un conjunto definido por un número finito de desigualdades lineales.
2. Cada punto extremo de la solución del poliedro $\mathcal{H}(X)$ está situado en X, y si se optimiza una función objetivo lineal sobre $\mathcal{H}(X)$, alguna solución en X será la solución óptima.
3. El conjunto $\mathcal{H}(X)$ está contenido en el conjunto de soluciones $\{x : \mathcal{D}x \leq q, x \geq 0\}$.

Propiedad 2.5. El valor óptimo L^* de la función objetivo del problema dual lagrangiano es igual al valor óptimo de la función objetivo del programa lineal mín$\{cx : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$.

La versión convexa del problema (P) mín$\{cx : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$ es y se le llama (CP). El teorema anterior 2.2 muestra que L^* es igual al valor óptimo de la función objetivo del problema convexo. ¿Cuál es la relación entre el conjunto de soluciones factibles del problema convexo (CP) y la relajación del problema lineal (LP)?

Obsérvese que de la Propiedad 2.4 se deduce que $\mathcal{H}(X)$ está contenido en el conjunto $\{x : \mathcal{D}x \leq q, x \geq 0\}$, el conjunto de soluciones del problema (CP) dado por $\{x : \mathcal{A}x = b, x \in \mathcal{H}(X)\}$ está contenido en el conjunto de soluciones de (LP) dado por $\{x : \mathcal{A}x = b, \mathcal{D}x \leq q, x \geq 0\}$. Puesto que optimizar la misma función objetivo en un espacio de solución más pequeño no puede mejorar el valor de la función objetivo, se tiene que $z^0 \leq L^*$.

Propiedad 2.6. En un problema entero expresado en forma de minimización, la cota inferior obtenida por la relajación lagrangiana es siempre tan grande como la cota obtenida por la relajación del problema lineal; es decir, $z^0 \leq L^*$.

¿En qué situaciones la cota de la relajación lagrangiana será igual a la cota de programación lineal? Si el subproblema lagrangiano satisface la propiedad de integralidad, la cota lagrangiana será igual a la cota de programación lineal. Se dice que el subproblema de Lagrange mín$\{dx : \mathcal{D}x \leq q, x \geq 0$ y entero$\}$ satisface la propiedad de integralidad si tiene una solución óptima entera para cada elección de coeficientes de la función objetivo incluso si se relajan las restricciones de integralidad sobre las variables x. Téngase en cuenta que esta condición supone que los problemas mín$\{cx+\mu(\mathcal{A}x-b) : \mathcal{D}x \leq q, x \geq 0$ y entero$\}$ y mín$\{cx+\mu(\mathcal{A}x-b) : \mathcal{D}x \leq q, x \geq 0\}$ tienen los mismos valores óptimos de la función objetivo para cada elección del multiplicador de Lagrange μ. Por ejemplo, si las restricciones $\mathcal{D}x \leq q$ son las restricciones de balance de flujo de un problema de flujo de coste mínimo (o cualquiera de sus casos especiales, como el flujo máximo, la ruta más corta y los problemas de asignación), el

problema mín$\{cx + \mu(\mathcal{A}x - b) : \mathcal{D}x \leq q, x \geq 0\}$ siempre tendrá una solución óptima entera y la imposición de restricciones de integralidad a las variables no aumentará el valor óptimo de la función objetivo.

Propiedad 2.7. Si el subproblema lagrangiano del problema de optimización (P) satisface la propiedad de integralidad, entonces $z^0 = L^*$.

Para problemas que satisfacen la propiedad de integralidad, resolver el problema dual lagrangiano es equivalente a resolver la relajación de programación lineal del problema. En estas situaciones, la técnica de relajación lagrangiana no proporciona una cota mejor que la relajación de programación lineal. No obstante, la técnica de relajación lagrangiana aún podría tener un valor considerable, porque resolver el problema dual lagrangiano podría ser más eficiente que resolver la relajación de programación lineal directamente. Los problemas de optimización de redes quizás conformen el dominio de problemas más útil para explotar este resultado porque el subproblema lagrangiano en estos casos a menudo resulta ser un problema de flujo de coste mínimo o una de sus variantes.

En la mayoría de los casos, el valor óptimo de la función objetivo L^* del problema dual de Lagrange será estrictamente menor que el valor óptimo de la función objetivo z^* del problema (P).

2.3 Generación de columnas

2.3.1 Introducción

Sea un problema de programación entera máx$\{cx : x \in X\}$ con una región factible X que puede escribirse como la intersección de dos o más conjuntos con estructura $X = \cap_{k=0}^{K} X^k$ para un $K \geq 1$. Tómese el caso en el que las restricciones tienen la forma:

$$A^1x^1 + A^2x^2 + \cdots + A^Kx^K = b \tag{2.49}$$

$$D^1x^1 \leq d_1 \tag{2.50}$$

$$\vdots$$

$$D^Kx^K \leq d_K \tag{2.51}$$

$$x^1 \in Z_+^{n_1}, \ldots, x^K \in Z_+^{n_K} \tag{2.52}$$

de manera que los conjuntos $X^k = \{x^k \in Z_+^{n_k} : D^k x^k \leq d_k\}$ son independientes para $k = 1, \ldots, K$, y sólo las restricciones $\sum_{k=1}^{K} A^k x^k = b$ asocian entre sí los diferentes conjuntos de variables.

Dada la función objetivo máx $\sum_{k=1}^{K} c^k x^k$, dos herramientas que explotan con éxito esta estructura son la generación de cortes, con la cual se trata de validar desigualdades para los subconjuntos X^k, $k = 1, \ldots, K$; y la relajación lagrangiana, con la que se dualizan las restricciones $\sum_{k=1}^{K} A^k x^k = b$ para obtener el problema dual:

$$\min_{u} L(u), \tag{2.53}$$

donde

$$L(u) = \text{máx}\{\sum_{k=1}^{K} (c^k - uA^k)x^k + ub : x^k \in X^k \text{ para } k = 1, \ldots, K\}, \tag{2.54}$$

y el cálculo de $L(u)$ se divide en K subproblemas distintos:

$$L(u) = \sum_{k=1}^{K} \text{máx}\{(c^k - uA^k)x^k : x^k \in X^k\} + ub. \tag{2.55}$$

A continuación, se presenta una tercera herramienta que explota la forma de programación entera anterior. El procedimiento esencialmente resuelve un problema equivalente de la siguiente forma:

$$\text{máx}\{\sum_{k=1}^{K} \gamma^k \lambda^k : \sum_{k=1}^{K} B^k \lambda^k = \beta, \lambda^k \geq 0 \text{ entero para } k = 1, \ldots K\} \tag{2.56}$$

donde cada matriz B^k tiene un gran número de columnas, una por cada punto factible en X^k, y cada vector λ^k contiene las correspondientes variables. Todas las demostraciones de los resultados y proposiciones que aparecen en esta sección se encuentran en el libro publicado por *Wolsey* [4] en 1998.

Véase, por ejemplo, una formulación alternativa de este tipo para el problema de ubicación de instalaciones no capacitadas. Las localizaciones $j = 1, \ldots, n$ se corresponden con los índices $k = 1, \ldots, K$. Si el depósito j satisface la demanda del conjunto cliente S, entonces se establece $\lambda_S^j = 1$ por cada subconjunto no vacío $S \subseteq M$ de

clientes. Esto conduce a la siguiente formulación:

$$\text{mín} \sum_{j \in N} \sum_{S \neq \phi} \left(\sum_{i \in S} c_{ij} + f_j \right) \lambda_S^j \tag{2.57}$$

$$\sum_{j \in N} \sum_{S \neq \phi,\, i \in S} \lambda_S^j = 1 \text{ para } i \in M \tag{2.58}$$

$$\sum_{S \neq \phi} \lambda_S^j \leq 1 \text{ para } j \in N \tag{2.59}$$

$$\lambda_S^j \in \{0, 1\} \text{ para } \phi \neq S \subseteq M, j \in N. \tag{2.60}$$

El coste λ_S^j es el coste de abrir el depósito j y atender a los clientes en S desde el depósito j. Las restricciones (2.58) imponen que se atienda a cada cliente, mientras que las restricciones (2.59) aseguran que al deposito j se le asigna como mucho un subconjunto de clientes. En la práctica las restricciones (2.60) son tipicamente innecesarias.

Luego los problemas que se desean resolver son problemas enteros con un número enorme de variables, donde las columnas a menudo se describen implicitamente como los vectores de incidencia de ciertos subconjuntos de un conjunto que recorre subconjuntos clientes. Estas grandes formulaciones de un problema entero se denominan problemas maestros. A continuación, se considera cómo resolver la relajación de los problemas maestros y comparar esta relajación con la obtenida por la relajación lagrangiana, o por el uso de planos de corte. Finalmente, se considera qué hacer cuando la solución del problema lineal no es entera, y se debe recurrir a la enumeración lo que lleva a un problema entero de generación de columnas o a algoritmos de ramificación y acotación.

2.3.2 Reformulación de Dantzig-Wolfe de un problema entero

Considérese el problema entero de la forma:

$$x = \text{máx}\{ \sum_{k=1}^{K} c^k x^k \ : \ \sum_{k=1}^{K} A^k x^k = b, \ x^k \in X^k \text{ para } k = 1, \ldots, K \} \tag{2.61}$$

donde $X^K = \{x^k \in Z_+^{n_k}; D^k x^k \leq d_k\}$ para $k = 1, \ldots, K$. Asumiendo que cada conjunto X^k contiene un enorme aunque finito conjunto de puntos $\{x^{k,t}\}_{t=1}^{T_k}$, se tiene que

$$X^k = \left\{ x^k \in R^{n_k} : \begin{array}{c} x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{x,t}, \\ \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \\ \lambda_{k,t} \in \{0,1\} \text{ para } t = 1, \ldots, T_k \end{array} \right\}. \tag{2.62}$$

Sustituyendo ahora por x^k se llega al problema entero maestro equivalente:

$$z = \text{máx} \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t}) \lambda_{k,t} \tag{2.63}$$

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b \tag{2.64}$$

$$\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \text{ para } k = 1, \ldots, K \tag{2.65}$$

$$\lambda_{k,t} \in \{0,1\} \text{ para } t = 1, \ldots, T_K \text{ y } k = 1, \ldots, K. \tag{2.66}$$

Retomando el problema de ubicación de instalaciones no capacitadas, supóngase que se parte de la formulación débil:

$$\text{mín} \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j \tag{2.67}$$

$$\sum_{j \in N} x_{ij} = 1 \text{ para } i \in M \tag{2.68}$$

$$\sum_{i \in M} x_{ij} \leq m y_j \text{ para } j \in N \tag{2.69}$$

$$x \in B^{|M| \times |N|}, \; y \in B^{|N|}. \tag{2.70}$$

Donde:

$$X^k = \{(x_{1k}, \ldots, x_{mk}, y_k) : \sum_{i \in M} x_{ik} \leq m y_k, \; x_{ik} \in B^1 \text{ para } i \in M, \; y_k \in \{0,1\}\}. \tag{2.71}$$

Los puntos en X^k son $\{(x_S^k, 1)\}_{S\subseteq M}$ donde x_S^k es el vector de incidencia de $S \subseteq M$, y $(0,0)$ con variables asociadas λ_S^k, ν^k que conducen al problema maestro entero:

$$\text{mín} \sum_{j\in N} \left[\sum_{S\neq\phi} \left(\sum_{i\in S} c_{ij} + f_i \right) \lambda_S^j + f_j \lambda_\phi^j \right] \tag{2.72}$$

$$\sum_{j\in N} \sum_{S\neq\phi,\, i\in S} \lambda_S^j = 1 \text{ para } i \in M \tag{2.73}$$

$$\sum_{S\neq\phi} \lambda_S^j + \lambda_\phi^j \nu^j = 1 \text{ para } j \in N \tag{2.74}$$

$$\lambda_S^j \in \{0,1\} \text{ para } S \subseteq M,\ j \in N, \nu^j \in \{0,1\} \text{ para } j \in N. \tag{2.75}$$

Obsérvese que $f_j \geq 0$, luego ν^j es dominante sobre la variable λ_ϕ^j y λ_ϕ^j puede eliminarse. Si se coge ν^j como variable de holgura en (2.59) entonces las formulaciones (2.57)-(2.60) y (2.72)-(2.75) son idénticas.

2.3.3 Resolución del problema maestro

Para resolver la relajación lineal del problema maestro entero se utiliza un algoritmo de generación de columnas llamado problema maestro de programación lineal.

$$z^{PML} = \text{máx} \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t}) \lambda_{k,t} \tag{2.76}$$

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b \tag{2.77}$$

$$\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \text{ para } k = 1, \ldots, K \tag{2.78}$$

$$\lambda_{k,t} \geq 0 \text{ para } t = 1, \ldots, T^k,\ k = 1, \ldots, K \tag{2.79}$$

donde hay una columna $\begin{pmatrix} c^k x \\ A^k x \\ e_k \end{pmatrix}$ para cada $x \in X^k$. Como variables duales asociadas a las restricciones (2.77) se usará $\{\pi_i\}_{i=1}^m$ y para las restricciones (2.78), llamadas restricciones de convexidad, se usarán como variables duales $\{\mu_k\}_{k=1}^K$.

La idea es resolver el problema lineal con el algoritmo simplex primal. Sin embargo, el paso de tasar y elegir una columna que entre en la base debe modificarse debido al

enorme número de columnas. En lugar de tasar las columnas una a una, se resuelve el problema de encontrar la columna con el mayor coste reducido, que es en sí mismo un conjunto de K problemas de optimización.

Inicialización. Supóngase que se dispone de un subconjunto de columnas (al menos una por cada k), el cual forma un problema maestro lineal restringido factible:

$$\tilde{z}^{PMLR} = \text{máx } \tilde{c}\tilde{\lambda} \tag{2.80}$$

$$\tilde{A}\tilde{\lambda} = \tilde{b} \tag{2.81}$$

$$\tilde{\lambda} \geq 0 \tag{2.82}$$

donde $\tilde{b} = \begin{pmatrix} b \\ 1 \end{pmatrix}$, el conjunto de columnas disponible genera la submatriz:

$$\tilde{A} = \begin{pmatrix} A^1x^{1,1} & \dots & A^1x^{1,T_1} & A^2x^{2,1} & \dots & A^1x^{1,T_2} & \dots & A^Kx^{K,1} & \dots & A^Kx^{K,T_K} \\ 1 & \dots & 1 & & & & & & & \\ & & & 1 & \dots & 1 & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & 1 & \dots & 1 \end{pmatrix} \tag{2.83}$$

y $\tilde{c}$, $\tilde{\lambda}$ son los costes y variables correspondientes. Al resolver (2.80)-(2.82) se obtiene una solución optima primal $\tilde{\lambda}^*$ y una solución óptima dual $(\pi, \mu) \in R^m \times R^K$.

Factibilidad primal. Cualquier solición factible para el problema (2.80)-(2.82) es factible para el problema (2.76)-(2.79). En particular, $\tilde{\lambda}^*$ es solución factible de (2.76)-(2.79), y por tanto $\tilde{z}^{PML} = \tilde{c}\tilde{\lambda}^* = \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k \leq z^{PML}$.

Comprobación de optimalidad para el problema maestro lineal. Es necesario comprobar si (π, μ) es solución factible dual de (2.76)-(2.79). Esto implica comprobar que se cumple $c^k x - \pi A^k x - \mu_k \leq 0$ para cada $x \in X^k$ de cada k, de cada columna. En lugar de examinar cada punto de forma separada, se examinan todos los puntos X^k implícitamente resolviendo el subproblema de optimización:

$$\zeta_k = \text{máx}\{(c^k - \pi A^k)x - \mu_k \,:\, x \in X^k\} \tag{2.84}$$

Criterio de parada. Si $\zeta_k = 0$ para $k = 1, \dots, K$, entonces la solución dual (π, μ) es la solución dual factible del problema maestro lineal y por tanto $z^{PML} \leq \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k$. Como el valor de la solución factible primal $\tilde{\lambda}$ es igual al de su cota superior, $\tilde{\lambda}$ es óptimo para el problema maestro lineal.

Generación de una nueva columna. Si $\zeta_k > 0$ para un k, entonces la columna correspondiente a la solución óptima $\tilde{x}^k$ del subproblema tiene un coste reducido positivo. Introduciendo la columna $\begin{pmatrix} c^k\tilde{x} \\ A^k\tilde{x} \\ e_k \end{pmatrix}$ se obtiene un nuevo problema maestro linear restringido que se puede reoptimizar (por ejemplo con el algoritmo simplex primal).

Una cota dual (superior). Del subproblema se tiene $\zeta_k \geq (c^k - \pi A^k)x - \mu_k$ para todo $x \in X^k$. O lo que es lo mismo $(c^k - \pi A^k)x - \mu_k - \zeta_k \leq 0$ para todo $x \in X^k$. Luego estableciendo $\zeta = (\zeta_1, \ldots, \zeta_K)$, se tiene que $(\pi, \pi + \zeta)$ es solución factible dual del problema maestro lineal. Luego:

$$z^{PML} \leq \pi b + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta_k. \tag{2.85}$$

Todas estas observaciones conducen directamente a un algoritmo para el problema maestro lineal que termina cuando $\zeta_k = 0$ para $k = 1, \ldots, K$. Sin embargo, como en la relajación lagrangiana, es posible terminar antes.

Un criterio de parada alternativo. Si la solución del subproblema $(\tilde{x}^1, \ldots, \tilde{x}^K)$ satisface las restricciones $\sum_{k=1}^{K} A^k x^k = b$, entonces $(\tilde{x}^1, \ldots, \tilde{x}^K)$ es óptima. Esto es así puesto que $\zeta_k = (c^k - \pi A^k)\tilde{x}^k - \mu_k$ implica que $\sum_k c^k\tilde{x}^k = \sum_k \pi A^k\tilde{x}^k + \sum_k \mu_k + \sum_k \zeta_k = \pi b + \sum_k \mu_k + \sum_k \zeta_k$. Luego la solución factible tiene el mismo valor que la cota superior de z^{PML}.

2.3.4 Eficacia del problema maestro lineal

¿Cómo de eficaz es la relajación del problema lineal del problema maestro? ¿Hay alguna esperanza de que resuelva el problema original entero?

Proposición 2.1.

$$z^{PML} = \text{máx} \left\{ \sum_{k=1}^{K} c^k x^k \ : \ \sum_{k=1}^{K} A^k x^k = b, x^j \in conv(X^k) \quad \text{para} \quad k = 1, \ldots, K \right\}. \tag{2.86}$$

Cuando un problema entero se puede descomponer, dos herramientas para resolverlo son la relajación lagrangiana y los planos de corte. Sea w_L el valor de la solución dual

lagrangiana cuando la restricción $\sum_{k=1}^{K} A^k x^k = b$ está dualizada, y sea z^C el valor obtenido cuando se añaden los planos de corte a la relajación del problema lineal entero usando un algoritmo de separación exacto para cada uno de los conjuntos $conv(X^k)$ para $k = 1, \ldots, K$. Tanto estos procedimientos como la generación de columnas son de cierta forma equivalentes en tanto en cuanto conducen hasta las mismas cotas de la solución dual.

Proposición 2.2.

$$z^{PML} = w_L = z^C \tag{2.87}$$

Como el subproblema resuelto tanto en el algoritmo de generación de columnas como en la relajación lagrangiana son problemas de optimización sobre X^k, puede visualizarse la generación de columnas como un algoritmo para resolver la relajación lagrangiana en el cual las variables duales π se actualizan resolviendo el problema maestro linal restringido. Este algoritmo se usa a menudo junto al algoritmo del subgradiante para resolver la relajación lagrangiana que se basa en un procedimiento de actualización mucho más simple.

Por otro lado, si se usan los planos de corte, aunque las cotas obtenidas sean potencialmente las mismas, se resuelven problemas de separación en $conv(X^K)$ en lugar de problemas de optimización. Aunque la complejidad de los problemas de optimización y los problemas de separación en $conv(X^k)$ sea teoricamente la misma, la elección depende de la dificultad relativa para resolver los dos problemas, así como de la convergencia de los algoritmos de generación de columnas y plano de corte en la práctica.

2.3.5 Resolución del problema del viajante simétrico mediante generación de columnas: un ejemplo ilustrativo

A continuación, se aplica el algoritmo anteriormente expuesto para resolver el problema maestro lineal de un problema en el cual hay sólo un subproblema. Considérese el problema del viajante simétrico que puede formularse como:

$$\text{mín} \left\{ \sum_{e \in E} c_e x_e \; : \; \sum_{e \in \delta(i)} x_e = 2 \text{ para } i \in N, \; x \in X^1 \right\}, \tag{2.88}$$

donde:

$$X^1 = \left\{ x \in Z_+^m : \begin{array}{c} \sum_{e\in\delta(1)} x_e = 2, \\ \sum_{e\in E(S)} x_e \leq |S| - 1 \text{ para } \phi \subset S \subset N \setminus \{1\}, \\ \sum_{e\in E} x_e = n \end{array} \right\}, \tag{2.89}$$

es el conjunto de vectores de incidencia de 1-árboles.

Escribiendo $x_e = \sum_{t:e\in E^t} \lambda_t$, donde $G^t = (N, E^t)$ es el t-ésimo 1-árbol, las restricciones que afectan al grado se convierten en $\sum_{e\in\delta(i)} x_e = \sum_{e\in\delta(i)} \sum_{t:e\in E^t} \lambda_t = \sum_t d_i^t \lambda_t = 2$ donde d_i^t es el grado del nodo i en el 1-árbol G^t. Por lo tanto, el correspondiente problema maestro lineal es:

$$\text{mín} \sum_{t=1}^{T_1} (cx^t)\lambda_t, \tag{2.90}$$

$$\sum_{t=1}^{T_1} d_i^t \lambda_t = 2 \text{ para } i \in N, \tag{2.91}$$

$$\sum_{t=1}^{T_1} \lambda_t = 1, \tag{2.92}$$

$$\lambda \in R_+^T, \tag{2.93}$$

al que se asocia variables duales $\{u_i\}_{i=1}^n$ en las restricciones de grado, y variables duales μ a las restricciones de convexidad. El subproblema correspondiente es:

$$\zeta_1 = \text{mín} \left\{ \sum_{e\in E} (c_e - u_i - u_j) x_e - \mu : x \in X^1 \right\}, \tag{2.94}$$

puesto que el coste reducido del 1-árbol $cx^t - \sum_{i\in N} d_i^t u_i - \mu = cx^t - \sum_{i\in N} u_i \sum_{e\in\delta(i)} x_e^t - \mu = \sum_{e\in E} (c_e - u_i - u_j) x_e^t - \mu$, donde x_e^t con $e \in E$ son las variables correspondientes a las aristas del 1-árbol G^t, y $e = (i, j)$ para $e \in E$.

Debido a que se trabaja con 1-árboles, $d_i^t = 2$ para todo t, y por tanto, (2.91) es dos veces (2.92), luego se puede desechar la restricción (2.92).

Considérese una instancia del problema del viajante simétrico con matriz de distan-

cias:

$$c_e = \begin{pmatrix} \cdot & 7 & 2 & 1 & 5 \\ & \cdot & 3 & 6 & 8 \\ & & \cdot & 4 & 2 \\ & & & \cdot & 9 \\ & & & & \cdot \end{pmatrix} \tag{2.95}$$

Se parte de un problema maestro lineal restringido que tiene 7 columnas correspondientes a un recorrido de longitud 28 y seis 1-árboles elegidos arbitrariamente:

$$\begin{array}{lccccccccc} \text{mín} & 28\lambda_1 & +25\lambda_2 & +21\lambda_3 & +19\lambda_4 & +22\lambda_5 & +18\lambda_6 & +28\lambda_7 & & \\ & 2\lambda_1 & +2\lambda_2 & +2\lambda_3 & +2\lambda_4 & +2\lambda_5 & +2\lambda_6 & +2\lambda_7 & = & 2 \\ & 2\lambda_1 & +2\lambda_2 & +2\lambda_3 & +1\lambda_4 & +1\lambda_5 & +2\lambda_6 & +3\lambda_7 & = & 2 \\ & 2\lambda_1 & +3\lambda_2 & +2\lambda_3 & +3\lambda_4 & +2\lambda_5 & +3\lambda_6 & +1\lambda_7 & = & 2 \\ & 2\lambda_1 & +2\lambda_2 & +3\lambda_3 & +3\lambda_4 & +3\lambda_5 & +1\lambda_6 & +1\lambda_7 & = & 2 \\ & 2\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +2\lambda_5 & +2\lambda_6 & +3\lambda_7 & = & 2 \\ & & & \lambda & \geq & 0. & & & & \end{array} \tag{2.96}$$

La solución es $\lambda = (0, 0, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ con coste 22,5 y su solución dual es $u = (\frac{151}{8}, \frac{1}{3}, -\frac{5}{3}, \frac{2}{3}, 0)$. La correspondiente matriz de costes reducidos para este subproblema es:

$$\begin{pmatrix} \cdot & -\frac{87}{8} & -\frac{91}{8} & -\frac{133}{8} & -\frac{111}{8} \\ & \cdot & \frac{19}{2} & \frac{33}{4} & 9 \\ & & \cdot & \frac{43}{4} & \frac{15}{2} \\ & & & \cdot & \frac{41}{4} \\ & & & & \cdot \end{pmatrix}. \tag{2.97}$$

El 1-árbol optimal es $x_{14} = x_{15} = x_{24} = x_{25} = x_{35} = 1$ con $\zeta = -\frac{23}{4}$. Por lo que $22,5 + \zeta = 16.75 \leq z^{LP} \leq 22.5$.

Comienza una nueva iteracción introduciendo este 1-árbol como una nueva columna en el problema maestro restringido con coste 22, y grados $(2, 2, 1, 2, 3)$. La solución del nuevo problema lineal es $\lambda = (0, 0, \frac{1}{3}, 0, 0, \frac{1}{3}, 0, \frac{1}{3})$ con coste 20.333 y solucion dual $u = (\frac{65}{6}, \frac{1}{3}, -\frac{5}{3}, \frac{2}{3}, 0)$.

La correspondiente matrix de costes reducidos para el subproblema es:

$$\begin{pmatrix} \cdot & -\frac{25}{8} & -\frac{43}{8} & -\frac{21}{2} & -\frac{35}{6} \\ & \cdot & \frac{13}{3} & 5 & \frac{23}{3} \\ & & \cdot & 5 & \frac{11}{3} \\ & & & \cdot & \frac{25}{3} \\ & & & & \cdot \end{pmatrix}. \tag{2.98}$$

El 1-árbol óptimo es $x_{13} = x_{14} = x_{23} = x_{24} = x_{35} = 1$ con $\zeta = -\frac{14}{3}$. La cota inferior es $20,333 - \frac{14}{3} = 15,667$ y no es tan bueno como el obtenido anteriormente. Por lo tanto, se tiene $16,75 \leq Z^{LP} \leq 30,333$.

De nuevo, se introduce este 1-árbol como una nueva columna en el problema maestro restringido con coste 14 y grados $(2, 2, 3, 2, 1)$. La nueva solución del problema lineal es $\lambda = (0, 0, 0, 0, 0, 0, 0, \frac{1}{2}, \frac{1}{2})$ con coste 18 y solución dual $u = (13, 0, -4, 0, 0)$.

La correspondiente matriz de costes reducidos para el subproblema es:

$$\begin{pmatrix} \cdot & -6 & -7 & -12 & -8 \\ & \cdot & 7 & 6 & 8 \\ & & \cdot & 8 & 6 \\ & & & \cdot & 9 \\ & & & & \cdot \end{pmatrix}. \tag{2.99}$$

El 1-árbol óptimo es $x_{14} = x_{15} = x_{23} = x_{24} = x_{35} = 1$ con $\zeta = -1$. La cota inferior de la solución aumenta en $18 - 1 = 17$. Como este 1-árbol es un tour, del principio de parada alternativo se deduce que es óptimo. De forma alternativa, se puede comprobar que el coste real es 17.

3 Problema de flujo de mínimo coste: algoritmo simplex en redes

3.1 Introducción

El método simplex para resolver problemas de programación lineal es quizás el algoritmo más poderoso para resolver problemas de optimización restringida. Debido a que los problemas de flujo de red tienen cierta estructura especial, es importante saber si el método simplex podría competir con otros métodos ”combinatorios” que explotan la estructura de red subyacente, como las muchas variantes del algoritmo de ruta más corta. El método simplex general, cuando se implementa de una manera que no explota la estructura de red subyacente, no es una solución competitiva para resolver problemas de flujo de mínimo coste. Afortunadamente, si se interpreta los conceptos centrales del método simplex apropiadamente como operaciones en redes, puede adaptarse y agilizarse el método para explotar la estructura del problema de flujo de mínimo coste, produciendo un algoritmo que es muy eficiente. El algoritmo simplex en redes es así una adaptación del método simplex.

El concepto central en el algoritmo simplex en redes es las soluciones de árbol de expansión, que son soluciones que se obtienen al fijar a cero la capacidad del flujo de cada arco que no está en un árbol de expansión. El problema de flujo de mínimo coste tiene siempre al menos una solución óptima de árbol de expansión y es posible encontrar una solución óptima de árbol de expansión ”moviéndose” de una solución a otra, sustituyendo en cada paso un arco del árbol por otro que no pertenece al árbol. Este método se conoce como algoritmo simplex en redes porque los árboles de expansión corresponden a las soluciones básicas factibles de programación lineal, y el movimiento de una solución de árbol de expansión a otra corresponde a una operación pivote del método simplex general. Todas las demostraciones de los resultados

y proposiciones que aparecen en este capítulo se encuentran en el libro publicado por *Ahuja*, *Magnanti* y *Orlin* [3] en 1993.

3.2 Problemas de flujo con coste mínimo

El modelo de flujo de mínimo coste es el más fundamental de todos los problemas de flujo en redes. El problema es fácil de enunciar: se desea determinar el envio de un servicio al menor coste a través de una red para satisfacer las demandas en ciertos nodos de los suministros disponibles en otros nodos.

Sea $G = (N, A)$ una red dirigida definida por un conjunto N de nodos y un conjunto A de m arcos dirigidos. Cada arco $(i, j) \in A$ tiene asociado un coste c_{ij} que denota el coste por unidad de flujo en ese arco, luego el coste del flujo varía linealmente con la cantidad de flujo. A cada arco $(i, j) \in A$ también se asocia una capacidad u_{ij} que es la cantidad máxima de flujo en cada arco y un límite inferior l_{ij} que es la cantidad mínima de flujo que atraviesa ese arco. A cada nodo $i \in N$ se le asocia un número entero $b(i)$ que representa la oferta/demanda. Si $b(i) > 0$, el nodo i es un nodo de suministro; si $b(i) < 0$, el nodo i es un nodo demandante con una demanda $-b(i)$; y si $b(i) = 0$, el nodo i es un nodo de transbordo. La variable de decisión en el problema de flujo de mínimo coste es el flujo x_{ij} en el arco $(i, j) \in A$. El problema de flujo de mínimo coste es un modelo de optimización que se formula así:

$$\text{Minimizar} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.1}$$

$$\text{s. a} \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \quad \text{para todo } i \in N, \tag{3.2}$$

$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \text{para todo } (i, j) \in A, \tag{3.3}$$

donde $\sum_{i=1}^{n} b(i) = 0$. El problema de flujo con mínimo coste en forma matricial es el siguiente:

$$\text{Minimizar } cx \tag{3.4}$$

$$\text{s. a} \quad \mathcal{N}x = b, \tag{3.5}$$

$$l \leq x \leq u. \tag{3.6}$$

donde $\mathcal{N}$ es una matriz $n \times m$ llamada matriz de incidencia de los arcos en los nodos del problema de flujo con coste mínimo. La columna $\mathcal{N}_{ij}$ tiene un $+1$ en la i-ésima fila, un -1 en la j-ésima fila; el resto de sus filas son cero.

Las restricciones en (3.2) son restricciones de balance de flujo. El primer término de esta restricción representa el flujo de salida total del nodo (es decir, el flujo que emana del nodo) y el segundo término representa el flujo de entrada total del nodo (es decir, el flujo que ingresa al nodo). La restricción del balance de flujo establece que la salida menos la entrada debe ser igual a la oferta/demanda de ese nodo. Si el nodo es un nodo de suministro, su flujo de salida excede su flujo de entrada; si el nodo es un nodo de demanda, su entrada excede su salida; y si el nodo es un nodo de transbordo, su flujo de salida es igual a su flujo de entrada. El flujo también debe satisfacer el límite inferior y las restricciones de capacidad (3.3), que son las restricciones de límite de flujo. Los límites de flujo típicamente modelan capacidades físicas o restricciones impuestas en los rangos de operación de los flujos. En la mayoría de las aplicaciones, los límites inferiores de los flujos en arco son cero; por lo tanto, si no se establecen límites inferiores para ningún problema, se asumen que tienen valor cero.

Se asume, en general, que los datos son números enteros, esto es el supuesto de integralidad. El cual, no es restrictivo para la mayoría de las aplicaciones porque siempre se puede transformar datos racionales en datos enteros multiplicándolos por una constante. Además, es necesario convertir números irracionales en números racionales para representarlos en un ordenador.

Se enuncia a continuación varias variantes del problema (3.1) - (3.3) que aparecen posteriormente.

Problema del camino más corto.

Para este problema se desea encontrar el camino de mínimo coste (o longitud) que va desde un nodo fuente s a un nodo sumidero t, cada arco $(i, j) \in A$ tiene un coste (longitud) asociado c_{ij}.

Problema de circulación.

El problema de circulación es un problema de flujo de coste mínimo con sólo nodos de transbordo; es decir, $b(i) = 0$ para todo $i \in N$. En este caso se desea encontrar un flujo factible que respete las cotas superior e inferior l_{ij} y u_{ij} impuestas sobre las capacidades de los flujos de los arcos x_{ij}. Dado que nunca se introduce ningún flujo exógeno en la red ni se extrae ningún flujo de ella, todo el flujo circula por la red. Se desea encontrar la circulación que tenga el coste mínimo.

Problemas de flujo con costes convexos.

En el problema de flujo con mínimo coste, el coste del flujo en cualquier arco varía linealmente con la cantidad de flujo. Los problemas de flujo de costes convexos tienen una estructura de costes más general: el coste es una función convexa de la cantidad de flujo.

Problemas de flujo multiservicio.

El problema de flujo con coste mínimo modela el flujo de un único servicio a través de una red. Los problemas de flujo multiservicio surgen cuando varios servicios utilizan la misma red subyacente. Los servicios pueden diferenciarse por sus características físicas o simplemente por sus pares de origen-destino. Los diferentes servicios tienen diferentes orígenes y destinos, y los servicios tienen restricciones de balance de flujo independientes en cada nodo. Sin embargo, comparten las capacidades de los arcos comunes que llevan varios servicios. De hecho, el problema esencial que aborda el problema del flujo multiservicio es la asignación de la capacidad de cada arco a los servicio individuales de una manera que minimice los costes generales del flujo.

A continuación, se enumeran varios tipos de transformaciones en redes será útil para posteriores desarrollos.

División de nodos.

Esta transformación divide cada nodo i en dos nodos i' e i'' correspondientes a las funciones de entrada y salida del nodo. Esta transformación reemplaza cada arco original (i, j) por un arco (i', j'') del mismo coste y capacidad. También agrega un arco (i'', i') de coste cero y capacidad infinita para cada i. La entrada del nodo i (es decir, el nodo i'') recibe todo el flujo de entrada del nodo, la salida (es decir, el nodo i') envía todo el flujo de salida del nodo, y el arco adicional (i'', i') lleva el flujo desde la entrada a la salida. La Figura 3.1 ilustra la red resultante cuando se lleva a cabo la transformación de división de nodos para todos los nodos de un red. Se define las ofertas/demandas de los nodos de la red transformada de acuerdo con los siguientes tres casos:

1. Si $b(i) > 0$, entonces $b(i'') = b(i)$ y $b(i') = 0$.
2. Si $b(i) < 0$, entonces $b(i'') = 0$ y $b(i') = b(i)$.

3. Si $b(i) = 0$, entonces $b(i') = b(i'') = 0$.

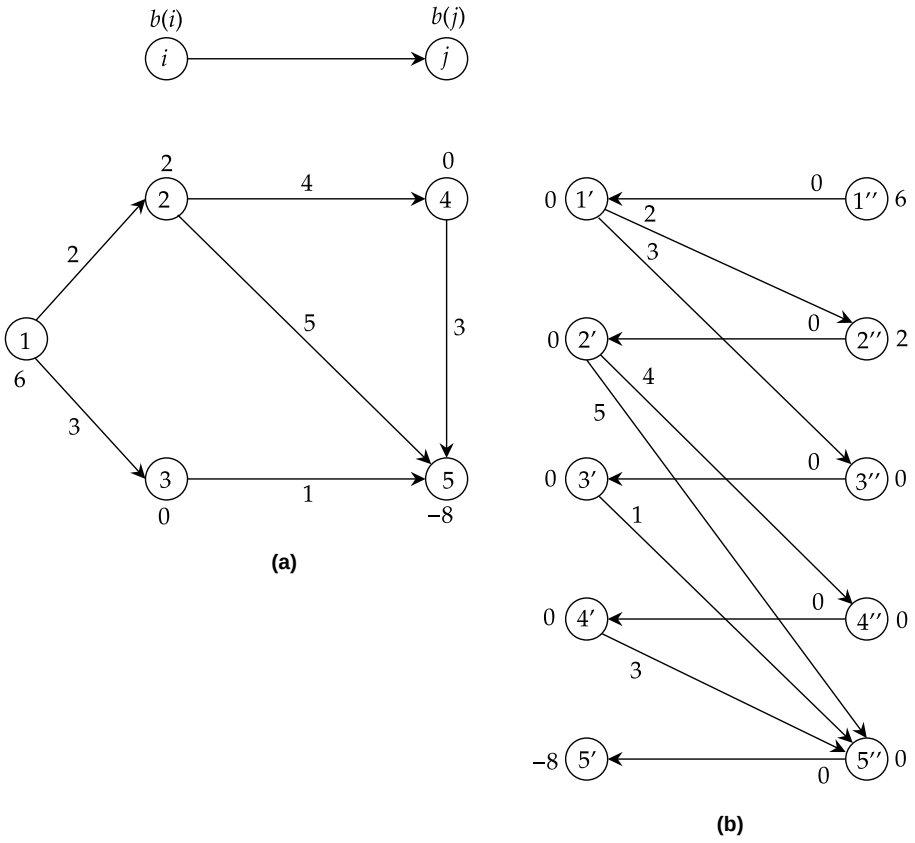

Figura 3.1: División de nodos: red original(a); red transformada (b).

Es fácil mostrar una correspondencia uno a uno entre un flujo en la red original y el flujo correspondiente en la red transformada; además, los flujos en ambas redes tienen el mismo coste.

La transformación de división de nodos permite modelar numerosas aplicaciones en una variedad de dominios de problemas prácticos. Por ejemplo, se puede usar la transformación para manejar situaciones en las que los nodos y los arcos tienen capacidades y costes asociados. En estas situaciones, cada unidad de flujo que pasa por un nodo i incurre en un coste c_i y el flujo máximo que puede pasar a través del nodo es u_i. Puede reducirse este problema a la forma estándar de "flujo de arco" del problema de flujo de red realizando la división de nodos y dejando que c_i y u_i sean el coste y la capacidad respectivamente del arco (i'', i').

Trabajar con costes reducidos

Supóngase que cada nodo $i \in N$ tiene asociado un número $\pi(i)$ que es el potencial del nodo. Respecto a los potenciales del nodo $\pi = (\pi(1), \pi(2), \dots, \pi(n))$, se define el coste reducido c_{ij}^{π} de un arco (i, j) como:

$$c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j) \tag{2.3}$$

Con frecuencia se trabaja con costes reducidos c_{ij}^{π} en lugar de los costes reales c_{ij}. Así, es importante entender la relación entre las funciones objetivo $z(\pi) = \sum_{(i,j)\in A} c_{ij}^{\pi} x_{ij}$ y $z(0) = \sum_{(i,j)\in A} c_{ij} x_{ij}$. Supóngase que inicialmente $\pi = 0$, luego se incrementa el potencial de nodo del nodo k hasta $\pi(k)$. La definición (2.3) de costes reducidos supone que este cambio reduce el coste reducido de cada unidad de flujo que sale del nodo k en $\pi(k)$ y aumenta el coste reducido de cada unidad de flujo que ingresa al nodo k en $\pi(k)$. Por tanto, la disminución total de la función objetivo es igual a $\pi(k)$ multiplicado por el flujo de salida del nodo k menos el flujo de entrada del nodo k. Por definición, la salida menos la entrada es igual a la oferta/demanda del nodo. En consecuencia, aumentar el potencial del nodo k en $\pi(k)$ disminuye el valor de la función objetivo en $\pi(k)b(k)$ unidades. Iterando para cada nodo se llega a:

$$z(0) - z(\pi) = \sum_{i\in N} \pi(i)b(i) = \pi b. \tag{3.7}$$

Para un potencial de nodo dado π, πb es una constante. Por tanto, un flujo que minimiza $z(\pi)$ también minimiza $\pi(0)$.

Propiedad 3.1. El problema de flujo de mínimo coste con costes en los arcos c_{ij} o c_{ij}^{π} tiene las mismas soluciones óptimas. Además, $z(\pi) = z(0) - \pi b$.

A continuación, se estudia los efectos de trabajar con costes reducidos en los costes de ciclos y caminos. Sea W un ciclo dirigido en G. Entonces:

$$\sum_{(i,j)\in W} c_{ij}^{\pi} = \sum_{(i,j)\in W} (c_{ij} - \pi(i) + \pi(j)), \tag{3.8}$$

$$= \sum_{(i,j)\in W} c_{ij} + \sum_{(i,j)\in W} (\pi(j) - \pi(i)), \tag{3.9}$$

$$= \sum_{(i,j)\in W} c_{ij}. \tag{3.10}$$

La última igualdad (3.10) resulta de que en cada ciclo dirijido W, la expresión $\sum_{(i,j)\in W}(\pi(j) - \pi(i))$ suma cero para cada nodo i en el ciclo W porque $\pi(i)$ ocurre una vez con signo

positivo y otra con signo negativo. De forma similar, si P es un camino dirigido desde el nodo k hasta el nodo l, entonces:

$$\sum_{(i,j)\in P} c_{ij}^{\pi} = \sum_{(i,j)\in P} (c_{ij} - \pi(i) + \pi(j)), \tag{3.11}$$

$$= \sum_{(i,j)\in P} c_{ij} - \sum_{(i,j)\in P} (\pi(i) + \pi(j)), \tag{3.12}$$

$$= \sum_{(i,j)\in P} c_{ij} - \pi(k) + \pi(l), \tag{3.13}$$

porque todos los $\pi(\cdot)$ correspondientes a los nodos en la ruta y distintos de los nodos terminales k e l, se cancelan entre sí en la expresión $\sum_{(i,j)\in P}(\pi(i) - \pi(j))$.

Propiedad 3.2.

1. Para todo ciclo dirigido W y para todo potencial π, $\sum_{(i,j)\in W} c_{ij}^{\pi} = \sum_{(i,j)\in W} c_{ij}$.
2. Para cualquier camino dirigido P del nodo k hasta el nodo l y para cualquier potencial de nodo π, $\sum_{(i,j)\in P} c_{ij}^{\pi} = \sum_{(i,j)\in P} c_{ij} - \pi(k) + \pi(l)$.

Trabajando con redes residuales.

Al diseñar, desarrollar e implementar algoritmos de flujo de red, a menudo es conveniente no medir el flujo en términos absolutos, sino en términos del incremento de flujo sobre alguna solución factible dada, típicamente, la solución en algún punto intermedio de un algoritmo. Hacerlo lleva a definir una nueva red auxiliar, conocida como red residual, que funciona como una ”red del flujo restante” para transportar el incremento de flujo. Se muestra que las formulaciones del problema en la red original y en la red residual son equivalentes, ambos dan una correspondencia biunívoca entre soluciones factibles a los dos problemas que preserva el valor del coste de las soluciones.

El concepto de red residual se basa en la siguiente idea intuitiva. Supóngase que el arco (i, j) lleva x^0 unidades de flujo. Entonces, se puede enviar $u_{ij} - x_{ij}^0$ unidades de flujo adicionales del nodo i al nodo j a lo largo del arco (i, j). Obsérvese también que se puede enviar hasta x^0 unidades de flujo del nodo j al nodo i por el arco (i, j), lo que equivale a cancelar el flujo existente en el arco. Mientras que enviar un flujo unitario del nodo i al nodo j en el arco (i, j) aumenta el coste del flujo en c_{ij} unidades, enviar flujo desde el nodo j al nodo i en el mismo arco disminuye el coste del flujo en c_{ij} unidades (ya que se ahorra el coste en el que se solía incurrir al enviar el flujo del nodo i al nodo j).

Usando estas ideas, se define la red residual con respecto a un flujo dado x^0 de la siguiente manera. Se reemplaza cada arco (i, j) en la red original por dos arcos, (i, j) y (j, i): el arco (i, j) tiene un coste c_{ij} y una capacidad residual $r_{ij} = u_{ij} - x^0_{ij}$, y el arco (j, i) tiene coste $-c_{ij}$ y capacidad residual $r_{ji} = x^0_{ij}$ (ver Figura 3.2). La red residual consiste sólo en los arcos con una capacidad residual positiva. Se usa la notación $G(x^0)$ para representar la red residual correspondiente al flujo x^0.

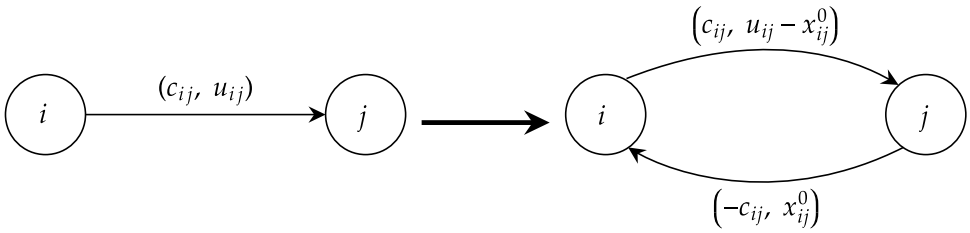

Figura 3.2: Construcción de la red residual $G(x^0)$.

Ahora, se muestra que todo flujo x en la red G corresponde a un flujo x' en la red residual $G(x^0)$. Se define el flujo $x' \geq 0$ de la siguiente manera:

$$x'_{ij} - x'_{ji} = x_{ij} - x^0_{ij}, \tag{3.14}$$

y

$$x'_{ij} x'_{ji} = 0. \tag{3.15}$$

La condición (3.15) supone que x'_{ij} y x'_{ji} no pueden ser ambos positivos al mismo tiempo. Si $x_{ij} \geq x^0_{ij}$, se establece $x'_{ij} = (x_{ij} - x^0_{ij})$ y $x'_{ji} = 0$. Obsérvese que si $x_{ij} < u_{ij}$, entonces $x'_{ij} \leq u_{ij} - x^0_{ij} = r_{ij}$. Por lo tanto, el flujo x'_{ij} satisface las restricciones de límite de flujo. De manera similar, si $x_{ij} < x^0_{ij}$, se establece $x'_{ij} = 0$ y $x'_{ji} = x^0_{ij} - x_{ij}$. Si $0 \leq x'_{ji} \leq x^0_{ij} = r_{ji}$, entonces el flujo x'_{ji} también satisface las restricciones de límite de flujo. Estas observaciones muestran que si x es un flujo factible en G, su correspondiente flujo x' es un flujo factible en $G(x^0)$.

A continuación, se establece una relación entre el coste de un flujo x en G y el coste del flujo correspondiente x' en $G(x^0)$. Sea c' los costes del arco en la red residual. Luego, para cada arco $(i, j) \in A$, $c'_{ij} = c_{ij}$ and $c'_{ji} = -c_{ij}$. Para un flujo x_{ij} en el arco (i, j) en la red original G, el coste del flujo en el par de arcos (i, j) y (j, i) en la red residual $G(x^0)$ es $c'_{ij}x'_{ij} + c'_{ji}x'_{ji} = c'_{ij}(x'_{ij} - x'_{ji}) = c_{ij}x_{ij} - c_{ij}x^0_{ij}$; la última igualdad se sigue de (3.14). Así se tiene que:

$$c'x' = cx - cx^0. \tag{3.16}$$

De manera similar, se puede mostrar el resultado inverso de que si x' es un flujo factible en la red residual $G(x^0)$, la solución dada por $x_{ij} = (x'_{ij} - x'_{ji}) + x^0_{ij}$ es un flujo factible en G. Además, los costes de estos dos flujos está relacionado por la igualdad $cx = c'x' + cx^0$.

Propiedad 3.3. Un flujo x es un flujo factible en la red G si y sólo si su correspondiente flujo x', definido por $x'_{ij} - x'_{ji} = x_{ij} + x^0_{ij}$, es factible en la red residual $G(x^0)$. Además, $cx = c'x' + cx^0$.

Una consecuencia importante de la Propiedad 3.3 es la flexibilidad que proporciona. En lugar de trabajar con la red G original, permite trabajar con la red residual $G(x^0)$ para algún x^0: una vez que se ha determinado una solución óptima en la red residual, se la puede convertir inmediatamente en una solución óptima en la red original.

3.3 Soluciones de árbol de expansión y soluciones sin ciclos

Para cualquier solución factible, x, un arco (i, j) es libre si $0 < x_{ij} < u_{ij}$ y es un arco restringido si $x_{ij} = 0$ o $x_{ij} = u_{ij}$. Puede aumentarse y disminuirse el flujo en un arco libre respetando las restricciones de flujo del arco. En la cota inferior de flujo de un arco restringido (i, j) $(x_{ij} = 0)$ sólo se puede aumentar el flujo. De manera similar, en la cota superior de flujo de un arco restringido (i, j) $(x_{ij} = u_{ij})$ sólo se puede disminuir el flujo. La solución x es una solución sin ciclos si la red no contiene ningún ciclo formado únicamente por arcos libres. En una solución sin ciclos, al aumentar el flujo en un ciclo de aumento (ver Sección 5.3.1) sólo se puede hacer en una dirección, puesto que en el ciclo siempre habrá algún arco que impedirá aumentar o disminuir el flujo de ese arco. Una solución x es factible y es un árbol de expansión asociado a la red (solución en árbol de expansión) si cada arco que no es del árbol es un arco restringido. En una solución de árbol de expansión, los arcos que no son del árbol pueden ser libres o restringidos. Con frecuencia, cuando se hace referencia a una solución de árbol de expansión, no se identifica explícitamente el árbol asociado; más bien, se entenderá por el contexto.

Los problemas de flujo de mínimo coste siempre tienen soluciones óptimas de árbol de expansión y sin ciclo. El algoritmo simplex en redes explota este resultado al restringir su búsqueda a sólo soluciones de árbol de expansión.

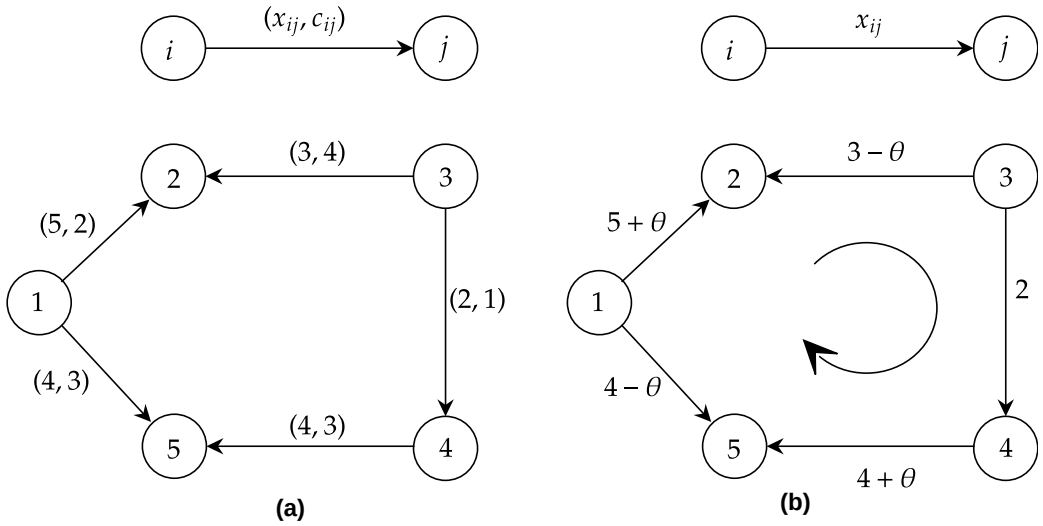

Figura 3.3: a) Solución factible; b) solución tras aumentar una cantidad θ de flujo en el ciclo.

Supóngase que los arcos no están capacitados ($u_{ij} = \infty$ para cada $(i, j) \in A$). La red que se muestra en la Figura 3.3 contiene flujo positivo alrededor de un ciclo. Se define la orientación del ciclo como la misma que la del arco $(4, 5)$. Se aumenta θ unidades de flujo a lo largo del ciclo en la dirección de su orientación. Como se muestra en la Figura 3.3, esto aumenta el flujo en los arcos a lo largo de la orientación del ciclo (arcos hacia adelante) en θ unidades y disminuye el flujo en los arcos opuestos a la orientación del ciclo (arcos hacia atrás) en θ unidades. El coste incremental por unidad para este cambio de flujo es la suma de los costes de los arcos hacia adelante menos la suma de los costes de los arcos hacia atrás en el ciclo, es decir,

$$\text{Aumento de coste por unidad de flujo: } \Delta = 2 + 1 + 3 - 4 - 3 = -1. \tag{3.17}$$

Dado que aumentar el flujo en el ciclo disminuye el coste, se establece θ lo más grande posible mientras se preserva la no negatividad de todos los flujos en los arcos. Se debe satisfacer las desigualdades $3 - \theta \geq 0$ y $4 - \theta \geq 0$, y por lo tanto se establece $\theta = 3$. En la nueva solución (en $\theta = 3$), algún arco en el ciclo tiene un flujo con valor cero, y además, esta solución es estrictamente menor que el valor de la solución inicial.

Si se cambia c_{12} de 2 a 5, el coste unitario del ciclo es $\Delta = 2$. En consecuencia, para mejorar al máximo el coste, se disminuye θ tanto como sea posible (satisfaciendo las restricciones $5+\theta \geq 0, 2+\theta \geq 0,$ y $4+\theta \geq 0,$ o $\theta \geq -2$) y se encuentra nuevamente la solución de menor coste con al menos un arco con flujo nulo. Puesto de otra manera:

para preservar todos los flujos en arco como no negativos, debe seleccionarse θ en el intervalo $-2 \leq \theta \leq 3$. Dado que la función objetivo depende linealmente de θ, se optimiza seleccionando $\theta = 3$ o $\theta = -2$, en este punto un arco en el ciclo tiene valor nulo. Se puede extender esto de la siguientes formas:

1. Si el incremento de coste $\Delta = 0$, no se obtiene mejora con las soluciones en el intervalo $-2 \leq \theta \leq 3$ y, por lo tanto, vuelve a elegirse una solución tan buena como la original, pero con el flujo de al menos un arco en el ciclo con valor cero.
2. Si se limita superiormente el flujo (por ejemplo no más de 6 unidades en cada arco), el rango de flujos que conserva la viabilidad (las restricciones de balance de flujo, las cotas inferior y superior de los flujos) es nuevamente un intervalo, en este caso: $-2 \leq \theta \leq 1$, y puede encontrarse una solución tan buena como la original eligiendo $\theta = -2$ o $\theta = 1$. Con estos valores de θ, la solución no tiene ciclos; es decir, algún arco en el ciclo tiene un flujo con valor cero.

Teorema 3.1 (Propiedad sin ciclos). *Si la función objetivo de un problema de flujo de mínimo coste está acotada inferiormente dentro de la región factible, entonces el problema siempre tiene una solución óptima sin ciclos.*

Los arcos libres en una solución sin ciclos definen un bosque (una colección de árboles cuyos nodos están separados). Si este bosque es un árbol de expansión, la solución sin ciclo ya es una solución de árbol de expansión. La Figura 3.4 (c) corresponde a un árbol de expansión de una solución sin ciclos.

Podría ser posible (y a menudo lo es) formar de varias formas un árbol de expansión a partir del conjunto de arcos. Agregar el arco $(3, 4)$ en lugar del arco $(2, 4)$ o $(3, 5)$ produciría otra solución de árbol de expansión. Por lo tanto, una solución sin ciclo dada puede corresponder a varios árboles de expansión. Sin embargo, dado que se asume que la red subyacente es conexa, siempre pueden agregarse algunos arcos restringidos a los arcos libres de una solución sin ciclos para producir un árbol de expansión, por lo que se establece el siguiente resultado:

Teorema 3.2 (propiedad del árbol de expansión). *Si la función objetivo de un problema de flujo de mínimo coste está acotado inferiormente dentro de la región factible, el problema siempre tiene una solución óptima de árbol de expansión.*

Una solución de árbol de expansión divide el conjunto de arcos A en tres subconjuntos: (1) T, los arcos en el árbol de expansión; (2) L, los arcos que no son del árbol cuyo flujo está restringido a cero; y (3) U, los arcos que no son del árbol cuyo flujo está

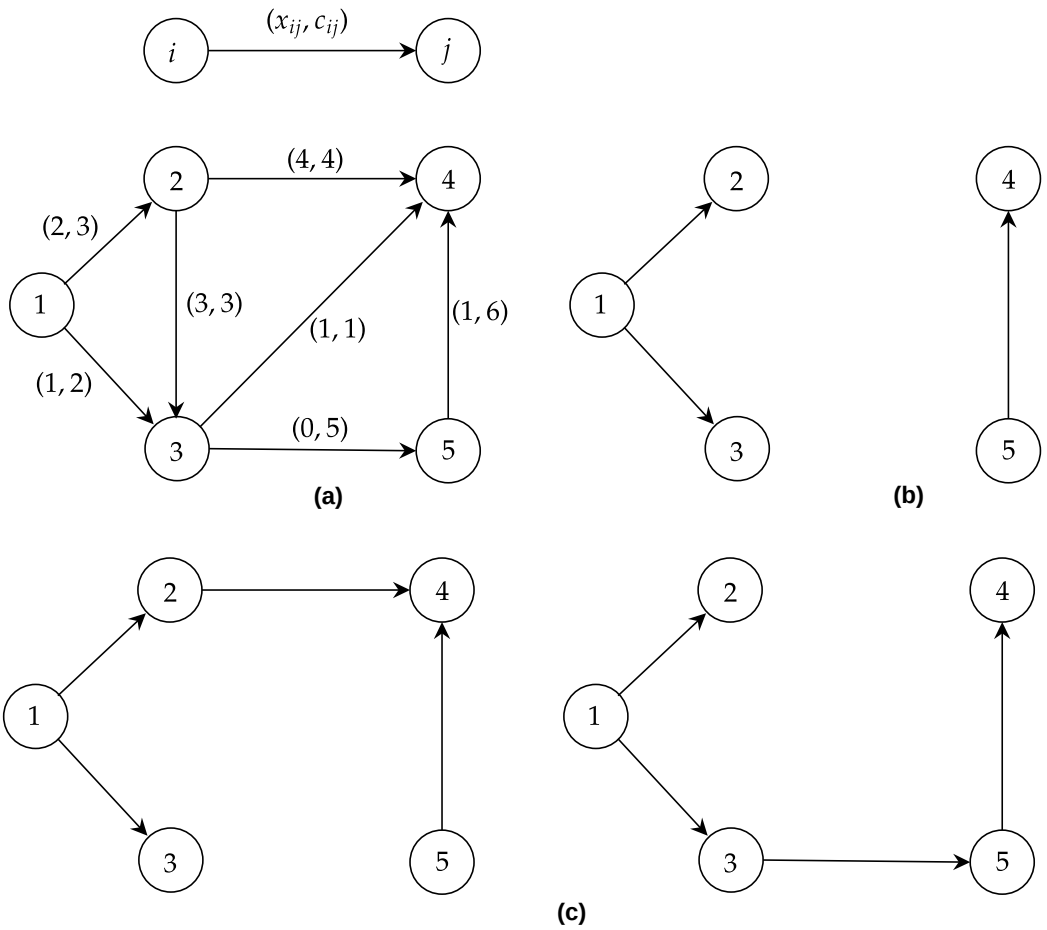

Figura 3.4: a) ejemplo de red; b) conjunto de aristas; c) dos soluciones de árboles de expansión.

restringido a las capacidades de flujo. El triplete (T, L, U) es una estructura de árbol de expansión.

Una estructura de árbol de expansión puede asociarse con una solución de árbol de expansión y también puede obtenerse una solución única de árbol de expansión correspondiente a una determinada estructura de árbol de expansión (T, L, U). Para hacerlo, se establece $x_{ij} = 0$ para todos los arcos $(i, j) \in L$, $x_{ij} = u_{ij}$ para todos los arcos $(i, j) \in U$, y luego se resuelven las ecuaciones de balance de flujo para determinar los valores de flujo de los arcos en T. Una estructura de árbol de expansión es factible si la solución de árbol de expansión asociada a ella satisface todos las restricciones de flujo de los arcos. En el caso especial en el que cada arco de una solución de árbol de

expansión es un arco libre, el árbol de expansión no es degenerado; en caso contrario sí es degenerado. Una estructura de árbol de expansión es óptima si la solución de árbol de expansión asociada a esta estructura es una solución óptima del problema de flujo de mínimo coste.

Teorema 3.3 (Condiciones de Optimalidad de Flujo de Mínimo Coste). *Una estructura de árbol de expansión (T, L, U) es una estructura de árbol de expansión óptima del problema de flujo de mínimo coste si es factible y para alguna elección de potenciales de nodo π, el arco de costes reducidos c_{ij}^{π} satisface las siguientes condiciones:*

$$c_{ji}^{\pi} = 0 \text{ para todo } (i, j) \in T. \tag{3.18}$$

$$c_{ji}^{\pi} \geq 0 \text{ para todo } (i, j) \in L. \tag{3.19}$$

$$c_{ji}^{\pi} \leq 0 \text{ para todo } (i, j) \in U. \tag{3.20}$$

Si $\pi(1) = 0$, las ecuaciones en (3.18) conllevan que $-\pi(k)$ denote la longitud de la ruta en el árbol desde el nodo 1 al nodo k. El coste reducido $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j)$ para un arco que no pertenece al árbol $(i, j) \in L$ denota el cambio en el coste del flujo resultante de enviar una unidad de flujo a través de la ruta en el árbol del nodo 1 al nodo i a través del arco (i, j), y luego de regreso al nodo 1 por la ruta del árbol de expansión que va desde el nodo j al nodo 1. La condición (3.19) implica que esta circulación de flujo no es rentable (es decir, no disminuye el coste) para cualquier arco no perteneciente al árbol en L. La condición (3.20) tiene una interpretación similar.

El algoritmo simplex en redes trabaja con una estructura de árbol de expansión factible y se mueve de una estructura de árbol de expansión a otra hasta que encuentra una estructura óptima. En cada iteración, el algoritmo sustituye un arco en el árbol de expansión por un arco ajeno al árbol que viola su condición de optimalidad. El algoritmo

1. agrega este arco al árbol de expansión, creando un ciclo negativo (que podría tener una capacidad residual nula),
2. envía el máximo flujo posible por este ciclo hasta que el flujo en al menos un arco del ciclo alcanza su cota inferior o superior, y
3. elimina un arco cuyo flujo ha alcanzado su cota inferior o superior, resultando en una nueva estructura de árbol de expansión.

Debido a su relación con el algoritmo simplex, esta operación de moverse de una estructura de árbol de expansión a otra se conoce como operación pivote, y las dos

estructuras de árboles generadores obtenidas en iteraciones consecutivas se llaman estructuras de árboles adyacentes.

3.4 Manteniendo una estructura de árbol de expansión

Dado que el algoritmo simplex en redes genera una secuencia de soluciones de árbol de expansión, para implementar el algoritmo de manera efectiva, es necesario poder representar árboles de expansión convenientemente en un ordenador realizar sus operaciones básicas de manera eficiente y, así actualizar la representación rápidamente cuando cambia el árbol de expansión.

El árbol parte de un nodo especifico llamado raíz. Aquí se asume que el nodo 1 es el nodo raíz. Se asocian tres índices por cada nodo i en el árbol: un índice predecesor, $pred(i)$, un índice profundidad $prof(i)$, y un índice rama, $rama(i)$.

Índice predecesor. Cada nodo i tiene una ruta única que lo conecta a la raíz. El índice $pred(i)$ almacena el primer nodo en esa ruta (que no sea el nodo i). Por convención, se establece el nodo predecesor al nodo raíz, el nodo 1, igual a cero. Un nodo j se denomina sucesor del nodo i si $pred(j) = i$. Un nodo hoja es un nodo sin sucesores. Los descendientes de un nodo i son el propio nodo i, sus sucesores, los sucesores de sus sucesores, etc.

Índice profundidad. Cada nodo i tiene una ruta única que lo conecta a la raíz. La profundidad del índice (i) almacena el número de arcos en esa ruta.

Índice rama. Los índices rama definen una secuencia de nodos que recorre los nodos de un árbol, comenzando en el nodo raíz y visitando los nodos en un orden "de arriba hacia abajo" y finalmente volviendo a la raíz.

Los índices rama son útiles para encontrar todos los descendientes de un nodo i. Se sigue la rama que comienza en ese nodo y se registran los nodos visitados, hasta que la profundidad del nodo visitado sea al menos tan grande como la del nodo i.

3.5 Cálculo de potencial de nodo y flujo en nodo

A medida que el algoritmo simplex en redes se mueve de un árbol de expansión al siguiente mantiene la condición de que el coste reducido de cada arco (i, j) en el árbol de expansión actual sea cero ($c_{ij} = 0$). Dada la estructura del árbol de expansión actual (T, L, U), el método determina valores para el π de los potenciales de nodo que satisfaga esta condición para los arcos del árbol.

Puede establecerse el valor de un potencial de nodo arbitrariamente porque agregar una constante k a cada potencial de nodo no altera el coste reducido de ningún arco. Esto es, para cualquier constante k, $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j) = c_{ij} - [\pi(i) + k] + [\pi(j) + k]$. Por conveniencia, se asumirá que $\pi(1) = 0$. Se calcula los potenciales de nodo restantes usando que el coste reducido de cada arco de árbol de expansión es cero. Esto es,

$$c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j) = 0 \quad \text{para cada } (i, j) \in T. \tag{3.21}$$

Si se conocen uno de los potenciales de nodo $\pi(i)$ o $\pi(j)$ en la ecuación (3.21), puede calcularse fácilmente el otro. En consecuencia, la idea del procedimiento es comenzar en el nodo 1 y desplegarse a lo largo de los arcos del árbol utilizando los índices de subprocesos para calcular otros potenciales de nodo. Al atravesar los nodos utilizando los índices de subprocesos, es seguro que siempre que el procedimiento visita un nodo k, ya ha evaluado el potencial de su predecesor, por lo que puede calcular $\pi(k)$ utilizando (3.21). El Algoritmo 1 da un enunciado formal del procedimiento de cálculo de potenciales.

$\pi(i) \leftarrow 0$;
$j \leftarrow rama(i)$;
mientras $j \neq 1$ **hacer**
  $i \leftarrow pred(j)$;
  si $(i, j) \in A$ **entonces**
    $\pi(j) \leftarrow \pi(i) - c_{ij}$
  fin
  si $(j, i) \in A$ **entonces**
    $\pi(j) \leftarrow \pi(i) + c_{ji}$
  fin
fin

Algoritmo 1: Procedimiento del cálculo de potenciales.

El ejemplo numérico que se muestra en la Figura 3.5 ilustra el procedimiento. Primero se establece $\pi(1) = 0$. La rama del nodo 1 es 2, luego se examina el nodo 2. Como el arco $(1, 2)$ conecta el nodo 2 con su predecesor, usando (3.21) se encuentra que $\pi(2) = \pi(1) - c_{12} = -5$. A continuación, se examina el nodo 5, que está conectado a su padre por el arco $(5, 2)$. Usando (3.21) se obtiene $\pi(5) = \pi(2) + c_{52} = -5 + 2 = -3$. De la misma manera se calcula el resto de los potenciales de los nodos; los números que se muestran junto a cada nodo en la Figura 3.5 especifican estos valores.

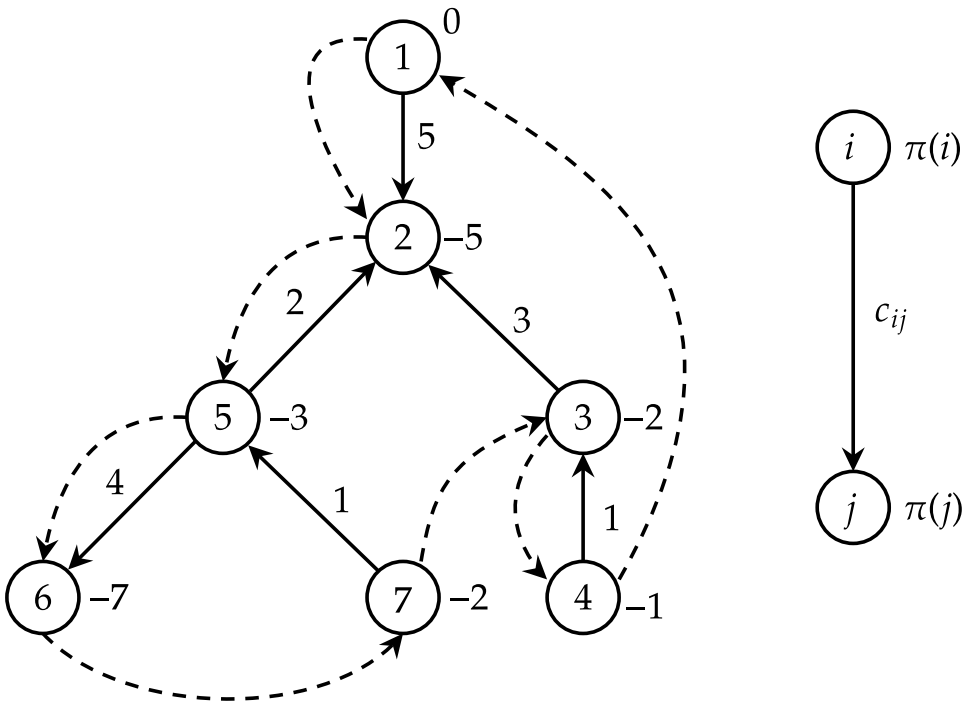

Figura 3.5: Calculo de los potenciales de nodo para un árbol de expansión.

Sea P la ruta en el árbol T que parte del nodo raíz 1 hasta el nodo k. Además, sean $\overline{P}$ y $\underline{P}$ los conjuntos de arcos hacia adelante y hacia atrás en P, respectivamente. El procedimiento de cálculo de potenciales supone que:

$$\pi(j) = \begin{cases} \pi(i) - c_{ij} & \text{si } (i,j) \in P \text{ es un arco hacia delante} \\ \pi(i) + c_{ji} & \text{si } (j,i) \in P \text{ es un arco hacia atrás} \end{cases}. \tag{3.22}$$

Esta observación implica que:

$$\pi(k) = \pi(k) - \pi(1) = -\sum_{(i,j)\in\overline{P}} c_{ij} + \sum_{(i,j)\in\underline{P}} c_{ij}. \tag{3.23}$$

En otras palabras, $\pi(k)$ es el coste negativo de enviar una unidad de flujo desde el nodo 1 al nodo k a lo largo de la ruta del árbol. Alternativamente, $\pi(k)$ es el coste de enviar una unidad de flujo del nodo k al nodo 1 a lo largo de la ruta del árbol. El procedimiento de cálculo de potenciales requiere tiempo $\mathcal{O}(1)$ por iteración y realiza $(n-1)$ iteraciones para evaluar el potencial de cada nodo. Por lo tanto, el procedimiento se ejecuta en tiempo $\mathcal{O}(n)$.

Una consecuencia importante del procedimiento de cálculo de potenciales es que el problema de flujo de mínimo coste tiene potenciales de nodo óptimos enteros siempre y cuando todos los costes de arco sean enteros. El problema de flujo de mínimo coste siempre tiene una solución óptima de árbol de expansión, los potenciales asociados a este árbol constituyen potenciales de nodo óptimos, que pueden determinarse usando el procedimiento calculo de potenciales. Este procedimiento de cálculo de potenciales implica que si todos los costes de arco son enteros, los potenciales de nodo también son enteros (esto es porque el procedimiento sólo realiza sumas y restas). Esta es la propiedad de integralidad dual, ya que los potenciales de nodo son las variables de programación lineal dual asociadas con el problema de flujo de mínimo coste.

Teorema 3.4 (Propiedad de integralidad dual). *Si todos los costes de arco son enteros, el problema de flujo de mínimo coste siempre tiene potenciales de nodo óptimos enteros.*

3.5.1 Calcular los flujos de arco

A continuación, considérese el problema de determinar los flujos en los arcos de árbol de una estructura de árbol de expansión. Primero se tratará la versión no capacitada del problema del flujo de mínimo coste. Puede suponerse que todos los arcos no pertenecientes al árbol tienen flujo nulo.

Si se elimina el arco (i, j) del árbol de expansión, el árbol se descompone en dos subárboles. Sea T_1 el subárbol que contiene el nodo i y sea T_2 el subárbol que contiene el nodo j. Entonces, $\sum_{k\in T_1} b(k)$ denota la oferta/demanda acumulada de nodos en T_1 (que debe ser igual a $-\sum_{k\in T_2} b(k)$ porque $\sum_{k\in T_1} b(k) + \sum_{k\in T_2} b(k) = 0$). En el árbol de expansión, el arco (i, j) es el único arco que conecta el subárbol T_1 con el subárbol T_2, por lo que debe llevar $\sum_{k\in T_1} b(k)$ unidades de flujo, ya que esta es la única forma de satisfacer las restricciones de balance de flujo.

Se dispone así de un método eficiente para calcular los flujos en todos los arcos de los árboles. Supóngase que (i, j) es un arco en el árbol y que el nodo j es un nodo hoja (el

caso en el que el nodo i es el nodo hoja es similar), el arco (i, j) debe transportar $-b(j)$ unidades de flujo. El Algoritmo 2 ofrece una descripción formal de este procedimiento.

$b'(i) \leftarrow b(i)$, para todo $i \in N$;
$T' \leftarrow T$;
para $(i, j) \in L$ **hacer**
  $x_{ij} \leftarrow 0$
fin
$j \leftarrow rama(i)$;
mientras $T' \neq \{1\}$ **hacer**
  seleccionar un nodo hoja j (que no sea el nodo 1) en el subárbol T';
  $i \leftarrow pred(j)$;
  si $(i, j) \in T'$ **entonces**
    $x_{ij} \leftarrow -b'(j)$
  en otro caso
    $x_{ij} \leftarrow b'(j)$
  fin
  $b'(i) \leftarrow b'(i) + b'(j)$;
  eliminar el nodo j y el arco indicente al este nodo y parte de T';
fin

Algoritmo 2: Procedimiento de cálculo de flujos

Este método para calcular los flujos asume que el problema de flujo de mínimo coste no está capacitado. Para la versión capacitada del problema, se agrega el siguiente enunciado inmediatamente después de la primera condición ($b'(i) := b(i)$ para todo $i \in N$) en el procedimiento de cálculo de flujos.

para $(i, j) \in U$ **hacer**
  $x_{ij} \leftarrow u_{ij}$;
  $b'(i) \leftarrow b'(i) - u_{ij}$;
  $b'(j) \leftarrow b'(j) + u_{ij}$;
fin

La inicialización de los flujos y la modificación de las ofertas/demandas $b(i)$ y $b(j)$ para los arcos (i, j) en U requiere un tiempo $\mathcal{O}(m)$. Ignorando el tiempo para seleccionar los nodos hoja de T cada iteración requiere tiempo $\mathcal{O}(1)$, lo que da como resultado un total de tiempo $\mathcal{O}(n)$. Una forma de identificar los nodos hoja en T es seleccionar nodos en el orden inverso al de los índices rama. Cada nodo aparece antes que sus descendientes en el recorrido de la rama. Este recorrido inverso examina cada nodo sólo después de

haber examinado todos los descendientes del nodo. Así, para el problema de flujo de mínimo coste sin capacidades, el procedimiento de cálculo de flujo se ejecuta en un tiempo $\mathcal{O}(m)$. Para la versión con capacidades del problema, el procedimiento también requiere un tiempo $\mathcal{O}(m)$.

El problema de flujo de mínimo coste siempre tiene una solución óptima de árbol de expansión. El flujo asociado con este árbol es un flujo óptimo y puede determinarse usando el procedimiento de cálculo de flujo. Este procedimiento implica que si las capacidades de todos los arcos y las ofertas/demandas de todos los nodos son enteras, los flujos de arco también son enteros (el procedimiento solo realiza sumas y restas).

Teorema 3.5 (Propiedad de integralidad primaria). *Si las capacidades de todos los arcos y las ofertas/demandas de todos los nodos son enteras, el problema de flujo de mínimo coste tiene siempre un flujo óptimo entero.*

Cada estructura de árbol de expansión (T, L, U) define un flujo x único. Si este flujo satisface las restricciones de flujo $0 \leq x_{ij} \leq u_{ij}$ para cada arco $(i, j) \in A$, la estructura del árbol de expansión es factible; de lo contrario, no es factible. El árbol de expansión T es degenerado si $x_{ij} = 0$ o $x_{ij} = u_{ij}$ para algún arco $(i, j) \in T$, y no degenerado en caso contrario. Para cada arco (i, j) en un árbol de expansión no degenerado se cumple $0 < x_{ij} < u_{ij}$.

3.6 Algoritmo simplex en redes

El algoritmo simplex en redes mantiene una estructura de árbol de expansión factible en cada iteración y transforma esta estructura sucesivamente en una estructura de árbol de expansión mejorada hasta que se vuelve óptima. El Algoritmo 4 especifica los pasos esenciales del método.

determinar una estructura de árbol factible inicial (T, L, U);
sea x el flujo y π el potencial de nodo asociado a la estructura del árbol;
mientras *algunos arcos no pertenecientes al árbol violen la condición de optimalidad* **hacer**
  seleccionar un arco de entrada (k, l) que viola la condición de optimalidad;
  añade el arco (k, l) al árbol y determinar el arco saliente (p, q);
  realizar una actualización de árbol y actualizar las soluciones x y π;
fin

Algoritmo 3: Algoritmo simplex en redes

3.6.1 Obtención de una estructura de árbol de expansión inicial

Se asume que para cada nodo $j \in N - \{1\}$, la red contiene los arcos $(1, j)$ y $(j, 1)$, con costes y capacidades suficientemente grandes. El árbol inicial T se construye de la siguiente manera. Se examina cada nodo j, distinto del nodo 1, uno por uno. Si $b(j) \geq 0$, se incluye el arco $(1, j)$ en T con un valor de flujo $b(j)$. Si $b(j) < 0$, se incluye el arco $(j, 1)$ en T con un valor de flujo de $-b(j)$. El conjunto L consta de los arcos restantes y el conjunto U está vacío. Puede calcularse fácilmente los potenciales de nodo para este árbol usando las ecuaciones $c_{ij} - \pi(i) + \pi(j) = 0$ para todo $(i, j) \in T$. Téngase en cuenta que $\pi(1) = 0$.

Si la red no contiene los arcos $(1, j)$ y $(j, 1)$ para cada nodo $j \in N - \{1\}$ (o no se desea agregar estos arcos por alguna razón), puede construirse una estructura de árbol de expansión inicial estableciendo primero un flujo factible en la red resolviendo un problema de flujo máximo y luego convirtiendo esta solución en una solución de árbol de expansión.

3.6.2 Condiciones de optimalidad

Sea (T, L, U) una estructura de árbol de expansión factible del problema de flujo de mínimo coste, y sea π los correspondientes potenciales de nodo. Para determinar si la estructura de árbol de expansión es óptima, se verifica si cumple las siguientes

condiciones:

$$c_{ij}^{\pi} \geq 0 \text{ para todo arco } (i,j) \in L, \tag{3.24}$$

$$c_{ij}^{\pi} \leq 0 \text{ para todo arco } (i,j) \in U. \tag{3.25}$$

Si la estructura del árbol de expansión satisface estas condiciones, es óptima y el algoritmo termina. De lo contrario, el algoritmo selecciona un arco no perteneciente al árbol que viola la condición de optimalidad para ser introducido en el árbol. Dos tipos de arcos son seleccionables para ingresar al árbol:

1. Cualquier arco $(i,j) \in L$ con $c_{ij}^{\pi} < 0$,
2. Cualquier arco $(i,j) \in U$ con $c_{ij}^{\pi} > 0$.

Para cualquier arco seleccionable (i,j), $|c_{ij}^{\pi}|$ es su violación. El algoritmo simplex en redes puede seleccionar cualquier arco elegible para ingresar en el árbol y aún así terminaría de manera finita (con algunas disposiciones para lidiar con la degeneración). Sin embargo, diferentes reglas para seleccionar el arco de entrada producen algoritmos con diferente comportamiento empírico y teórico. Son posibles muchas reglas diferentes, llamadas reglas de pivote, para elegir el arco de entrada. Las siguientes reglas son las más ampliamente adoptadas.

Regla de pivote de Dantzig. Esta regla selecciona en cada iteracción el arco con mayor violación puesto que causa la máxima disminución en la función objetivo por unidad de flujo. Si el aumento promedio en el flujo del arco seleccionado fuera el mismo para todos los arcos entonces la introducción de este arco en el árbol de expansión causa la máxima disminución por pivote. Esta elección del arco de entrada tiende a producir disminuciones relativamente grandes en la función objetivo por iteración y, como resultado, el algoritmo realiza menos iteraciones que otras opciones para la regla de pivote. Sin embargo, esta regla tiene un gran inconveniente: el algoritmo debe considerar cada arco no perteneciente al árbol para identificar el arco con la máxima violación y hacerlo requiere mucho tiempo. Por lo tanto, aunque este algoritmo generalmente realiza menos iteraciones que otras implementaciones, el tiempo de ejecución del algoritmo no es atractivo.

Primera regla de pivote de arco elegible. Para implementar esta regla, se escanea la lista de arcos secuencialmente y se selecciona el primer arco apropiado para ingresar en el árbol. En una versión popular de esta regla, se examina la lista de la forma en la que a continuación se ejemplifica: en una iteración, si se encuentra que el quinto arco en la lista de arcos es el primer arco apropiado, en la siguiente iteración

comienza a escanearse la lista de arcos desde el sexto arco. Si se llega al final de la lista de arcos mientras se está realizando alguna iteración, se continua examinando la lista de arcos desde el principio. Una característica interesante de esta regla de pivote es que identifica rápidamente el arco de entrada. Esta regla de pivote tiene un inconveniente de contrapeso: con ella, el algoritmo generalmente realiza más iteraciones que con otras reglas de pivote porque cada operación de pivote produce una disminución relativamente pequeña en el valor de la función objetivo. El tiempo de ejecución del algoritmo no es muy atractivo, aunque la regla produce una implementación más eficiente que la regla de pivote de Dantzig.

Lista de candidatos según la regla del pivote. El algoritmo selecciona el arco de entrada usando un procedimiento de dos fases que consta de iteraciones mayores e iteraciones menores. En una iteración mayor se construye una lista de candidatos de arcos elegibles. Una vez construida esta lista, se realiza una serie de iteraciones menores; en cada una de estas iteraciones, se selecciona un arco elegible de la lista de candidatos que causa la máxima disminución por unidad de flujo en la función objetivo.

En una iteración mayor, se construye la lista de candidatos de la siguiente manera. Primero se examinan los arcos que parten del nodo 1 y se agregan arcos elegibles a la lista de candidatos. Este proceso se repite para los nodos $2, 3, \ldots$, hasta que la lista haya alcanzado el máximo tamaño permitido o se hayan examinado todos los nodos. La siguiente iteración principal comienza con el nodo donde terminó la iteración principal anterior.

El método de la lista de candidatos ofrece una flexibilidad considerable para ajustarse con precisión a la clase de problema. Al establecer el tamaño máximo permitido de la lista de candidatos de manera adecuada y al especificar el número de iteraciones menores que se realizarán dentro de una iteración principal, puede obtenerse numerosas reglas de pivote diferentes.

A continuación, se estudia el proceso de elección del arco que abandona la estructura del árbol de expansión en cada paso del algoritmo simplex en redes.

3.6.3 Arcos que abandonan el árbol

Se selecciona el arco de entrada (k, l), la inclusión de este arco al árbol T crea un ciclo pivote W. El ciclo pivote consiste en la única ruta en el árbol T que va desde

el nodo k al nodo l, junto con el arco (k, l). Se define la orientación del ciclo como la misma que (k, l) si $(k, l) \in L$ y opuesta a la de (k, l) si $(k, l) \in U$. Sean $\overline{W}$ y $\underline{W}$ los conjuntos de arcos hacia adelante (arcos de W con su misma orientación) y arcos hacia atrás (arcos de W con la orientación contraria a W) en el ciclo pivote. El envío de flujo adicional por el ciclo pivote W en la dirección de su orientación disminuye estrictamente el coste de la solución actual en una tasa de $|c^{\pi}_{kl}|$ por unidad. Se aumenta el flujo tanto como sea posible hasta que uno de los arcos en el ciclo pivote alcance su cota inferior o superior. Aumentar el flujo a lo largo de W aumenta el flujo en los arcos hacia adelante y disminuye el flujo en los arcos hacia atrás. En consecuencia, el cambio de flujo máximo δ_{ij} en un arco $(i, j) \in W$ que satisface las restricciones de límite de flujo es:

$$\delta_{ij} = \begin{cases} u_{ij} - x_{ij} & \text{si } (i, j) \in \overline{W}, \\ x_{ij} & \text{si } (i, j) \in \underline{W}. \end{cases} \tag{3.26}$$

Puede aumentarse $\delta = \text{mín}\{\delta_{ij} : (i, j) \in W\}$ unidades de flujo a lo largo de W para mantener la viabilidad. Cualquier arco $(i, j) \in W$ que defina δ ($\delta = \delta_{ij}$) es un arco de bloqueo. Se aumenta δ unidades de flujo y se selecciona un arco (p, q) con $\delta_{pq} = \delta$ como arco saliente. Una iteración pivote es una iteración no degenerada si $\delta > 0$ y es una iteración degenerada si $\delta = 0$. Una iteración degenerada ocurre sólo si T es un árbol de expansión degenerado. Si dos arcos tienen el mismo valor de δ, el siguiente árbol de expansión se degenerará.

El paso crucial para identificar el arco saliente es identificar el ciclo de pivote. Si $P(i)$ denota la única ruta en el árbol desde un nodo i al nodo raíz, este ciclo consta de los arcos $\{(k, l)\} \cup P(k) \cup P(l) - (P(k) \cap P(l))$. En otras palabras, W consiste en el arco (k, l) y las porciones disjuntas de $P(k)$ y $P(l)$. Los índices predecesor permiten identificar el ciclo que sigue. Primero, se designan todos los nodos de la red como "sin marcar". Luego se comienza en el nodo k y, usando los índices predecesor, se trazan la ruta desde este nodo hasta la raíz y se marcan todos los nodos en esta ruta. Luego partiendo del nodo l se trazan los índices predecesor hasta que se encuentra un nodo marcado w. El nodo w es el primer ancestro común de los nodos k e l; se trata del vértice esquina del ciclo W. El ciclo W contiene las porciones de las rutas $P(k)$ y $P(l)$ hasta el nodo w, junto con el arco (k, l). Este método identifica el tiempo del ciclo W en $\mathcal{O}(n)$, y por lo que es eficiente. Sin embargo, tiene el inconveniente de retroceder a lo largo de esos arcos de $P(k)$ que no están en W. Si el ciclo de pivote está lejos de su raíz, entonces rastrear los nodos hasta la raíz será ineficaz. Idealmente, se desea identificar el ciclo W en el tiempo proporcional a $|W|$. El uso simultáneo de los índices profundidad y predecesor permite alcanzar este objetivo (veáse Algoritmo 4).

```
i ← k y j ← l;
mientras i ≠ j hacer
    si prof(i) > prof(j) entonces
        i ← pred(i)
    en otro caso
        si prof(j) > prof(i) entonces
            j ← pred(j)
        en otro caso
            i ← pred(i) y j ← pred(j)
        fin
    fin
fin
```

Algoritmo 4: Procedimiento para identificar el ciclo pivote.

Este método escanea los arcos en el ciclo de pivote W dos veces. Durante la primera exploración, se identifica el vértice esquina del ciclo y también se identifica el máximo flujo posible que puede aumentarse a lo largo de W. En la segunda exploración, se aumenta el flujo. Toda la operación de cambio de flujo requiere en el peor de los casos un tiempo $\mathcal{O}(n)$, pero normalmente examina solo un pequeño subconjunto de nodos (y arcos).

3.6.4 Actualización del árbol

Cuando para un arco entrante dado (k, l) el algoritmo simplex en redes ha determinado un arco saliente (p, q) actualiza la estructura del árbol. Si el arco de salida es el mismo que el arco de entrada, lo que sucede cuando $\delta = \delta_{kl} = u_{kl}$, el árbol no cambia. En este caso, el arco (k, l) se mueve del conjunto L al conjunto U, o viceversa. Si el arco saliente difiere del arco entrante, el algoritmo debe realizar cambios más extensos. En este caso, el arco (p, q) se convierte en un arco no perteneciente al árbol dependiendo de si $x_{pq} = 0$ o $x_{pq} = u_{pq}$ (flujo actualizado) en su cota inferior o superior. Agregar el arco (k, l) al árbol de expansión actual y eliminar el arco (p, q) crea un nuevo árbol de expansión.

Para el nuevo árbol de expansión, los potenciales de los nodos también cambian; pueden actualizarse de la siguiente manera. La eliminación del arco (p, q) del árbol anterior divide el conjunto de nodos en dos subárboles. Uno, T_1, que contiene el nodo raíz, y el otro, T_2, que no contiene el nodo raíz. Téngase en cuenta que el subárbol T_2 parte del nodo p o del nodo q. El arco (k, l) tiene un extremo en T_1 y el otro en T_2.

Como es fácil de verificar, las condiciones $\pi(1) = 0$ y $c_{ij} - \pi(i) + \pi(j) = 0$ implican para todos los arcos en el nuevo árbol que los potenciales de los nodos en el subárbol T_1 permanecen sin cambios, y los potenciales de los nodos en el subárbol T_2 cambian en una cantidad constante. Si $k \in T_1$ e $l \in T_2$, todos los potenciales de nodo en T_2 aumentan en $-c^{\pi}_{kl}$; si $l \in T_1$ y $k \in T_2$, aumentan en la cantidad c^{π}_{kl}. Usando los índices de profundidad y de rama, el método descrito en el Algoritmo 5 actualiza los potenciales de nodo rápidamente.

si $q \in T_2$ **entonces**
  $y \leftarrow q$
en otro caso
  $y \leftarrow p$
fin
si $k \in T_1$ **entonces**
  $intercambio \leftarrow -c^{\pi}_{kl}$
en otro caso
  $intercambio \leftarrow c^{\pi}_{kl}$
fin
$\pi(y) \leftarrow \pi(y) + intercambio$;
$z \leftarrow rama(y)$;
mientras $prof(z) > prof(y)$ **hacer**
  $\pi(z) = \pi(z) + intercammbio$;
  $z \leftarrow rama(z)$;
fin

Algoritmo 5: Actualización del potencial de nodo en una operación pivote.

El último paso en la actualización del árbol es volver a calcular los distintos índices del árbol. Este paso es bastante complicado pero es posible actualizar los índices del árbol en tiempo $\mathcal{O}(n)$. De hecho, el tiempo necesario para actualizar los índices del árbol es $\mathcal{O}(|W| + \text{mín}\{|T_1|, |T_2|\})$, que suele ser mucho menor que n.

3.6.5 Terminación

El algoritmo simplex en redes se mueve de una estructura de árbol de expansión factible a otra hasta que obtiene una estructura de árbol de expansión que satisface la condición de optimalidad (3.18)-(3.20). Si cada operación de pivote en el algoritmo no es degenerada, se puede mostrar que el algoritmo es finito. $|c^{\pi}_{kl}|$ es la disminución neta en el coste por flujo unitario enviado alrededor del ciclo de pivote W. En un pivote no degenerado (para el cual $\delta > 0$), el coste de la nueva estructura de árbol de expansión

es $\delta|c_{kl}^{\pi}|$ unidades menos que el coste de la estructura de árbol de expansión anterior. Dado que cualquier red tiene un número finito de estructuras de árbol de expansión y cada estructura de árbol de expansión tiene un coste asociado único, el algoritmo simplex en redes encuentra cualquier estructura de árbol de expansión como máximo una vez y, por lo tanto, terminará de manera finita. Sin embargo, los pivotes degenerados plantean una dificultad teórica: es posible que el algoritmo no termine de forma finita a menos que se realicen pivotes con cuidado. En la siguiente sección se discute una implementación especial, llamada implementación de árbol de expansión altamente factible, que garantiza la convergencia finita del algoritmo simplex en redes incluso para problemas que están degenerados.

Se hace uso del ejemplo de la Figura 3.6 (a) para ilustrar el algoritmo simplex en redes. La Figura 3.6 (b) muestra una solución factible de árbol de expansión para el problema. Para esta solución, $T = \{(1,2),(1,3),(2,4),(2,5),(5,6)\}$, $L = \{(2,3),(5,4)\}$ y $U = \{(3,5),(4,6)\}$. En esta solución, el arco $(3,5)$ tiene una violación positiva, que es una unidad. Se introduce este arco en el árbol creando un ciclo cuyo vértice es el nodo 1. Dado que el arco $(3,5)$ está en su cota superior, la orientación del ciclo es opuesta a la del arco $(3,5)$. Los arcos $(1,2)$ y $(2,5)$ son arcos hacia adelante en el ciclo y los arcos $(3,5)$ y $(1,3)$ son arcos hacia atrás. El aumento máximo de flujo permitido por los arcos $(3,5)$, $(1,3)$, $(1,2)$ y $(2,5)$ es, respectivamente, 3, 3, 2 y 1 unidades. En consecuencia, $\delta = 1$ y se aumenta una unidad de flujo a lo largo del ciclo. El aumento incrementa el flujo en los arcos $(1,2)$ y $(2,5)$ en una unidad y disminuye el flujo en los arcos $(1,3)$ y $(3,5)$ en una unidad. El arco $(2,5)$ es el arco de bloqueo único y, por lo tanto, se selecciona para salir del árbol. Al soltar el arco $(2,5)$ del árbol se obtienen dos subárboles: T_1 que consta de los nodos 1, 2, 3, 4 y T_2 que consta de los nodos 5 y 6. Introduciendo el arco $(3,5)$, se obtiene nuevamente un árbol de expansión, véase la Figura 3.6 (c). En este árbol de expansión, los potenciales de nodo de los nodos 5 y 6 son una unidad menor que los del árbol de expansión anterior.

En la solución de árbol de expansión factible que se muestra en la Figura 3.6 (c), $L = \{(2,3),(5,4)\}$ y $U = \{(2,5),(4,6)\}$. En esta solución, el arco $(4,6)$ es el único arco elegible: su violación es igual a una unidad. Por lo tanto, se introduce el arco $(4,6)$ en el árbol. La Figura 3.6 (c) muestra el ciclo resultante y su orientación. Puede aumentarse una unidad de flujo adicional a lo largo de la orientación de este ciclo. Al enviar este flujo, se encuentra que el arco $(3,5)$ es un arco de bloqueo, por lo que se elimina este arco del árbol de expansión actual. La Figura 3.6 (d) muestra el nuevo árbol de expansión. Como puede verificarse, esta solución no tiene un arco elegible y, por lo tanto, el algoritmo simplex en redes termina con esta solución.

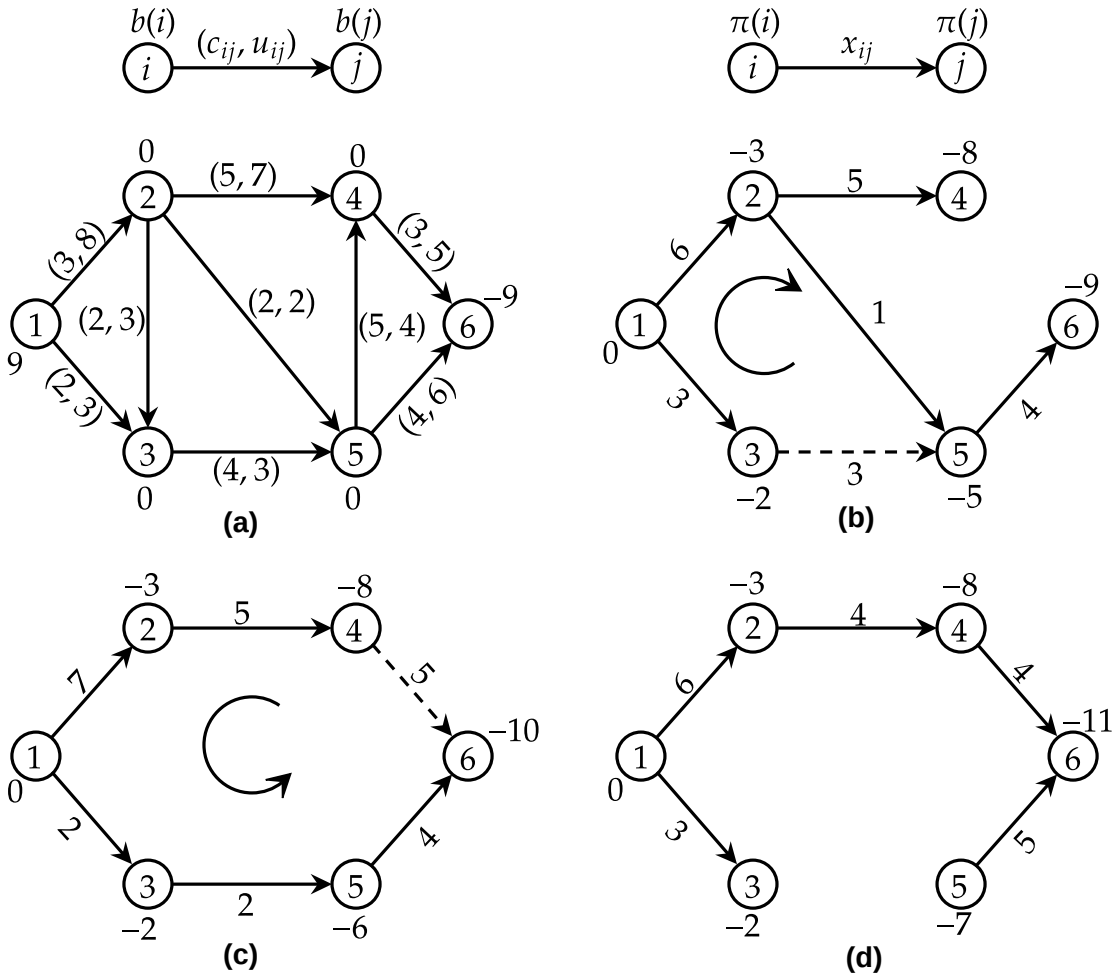

Figura 3.6: Ejemplo numérico para el algoritmo simplex en redes.

3.7 Árbol de expansión altamente factible

El algoritmo simplex en redes no necesariamente termina en un número finito de iteraciones a menos que se imponga alguna restricción adicional sobre la elección de los arcos de entrada y salida. Al mantener un tipo especial de árbol de expansión, el árbol de expansión altamente factible, el algoritmo simplex en redes termina de manera finita; además, en la práctica se ejecuta más rápido.

Sea (T, L, U) una estructura de árbol de expansión para un problema de flujo de mínimo coste con datos integrales. Los arcos de los árboles apuntan hacia arriba (hacia la raíz) o apuntan hacia abajo (dirección contraria a la raíz). Se establecen dos defini-

ciones alternativas de un árbol de expansión altamente factible.

1. Un árbol de expansión T es altamente factible si cada arco de árbol con flujo cero apunta hacia arriba y cada arco de árbol cuyo flujo es igual a su capacidad apunta hacia abajo.
2. Un árbol de expansión T es altamente factible si puede enviarse una cantidad positiva de flujo desde cualquier nodo a la raíz a lo largo de la ruta del árbol sin violar ningúna restricción de flujo.

Si un árbol de expansión T es altamente factible entonces la estructura de árbol de expansión (T, L, U) también es altamente factible.

La Figura 3.7 (a) da un ejemplo de un árbol de expansión altamente factible, y la Figura 3.7 (b) ilustra un árbol de expansión poco factible. El árbol de expansión que se muestra en la Figura 3.7 (b) es poco factible porque el arco $(3, 5)$ tiene flujo cero y apunta hacia abajo. En este árbol de expansión, no puede enviarse ningún flujo adicional desde los nodos 5 y 7 a la raíz a lo largo de la ruta del árbol.

Al implementar el algoritmo simplex en redes para que siempre mantenga un árbol de expansión altamente factible, primero debe encontrarse un árbol de expansión inicial altamente factible. Un árbol de expansión no degenerado siempre es altamente factible; un árbol de expansión degenerado podría ser o no muy factible. El algoritmo simplex en redes crea un árbol de expansión degenerado a partir de un árbol de expansión no degenerado siempre que dos o más arcos se identifican como arcos salientes y se descarta sólo uno de ellos. Por lo tanto, el algoritmo debe seleccionar el arco saliente con cuidado para que el siguiente árbol de expansión sea altamente factible.

Supóngase que se parte de un árbol de expansión altamente factible y, durante una operación de pivote, el arco (k, l) entra en el árbol de expansión. Primero considérese el caso en el que (k, l) es un arco que no pertenece al árbol cuyo valor coincide con su cota inferior. Supóngase que W es el ciclo de pivote formado al agregar el arco (k, l) al árbol de expansión y que el nodo w es el vértice esquina del ciclo W; es decir, w es el primer ancestro común de los nodos k y l. Sea la orientación del ciclo la misma que la del arco (k, l). Después de aumentar δ unidades de flujo a lo largo del ciclo de pivote, el algoritmo identifica los arcos de bloqueo (aquellos arcos (i, j) en el ciclo que satisfacen $\delta_{ij} = \delta$). Si el arco de bloqueo es único, se selecciona para salir del árbol de expansión. Si el ciclo contiene más de un arco de bloqueo, el siguiente árbol de expansión será degenerado (el flujo en algunos arcos de árbol coincidirá con sus cotas inferiores o superiores).

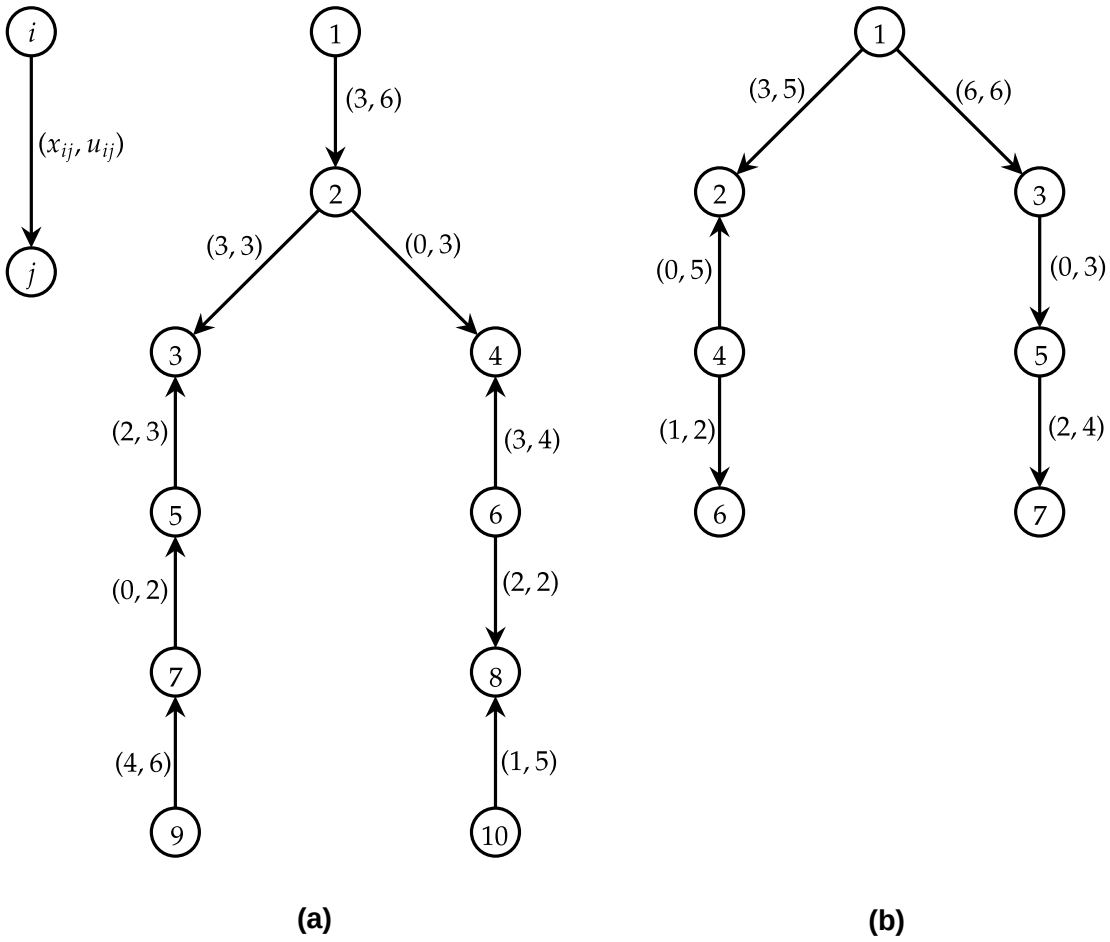

Figura 3.7: Árboles de expansión factibles: a) altamente factible; b) no altamente factible.

Regla del arco saliente. Se selecciona el arco saliente como el último arco de bloqueo encontrado al atravesar el ciclo de pivote W a lo largo de su orientación comenzando en el vértice esquina w.

Para ilustrar la regla del arco saliente, se parte del árbol de expansión altamente factible de la Figura 3.8. Sea $(9, 10)$ el arco de entrada, el ciclo de pivote es $10 - 8 - 6 - 4 - 2 - 3 - 5 - 7 - 9 - 10$ y el vértice esquina es el nodo 2. Este pivote está degenerado porque los arcos $(2, 3)$ y $(7, 5)$ bloquean cualquier flujo en el ciclo de pivote. Al atravesar el ciclo de pivote que comienza en el nodo 2, uno se topa con el arco $(7, 5)$ más tarde que con el arco $(2, 3)$; luego se selecciona el arco $(7, 5)$ como el arco saliente.

A continuación, se muestra que la regla del arco saliente garantiza que en el siguiente

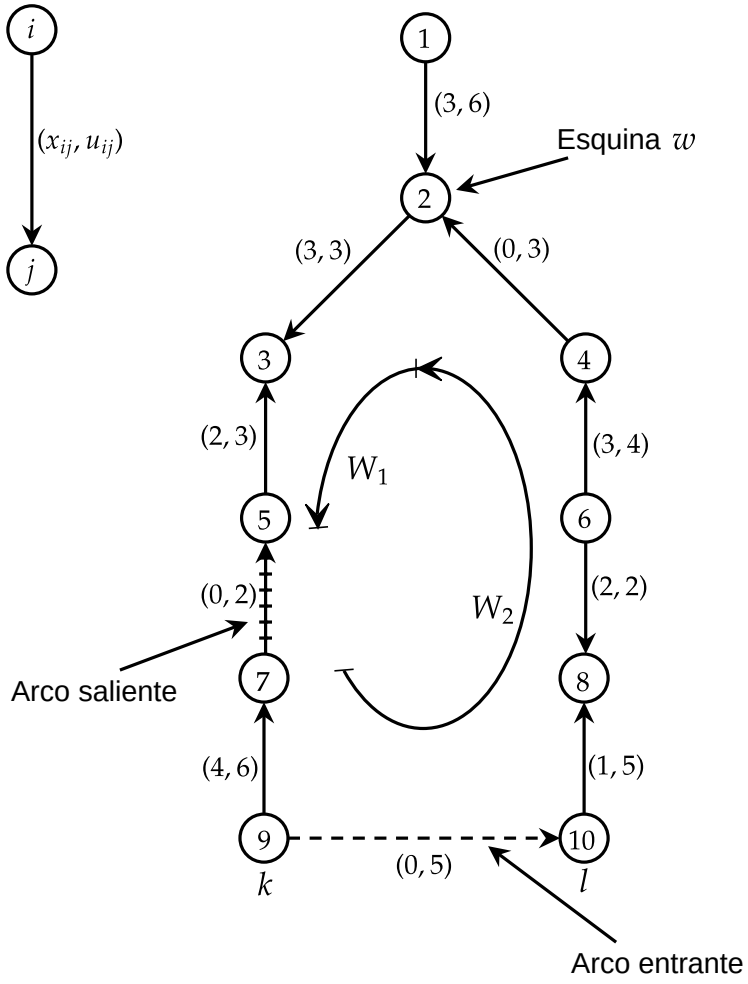

Figura 3.8: Selección del arco saliente.

árbol de expansión todo nodo del ciclo W puede enviar una cantidad positiva de flujo al nodo raíz. Sea (p, q) el arco seleccionado por la regla del arco saliente. Sea W_1 el segmento del ciclo W entre el vértice esquina w y el arco (p, q) donde se recorre el ciclo siguiendo su orientación. Sea $W_2 = W - W_1 - \{(p, q)\}$. Las orientaciones de los segmentos W_1 y W_2 se toman de tal manera que sean compatibles con W (veáse la Figura 3.8). Se usa la siguiente propiedad sobre los nodos en el segmento W_2.

Propiedad 3.4. Cada nodo del segmento W_2 puede enviar una cantidad positiva de flujo a la raíz del siguiente árbol de expansión.

Esta observación se deriva del hecho de que el arco (p, q) es el último arco de bloqueo en W; luego, ningún arco en W_2 se bloquea y cada nodo de este segmento puede

enviar una cantidad positiva de flujo a la raíz a través del nodo w a lo largo de la orientación de W_2. Si el arco saliente no satisface la regla del arco saliente, ningún nodo en el segmento W_2 puede enviar una cantidad positiva de flujo a la raíz; por lo tanto, el próximo árbol de expansión no será altamente factible.

Propiedad 3.5. Cada nodo del segmento W_1 puede enviar una cantidad positiva de flujo a la raíz del siguiente árbol de expansión.

Se muestra esta observación considerando dos casos. Si el pivote anterior era un pivote no degenerado, el pivote aumenta una cantidad positiva de flujo δ a lo largo de los arcos en W_1. En consecuencia, después del aumento, todo nodo en el segmento W_1 puede enviar de vuelta una cantidad positiva de flujo en dirección opuesta a la orientación de W_1 a través del vértice esquina w (cada nodo puede enviar al menos δ unidades al vértice esquina y luego al menos parte de este flujo a la raíz ya que el árbol de expansión anterior era altamente factible). Si el pivote anterior era un pivote degenerado, W_1 debe estar contenido en el segmento de W entre el nodo w y el nodo k porque la Propiedad 3.5 supone que cada nodo en la ruta desde el nodo l al nodo w puede enviar una cantidad positiva de flujo a la raíz antes del pivote, por lo que ningún arco en esta trayectoria puede ser un arco de bloqueo en un pivote degenerado. Todo nodo en W_1 antes del pivote podría enviar una cantidad positiva de flujo a la raíz y, por lo tanto, dado que el pivote no cambia los valores de flujo, cada nodo en W_1 debe poder enviar una cantidad positiva de flujo a la raíz también después del pivote. Luego en el siguiente árbol de expansión cada nodo del ciclo W puede enviar una cantidad positiva de flujo al nodo raíz.

A continuación, se muestra que en el siguiente árbol de expansión, los nodos que no pertenecen al ciclo W también pueden enviar una cantidad positiva de flujo a la raíz. En el árbol de expansión anterior (antes del aumento), cada nodo j podría enviar una cantidad positiva de flujo a la raíz y si la ruta del árbol desde el nodo j no pasa por el ciclo W, la misma ruta está disponible para llevar una cantidad positiva de flujo en el siguiente árbol de expansión. Si la ruta del árbol desde el nodo j pasa por el ciclo W, el segmento de esta ruta del árbol al ciclo W está disponible para transportar una cantidad positiva de flujo en el siguiente árbol de expansión y una vez que una cantidad positiva de flujo alcanza el ciclo W, luego, puede enviarse (o parte de él) al nodo raíz. Esta conclusión completa la prueba de que el próximo árbol de expansión es altamente factible.

Compruébese, ahora la finitud del algoritmo simplex en redes. Cada pivote no degenerado disminuye estrictamente el valor de la función objetivo, el número de pivotes no degenerados es finito. Sin embargo, el algoritmo también puede tomar pivotes dege-

nerados. Se muestra a continuación que el número de pivotes degenerados entre dos pivotes no degenerados cualesquiera está limitado de forma finita. Supóngase que el arco (k, l) entra en el árbol de expansión en su cota inferior y, al hacerlo, define un pivote degenerado. En este caso, el arco saliente pertenece a la trayectoria del árbol desde el nodo k hasta el vértice w. El nodo k se encuentra en el subárbol T_2 y los potenciales de todos los nodos en T_2 cambian en una cantidad c_{kl}^{π}. Dado que $c_{kl}^{\pi} < 0$, este pivote degenerado disminuye estrictamente la suma de todos los potenciales de nodo (que es entera). Puesto que ningún potencial de nodo puede caer por debajo de $-nC$, el número de pivotes degenerados sucesivos es finito.

Hasta ahora se ha asumido que los arcos de entrada tienen siempre el valor de sus cotas inferiores. Si el arco de entrada (k, l) tiene el valor de su cota superior, se define la orientación del ciclo W como opuesta a la orientación del arco (k, l). El criterio para seleccionar el arco saliente permanece sin cambios: el arco saliente es el último arco de bloqueo que se encuentra al atravesar W a lo largo de su orientación comenzando en el vértice w. En este caso, el nodo l está contenido en el subárbol T_2 y, por lo tanto, después del pivote, los potenciales de todos los nodos T_2 disminuyen en la cantidad $c_{kl}^{\pi} > 0$; en consecuencia, el pivote vuelve a reducir la suma de los potenciales de los nodos.

3.7.1 Algoritmo simplex en redes para el problema del camino más corto

Se estudia a continuación la versión del problema de la ruta más corta en la que desea enviar una unidad de flujo desde la fuente a todos los demás nodos de la red a lo largo de rutas de mínimo coste. Puede formularse esta versión del problema de la ruta más corta como el siguiente modelo de flujo de mínimo coste:

$$\text{mín} \sum_{(i,j)\in A} c_{ij} x_{ij}, \tag{3.27}$$

$$\text{s. a} \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} n-1 & \text{para } i = s \\ -1 & \text{para todo } i \in N - \{s\} \end{cases}, \tag{3.28}$$

$$x_{ij} \leq 0 \qquad \text{para todo } (i, j) \in A. \tag{3.29}$$

Si la red contuviera un ciclo dirigido de coste negativo, esta formulación tendría una solución ilimitada ya que podría enviarse una cantidad infinita de flujo a lo largo de este ciclo sin violar ninguna de las restricciones (los flujos de arco no tienen cotas

superiores). El algoritmo simplex en redes que se describe detecta los ciclos negativos, y si la red no los contiene, determina las distancias de ruta más cortas.

Sea árbol de salida dirigido aquel árbol que conecta el nodo fuente s con el resto de nodos del grafo mediante caminos dirigidos y únicos para cada nodo. El problema de la ruta más corta tiene una solución de árbol de expansión. Debido a que el nodo s es el único nodo de origen y todos los demás nodos son nodos de demanda, el árbol de expansión debe de ser un árbol de salida dirigido. En un árbol de salida dirigido, cada nodo que no sea la fuente tiene exactamente un arco entrante pero podría tener varios arcos salientes (véase la Figura 3.9). Dado que cada nodo, excepto el nodo s, tiene demanda unitaria, el flujo del arco (i, j) es $|D(j)|$ ($D(j)$ es el conjunto de descendientes del nodo j en el árbol de expansión y, por definición, este conjunto incluye al node j). Por lo tanto, cada árbol del problema de la ruta más corta no es degenerado y, en consecuencia, el algoritmo simplex en redes nunca realizará pivotes degenerados.

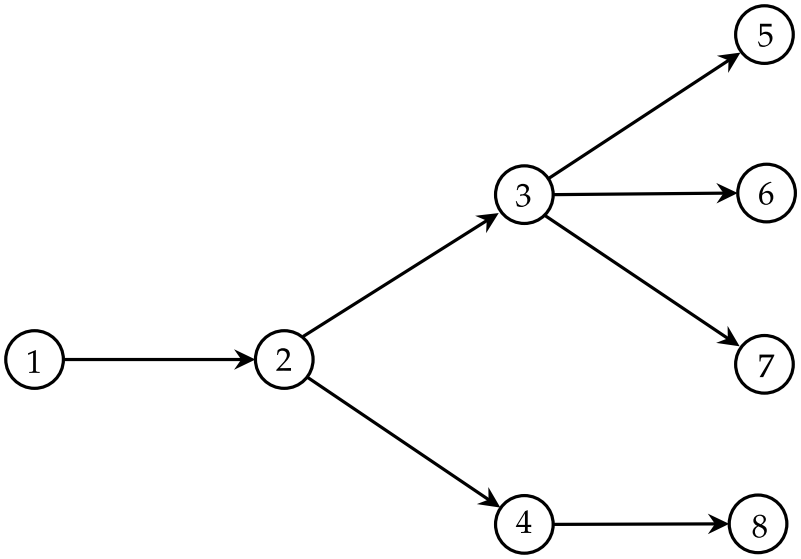

Figura 3.9: Ejemplo de árbol de salida.

Cualquier árbol de expansión del problema de la ruta más corta contiene una ruta dirigida única desde el nodo s hasta cada uno del resto de los nodos. Sea $P(k)$ la ruta desde el nodo s al nodo k. Se obtienen los potenciales de nodo correspondientes al árbol T estableciendo $\pi(s) = 0$ y luego usando la ecuación $c_{ij} - \pi(i) - \pi(j) = 0$ para cada arco $(i, j) \in T$ partiendo desde el nodo s (ver Figura 3.10). En un árbol de salida dirigido del árbol de expansión $\pi(k) = -\sum_{(i,j)\in P(k)} c_{ij}$ es la longitud del camino $P(k)$.

Dado que las variables en la formulación de flujo de mínimo coste del problema de la

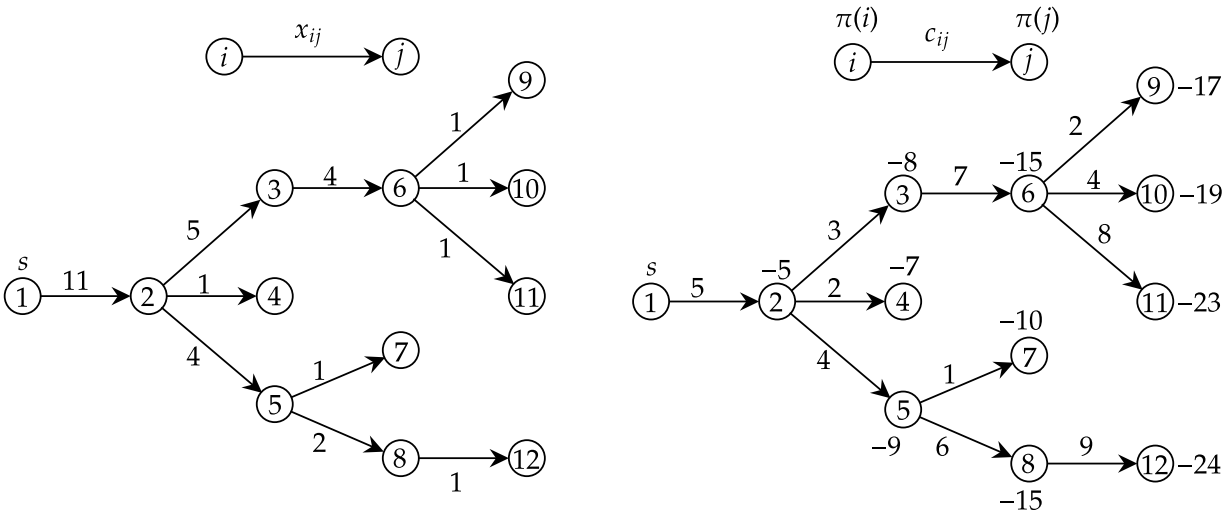

Figura 3.10: Cálculo de los potenciales de nodo.

ruta más corta no tienen cotas superiores, cada arco que no pertenece al árbol tiene el valor de su cota inferior. El algoritmo selecciona un arco (k, l) no perteneciente al árbol con un coste reducido negativo para introducirlo en el árbol de expansión. La adición del arco (k, l) al árbol crea un ciclo que se orienta en la misma dirección que el arco (k, l). Sea w el vértice esquina de este ciclo. La Figura 3.11 ilustra esto. En este ciclo, cada arco desde el nodo l al nodo w es un arco hacia atrás y cada arco desde el nodo w al nodo k es un arco hacia adelante. En consecuencia, el arco saliente estaría en el segmento del nodo l al nodo w. De hecho, el arco saliente sería el arco $(pred(l), l)$ porque este arco tiene el valor de flujo más pequeño entre todos los arcos en el segmento desde el nodo l al nodo w. Luego, el algoritmo aumentaría los potenciales de los nodos en el subárbol partiendo del nodo l con una cantidad de $|c^{\pi}_{kl}|$, actualizaría los índices del árbol y repetiría los cálculos hasta que todos los arcos ajenos al árbol tengan costes reducidos no negativos. Cuando el algoritmo termina, el árbol final sería un árbol de ruta más corta (un árbol en el que la ruta dirigida desde los nodos a todos los demás nodos es la ruta más corta).

Recuérdese, que al implementar el algoritmo simplex en redes para el problema de flujo de mínimo coste, se mantiene los valores de flujo para todos los arcos porque se necesitan estos valores para identificar el arco saliente. Sin embargo, para el problema de la ruta más corta, puede determinarse el arco saliente sin considerar los valores de flujo. Si (k, l) es el arco de entrada, entonces $(pred(l), l)$ es el arco de salida. Por lo tanto, el algoritmo simplex en redes para el problema de la ruta más corta no necesita

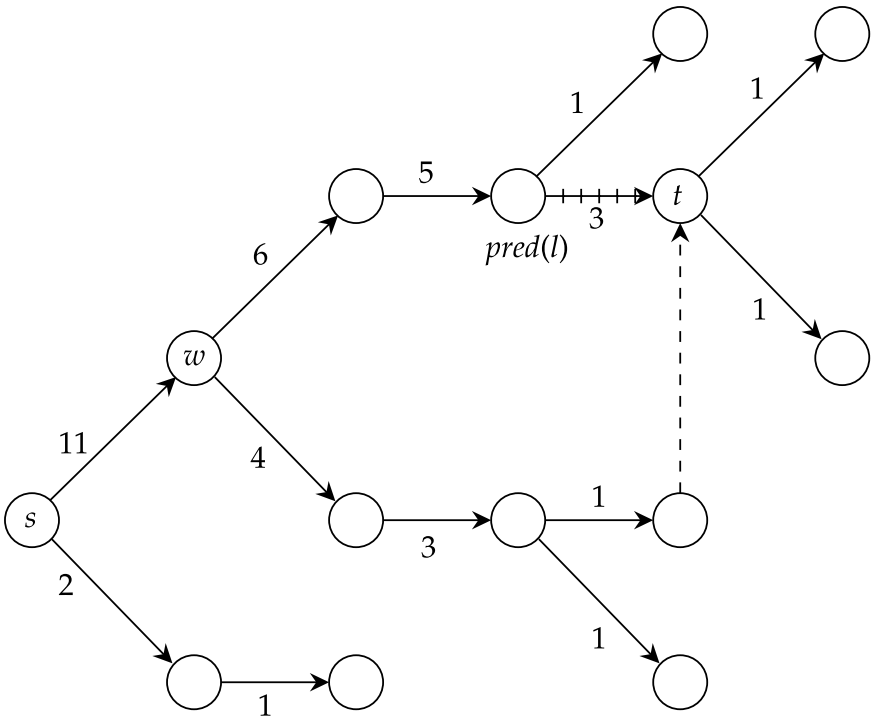

Figura 3.11: Elección del arco saliente.

mantener los flujos de arco. Además, la actualización de los índices del árbol es más sencilla para el problema de la ruta más corta.

El algoritmo simplex en redes aplicado al problema del camino más corto es similar al Algoritmo 6. Sea $d(i) = -\pi(i)$ la etiqueta de distancia válida, que representa la distancia de un camino dirigido desde la fuente s hasta el nodo i. El algoritmo simplex en redes busca un arco que satisfaga la condición $\pi(j) < \pi(i) - c_{ij}$, y establece la etiqueta de distancia válida del nodo j igual a $d(i)+c_{ij}$. En cada iteración, el algoritmo simplex en redes selecciona un arco (i, j) con $c^{\pi}_{ij} < 0$, siendo $c^{\pi}_{ij} = c_{ij} - \pi(i) + \pi(j) = c_{ij} + d(i) - d(j)$. Luego, el algoritmo aumenta el potencial de cada nodo en el subárbol que parte del nodo j en una cantidad $|c^{\pi}_{ij}|$ que equivale a disminuir la etiqueta de distancia válida de todos los nodos en el subárbol enraizado en el nodo j en una cantidad $|c^{\pi}_{ij}|$.

$d(s) \leftarrow 0$ y $pred(s) \leftarrow 0$;
$d(j) \leftarrow \infty$ para cada $j \in N - \{s\}$;
mientras *algún arco* (i, j) *satisfaga* $d(j) > d(i) + c_{ij}$ **hacer**
  $d(j) \leftarrow d(i) + c_{ij}$;
  $pred(j) \leftarrow i$;
fin

Algoritmo 6: Algoritmo de corrección del etiquetado.

Si la red no contiene ningún ciclo negativo, el algoritmo simplex en redes termina con un árbol de ruta más corto. Cuando la red contiene algún ciclo negativo, el algoritmo eventualmente encuentra una situación como la que se muestra en la Figura 3.12. Este tipo de situación ocurre sólo cuando el nodo inicial del arco de entrada (k, l) pertenece a $D(l)$, el conjunto de descendientes del nodo l. El algoritmo simplex en redes puede detectar esta situación fácilmente sin ningún aumento significativo en su esfuerzo computacional: después de introducir un arco (k, l), el algoritmo actualiza los potenciales de todos los nodos en $D(l)$; en ese momento, puede verificar si $k \in D(l)$ y, de ser así, terminar. En este caso, rastrear los índices predecesores producirá un ciclo negativo.

La versión genérica del algoritmo simplex en redes para el problema de la ruta más corta se ejecuta en un tiempo pseudopolinomial. Este resultado se deriva de que:

1. para cada nodo i, $-nC \leq \pi(i) \leq nC$ (porque la longitud de cada camino dirigido desde s a el nodo i se encuentra entre $-nC$ y nC), y
2. cada iteración aumenta el valor de al menos un potencial de nodo.

Hay, sin embargo, implementaciones que resuelven en tiempo polinomial, alguna de ellas son la regla de pivote del primer arco elegible, la regla de pivote de Dantzig o la regla del pivote escalado.

3.7.2 Algoritmo simplex en redes para el problema de flujo máximo

El problema de flujo máximo puede considerarse una versión particular del problema de flujo de mínimo coste, obtenido al introducir un arco adicional (t, s) con coeficiente de coste $c_{ts} = -1$ y una cota superior $u_{ts} = \infty$, y estableciendo $c_{ij} = 0$ para todos los arcos (i, j) en A. Para simplificar la notación, de ahora en adelante asúmase que A

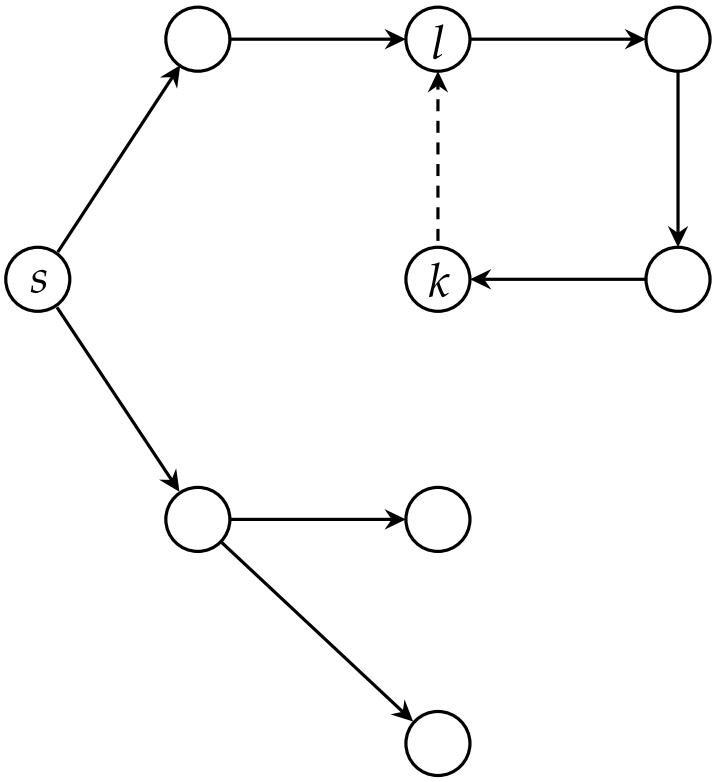

Figura 3.12: Detectando un ciclo negativo en la red.

representa el conjunto $A \cup \{(t, s)\}$. La formulación del problema es pues:

$$\text{minimizar} \quad -x_{ts}, \tag{3.30}$$

$$\text{s. a} \quad \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = 0 \qquad \text{para todo } i \in N, \tag{3.31}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{para todo } (i, j) \in A. \tag{3.32}$$

Minimizar $-x_{ts}$ equivale a maximizar x_{ts}, lo que equivale a maximizar el flujo neto enviado desde la fuente al sumidero, ya que este flujo regresa a la fuente a través del arco (t, s). Esto explica por qué la entrada es igual a la salida para todos los nodos de la red, incluidos los nodos fuentes y sumideros.

En cualquier solución de árbol de expansión factible que lleve una cantidad positiva de flujo desde la fuente al sumidero ($x_{ts} > 0$), el arco (t, s) debe de estar en el árbol de expansión. En consecuencia, el árbol de expansión para el problema de flujo máximo consta de dos subárboles de G unidos por el arco (t, s) (ver Figura 3.13). Sean T_s y T_t los subárboles que contienen los nodo s y t.

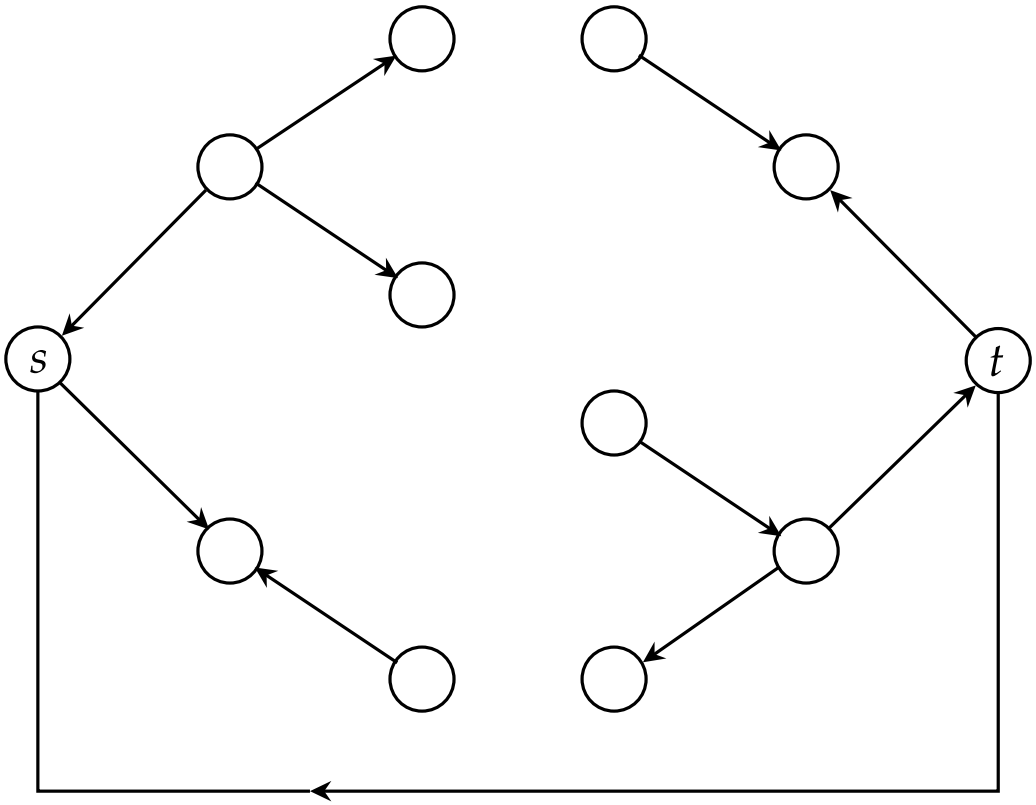

Figura 3.13: Árbol de expansión para el problema de máximo flujo.

Los potenciales de nodo correspondientes a un árbol de expansión factible del problema de flujo máximo se obtienen de la siguiente manera. Dado que puede establecerse el potencial de nodo arbitrariamente, sea $\pi(t) = 0$. Además, puesto que el coste reducido del arco (t, s) debe ser cero, $0 = c^{\pi}_{ts} = c_{ts} - \pi_t + \pi_s = -1 + \pi_s$, luego, $\pi_s = 1$. Dado que

1. el coste reducido de cada arco en T_s y T_t debe ser cero, y
2. los costes de estos arcos son también cero,

los potenciales de nodo tienen los siguientes valores: $\pi(i) = 1$ para todo nodo $i \in T_s$, y $\pi(i) = 0$ para todo nodo $i \in T_t$.

Cada solución de árbol de expansión del problema de flujo máximo define un corte $s-t$ ($[S, \overline{S}]$) en la red original obtenida estableciendo $S = T_s$ y $\overline{S} = T_t$. Cada arco de este corte es un arco sin árbol; su flujo tiene valor cero o es igual a la capacidad del arco. Para cada arco hacia adelante (i, j) en el corte, $c_{ij}^{\pi} = -1$, y para cada arco hacia atrás (i, j) en el corte, $c_{ij}^{\pi} = 1$. Además, para todo arco $(i, -j)$ que no está en el corte, $c_{ij}^{\pi} = 0$. En consecuencia, si cada arco hacia adelante en el corte tiene un valor de flujo igual a la capacidad del arco y cada arco hacia atrás tiene flujo cero, esta solución de árbol de expansión satisface las condiciones de optimalidad (3.18)-(3.20) y, por lo tanto, debe ser óptima. Por otro lado, si en la solución actual del árbol de expansión, algún arco hacia adelante en el corte tiene un flujo nulo o el flujo en algún arco hacia atrás es igual a la capacidad del arco, todos estos arcos tienen una violación unitaria. Por lo tanto, puede seleccionarse cualquiera de estos arcos para ingresar al árbol de expansión. Supóngase que se selecciona el arco (k, l). La introducción de este arco en el árbol crea un ciclo que contiene el arco (t, s) como un arco hacia adelante (vea la Figura 3.14). El algoritmo aumenta el flujo máximo posible en este ciclo e identifica un arco de bloqueo. Dejar caer este arco nuevamente crea dos subárboles unidos por el arco (t, s). Este nuevo árbol constituye un árbol de expansión para la próxima iteración.

Este algoritmo es un algoritmo de camino incremental: la estructura de árbol permite determinar la ruta desde la fuente hasta el sumidero con mucha facilidad. En este sentido, el algoritmo simplex en redes tiene una ventaja sobre otros tipos de algoritmos de camino incremental para el problema de flujo máximo. Sin embargo, debido a la degeneración, es posible que el algoritmo simplex en redes no envíe una cantidad positiva de flujo desde la fuente al sumidero en cada iteración.

Teorema 3.6 (Teorema de flujo máximo y mínimo corte). *El valor máximo del flujo desde la fuente s hasta el sumidero t en una red capacitada es igual a la mínima capacidad entre todos los cortes $s - t$.*

El algoritmo simplex en redes para el problema de flujo máximo proporciona una prueba del Teorema 3.6. El algoritmo termina cuando se capacita cada arco hacia adelante en el corte y cada arco hacia atrás tiene un flujo nulo. Esta condición de terminación implica que el valor de flujo máximo actual es igual a la capacidad del corte $s - t$ definido por los subárboles T_s y T_t, y por lo tanto el valor de un flujo máximo desde el nodo s al nodo t es igual a la capacidad del corte mínimo $s - t$.

Así como la mecánica del algoritmo simplex en redes se vuelve más simple en el contexto del problema de flujo máximo, también lo hace en el contexto de un árbol de expansión altamente factible. Si se designa el sumidero como nodo raíz, la definición de un árbol de expansión altamente factible implica que pueda enviarse una cantidad

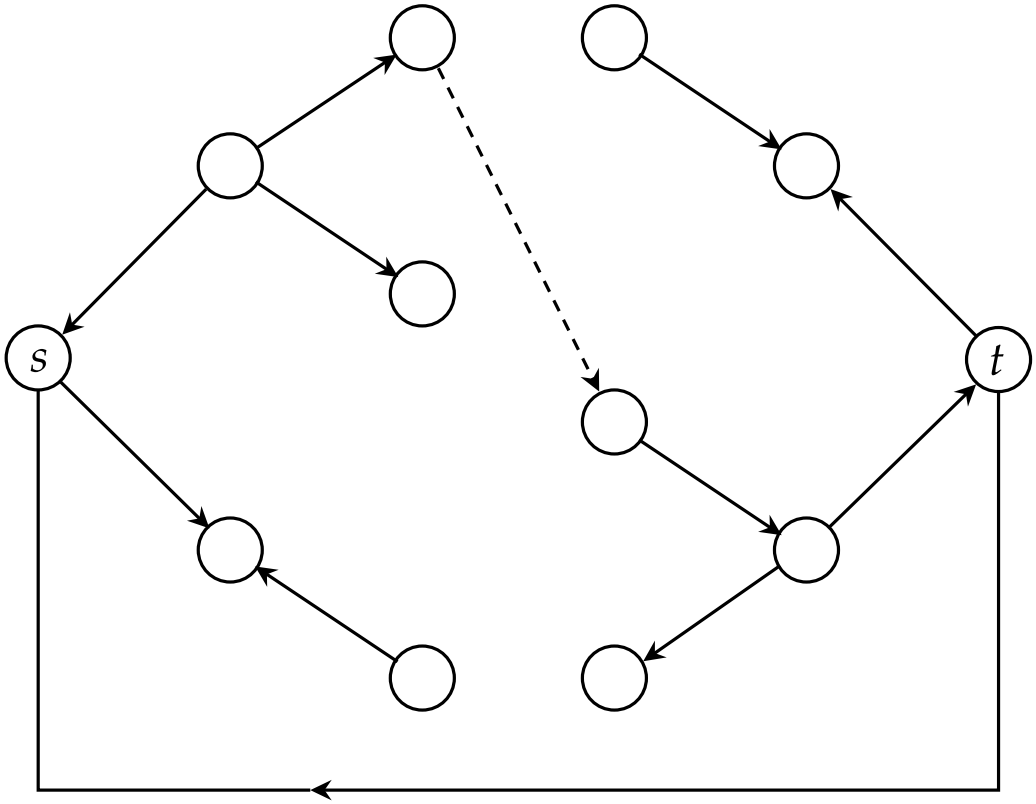

Figura 3.14: Formación de un ciclo.

positiva de flujo desde cada nodo en T_t al nodo sumidero t sin violar ninguno de las restricciones de flujo. Por lo tanto, todo arco en T_t cuyo valor de flujo es cero debe apuntar hacia el nodo sumidero t y todo arco en T_t cuyo valor de flujo, que es igual a la cota superior del arco, debe apuntar en sentido contrario al nodo t. Además, el criterio del arco saliente se reduce a seleccionar un arco de bloqueo en el ciclo de pivote que está más alejado del nodo sumidero cuando se atraviesa el ciclo en la dirección del arco (t, s) comenzando desde el nodo t. Cada pivote degenerado selecciona un arco entrante que incide en algún nodo de T_t. La observación anterior implica que cada arco de bloqueo debe ser un arco en T_s. En consecuencia, cada pivote degenerado aumenta el tamaño de T_t, por lo que el algoritmo puede realizar como máximo n pivotes degenerados consecutivos. Podría darse que el problema de flujo de coste mínimo no satisficiera esto: para el problema más general, el número de pivotes degenerados consecutivos puede ser exponencialmente grande.

La discusión anterior muestra que cuando se implementa para mantener un árbol de

expansión altamente factible, el algoritmo simplex en redes realiza $\mathcal{O}(n^2U)$ iteraciones para el problema de flujo máximo. Este resultado se deriva del hecho de que el número de pivotes no degenerados es como máximo nU, cota superior del valor de flujo máximo. Esta magnitud en el número de iteraciones no es polinomial, por lo que no es satisfactorio desde una perspectiva teórica.

3.7.3 Análisis de sensibilidad

El propósito del análisis de sensibilidad es determinar cambios en la solución óptima del problema de flujo de mínimo coste como producto de cambios en los datos (vector de oferta/demanda, capacidad o coste de cualquier arco).

El análisis de sensibilidad adopta el siguiente enfoque básico. Primero se determina el efecto de un cambio dado en los datos sobre la viabilidad y la optimización de la solución asumiendo que la estructura del árbol de expansión permanece sin cambios. Si el cambio afecta a la optimalidad de la estructura del árbol de expansión, se realizan pivotes (primarios) para lograr la optimización. Siempre que el cambio destruye la viabilidad de la estructura del árbol de expansión, se realizan pivotes duales para lograr su viabilidad.

Sea x^* una solución óptima del problema de flujo de mínimo coste, (T^*, L^*, U^*) su correspondiente estructura de árbol de expansión y π^* denota los potenciales de nodo correspondientes. Primero se considera el análisis de sensibilidad con respecto a los cambios en los coeficientes de coste.

Coste del análisis de sensibilidad

Supóngase que el coste de un arco (p, q) aumenta en λ unidades. El análisis será distinto si el arco (p, q) está en el árbol o no.

Caso 1. El arco (p, q) no está en el árbol.

En este caso, cambiar el coste del arco (p, q) no cambia los potenciales de nodo de la estructura del árbol de expansión. El coste reducido modificado del arco (p, q) es $c_{pq}^{\pi^*} + \lambda$. Si el coste reducido modificado satisface la condición (3.19) o (3.20) (la que sea apropiada) la estructura del árbol de expansión sigue siendo óptima. De lo contrario, vuelve a optimizarse la solución utilizando el algoritmo simplex en redes con (T^*, L^*, U^*) como estructura de árbol de expansión inicial.

Caso 2. El arco (p, q) está en el árbol.

En este caso, cambiar el coste del arco (p, q) cambia algunos potenciales de nodo. Si el arco (p, q) es un arco que apunta hacia arriba en el árbol de expansión, los potenciales de todos los nodos en $D(p)$ aumentan en λ, y si (p, q) es un arco que apunta hacia abajo, los potenciales de todos los nodos en $D(q)$ disminuyen en λ. Nótese que estos cambios alteran los costes reducidos de aquellos arcos ajenos al árbol que pertenecen al corte $([D(q), \overline{D}(q)])$. Si todos los arcos que no son de árbol aún satisfacen la condición de optimalidad, la estructura actual del árbol de expansión sigue siendo óptima; de lo contrario, vuelve a optimizarse la solución utilizando el algoritmo simplex en redes.

Análisis de sensibilidad de oferta/demanda

Supóngase que el vector de oferta/demanda $b(k)$ del nodo k aumenta en λ y la oferta/demanda $b(l)$ de otro nodo l disminuye en λ (dado que: $\sum_{i \in N} b(i) = 0$, los suministros de dos nodos deben cambiar simultáneamente, en igual magnitud y direcciones opuestas). Las restricciones de balance de flujo requieren que deba enviarse λ unidades de flujo desde el nodo k al nodo l. Sea P la ruta de árbol única del nodo k al nodo l. Sean $\overline{P}$ y $\underline{P}$, los conjuntos de arcos en P con la misma y opuesta dirección a la trayectoria respectivamente. El cambio de flujo máximo δ_{ij} en un arco $(i, j) \in P$ que respeta las restricciones de flujo es

$$\delta_{ij} = \begin{cases} u_{ij} - x_{ij} & \text{si } (i, j) \in \overline{P}, \\ x_{ij} & \text{si } (i, j) \in \underline{P}. \end{cases} \tag{3.33}$$

Sea

$$\delta = \text{mín}\{\delta_{ij} : (i, j) \in P\}. \tag{3.34}$$

Si $\lambda \leq \delta$, se envía λ unidades de flujo desde el nodo k al nodo l a lo largo de la ruta P. La solución modificada es factible para el problema modificado y dado que la modificación en $b(i)$ no afecta a la optimalidad de la solución, la solución resultante debe ser una solución óptima del problema modificado.

Si $\lambda > \delta$, no puede enviarse λ unidades de flujo desde el nodo k al nodo l a lo largo de los arcos del árbol de expansión y preservar su viabilidad. En este caso se envía δ unidades de flujo a lo largo de P y se reduce λ a $\lambda - \delta$. Sea x' el flujo actualizado, se realiza un pivote dual para obtener un nuevo árbol de expansión que podría permitir

el envío de flujo adicional desde el nodo k al nodo l a lo largo de la ruta del árbol. En un pivote dual, primero se decide la variable saliente y luego se identifica una variable entrante. Sea (p, q) un arco en $\boldsymbol{P}$ que impide enviar flujo adicional desde el nodo k al nodo l. Si $(p, q) \in \overline{P}$, entonces $x'_{pq} = u_{pq}$ y si $(p, q) \in \underline{P}$, entonces $x'_{pq} = 0$. Se elimina el arco (p, q) del árbol de expansión dividiendo el conjunto de nodos en dos subárboles. Sea S el subárbol que contiene el nodo k y $\overline{S}$ el subárbol que contiene el nodo l. Dado que se desea enviar flujo adicional a través del corte $[S, \overline{S}]$, los arcos elegibles para entrar en el árbol serían los arcos hacia adelante en el corte en su cota inferior o los arcos hacia atrás en sus cortes superiores. Si la red no contiene arcos capacitados no puede enviarse ningún flujo adicional desde el nodo k al nodo l y el problema modificado es inviable. Si la red contiene arcos capacitados, entonces, entre estos arcos, se selecciona un arco, por ejemplo (g, h), cuyo coste reducido tiene la menor magnitud. Se introduce el arco (g, h) en el árbol de expansión y se actualizan los potenciales de nodo.

Luego se intenta nuevamente enviar $\lambda' = \lambda - \delta$ unidades de flujo desde el nodo k al nodo l en la ruta del árbol. Si se logra, se termina; de lo contrario, se envía el máximo flujo posible y se realiza otro doble pivote para obtener una nueva estructura de árbol de expansión. Se repiten estos cálculos hasta que se establece un flujo factible en la red o se descubre que el problema modificado no es factible.

Análisis de sensibilidad de la capacidad

Finalmente, se considera el análisis de sensibilidad con respecto a las capacidades del arco. Considérese el caso en el que la capacidad de un arco (p, q) aumenta en λ unidades. Siempre que se aumenta la capacidad de cualquier arco, la solución óptima anterior sigue siendo factible; para determinar si esta solución sigue siendo óptima, deben verificar las condiciones de optimalidad (3.18) - (3.20). Esté el arco en el árbol o no lo esté, aumentar u_{pq} en λ unidades no afecta la condición de optimalidad. Sin embargo, si el arco (p, q) con el valor de su cota superior no pertenece al árbol y su capacidad aumenta en λ unidades, la condición de optimalidad (3.20) dicta que se debe aumentar el flujo en el arco en λ unidades. Hacerlo crea un exceso de unidades λ en el nodo q y un déficit de unidades λ en el nodo p. Para lograr la viabilidad, debe enviarse unidades λ del nodo q al nodo p. Se logra este objetivo utilizando el método descrito anteriormente en el análisis de sensibilidad de oferta/demanda.

3.7.4 Unimodularidad

Sea $\mathcal{A}$ una matriz $p \times q$ con elementos enteros y p filas linealmente independientes (el rango de la matriz es p). La matriz $\mathcal{A}$ es unimodular si el determinante de cada matriz base $\mathcal{B}$ de $\mathcal{A}$ tiene valor $+1$ o -1 ($det(\mathcal{B}) = \pm 1$). Una submatriz $p \times p$ de $\mathcal{A}$ es una matriz base si sus columnas son linealmente independientes. El siguiente resultado clásico muestra la relación entre la unimodularidad y la solubilidad entera de problemas lineales.

Teorema 3.7 (Teorema de unimodularidad). *Sea $\mathcal{A}$ una matriz entera con filas linealmente independientes. Entonces las siguientes tres condiciones son equivalentes:*

(a) $\mathcal{A}$ es unimodular
(b) Cada base de solución factible definida por las restricciones $\mathcal{A}x = b, x \geq 0$, es entero si el vector b es también entero.
(c) Cada matriz base $\mathcal{B}$ de $\mathcal{A}$ tiene una inversa entera $\mathcal{B}^{-1}$.

Este resultado muestra cuando un problema lineal de la forma

$$\text{mín} \quad cx \tag{3.35}$$
$$\text{s. a} \quad \mathcal{A}x = b, x \geq 0 \tag{3.36}$$

tiene soluciones óptimas enteras para todos los vectores enteros del lado derecho b y para todos los vectores de costes c. Los problemas de flujo de red son la clase más importante de modelos que satisfacen esta propiedad de integralidad. Para establecer una conexión formal entre los flujos de red y los resultados incorporados en este teorema, se considera otra clase de matrices digna de mención.

Las matrices totalmente unimodulares son una subclase importante de matrices unimodulares. Una matriz $\mathcal{A}$ es totalmente unimodular si cada submatriz cuadrada de $\mathcal{B}$ tiene determinante 0 o ± 1. Toda matriz totalmente unimodular $\mathcal{A}$ es unimodular porque cada matriz base $\mathcal{B}$ debe tener determinante ± 1 (un determinante de valor cero implicaría la dependencia lineal de las columnas de $\mathcal{B}$). Sin embargo, una matriz unimodular no necesita ser totalmente unimodular. Las matrices totalmente unimodulares son importantes, en gran parte, porque las matrices de restricción de los problemas de flujo de mínimo coste son totalmente unimodulares.

Teorema 3.8. *La matriz de incidencia arco-nodo N de una red dirigida es totalmente unimodular.*

Este resultado (Teorema 3.8), combinado con el Teorema 3.7, proporciona una prueba de la propiedad de integralidad de los flujos de red: los modelos de flujo de red tienen soluciones óptimas enteras porque cada matriz de incidencia de arco de nodo es totalmente unimodular y, por lo tanto, unimodular. Las matrices de restricciones para muchas extensiones del problema de flujo de red básico, por ejemplo, los flujos multiservicios, no son unimodulares. Por lo tanto, no se esperaría que las soluciones óptimas de estos modelos fueran enteras incluso cuando todos los datos subyacentes son enteros. Por lo tanto, para encontrar soluciones enteras en estos problemas, se necesita confiar en métodos de programación entera. Una de las principales razones por las que puede resolverse el problema de flujo de mínimo coste de manera tan eficiente y aún obtener soluciones enteras es porque, como se refleja en la propiedad de integralidad, las soluciones básicas factibles de la formulación de programación lineal de este problema son enteras siempre que los datos subyacentes son enteros.

Las propiedades de unimodularidad brindan un resultado muy fuerte: se garantiza que cualquier solución básica factible será entera siempre que el vector b del lado derecho sea entero.

4 Problema de flujo multiservicio

4.1 Introducción

En este capítulo se estudia el problema introducido en Sección 3.2 y todas las demostraciones de los resultados y proposiciones se encuentran en el libro publicado por *Ahuja*, *Magnanti* y *Orlin* [3] en 1993. Sea x_{ij}^k el flujo del servicio k en el arco (i, j), y sea x^k y c^k el vector de flujo y el vector de coste por unidad de flujo en el servicio k. Haciendo uso de esta notación se formula el problema de flujo multiservicio como:

$$\text{Minimizar} \quad \sum_{1 \le k \le K} c^k x^k \tag{4.1}$$

$$\text{s. a} \quad \sum_{1 \le k \le K} x_{ij}^k \le u_{ij} \quad \text{para todo } (i, j) \in A, \tag{4.2}$$

$$\mathcal{N} x^k = b^k \quad \text{para } k = 1, 2, \cdots, K, \tag{4.3}$$

$$0 \le x_{ij}^k \le u_{ij}^k \quad \text{para todo } (i, j) \in A \text{ y todo } k = 1, 2, \cdots, K. \tag{4.4}$$

Esta formulación presenta una colección de K restricciones de balance de flujo (4.3), que modela el flujo de cada servicio $k = 1, 2, \ldots, K$. El "paquete" de restricciones (4.2) liga todos los servicios, restringiendo el flujo total $\sum_{1 \le k \le K} x_{ij}^k$ de todos los servicios en cada arco (i, j) a como mucho u_{ij}. Nótese que también se impone límite a la capacidad del flujo individual u_{ij}^k del servicio k en el arco (i, j).

A veces, será más conveniente tomar el paquete de restricciones (4.2) como igualdades en lugar de desigualdades. En estos casos, se introducen las variables de holgura s_{ij} y se escribe el paquete de restricciones de capacidad de flujo como:

$$\sum_{1 \le k \le K} x_{ij}^k + s_{ij} = u_{ij} \quad \text{para todo } (i, j) \in A. \tag{4.5}$$

La variable de holgura s_{ij} para el arco (i, j) mide la capacidad no utilizada de ese arco.

4.1.1 Supuestos

Nótese que el modelo (4.1)-(4.4) impone capacidades a los arcos pero no a los nodos. Esto no se traduce en una falta de generalidad dado que usando la división de nodos (3.2) puede usarse esta formulación para modelizar situaciones con capacidades en los nodos también. Existen otras 3 características de los nodos que merece la pena mencionar.

- **Supuesto de bienes homogéneos.** Se asume que cada unidad de flujo de cada servicio usa una unidad de capacidad en cada arco. Un modelo más general permitiría que la unidad de flujo de cada servicio k consumiese una cantidad de capacidad σ_{ij}^k asociada con cada arco (i, j) y sustituir la restricción de capacidad total (4.2) por una más general restricción de disponibilidad de recursos $\sum_{i \leq k \leq K} \sigma_{ij}^k x_{ij}^k \leq u_{ij}$.
- **Sin suposición de congestión.** Se tienen capacidades duras (fijas) en cada arco y, por tanto, el coste de cada arco es lineal respecto al flujo de cada arco.
- **Supuesto de bienes indivisibles.** El modelo (4.1)-(4.4) asume que las variables de flujo pueden ser fraccionales. En algunas aplicaciones prácticas, este supuesto es adecuado, en otros contextos, sin embargo, las variables deben ser valores enteros. Aún en estos casos el modelo puede ser útil dado que el modelo de programación lineal puede ser una buena aproximación del modelo de programación lineal entera, o se puede usar el modelo de programación lineal como una relajación del modelo de programación lineal entera y modificar el proceso de ramificación y corte o algún otro tipo de enfoque de enumeración.
  Una característica importante y distintiva de los problemas de flujo monoservicio y multiservicio es la integralidad de las soluciones. Los problemas de flujo monoservicio siempre tienen soluciones enteras cuando los datos de oferta, demanda y capacidad son a su vez enteros. Los problemas multiservicio no satisfacen esto.

El problema de flujo multiservicio se puede ver como un problema de asignación de capacidad. Todos los servicios compiten por la capacidad u_{ij} de cada arco (i, j) de la red. Cualquier solución óptima del problema de flujo multiservicio preasignará un flujo específico en cada arco (i, j) en cada servicio, este flujo se corresponde con la capacidad de ese servicio. Se pueden asignar estas capacidades a cada servicio y resolver el problema como múltiples problemas de flujo monoservicio con facilidad. Los métodos de asignación de recursos son útiles para implementar esto. Empiezan por asignar las capacidades a los servicios, luego utilizan la información obtenida en los

problemas monoservicio resultantes para reasignar las capacidades de una manera que mejore el coste general del sistema. (En la Sección 4.8 se muestra cómo resolver el problema de flujo multiservicio utilizando el método de dirección de recursos.)

Los métodos de particionamiento aprovechan el hecho de que el problema de flujo multiservicio es lineal con problemas de flujo de red integrados. Para resolver cualquier problema monoflujo, puede usarse el método simplex en redes (Capítulo 3), que funciona generando un árbol generador de soluciones. Las soluciones del árbol generador corresponden a soluciones factibles del problema de flujo de mínimo coste.

¿Se puede adoptar una metodología similar para resolver el problema de flujo multiservicio? ¿Se puede de alguna manera usar soluciones del árbol generador para las restricciones de capacidad $\mathcal{N}x^k = b^k$? El método de partición (sección 4.9) permite responder a ambas preguntas afirmativamente. Mantiene una base de programación lineal que se compone de bases (árboles generadores) de los problemas monoflujo, así como arcos adicionales que se requieren para "amarrar" estas soluciones y adaptarlas a las restricciones de capacidad.

4.2 Condiciones de optimalidad

Se asume que las variables de flujo x_{ij}^k no tiene cota superior, esto es, todo $u_{ij}^k = +\infty$ en la formulacion (4.2) y (4.4).

Puesto que el problema de flujo multiservicio es lineal, se puede usar las condiciones de optimización de la programación lineal para caracterizar las soluciones óptimas al problema.

La formulación de programación lineal del problema (4.1)-(4.4) tiene una restricción de capacidad para cada arista (i, j) de la red (4.2) y una restricción de balance de flujo para cada combinación nodo-servicio (4.3), el problema lineal dual tiene dos tipos de variables duales: el precio w_{ij} de cada arista (i, j) y un potencial de nodo $\pi^k(i)$ para la combinación del servicio k y el nodo i. Usando estas variables duales, se definen los costes reducidos $c_{ij}^{\pi,k}$ de cada arista (i, j) con respecto al servicio k:

$$c_{ij}^{\pi,k} = c_{ij}^k + w_{ij} - \pi^k(i) + \pi^k(j). \tag{4.6}$$

En notación matricial, se define como $c^{\pi,k} = c^k + w - \pi^k\mathcal{N}$.

Si se considera un servicio fijo k, este coste reducido es similar al coste reducido utilizado anteriormente (2.3) para el problema de flujo de mínimo coste (Capítulo 3) ; la diferencia es que ahora se suma el precio de la arista w_{ij} al coste c_{ij}^k. Nótese como las restricciones de capacidad (4.2) relacionan las variables x_{ij}^k, sin ellas serían independientes. Los costes de las aristas w_{ij} relacionan los costes reducidos por servicio y evitan que sean independientes.

Para caracterizar las soluciones óptimas del problema del flujo multiservicio, primero se escribe el problema dual del flujo multiservicio (4.1)-(4.4):

$$\text{Maximizar} \qquad -\sum_{(i,j)\in A} u_{ij} w_{ij} + \sum_{k=1}^{k} b^k \pi^k \tag{4.7}$$

$$\text{s. a} \quad c_{ij}^{\pi,k} = c_{ij}^k + w_{ij} - \pi^k(i) + \pi^k(j) \geq 0 \quad \text{para todo } (i,j) \in A \text{ y todo } k = 1, \ldots, K, \tag{4.8}$$

$$w_{ij} \geq 0 \quad \text{para todo } (i,j) \in A. \tag{4.9}$$

Las condiciones de optimalidad de la programación lineal, llamadas condiciones (óptimas) de holgura complementaria, establecen que una solución factible primal x y una solución factible dual (w, π_k) son óptimas para los problemas respectivos si y sólo si el servicio de cada variable primal (dual) y la holgura en la correspondiente restricción dual (primal) es cero. Las condiciones de holgura complementaria para el par primal-dual del problema de flujo multiservicio asumen la siguiente forma especial. (Aquí se usa y_{ij}^k para denotar un valor específico de la variable de flujo x_{ij}^k.)

Teorema 4.1 (Condiciones de holgura del problema de flujo multiservicio). *Los flujos de cada servicio y_{ij}^k son óptimos en el problema de flujo multiservicio* (4.1)-(4.4) *con cada $u_{ij}^k = +\infty$ si y sólo si son factibles para alguna elección de costes (no negativos) de aristas w_{ij} y potenciales de nodos (sin restricciones en el signo) $\pi^k(i)$, los costes reducidos y los flujos de las aristas satisfacen las siguientes condiciones de holgura:*

$$(a) \quad w_{ij}\left(\sum_{1\leq k\leq K} y_{ij}^k - u_{ij}\right) = 0 \quad \textit{para toda arista } (i,j) \in A. \tag{4.10}$$

$$(b) \quad c_{ij}^{\pi,k} \geq 0 \quad \textit{para toda arista } (i,j) \in A \textit{ y} \tag{4.11}$$

$$\textit{todos los servicios } k = 1, 2, \ldots, K. \tag{4.12}$$

$$(c) \quad c_{ij}^{\pi,k} y_{ij}^k = 0 \quad \textit{para toda arista } (i,j) \in A \textit{ y} \tag{4.13}$$

$$\textit{todos los servicios } k = 1, 2, \ldots, K. \tag{4.14}$$

Los costes de aristas óptimos y potenciales de nodos óptimos son cualquier conjunto de costes de aristas y potenciales de nodos que satisfacen las condiciones de holgura. El siguiente teorema muestra la conexión entre los problemas de flujo multiservicio y monoservicio.

Teorema 4.2 (Dualización parcial). *Sea y_{ij}^k los flujos óptimos y w_{ij} los costes óptimos de las aristas para el problema del flujo de multiservicio* (4.1)-(4.4). *Luego, para cada servicio k, las variables de flujo y_{ij}^k para $(i,j) \in A$ resuelven el siguiente problema de flujo de coste mínimo (no capacitado):*

$$\text{mín} \left\{ \sum_{(i,j)\in A} \left(c_{ij}^k + w_{ij} \right) x_{ij}^k \,:\, \mathcal{N} x^k = b, x_{ij}^k \geq 0 \text{ para todo } (i,j) \in A \right\}. \tag{17.3}$$

4.3 Relajación Lagrangiana

Para aplicar la relajación lagrangiana al problema de flujo multiservicio, se asocian multiplicadores w_{ij} lagrangianos no negativos a las restricciones (4.2), dando lugar al siguiente subproblema lagrangiano:

$$L(w) = \text{mín} \sum_{1\leq k\leq K} c^k x^k + \sum_{(i,j)\in A} w_{ij} \left(\sum_{1\leq k\leq K} x_{ij}^k - u_{ij} \right), \tag{4.15}$$

o, de forma equivalente,

$$L(w) = \text{mín} \sum_{1\leq k\leq K} \sum_{(i,j)\in A} \left(c_{ij}^k + w_{ij} \right) x_{ij}^k - \sum_{(i,j)\in A} w_{ij} u_{ij} \tag{4.16}$$

$$\text{s. a} \quad \mathcal{N} x^k = b^k \quad \text{para todo } k = 1, \ldots, K, \tag{4.17}$$

$$x_{ij}^k \geq 0 \quad \text{para todo } (i,j) \in A \text{ y todo } k = 1, 2, \ldots, K. \tag{4.18}$$

Nótese que dado el término $-\sum_{(i,j)\in A} w_{ij} u_{ij}$ en la función objetivo del subproblema lagrangiano es siempre una constante. Sean cuales fuere los multiplicadores lagrangianos, este término es constante y, por tanto, se puede ignorar. En la función objetivo resultante del subproblema lagrangiano cada variable x_{ij}^k tiene un coste $c_{ij}^k + w_{ij}$. Dado que en este problema ninguna de las restricciones contienen las variables de flujo de más de un servicio, el problema se puede descomponer en problemas de flujo de mínimo coste separados entre sí, uno para cada servicio. En consecuencia, aplicando el procedimiento de optimización de subgradiante de la Sección 2.2.6, se puede alternativamente:

1. Hallar el conjunto de flujos de mínimo coste (para valores fijos de los multiplicadores lagrangianos w) con los coeficientes de coste $c_{ij}^k + w_{ij}$.
2. Actualizar los multiplicadores con el algoritmo descrito en la Sección 2.2.6.

En este último caso, si y_{ij}^k denota la solución optima del subproblema de flujo de coste mínimo cuando los multiplicadores lagrangianos tienen el valor w_{ij}^q en la q-ésima iteración, actualizando la fórmula del subgradiante queda:

$$w_{ij}^{q+1} = \left[w_{ij}^q + \theta_q \left(\sum_{1 \le k \le K} y_{ij}^k - u_{ij} \right) \right]^+ . \tag{4.19}$$

La notación $[\alpha]^+$ denota la parte positiva de α osea, máx$(\alpha, 0)$. El escalar θ_q es el tamaño del paso que especifica como de lejos se mueve uno de la solución w_{ij}^q. Nótese que la formula actualizadora hace lo siguiente:

1. Si la solución y_{ij}^k del subproblema usa más de la capacidad u_{ij} de la arista, incrementa el multiplicador w_{ij}^q una cantidad $\left(\sum_{1 \le k \le K} y_{ij}^k - u_{ij} \right)$ en la arista (i, j).
2. Si la solución y_{ij}^k del subproblema usa menos de la capacidad disponible en la arista, reduce el multiplicador de lagrangiano una cantidad $\left(\sum_{1 \le k \le K} y_{ij}^k \right)$ en la arista (i, j).
3. Sin embargo, si este decrecimiento hace que el multiplicador w_{ij}^{q+1} se haga negativo, entonces se reduce el valor del multiplicador sólo a cero.

Se elige el tamaño del paso θ_q para las interaciones $q = 1, 2, \ldots, K$, de acuerdo con el procedimiento descrito en la Sección 2.2.6.

El método del subgradiante (Sección 2.2.6) para el problema de flujo multiservicio es un procedimiento para resolver el problema lineal (4.1)-(4.4) que es capaz de hacer uso de la relajación de las restricciones de balance de flujo. En el Teorema 2.2 se señaló que siempre que se aplica la relajación lagrangiana a cualquier problema lineal, como es el caso del problema de flujo multiservicio, el valor óptimo $L^* = \text{máx}_{w \ge 0} L(w)$ del problema dual lagrangiano es igual al valor óptimo de la función objetivo z^* del problema lineal.

El uso de la optimización subgradiante para resolver el problema de multiplicadores lagrangianos es atractivo por varias razones. Primero, como se acaba de señalar, este planteamiento permite explotar la estructura de la red de flujo. En segundo lugar, las

fórmulas para actualizar los multiplicadores lagrangianos w_{ij} computacionalmente son sencillas y muy fáciles de codificar en un programa de ordenador. Sin embargo, este tipo de solución también tiene algunas limitaciones. Para asegurar la convergencia, son necesarios pasos pequeños; como resultado, el método no converge muy rápido. En segundo lugar, el método es de base dual y, por lo tanto, aunque converge a las variables duales óptimas w_{ij}, las soluciones óptimas y_{ij}^k de los subproblemas no tienen por qué coincidir con la solución óptima del problema de flujo multiservicio. De hecho, aunque se ha demostrado que si se establecen los valores óptimos de los multiplicadores lagrangianos (es decir, las variables duales óptimas para las restricciones de capacidad en el problema lineal (4.1)-(4.4)), los flujos óptimos y_{ij}^k resuelven el subproblema lagrangiano (4.15)-(4.18). Estos subproblemas también pueden tener otras soluciones óptimas que no satisfacen las restricciones de capacidad.

Supóngase que se resuelve el subproblema de Lagrange para cualquier elección de los multiplicadores mediante el algoritmo simplex en redes (Capítulo 3). Al hacerlo, se obtiene una solución óptima de árbol generador para cada servicio, que corresponde a una base $\mathcal{B}^k$ de las restricciones (4.17) de capacidad de la red. Se puede extender esta base a una base general del problema lineal (4.1)-(4.4), como se muestra en la Figura 4.1.

La matriz de identidad en la fila superior de esta matriz corresponde a las variables de holgura s_{ij} en la igualdad (4.5) de las restricciones de capacidad. Obsérvese que en la solución correspondiente a la base $\mathcal{B}$:

- Cada variable de flujo x^k para $k = 1, 2, \ldots, K$ es igual a la solución y^k del subproblema lagrangiano.
- los valores de las variables de holgura vienen dados por $s_{ij} = u_{ij} - \sum_{1 \le k \le K} y_{ij}^k$.

Esta base es factible si todas las variables de holgura son no negativas. Si alguna de ellas es negativa la base es inviable. En cualquier caso, se puede aplicar métodos de programación lineal estándar (ya sea el método simplex primal o el método simplex dual) usando esta base como una solución inicial para resolver el problema.

Esta base proporciona una solución inicialmente atractiva para el método símplex porque permite explotar la estructura de red y los costes de flujo para generar una solución inicial ”inteligente”. Para problemas que están ”capacitados modestamente”, la base puede ser una muy buena aproximación de la solución óptima del problema y, por lo tanto, puede mejorar en gran medida el rendimiento del método simplex.

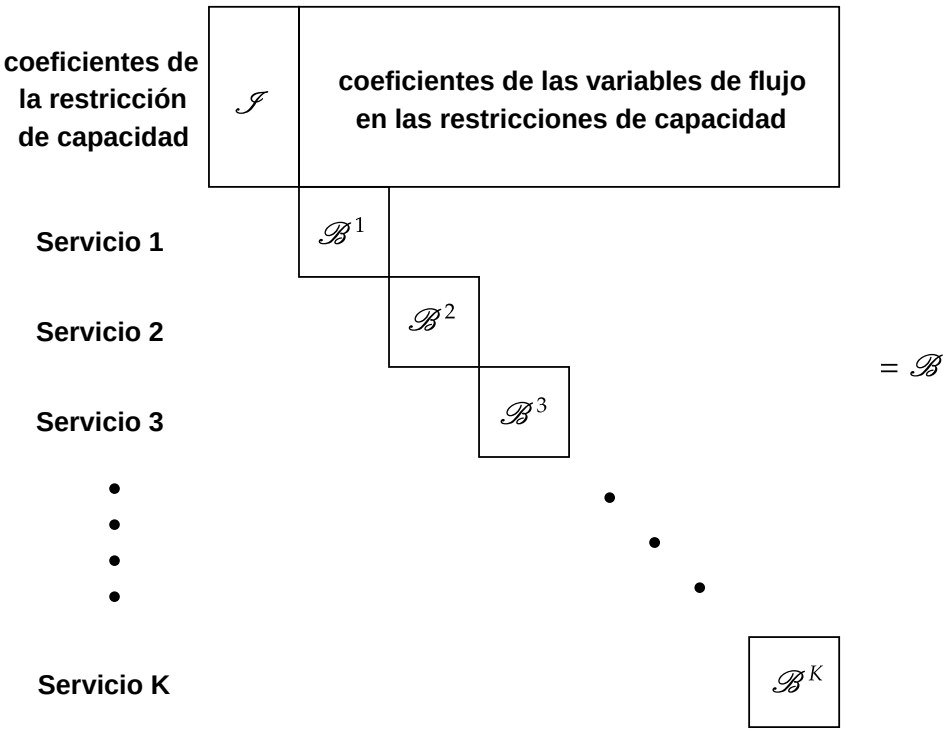

Figura 4.1: Base formado por las bases de los subproblemas (4.1)-(4.4) de cada servicio.

A continuación, se considera un planteamiento alternativo, conocido como descomposición de Dantzig-Wolfe, para resolver el problema dual de Lagrange. Este algoritmo requiere considerablemente más trabajo (la solución de un problema lineal) en cada iteración para actualizar los multiplicadores de Lagrange, pero ha demostrado converger más rápido que el procedimiento de optimización subgradiente para varias clases de problemas. En lugar de describir el procedimiento de descomposición de Dantzig-Wolfe como una variación de la relajación lagrangiana, primero se desarrolla desde un punto de vista de programación lineal a gran escala que proporciona una perspectiva algo diferente.

4.4 Generación de columnas

Considérese un caso especial del problema de flujo multiservicio: cada servicio k tiene un solo nodo fuente s^k y un solo nodo sumidero t^k y un flujo de d^k unidades. No se impone límite a la capacidad de flujo de los servicios individualmente salvo las restricciones de capacidad. Por lo tanto, para cada servicio k, las restricciones del subproblema $\mathcal{N}x^k = b^k$, $x^k \geq 0$ definen un problema de camino más corto. En este modelo, para cualquier elección w_{ij} de los multiplicadores de Lagrange, la relajación lagrangiana de las restricciones de capacidad requiere la solución de un problema de ruta más corta por cada servicio.

4.4.1 Reformulación de flujos en ruta

En la formulación de los problemas de flujo de red se puede definir el flujo en arcos o definir el flujo en caminos y en ciclos. Por "flujo en arco" se entiende un vector $x = \{x_{ij}\}$ que satisface las siguientes restricciones:

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = -e(i) \qquad \text{para todo } i \in N, \tag{4.20}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{para todo } (i,j) \in A. \tag{4.21}$$

donde $\sum_{i=1}^{n} e(i) = 0$. En este modelo se ha reemplazado la oferta/demanda $b(i)$ de cada nodo i por otro término, $-e(i)$. $e(i)$ es el desequilibrio del nodo y representa lo que entra menos lo que sale del nodo i. Si entra más de lo que sale $e(i) > 0$ se dice que i es un nodo en exceso. Si sale más de lo que entra $e(i) < 0$ se dice que i es un nodo en deficit. Si entra tanto como sale se dice que i es un nodo balanceado. Si $e = -b$ el flujo x es factible para el problema de flujo de mínimo coste.

En la formulación de flujo de arco las variables de decisión básicas son los flujos x_{ij} en los arcos $(i,j) \in A$. La formulación de flujos en camino o ciclo empieza con una enumeración de los caminos P entre cualquier par de nodos y la colección de todos los ciclos $\mathcal{W}$. La variable de decisión en la formulación del flujo en camino y del flujo en ciclo son $f(P)$ y $f(W)$. Se define estas variables para cada camino $p \in \mathcal{P}$ y cada ciclo $W \in \mathcal{W}$.

El flujo x_{ij} en el arco (i,j) es igual a la suma de los flujos $f(P)$ y $f(W)$ para todos los caminos P y ciclos W que contienen a este arco. Sea $\delta_{ij}(P) = 1$ si $(i,j) \in P$ y

$\delta_{ij}(P) = 0$ en caso contrario. Entonces

$$x_{ij} = \sum_{P \in \mathcal{P}} \delta_{ij}(P) f(P) + \sum_{W \in \mathcal{W}} \delta_{ij}(W) f(\mathcal{W}). \tag{4.22}$$

Por tanto, los flujos en caminos y los flujos en ciclos determinan de forma única los flujos en arcos.

Teorema 4.3 (Teorema de la descomposición de flujo). *Cada flujo en camino o en ciclo tiene una única representación como flujos no negativos en arcos. A su vez, con las siguientes dos propiedades cada flujo de arco no negativo x se puede representar como un flujo en camino y ciclo (no necesariamente de forma única):*

1. *Cada camino dirigido con flujo positivo conecta un nodo deficitario con un nodo en exceso.*
2. *Al menos $n + m$ caminos y ciclos tienen flujo no negativo; además, como mucho m ciclos tienen flujo no negativo.*

Sea x un flujo para el cual $e(i) = 0$ por cada $i \in N$. Cuando se aplica el algoritmo de descomposición de flujo a una circulación, cada iteración describe un ciclo dirigido que consta únicamente de arcos con flujo positivo y, posteriormente, reduce el flujo en al menos un arco a cero. En consecuencia, una circulación se descompone en flujos a lo largo de como máximo m ciclos dirigidos.

Propiedad 4.1. Una circulación x puede representarse como un flujo en ciclo a lo largo de la mayoría de los m ciclos dirigidos.

El Teorema 4.3 tiene varias consecuencias importantes. Por ejemplo, permite comparar dos soluciones de un problema de flujo de red de una manera conveniente y mostrar cómo se puede construir una solución a partir de otra mediante una secuencia de operaciones simples.

Para definir los ciclos de aumento es primero necesario introducir los arcos dirigidos hacia delante y los arcos dirigidos hacia atrás. Los arcos dirigidos hacia delante son aquellos que se corresponden con la dirección del flujo y los arcos dirigidos hacia atras aquellos contrarios a este, véase la Figura 4.2.

Un ciclo W (no necesariamente dirigido) de G es un ciclo de aumento con respecto al flujo x si al aumentar una cantidad positiva de flujo $f(W)$ por el ciclo, el flujo sigue siendo factible. Este crecimiento aumenta el flujo en los arcos dirigidos hacia adelante en el ciclo W y disminuye el flujo en los arcos dirigidos hacia atrás en el

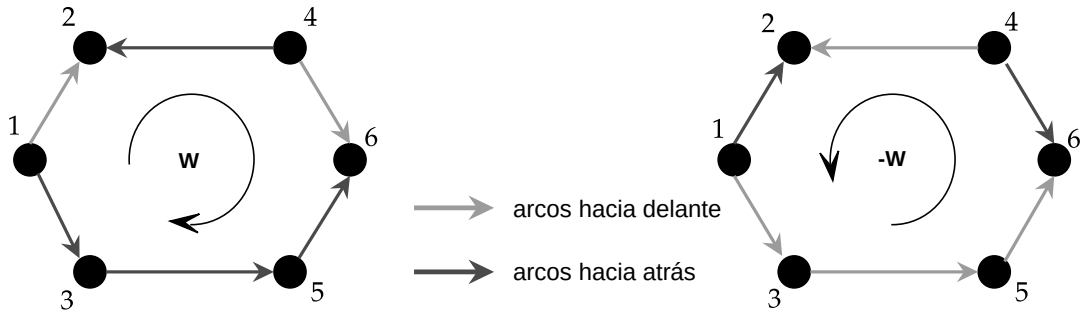

Figura 4.2: Ejemplos de arcos hacia delante y arcos hacia atrás en dos ciclos.

ciclo. Por lo tanto, un ciclo W es un ciclo de aumento de G si $x_{ij} \leq u_{ij}$ para cada arco dirigido hacia adelante (i,j) y $x_{ij} \geq 0$ para cada arco dirigido hacia atrás (i,j). A continuación, se amplía la notación de $\delta_{ij}(W)$ para ciclos que no están necesariamente dirigidos. Se define $\delta_{ij}(W) = 1$ si (i,j) es un arco dirigido hacia adelante en el ciclo W, $\delta_{ij}(W) = -1$ si (i,j) es un arco dirigido hacia atrás en el ciclo W, e igual a 0 en caso contrario.

En términos de redes residuales, cada ciclo de aumento W con respecto a un flujo x corresponde a un ciclo dirigido W en $G(x)$, y viceversa. Se define el coste de un ciclo de aumento W como $c(W) = \sum_{(i,j)\in W} c_{ij}\delta_{ij}(W)$. El coste de un ciclo de aumento representa el cambio en el coste de una solución factible si se aumenta en 1 unidad el flujo a lo largo del ciclo. El cambio en el coste de flujo para aumentar $f(W)$ unidades a lo largo del ciclo W es $c(W)f(W)$.

Supóngase que x y x^0 son dos soluciones factibles cualesquiera del problema de flujo con mínimo coste. Alguna circulación factible x^1 en G^0 fatisface la propiedad $x = x^0 + x^1$. La Propiedad 4.1 supone que se puede representar la circulación x^1 como flujos en ciclo $f(W_1), f(W_2), \ldots, f(W_r)$, con $r \leq m$. Nótese que cada uno de los ciclos $W_1, W_2, \ldots, W_r$ es un ciclo de aumento en $G(x^0)$. Además,

$$\sum_{(i,j)\in A} c_{ij}x_{ij} = \sum_{(i,j)\in A} c_{ij}x^0_{ij} + \sum_{(i,j)\in G(x^0} c_{ij}x^1_{ij} \tag{4.23}$$

$$= \sum_{(i,j)\in A} c_{ij}x^0_{ij} + \sum_{(i,j)\in G(x^0} c_{ij}\left[\sum_{k=1}^{r} \delta_{ij}(W_k)f(W_k)\right] \tag{4.24}$$

$$= \sum_{(i,j)\in A} c_{ij}x^0_{ij} + \sum_{k=1}^{r} c(W_k)f(W_k). \tag{4.25}$$

Teorema 4.4 (Teorema del ciclo de aumento). *Sean x y x^0 dos soluciones posi-*

bles de un problema de flujo en red. Entonces x es igual a x^0 más el flujo en como máximo m ciclos dirigidos de $G(x^0)$. Además, el coste de x es igual al coste de x^0 más el coste del flujo en estos ciclos de aumento.

Teorema 4.5 (Teorema de Optimidad de Ciclo Negativo). *Una solución factible x^* del problema de flujo de mínimo coste es una solución óptima si y sólo si la red residual $G(x^*)$ no contiene un ciclo dirigido de coste negativo.*

Se reformula el problema de flujo multiservicio utilizando flujos de trayectoria y flujos de ciclo en lugar de flujos de arco. Para simplificar más, supóngase que para cada servicio el coste de cada ciclo W en la red subyacente es no negativo. El problema satisface esta condición, por ejemplo, si los costes de flujo en cada arco son no negativos. Si se impone esta condición de coste de ciclo no negativo, entonces en alguna de las soluciones óptimas del problema, el flujo en cada ciclo es cero, por lo que se puede eliminar las variables de flujo de los ciclos. Así se podrá representar cualquier solución potencialmente óptima como la suma de los flujos en las rutas dirigidas.

Para cada servicio k, sea P^k la colección de todas las rutas dirigidas desde la fuente s^k hasta el sumidero t^k en la red $G = (N, A)$. En la formulación de flujo de ruta, cada variable de decisión $f(P)$ es el flujo en alguna ruta P para el k-ésimo servicio, se define esta variable para cada ruta P en P^k.

Sea $\delta_{ij}(P)$ una variable indicadora. Indica la trayectoria en arco, es decir, $\delta_{ij}(P)$ es igual a 1 si la trayectoria P contiene al arc(i, j), y es 0 en caso contrario. El Teorema 4.3 establece que un flujo en arco x_{ij}^k óptimo siempre se puede descomponer en flujos de trayectoria $f(P)$ de la siguiente manera:

$$f(P) = x_{ij}^k \sum_{P \in P^k} \delta_{ij}(P) f(P). \tag{4.26}$$

Sea $c^k(P) = \sum_{(i,j)\in A} c_{ij}^k \delta_{ij}(P) = \sum_{(i,j)\in P} c_{ij}^k$ que denota el coste de la unidad de flujo en la trayectoria $P \in P^K$ con respecto al servicio k. Para cada servicio k, si se sustituyen las variables de flujo del arco en la función objetivo, se intercambia el orden de las sumas y se reordenan términos, se tiene que:

$$\sum_{(i,j)\in A} c_{ij}^k x_{ij}^k = \sum_{(i,j)\in A} c_{ij}^k \left[\sum_{P \in P^k} \delta_{ij}(P) f(P) \right] = \sum_{P \in P^k} c^k(P) f(P). \tag{4.27}$$

Esto muestra que se puede expresar el coste de cualquier solución como el coste de los flujos de arco o el coste de los flujos de ruta.

Al sustituir las variables de ruta en la formulación de flujo multiservicio, se obtiene la siguiente formulación de flujo de ruta:

$$\text{Minimizar} \quad \sum_{1\le k\le K} \sum_{P\in P^k} c^k(P) f(P) \tag{4.28}$$

$$\text{s. a} \quad \sum_{1\le k\le K} \sum_{P\in P^k} \delta_{ij}(P) f(P) \le u_{ij} \quad \text{para todo } (i,j)\in A, \tag{4.29}$$

$$\sum_{P\in P^k} f(P) = d^k \quad \text{para todo } k = 1,2,\dots,K, \tag{4.30}$$

$$f(P) \le 0 \quad \text{para todo } k = 1,2,\dots,K \text{ y todo } P \in P^k. \tag{4.31}$$

Al formular este problema, se ha usado Teorema 4.3 que establece que se puede descomponer cualquier arco de flujo factible del sistema $\mathcal{N}x^k = b^k$ en un conjunto de flujos de trayectoria y flujos de ciclo de tal manera que los trayectos satisfacen la condición de balance de flujo (4.30).

La formulación de flujo de ruta del problema de flujo multiservicio tiene una estructura de restricciones muy simple. El problema tiene una única restricción (4.29) para cada arco (i,j) que establece que la suma de los flujos de la trayectoria que atraviesan el arco es como máximo u_{ij} (la capacidad del arco). Además, el problema tiene una única restricción (4.30) para cada servicio k que establece que el flujo total en todas las rutas que conectan la fuente s^k y el sumidero t^k del servicio k debe ser igual a la demanda d^k de este servicio. Para una red con n nodos, m arcos y K servicios, la formulación de flujo de ruta contiene $m+K$ restricciones (además de las restricciones de no negatividad impuestas a los valores de flujo de ruta (4.31)). Por el contrario, la formulación en arco (4.28) contiene $m + nK$ restricciones puesto que contiene una restricción de balance de flujo para cada combinación de nodo y servicio. Cómo sólo aparece una ruta en una de las restricciones (4.30), se puede aplicar una versión especifica del método simplex, conocido como método simplex de cota superior generalizada, para resolver la formulación de flujo de ruta de manera más eficiente. Este método esencialmente resuelve el problema como si sólo tuviera m restricciones de capacidad.

Esta disminución en el número de restricciones tiene su precio, la formulación de flujo de ruta tiene una variable para cada ruta que, por servicio, conecta una fuente y un sumidero. El número de variables suele ser enorme, creciendo exponencialmente con el tamaño de la red. Pero se podría esperar que sólo muy pocos de los caminos recorran la solución óptima al problema. De hecho, se sabe que, como máximo, las $K + m$ trayectorias llevan un flujo positivo en alguna solución óptima al problema. Si se conociera el conjunto óptimo de caminos, o un muy buen conjunto de caminos, se

podría obtener una solución óptima (es decir, valores para los flujos de camino) resolviendo un problema lineal que contenga sólo los servicios con dos o más conjuntos de caminos.

4.4.2 Condiciones de optimalidad

El método de programación lineal simplex revisado mantiene una base en cada paso, y el uso de esta base determina un vector de multiplicadores simplex para las restricciones. Dado que la fórmula de flujo de trayectoria (4.28)-(4.31) contiene una restricción de capacidad por cada arco (4.29) y una restricción de demanda (4.30) para cada servicio, el problema lineal dual tiene una variable dual w_{ij} para cada arco (este es el mismo coste por arco que se ha introducido antes) y otra variable dual σ^k para cada servicio $k = 1, 2, \ldots, K$. Con respecto a estas variables duales, el coste reducido $c_P^{\sigma,w}$ para cada variable de flujo de ruta $f(P)$ es:

$$c_P^{\sigma,w} = c^k(P) + \sum_{(i,j)\in P} w_{ij} - \sigma^k. \tag{4.32}$$

Y las condiciones de holgura complementaria (4.10)-(4.14) para la formulación del arco del problema original asumen la siguiente forma:

Condiciones de holgura complementaria del flujo en camino. Los flujos como trayectoria de los servicios $f(P)$ son óptimos en la formulación del flujo de la trayectoria (4.28)-(4.31) del problema de flujo multiservicio si y sólo si para algunos precios de arco w_{ij} y precios de servicios σ^k, los costes reducidos y los flujos de arco satisfacen las siguientes condiciones de holgura complementarias:

$$\text{(a)} \qquad w_{ij}\left[\sum_{1\le k\le K}\sum_{P\in P^k}\delta(P)f(P) - u_{ij}\right] = 0 \text{ para todo } (i,j)\in A. \tag{4.33}$$

$$\text{(b)} \qquad c_P^{\sigma,w} \ge 0 \text{ para todo } k = 1, 2, \ldots, K \text{ y todo } P \in P^k. \tag{4.34}$$

$$\text{(c)} \qquad c_P^{\sigma,w} f(P) = 0 \text{ para todo } k = 1, 2, \ldots, K \text{ y todo } P \in P^k. \tag{4.35}$$

Estas condiciones de optimalidad tienen una interpretación muy atractiva e intuitiva. La condición (4.33) establece que el precio w_{ij} del arco (i, j) es cero si la solución óptima $f(P)$ no usa toda la capacidad u_{ij} del arco. Es decir, si la solución óptima no usa completamente la capacidad de ese arco, se podría ignorar la restricción (no ponerle un coste).

Dado que el coste $c^k(P)$ del camino P es sólo la suma del coste de los arcos contenidos en ese camino, es decir, $c^k(P) = \sum_{(i,j)\in P} c^k_{ij}$, se puede escribir el coste reducido del camino P como:

$$c^{\sigma,w}_P = \sum_{(i,j)\in P} (c^k_{ij} + w_{ij}) - \sigma^k. \tag{4.36}$$

Es decir, el coste reducido de la ruta P es sólo el coste de esa ruta con respecto a los costes modificados $c^k_{ij} + w_{ij}$ menos el coste del servicio σ. La condición de holgura complementaria (4.34) establece que el coste de la ruta modificada $\sum_{(i,j)\in P}(c^k_{ij} + w_{ij})$ para cada ruta que conecta el nodo fuente s_k y el nodo sumidero t_k del servicio k debe ser al menos tan grande como el coste σ^k del servicio. La condición (4.35) implica que el coste reducido $c^{p,w}_P$ debe ser cero para cualquier ruta P que lleve flujo en la solución óptima (para la cual el flujo $f(P)$ es positivo); es decir, el coste modificado $\sum_{(i,j)\in P}(c^k_{ij} + w_{ij})$ de esta ruta debe ser igual al coste σ^k del servicio. Por tanto, las condiciones (4.34) y (4.35) implican que

σ^k es la distancia de ruta más corta desde el nodo s^k al nodo t^k con respecto a los costes modificados $c^k_{ij}+w_{ij}$ y en la solución óptima cada ruta que va desde el nodo s^k al nodo t^k y que lleva un flujo positivo debe ser la ruta más corta con respecto a los costes modificados.

Este resultado muestra que los costes del arco w_{ij} permiten descomponer el problema de flujo multiservicio en un conjunto de problemas de ruta más corta de costes "modificados" independientes.

4.5 Procedimiento de solución de generación de columnas

Hasta este punto, se ha reformulado el problema del flujo multiservicio como un problema lineal a gran escala con una enorme cantidad de columnas y con una variable de flujo por cada ruta que conecta la fuente y el sumidero de un servicio. También se ha mostrado cómo caracterizar cualquier solución óptima de esta formulación en términos de las variables duales de programación lineal w_{ij} y σ^k, interpretando estas condiciones como condiciones de camino más corto con respecto a los costes del arco modificado $c^k_{ij} + w_{ij}$. A continuación, se muestra cómo resolver el problema mediante un procedimiento de solución conocido como generación de columnas.

La idea clave en la generación de columnas nunca es enumerar explícitamente todas

las columnas de la formulación del problema, sino generarlas sólo "según sea necesario". El método simplex revisado de programación lineal, que mantiene una base $\mathcal{B}$ en cada iteracción, es perfectamente adecuado para llevar a cabo esta estrategia algorítmica. Utiliza esta base para definir un conjunto de multiplicadores simplex π mediante el cálculo matricial $\pi\mathcal{B} = c_B$ (en esta aplicación, los multiplicadores son w y σ). Es decir, el método define los multiplicadores simplex de modo que los costes reducidos c_B^π de las variables básicas sean cero; es decir, $c_B^\pi = c_\mathcal{B} - \pi\mathcal{B}$. Para encontrar los multiplicadores del simplex, el método no requiere información sobre las columnas (variables) que no estén en la base. Luego usa los multiplicadores para calcular el precio de las columnas no básicas, es decir, calcular sus costes reducidos. Si algún coste reducido es negativo (asumiendo una formulación de minimización), el método introducirá una variable no básica en la base en lugar de una de las variables básicas actuales, volverá a calcular los multiplicadores simplex π y luego repetirá estos cálculos. Para usar el método de generación de columnas, las columnas deben tener propiedades estructurales que permitan realizar las operaciones de tasación sin examinar explícitamente cada columna.

Cuando se aplica a la formulación de flujo de ruta del problema de flujo multiservicio, en cualquier paso con respecto a la base actual (que se compone de un conjunto de columnas, o variables de ruta, para el problema), el método simplex revisado define los multiplicadores simplex w_{ij} y σ^k para que el coste reducido de cada variable en la base sea cero. Por lo tanto, si una ruta P que conecta la fuente s^k y el sumidero t^k para el servicio k es una de las variables básicas, entonces $c_P^{\sigma,w} = 0$, o equivalentemente, $\sum_{(i,j)\in P}(c_{ij}^k + w_{ij}) = \sigma^k$. Por lo tanto, el método simplex revisado determina los multiplicadores simplex w_{ij} y σ^k para que satisfagan las siguientes ecuaciones:

$$\sum_{(i,j)\in P}\left(c_{ij}^k + w_{ij}\right) = \sigma^k \text{ por cada camino } P \text{ en la base.}$$

Obsérvese que, dado que cada base consta de $K+m$ trayectos, cada base da lugar a $K+m$ de estas ecuaciones. Además, las ecuaciones contienen $K+m$ incognitas (es decir, m costes de arco w_{ij} y K distancias de trayectoria más cortas σ^K). El método simplex revisado usa cálculos matriciales para resolver las ecuaciones $K+m$ y determina los valores únicos de los multiplicadores del simplex.

La condición de holgura complementaria (4.35) dicta que $c_P^{\sigma,w} f(P) = 0$ para cada camino P en la red. Dado que cada camino P en la base satisface la condición $c_P^{\sigma,w} = 0$ puede enviarse cualquier cantidad de flujo en él y aún así satisfacer la condición (4.35). Para satisfacer esta condición para una trayectoria P que no está en la base, se establece $f(P) = 0$. A continuación, considérese la condición de holgura complementaria

(4.33). Si la variable de holgura $s_{ij} = \left[\sum_{1\le k\le K}\sum_{(i,j)\in P^k}\delta_{ij}(P)f(P) - u_{ij}\right]$ no está en la base, $s_{ij} = 0$, entonces la solución satisface (4.33). Por otro lado, si la variable de holgura s_{ij} está en la base, su coste reducido, que es igual a $0 - w_{ij}$, es cero, lo que supone que $w_{ij} = 0$ y la solución satisface (4.33). Por lo tanto, se ha demostrado que la solución definida por la base actual satisface las condiciones (4.33) y (4.35); es óptima si satisface la condición (4.34) (el coste reducido de cada variable de flujo de ruta es no negativo). ¿Cómo se podría verificar esta condición? Es decir, comprobar que para cada servicio k,

$$c_P^{\sigma,w} = \sum_{(i,j)\in P}\left(c_{ij}^k + w_{ij}\right) - \sigma^k \ge 0 \quad \text{para cada } P \in P^k, \tag{4.37}$$

o equivalentemente

$$\min_{P\in P^k}\sum_{(i,j)\in P}(c_{ij}^k + w_{ij}) \ge \sigma^k. \tag{4.38}$$

Como se ha señalado, el lado izquierdo de esta desigualdad es sólo la longitud del camino más corto que conecta los nodos fuente y sumidero, s_k y t_k, del servicio k con respecto a los costes modificados $c_{ij}^k + w_{ij}$. Por lo tanto, para ver si los precios del arco junto con las distancias de la trayectoria actual σ^k satisfacen las condiciones de holgura complementarias, se resuelve un problema del camino más corto para cada servicio k. Si para todas los servicios k, la longitud de la trayectoria más corta para ese servicio es al menos tan grande como σ^k, se satisface la condición de holgura complementaria (4.34).

De lo contrario, si para algún servicio k, Q denota la ruta más corta con respecto a los costes modificados actuales $c_{ij}^k + w_{ij}$ y el coste reducido de la ruta Q es menor que la longitud σ^k de la ruta de coste mínimo desde el conjunto P^k, entonces:

$$c_Q^{\sigma,w} = \sum_{(i,j)\in Q}\left(c_{ij}^k + w_{ij}\right) - \sigma^k < 0. \tag{4.39}$$

En términos del problema lineal (4.28)-(4.31), la ruta Q tiene un coste reducido negativo, por lo que se puede usar de manera provechosa en el problema lineal en lugar de una de las rutas P en la base actual $\mathcal{B}$. Es decir, utilizando los pasos habituales del método simplex, se realiza un cambio de base introduciendo la ruta Q en la base actual. Hacerlo permite determinar un nuevo conjunto de precios de arco w_{ij} y una nueva distancia de trayectoria más corta modificada σ^k entre los nodos fuente y sumidero del servicio k. Se eligen los valores de estas variables para que el coste reducido de cada variable básica sea cero. Es decir, usando operaciones matriciales, se resuelve

una vez más el sistema $c_P^{\sigma,w} = \sum_{(i,j)\in P}(c_{ij}^k + w_{ij} - \sigma^k) = 0$ en las variables w_{ij} y σ^k. Entonces, como antes, se resolve un problema de ruta más corta para cada servicio k y se vería si alguna ruta tiene una longitud más corta que σ^k. Si es así, se introduce esta ruta en la base y se continua alternativamente

1. encontrando nuevos valores para los costes del arco w_{ij} y para las longitudes de las rutas σ^k, y
2. resolviendo problemas de la ruta más corta.

Esta discusión muestra cómo se determinaría la variable a introducir en la base en cada paso. El resto de los pasos para implementar el método símplex (por ejemplo, determinar la variable a eliminar de la base en cada paso) son los mismos que los de la implementación habitual, por lo que no se especifican más detalles.

4.5.1 Determinación de cotas inferiores

Sea z^* el valor óptimo de la función objetivo del problema de flujo multiservicio (4.28)-(4.31) y sea z^{lp} el valor óptimo de la función objetivo en cualquier paso de la resolución de la formulación del flujo de trayectoria del problema (4.28)-(4.31) mediante el método simplex. Dado que z^{lp} corresponde a una solución factible al problema, $z^* \leq z^{lp}$. Como se ha visto en la discusión de la técnica de relajación lagrangiana en la Sección 4.3, para cualquier elección de los costes del arco w, el valor óptimo $L(w)$ del subproblema lagrangiano es una cota inferior en z^*. Por lo tanto, supóngase que en cualquier momento durante el transcurso del algoritmo, se resuelve el subproblema lagrangiano con respecto a los precios actuales del arco w_{ij}. Es decir, se resuelven las trayectorias más cortas $l^k(W)$ para todos los servicios k con respecto a los costes modificados $c_{ij}^k + w_{ij}$. (Obsérvese que este es el mismo cálculo que se realiza al tasar las columnas para el método simplex). Luego, a partir de (4.16)-(4.18) el valor $L(w)$ del subproblema de Lagrange es:

$$L(w) = \sum_{k=1}^{K} l^k(w) - \sum_{(i,j)\in A} w_{ij}u_{ij}, \tag{4.40}$$

y por la teoría de la relajación lagrangiana,

$$L(w) \leq z^* \leq z^{lp}. \tag{4.41}$$

Luego, al implementar el procedimiento de generación de columnas tasando las columnas y encontrando las distancias de las trayectorias más cortas $l^k(w)$ se obtiene como subproducto una cota inferior en el valor de la función objetivo. Esta cota inferior permite juzgar la calidad de la solución actual en la técnica de generación de columnas y, a menudo, finalizar el procedimiento sin más cálculos si la diferencia entre el valor de la solución z^{lp} y la cota inferior $L(w)$ es suficientemente pequeña. Nótese que dado que en cada paso del método simplex, el valor objetivo z^{lp} del problema permanece igual o disminuye, la cota superior no aumenta monótonamente de un paso a otro. Por otro lado, el valor objetivo $L(w)$ del subproblema lagrangiano no necesita disminuir de un paso a otro, por lo que en cualquier punto del algoritmo se usaría el mayor de los valores $L(w)$ generados en todos los pasos anteriores como la mejor cota inferior.

4.6 Descomposición de Dantzig-Wolfe

Se tienen K pricers y un master que están resolviendo conjuntamente el problema de flujo multiservicio. El master resuelve la formulación de camino (4.28)-(4.31) del problema, al cual se le llama problema maestro. El master trabaja sólo con un subconjunto de las columnas, a este problema se le llama problema maestro restringido. La formulación de camino del problema de flujo multiservicio tiene $m + k$ restricciones:

1. Una por cada servicio k especificando que el servicio tiene un flujo d^k.
2. Una por cada arco (i, j) especificando que el flujo total que pasa por ese arco es como mucho u_{ij}.

El master resuelve el problema maestro restringido y necesita averiguar si la solución es óptima para el problema original o si hay otras columnas con costes reducidos negativos. Para esto el master calcula el conjunto óptimo de multiplicadores simplex (costes) del problema maestro restringido. Esto es, encuentra los precios de los arcos w_{ij} y la longitud del camino σ^k.

Una vez el master ha establecido los costes, el pricer del servicio k determina el camino más barato sabiendo que cada arco tiene un peaje w_{ij} además del coste c^k_{ij}. Si el coste de este camino más corto es menor que σ^k, el pricer transmite esta solución al master

como una mejora de la solución. El pricer resuelve el subproblema:

$$\text{mín} \quad \sum_{1\leq k\leq K} \sum_{P\in P^K} \left[c^k(P) + \sum_{(i,j)\in P} w_{ij} \right] \tag{4.42}$$

$$\text{s. a} \quad \sum_{P\in P^k} f(P) = d^k \text{ para todo } k = 1, 2, \dots, K, \tag{4.43}$$

$$f(P) \geq 0 \text{ para todo } k = 1, 2, \dots, K \text{ y todo } P \in P^K. \tag{4.44}$$

donde $c^k(P) = \sum_{(i,j)\in P} c^k_{ij}$. Así cada pricer resuelve un problema de camino más corto con los pesos facilitados por el master y reportan si encuentran un camino más corto para su correspondiente servicio que el que está ya usando el master.

El algoritmo mantiene una base de programación lineal e introduce una columna en cada iteracción. Descarta cualquier columna que abandona la base pero dado que generar columnas es un proceso costoso y estas columnas pueden tener un valor negativo en futuras iteraciones, se almacenan. Así, el algoritmo, en general, realiza más de un cambio en la base, resuelve el problema maestro restringido, halla los nuevos costes y se resuelven de nuevo los subproblemas de camino más corto.

Cada subproblema de camino más corto se puede resolver de forma paralela. El algoritmo es finito, cada columna del problema maestro corresponde a un camino entre la cantidad finita de posibles caminos. Eventualmente el algoritmo generará todas las columnas del problema maestro.

Los K subproblemas corresponden a la relajación lagrangiana con multiplicadores w_{ij} en cada arco (i, j). El master fija los multiplicadores y resuelve el problema de multiplicadores de Lagrange. La descomposición de Dantzig-Wolfe es un método eficiente para resolver la relajación lagrangiana si se mide la eficiencia mediante el número de iteraciones de un algoritmo. El algoritmo de descomposición de Dantzig-Wolfe siempre mantiene una solución factible del problema. Como se vio en la Sección 4.5 la solución del subproblema aporta una cota de cómo de alejada está la solución factible de la óptima. Se puede parar el algoritmo en cualquier iteración y obtener no sólo una solución factible sino una medida de cuan alejado está la solución factible de la solución óptima.

4.7 Descomposición de la directiva de recursos

El método de la directiva de recursos, en lugar de utilizar costes para descomponer el problema, asigna la capacidad de los arcos a los servicios. Cuando se aplica a la formulación del problema (17.1), asigna $r_{ij}^k \leq u_{ij}^k$ unidades de la capacidad u_{ij} del arco (i, j) al servicio k, produciendo el siguiente problema de directiva de recursos:

$$z = \text{mín} \sum_{1 \leq k \leq K} c^k x^k \tag{4.45}$$

$$\sum_{1 \leq k \leq K} r_{ij}^k \leq u_{ij} \text{ para todo } (i, j) \in A, \tag{4.46}$$

$$\mathcal{N} x^k = b^k \text{ para } k = 1, 2, \ldots, K, \tag{4.47}$$

$$0 \leq x_{ij}^k \leq r_{ij}^k \text{ para todo } (i, j) \in A \text{ y todo } k = 1, 2, \ldots, K. \tag{4.48}$$

La restricción (4.46) asegura que la asignación total de recursos para el arco (i, j) no exceda la capacidad de ese arco. Sea $r = (r_{ij}^k)$ el vector de asignaciones de recursos.

Propiedad 4.2. El problema de la directiva de recursos (4.45)-(4.48) es equivalente al problema original del flujo multiservicio (4.1)-(4.4) en el sentido de que

1. si (x, r) es factible en el problema de la directiva de recursos, entonces x es factible y la función tiene el mismo valor objetivo que en el problema original, y
2. si x es factible en el problema original y $r = x$, entonces (x, r) es factible y la función objetivo tiene el mismo valor que en el problema de directiva de recursos.

Véase el siguiente método para resolver el problema de la directiva de recursos (4.45)-(4.48). En lugar de resolver el problema eligiendo los vectores r y x simultáneamente, se eligen secuencialmente. Primero se fijan las asignaciones de recursos r_{ij}^k y luego se elige el flujo x_{ij}^k. Sea $z(r)$ el valor óptimo del problema de la directiva de recursos para un valor fijo de la asignación de recursos r y considérese el siguiente problema derivado de asignación de recursos:

$$\text{Minimizar} \quad z(r) \tag{4.49}$$

$$\text{s. a} \quad \sum_{1 \leq k \leq K} r_{ij}^k \leq u_{ij} \quad \text{para todo } (i, j) \in A, \tag{4.50}$$

$$0 \leq r_{ij}^k \leq u_{ij}^k \quad \text{para todo } (i, j) \in A \text{ y todo } k = 1, 2, \ldots, K. \tag{4.51}$$

La función objetivo $z(r)$ para este problema es complicada. Se conoce su valor sólo implícitamente como la solución de un problema de optimización en las variables de flujo x_{ij}^k. Además, para cualquier valor fijo de las variables de recursos r_{ij}^k, el problema de la directiva de recursos se descompone en un subproblema de flujo de red separado para cada servicio. Es decir, $z(r) = \sum_{k \in K} z^k(r^k)$ con el valor $z^k(r^k)$ del k-ésimo subproblema dado por

$$z^k(r^k) = \text{mín} \sum_{1 \le k \le K} c^k x^k \tag{4.52}$$

$$\text{s. a} \quad \mathcal{N} x^k = b^k \quad \text{para todo } k = 1, 2, \dots, K. \tag{4.53}$$

$$0 \le x_{ij}^k \le r_{ij}^k \quad \text{para todo } (i, j) \in A \text{ y para todo } k = 1, 2, \dots, K. \tag{4.54}$$

Propiedad 4.3. El problema de la directiva de recursos (4.45)-(4.48) es equivalente al problema de asignación de recursos (4.49)-(4.51) en el sentido de que

1. si (x, r) es factible en el problema de la directiva de recursos, entonces r es factible en el problema de la asignación de recursos y $z(r) \le cx$, y
2. si r es factible en el problema de asignación de recursos, entonces para algún vector x, (x, r) es factible en el problema original y $cx = z(r)$.

Las propiedades (4.45) y (4.46) implican que en lugar de resolver el problema del flujo multiservicio directamente, se puede descomponer en un problema de asignación de recursos con una estructura de restricciones muy simple con una única restricción de desigualdad pero con una función objetivo complicada $z(r)$. Para encontrar $z(r)$ para cualquier elección del vector de asignación de recursos r, se necesita resolver K problemas de flujo monoservicio.

Otra forma de ver la función objetivo $z(r)$ es como el coste del problema lineal (4.45)-(4.48) en función de los parámetros r de la parte derecha. Es decir, cualquier valor r para el vector de asignación define los valores de los parámetros del lado derecho para este problema lineal.

Propiedad 4.4. Sea R el conjunto de asignaciones para el cual el problema lineal minimiza $\{cx : \mathcal{A}x = b, 0 \le x \le r\}$. Sea $z(r)$ el valor de este problema lineal en función del parámetro r del lado derecho. En el conjunto R, la función objetivo $z(r)$ es una función convexa, lineal a trozos de r.

4.8 Resolver los modelos de directiva de recursos

Dado que la función $z(r)$ no es diferenciable (porque es lineal a trozos), no se puede usar métodos de gradiente para resolver el problema de asignación de recursos. En cambio, se podría utilizar otros métodos. Por ejemplo, podría buscarse una mejora local en $z(r)$ utilizando un método heurístico. Podría usarse un "método de arco por vez" sumando 1 a $r_{pq}^{k'}$ y restando 1 de $r_{pq}^{k''}$ para un arco (p, q) en dos servicios k' y k'', eligiendo el arco y los servicios en cada paso en base a algún criterio (por ejemplo, las opciones que den la mayor disminución en el valor de la función objetivo en cada paso). Este método es fácil de implementar pero no asegura la convergencia a una solución óptima. Puede visualizarse como cambiar la asignación de recursos en cada paso utilizando la fórmula:

$$r \leftarrow r + \theta\gamma, \tag{4.55}$$

con una longitud de paso de $\theta = 1$ y una dirección de movimiento $\gamma = (\gamma_{ij}^{k})$ dada por $\gamma_{pq}^{k'} = 1$, $\gamma_{pq}^{k''} = -1$, y $\gamma_{pq}^{k} = 0$ para todas las demás combinaciones de arco-servicio. Sin embargo, tomando ideas prestadas de la optimización subgradiente, se podría usar un planteamiento de optimización eligiendo la dirección del movimiento γ como un subgradiente correspondiente a la asignación de recursos r (ver Figura 4.3). Un método sería buscar una dirección de movimiento γ, y una longitud de paso θ que simultáneamente mantengan la viabilidad y aseguren la convergencia hacia una solución óptima r del problema de asignación de recursos (4.49)-(4.51). Se adoptará un método de dos pasos.

- Primero, se determinará una dirección de subgradiente y una longitud de paso que asegure la convergencia, siempre que no se imponga restricciones a las asignaciones.
- Si moverse a $r + \theta\gamma$ da una solución inviable, se transforma r en un punto r' que mantenga la factibilidad y asegure la convergencia.

Para aplicar esta metodología, se debe poder responder dos preguntas básicas:

1. ¿Cómo se puede encontrar un subgradiante γ de $z(r)$ en un punto r dado?
2. ¿Cómo se podría transformar una asignación de recursos no factible $r + \theta\gamma$ de modo que sea factible para el problema de elección de recursos, es decir, satisfaga las restricciones $\sum_{1 \le k \le K} r_{ij}^{k} \le u_{ij}$ para todo $(i, j) \in A$, y $0 \le r_{ij}^{k} \le u_{ij}^{k}$ para todo $(i, j) \in A$ y todo $k = 1, 2, \ldots, K$?

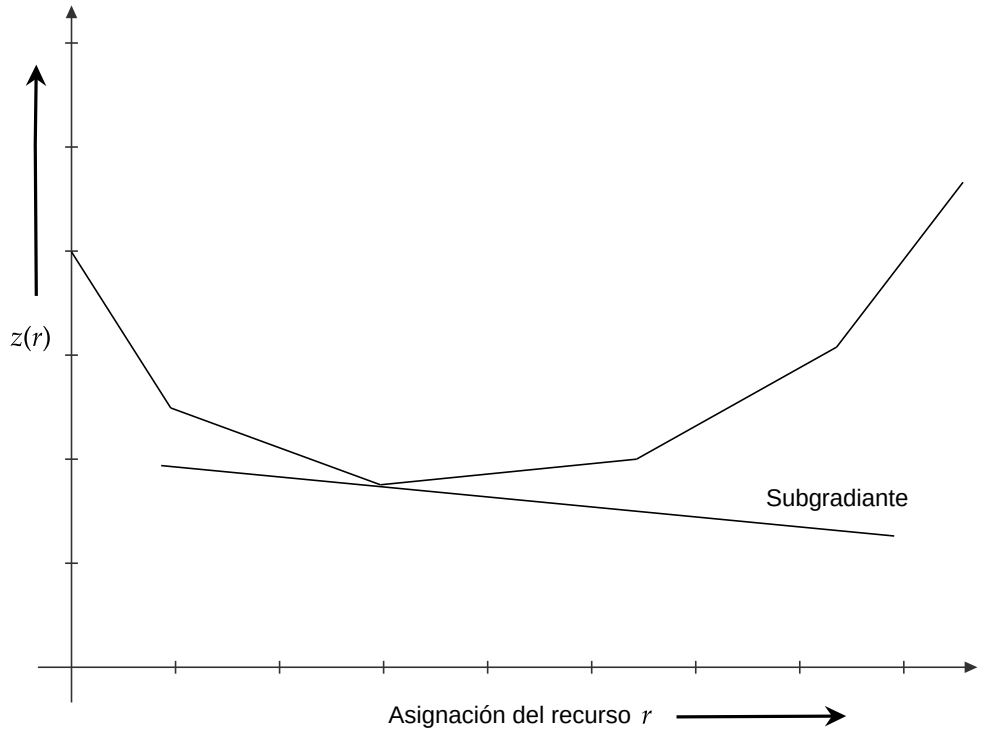

Figura 4.3: Subgradiante de la función z(r)

Para responder a 1, recuérdese que, como se muestra en la Figura 4.3, un subgradiente γ de $z(r)$ en el punto $r = \bar{r}$ es cualquier vector que satisface la condición

$$z(r) \geq z(\bar{r}) + \gamma(r - \bar{r}) \quad \text{para todo } r = (r^1, r^2, \dots, r^K) \text{ con } r^k \in R^k. \tag{4.56}$$

En esta expresión, R^k es el conjunto de asignaciones de recursos para el servicio k para el cual el subproblema (4.49)-(4.51) es factible. La siguiente propiedad muestra que para encontrar cualquiera de esos subgradiantes, se puede trabajar con cada una de las funciones constituyentes $z^k(r^k)$ de forma independiente.

Propiedad 4.5. Sea γ^k para $k = 1, 2, \dots K$ un subgradiante de $z^k(r^k)$ en el punto $\bar{r}^K$. Luego $\gamma = (\gamma^1, \gamma^2, \dots, \gamma^k)$ es un subgradiante de $z(r)$ en $\bar{r} = (\bar{r}^1, \bar{r}^2, \dots, \bar{r}^K)$.

Este resultado muestra que para obtener un subgradiente de $z(r)$, se puede encontrar un subgradiente de cada función $z^k(r^k)$, lo cual requiere información sobre la sensibilidad de la solución de un problema de flujo en red (17.9) con cambios en las cotas superiores r_{ij}^k de los flujos en arco.

Teorema 4.6 (Condiciones de Optimalidad de Holgura Complementaria). *Una solución factible x^* es una solución óptima del problema de flujo de mínimo coste si y sólo si para algún conjunto de potenciales de nodo π, los costes reducidos y los flujos satisfacen las siguientes condiciones de optimalidad para cada arco $(i,j) \in A$:*

$$Si \quad c_{ij}^{\pi} > 0, \quad luego \quad x_{ij}^* = 0. \tag{4.57}$$

$$Si \quad 0 < x_{ij}^* < u_{ij}, \quad luego \quad c_{ij}^{\pi} = 0. \tag{4.58}$$

$$Si \quad c_{ij}^{\pi} < 0, \quad luego \quad x_{ij}^* = u_{ij}. \tag{4.59}$$

Propiedad 4.6. Considérese el problema de flujo de red $z(q) = \text{mín}\{cx : \mathcal{N}x = b, 0 \leq x \leq q\}$, con un vector paramétrico q con cotas superiores no negativas en los flujos en arco. Supóngase que x^* es una solución óptima a este problema cuando $q = q^*$, y que x^* junto con el vector de coste reducido c^{π} satisfacen las condiciones de optimalidad del flujo de mínimo coste del Teorema (4.6). Se definen los vectores $\mu = (\mu_{ij})$ estableciendo

$$\mu_{ij} = \begin{cases} 0 & \text{si } x_{ij}^* < q_{ij}^* \\ c_{ij}^{\pi} & \text{si } x_{ij}^* = q_{ij}^* \end{cases} \tag{4.60}$$

Entonces, para cualquier vector no negativo q' para el cual el problema es factible, $z(q') \geq z(q^*) + \mu(q' - q^*)$. Es decir, μ es un subgradiente de la función objetivo paramétrica $z(\mu)$ en $\mu = \mu^*$.

Para aplicar la Propiedad 4.56, se resuelve cada subproblema (4.52)-(4.54). Después de resolver el subproblema del k-ésimo servicio con $q = r^k$, se establece γ^k igual al vector μ. Luego se usa la propiedad 4.56 y se establece $\gamma = (\gamma^1, \gamma^2, \ldots, \gamma^K)$. Estos resultados muestran cómo encontrar un subgradiente utilizando cualquier algoritmo de flujo de coste mínimo que produzca costes reducidos c^{π} así como un vector de flujo óptimo.

Una vez que se haya obtenido un subgradiante γ de $z(r)$, podría usarse la fórmula de subgradiante $r \leftarrow r + \theta\gamma$ para encontrar la nueva asignación de recursos r. Si el vector de asignación de recursos resultante r es factible (satisface las restricciones $\sum_{1 \leq k \leq K} r_{ij}^k \leq u_{ij}$), se mueve a este punto. Sin embargo, si la nueva asignación de recursos r no es factible, es necesario modificarla para asegurar la factibilidad, se mueve en su lugar a algún otro punto $\bar{r}$. Un método que asegura que el algoritmo converge con la elección apropiada de tamaños de paso (como se discutió en el Sección 2.2.6) es elegir r como el punto factible que está lo más cerca posible de $\bar{r}$ en el sentido de minimizar la cantidad $\sum_{1 \leq k \leq K} \sum_{(i,j) \in A} (r_{ij}^k - \bar{r}_{ij}^k)^2$.

4.9 Partición base

El algoritmo puede utilizar la estructura de árbol de expansión de la base de programación lineal para acelerar los cálculos matriciales. El problema del flujo multiservicio es un problema lineal que combina dos conjuntos de restricciones:

1. restricciones de conservación de flujo independientes $\mathcal{N}x^k = b^k$ para cada servicio $k = 1, 2, \ldots, K$, y
2. restricciones de capacidad que vinculan los servicios a través de capacidades de arco compartidas.

Debido a esta subestructura de red subyacente, es planteable si también podría usarse alguna forma de estructura de árbol de expansión para implementar el método simplex para el problema de flujo multiservicio.

Considérese cualquier base $\mathcal{B}$ de la formulación de programación lineal (4.1)-(4.4) del problema de flujo multiservicio. Dado que cada columna de $\mathcal{B}$ corresponde al flujo en un arco de algún servicio, las columnas de la base definen un conjunto de subgrafos, uno para cada servicio. Sea $\mathcal{M}^k$ la submatriz de $\mathcal{B}$ correspondiente al servicio k. Obsérvese que cada matriz $\mathcal{M}^k$ debe contener una base $\mathcal{B}^k$ de la matriz de incidencia de arco-nodo $\mathcal{N}^k$. Los subgrafos correspondientes a cualquier base son árboles de expansión junto con algunos arcos adicionales. Además, dado que las restricciones del problema de flujo multiservicio tienen forma de bloques, también la tiene la base $\mathcal{B}$. La Figura 4.4 muestra la estructura de la base $\mathcal{B}^k$.

En lugar de trabajar directamente con la matriz base $\mathcal{B}$, mediante un cambio en las variables, se transformará esta matriz en otra matriz $\mathcal{B}'$ para poder realizar los pasos del método simplex de manera mucho más eficiente utilizando la estructura de red subyacente del problema. Supóngase que se establecen las restricciones de capacidad como restricciones de igualdad (4.5) agregando la variable de holgura no negativa s_{ij} a la restricción de capacidad para el arco (i, j). La variable s_{ij} denota la capacidad de flujo no utilizada en el arco (i, j). Considérese el sistema de balance de flujo y restricciones de capacidad como base del problema de flujo multiservicio, que se escribe como $\mathcal{B}x_B = b$. En esta expresión, $b = (u, b^1, b^2, \ldots, b^K)$ es el vector de los coeficientes del lado derecho de las ecuaciones y x_B es el conjunto de variables correspondientes a las variables básicas. Estas variables incluyen tanto las variables de flujo x_{ij}^k como las variables de holgura s_{ij} para las restricciones de capacidad.

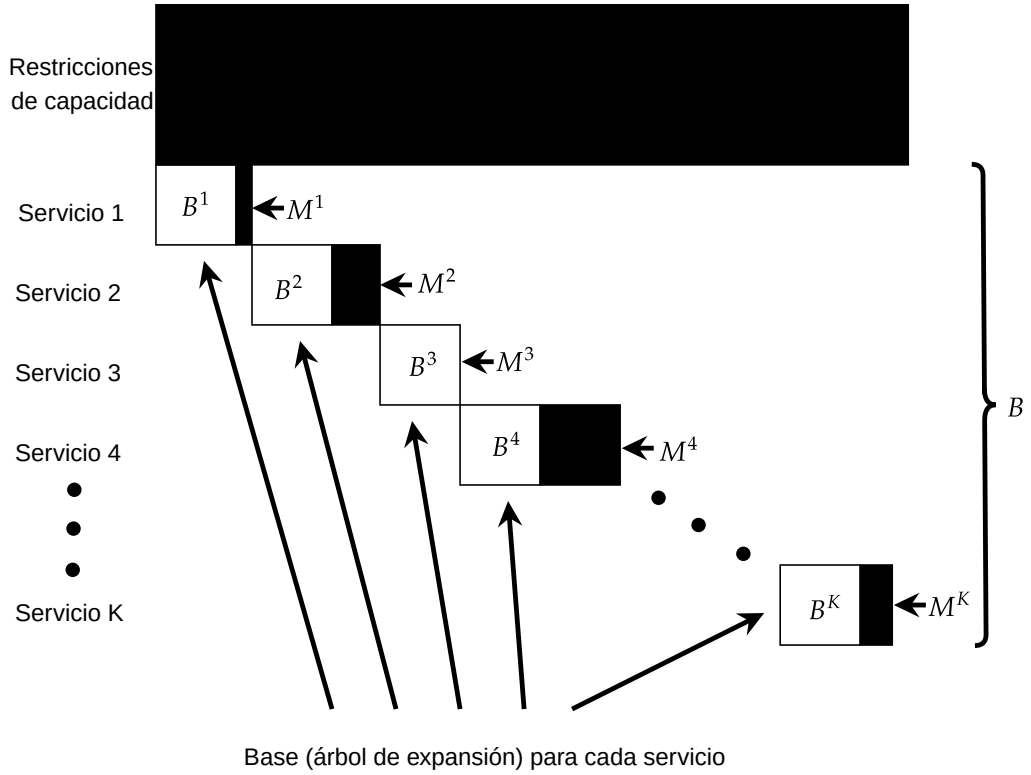

Figura 4.4: Partición de una base de programación lineal del problema de flujo multiservicio: base original.

Para simplificar el sistema $\mathcal{B}x_B = b$, supóngase que se hace un cambio en las variables dejando $\mathcal{D}y_B = x_B$ para una matriz no singular $\mathcal{D}$ seleccionada apropiadamente. Si se sustituye $\mathcal{D}y_B$ por x_B en el sistema $\mathcal{B}x_B = b$, y $\mathcal{B}' = \mathcal{B}\mathcal{D}$, el sistema se convierte en $\mathcal{B}(\mathcal{D}y_B) = (\mathcal{B}\mathcal{D})y_B = \mathcal{B}'y_B = b$. Supóngase que se puede hacer este cambio en las variables de modo que la matriz $\mathcal{B}'$ tenga la estructura especial que se muestra en la Figura 4.5; en esta estructura especial, solo las variables y que aparecen en las restricciones de balance de flujo para el servicio k son las correspondientes a la base $\mathcal{B}^k$.

En el sistema transformado, sea $\mathcal{W}$ la matriz de las restricciones de capacidad que no corresponden a ninguna de las columnas en las bases $\mathcal{B}^k$ para $k = 1, 2, \ldots, K$. Se hace referencias a $\mathcal{W}$ como el trabajo base. En la representación pictórica de la base $\mathcal{B}'$ en la Figura 4.5, se ha reorganizado las columnas de la base de modo que aquellas en la base de trabajo aparezcan primero.

Dos observaciones más:

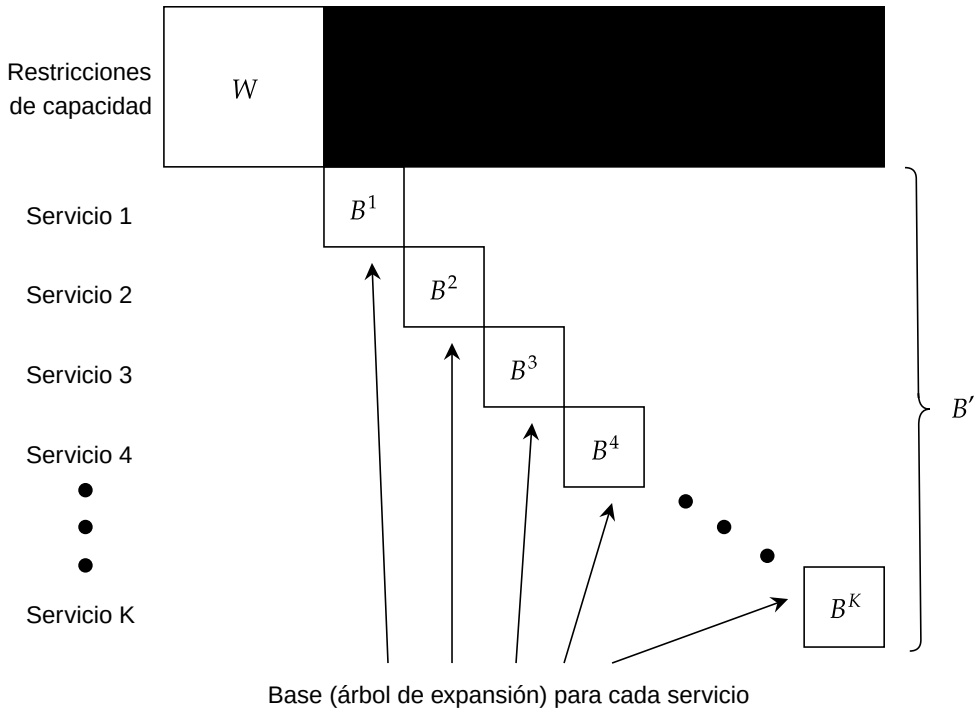

Figura 4.5: Partición de una base de programación lineal del problema de flujo multiservicio: base después de cambio de variables y reordenamiento de columnas.

1. Dado que un cambio en las variables, que corresponde a las operaciones de columna en la matriz $\mathcal{B}$, no cambia que esta matriz no es singular, la matriz $\mathcal{B}'$ tampoco es singular. Este hecho implica que la base de trabajo $\mathcal{W}$ no es singular.
2. Dado que las operaciones de columna en la matriz $\mathcal{B}$ no cambian la solución del sistema $\pi\mathcal{B} = c_B$, el vector de los multiplicadores simplex determinados por el sistema $\pi\mathcal{B}' = c_{B'}$ es el mismo que los determinados por el sistema original $\pi\mathcal{B} = c_B$. (Cuando se realizan las operaciones de columna, se cambia tanto $\mathcal{B}$ como c_B; $c_{B'}$ denota el resultado de las operaciones de columna en los costes c_B).

Estas dos observaciones y el cambio en las variables dan todos los ingredientes necesarios para implementar el método simplex de partición de bases para el problema de flujo multiservicio. Recuérdese que el método simplex requiere de dos tipos de cálculos matriciales:

1. Resolver un sistema de la forma $\mathcal{B}x_B = b$ o $\mathcal{B}x_B = \mathcal{A}_j$ para alguna columna $\mathcal{A}_j$

de la matriz de coeficientes $\mathcal{A}$ del modelo de programación lineal (se hace este cálculo en cada paso antes de introducir la columna $\mathcal{A}_j$ en la base).
2. Resolver un sistema de la forma $\pi\mathcal{B} = c_B$ para encontrar el vector π de multiplicadores simplex.

En lugar de hacer estos cálculos de la matriz $\mathcal{B}$ directamente, se usará la matriz $\mathcal{B}'$. Los sistemas $\pi\mathcal{B} = c_B$ y $\pi\mathcal{B}' = c_{B'}$ tienen las mismas soluciones. Supóngase que se divide el vector de multiplicadores simplex como $\pi = (\pi^0, \pi_1, \ldots, \pi^K)$. El vector π^0 contiene los multiplicadores simplex para las restricciones de capacidad y el vector π^k contiene los multiplicadores simplex correspondientes a las restricciones del balance de flujo. $\mathcal{N}x^k = b^k$ para el servicio k. Sea c_W el subvector de c_B correspondiente a las columnas de W. Se resuelve el sistema $\pi\mathcal{B}' = c_{B'}$ de manera eficiente mediante el siguiente procedimiento de sustitución hacia adelante. Primero se resuelve el subsistema $\pi\mathcal{B}' = c_{B'}$ resolviendo un sistema de ecuaciones lineales. Si se sustituye estos valores en el sistema $\pi\mathcal{B}' = c_{B'}$, el sistema se descompone en subsistemas separados para cada servicio k con coeficientes modificados $\hat{c}_{B'}$ para $c_{B'}$. Además, los cálculos para cada subsistema son exactamente los necesarios para encontrar los multiplicadores simplex de un problema de flujo de red de un sólo servicio.

Para resolver un sistema de la forma $\mathcal{B}x_B = b$, primero se resuelve el sistema relacionado $\mathcal{B}'y_B = b$ usando sustitución hacia adelante. Es decir, si se divide el vector y_B en $y_B = (y^0, y^1, y^2, \ldots, y^K)$ correspondiente a las columnas $W, \mathcal{B}^1, \mathcal{B}^2, \ldots, \mathcal{B}^K$ de $\mathcal{B}'$, primero se resuelve los subsistemas independientes $\mathcal{B}^k y^k = b^k$. Luego se sustituye los valores de las variables $y^1, y^2, \ldots, y^K$ en el sistema $\mathcal{B}'y = b$, dejando un sistema reducido de la forma $\mathcal{W}y^0 = u'$ (aquí u' denota las capacidades de arco modificadas que se obtiene después de sustituir los valores de las variables de flujo $y^1, y^2, \ldots, y^K$ en las restricciones de capacidad transformadas). Después de resolver, en este sistema de ecuaciones lineales, se usa la transformación entre x_B e y_B (es decir, $x_B = \mathcal{D}y_B$) para encontrar los valores de las variables x_B.

Supóngase que se aplica el procedimiento a un problema con p restricciones de capacidad (p podría ser igual al número m de arcos si todos tienen restricciones de capacidad, o podría ser mucho menor que m). En lugar de resolver sistemas de la forma $\mathcal{B}x = b$ en la matriz completa $\mathcal{B}$ que tiene dimensión $(p + K) \times (p + K)$, se hacen K cálculos en árbol y se resuelve un sistema $p \times p$ más pequeño $\mathcal{W}y^0 = u'$. De manera similar, en lugar de resolver un sistema grande $(p+K)\times(p+K)$ de la forma $\pi\mathcal{B} = c_B$ se resuelve un solo sistema $\mathcal{W}y^0 = u'$ de dimensiones $p \times p$ y se hace K cálculos en árbol.

El resto de los pasos del método simplex (tasar las columnas para encontrar la variable

de entrada en cada paso o realizar la prueba de ratio para encontrar la variable de salida) son los del método general.

Para completar esta breve descripción del método de partición de base, se muestra cómo hacer el cambio en las variables $x_B = \mathcal{D}y_B$ necesario para eliminar cualquier columna $\mathcal{M}^k_{ij}$ que pertenezca a $\mathcal{M}^k$ pero no a $\mathcal{B}^k$ de las ecuaciones de balance de flujo para el servicio k de modo que las únicas variables restantes en estas ecuaciones serán las correspondientes a la base $\mathcal{B}^k$.

Considérese el siguiente ejemplo, que corresponde al sistema $\mathcal{M}^k x^k = b^k$ (para simplificar, se elimina el índice de servicios k y se usa x_{ij} en lugar de x^k_{ij}):

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}\!\!\begin{bmatrix} +1 & +1 & & & & & & \\ -1 & & & & & & & \\ & -1 & +1 & +1 & & & & \\ & & -1 & & & & +1 & \\ & & & & -1 & -1 & & \\ & & & & & +1 & -1 & \end{bmatrix}\begin{bmatrix} x_{12} \\ x_{13} \\ x_{34} \\ x_{35} \\ x_{65} \\ x_{46} \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 1 \\ 2 \\ 4 \\ 3 \end{bmatrix} \tag{4.61}$$

Sea $\mathcal{M}_{ij}$ la columna de este sistema correspondiente a la variable x_{ij}. Las columnas $\mathcal{M}_{12}$, $\mathcal{M}_{13}$, $\mathcal{M}_{34}$, $\mathcal{M}_{35}$ y $\mathcal{M}_{65}$ forman una base para este sistema y $\mathcal{M}_{46}$ es la columna no básica que se desea eliminar. Obsérvese que $\mathcal{M}_{46} = -\mathcal{M}_{34} + \mathcal{M}_{35} - \mathcal{M}_{65}$, que en términos de la red subyacente da la representación del arco $(4, 6)$ en términos del árbol de expansión definido por los arcos $(1, 2)$, $(1, 3)$, $(3, 4)$, $(3, 5)$ y $(6, 5)$. Ahora supóngase que se hace el cambio en las variables $x_{34} = y_{34} - y_{46}$, $x_{35} = y_{35} + y_{46}$, $x_{65} = y_{65} - y_{46}$, $x_{12} = y_{12}$, $x_{13} = y_{13}$ y $x_{46} = y_{46}$. Entonces se encuentra que

$$\mathcal{M}_{12}x_{12} + \mathcal{M}_{13}x_{13} + \mathcal{M}_{34}x_{34} + \mathcal{M}_{35}x_{35} + \mathcal{M}_{65}x_{65} + \mathcal{M}_{46}x_{46} \tag{4.62}$$

$$= \mathcal{M}_{12}y_{12} + \mathcal{M}_{13}y_{13} + \mathcal{M}_{34}(y_{34} - y_{46}) + \mathcal{M}_{35}(y_{35} + y_{46}) + \mathcal{M}_{65}(y_{65} - y_{46}) + \mathcal{M}_{46}y_{46} \tag{4.63}$$

$$= \mathcal{M}_{12}y_{12} + \mathcal{M}_{13}y_{13} + \mathcal{M}_{34}y_{34} + \mathcal{M}_{35}y_{35} + \mathcal{M}_{65}y_{65}. \tag{4.64}$$

Es decir, se ha eliminado la columna $\mathcal{M}_{46}$ del sistema. Las nuevas variables y_{34}, y_{35} e y_{65} miden tanto los flujos x_{ij} en estos arcos como el flujo x_{46} "desviado" del arco $(4, 6)$ a estos arcos en la trayectoria $4 - 3 - 5 - 6$.

En general, si se tienen varias columnas $\mathcal{M}_{ij}$ que se desean eliminar del sistema, y en el grafo subyacente al arco (i, j) forma un ciclo único cuando se agrega al árbol

de expansión T correspondiente a la base del sistema, sea $\gamma_{ij}^{pq} = \pm 1$ ó 0, dependiendo de si el arco (p, q) del árbol pertenece o no a este ciclo (el signo de cada ± 1 depende de la orientación del arco del árbol (p, q) relativo al arco (i, j)). Para transformar las variables, se establece $x_{pq} = y_{pq} - \sum \gamma_{ij}^{pq} y_{pq}$ con la suma de los índices (i, j) para todas las columnas $\mathcal{M}_{ij}$ que se desean eliminar. También se establece $x_{ij} = y_{ij}$ para todos los arcos (i, j) que no están en el árbol T.

Para implementar este procedimiento para el problema de flujo multiservicio, se aplican estos cálculos de red y el cambio de variables para cada una de las matrices $\mathcal{M}^k$. Estos cálculos dan la matriz $\mathcal{B}'$ y luego se aplican las operaciones discutidas anteriormente.

5 Métodos basados en los flujos en redes para el problema del óptimo enrutamiento de tuberías en diseños navales

5.1 Introduction

En la actualidad existe una gran cantidad de situaciones prácticas en las que es necesario diseñar redes de forma óptima. Las infraestructuras industriales requieren de reglas de diseño y restricciones de trazado de alto nivel de redes complejas que involucran diversos servicios independientes que deben competir por espacios de trazado escasos y respetando la compatibilidad y fabricabilidad de los elementos diseñados y trazados. Aunque en el mercado existen algunas herramientas comerciales que facilitan el desarrollo de esta labor, la forma de trabajar con ellas sigue siendo muy manual, requiriendo que un técnico especialista dedique multitud de horas dibujando y modelizando en 3D. Este capítulo trata el problema de trazar canalizaciones en ingeniería naval, aunque tiene aplicabilidad directa en otras situaciones como en el uso de infraestructuras inteligentes.

El routing automático no es un concepto nuevo, y ya se ha tratado de abordar en contadas ocasiones con éxito escaso, debido a la complejidad del problema a resolver y las múltiples variables implicadas. No hace muchos años, la capacidad de procesamiento de los ordenadores era demasiado limitada para poder abordar este tipo de problemas, pero el avance en la computarización y sistemas de almacenamiento de datos han abierto la puerta a la resolución del problema de manera eficiente, y esto ha provocado la aparición de iniciativas no solo en el ámbito de las instalaciones de

tuberías, sino también dentro del diseño de canalizaciones y rutado eléctrico.

En el mercado existen algunas herramientas software que facilitan el desarrollo de esta labor, siendo las más comunes: AVEVA, FORAN y SMART3D. En cualquier caso los algoritmos que subyacen bajo estos módulos que intentan la automatización, son deterministas y ofrecen una única solución óptima (entendiéndose "óptima" desde el punto de vista de lo que el algoritmo tiene programado como "óptimo", pero no desde el punto de vista del diseñador), que es siempre la misma con las mismas variables de entrada, solución que en muchas ocasiones no resulta válida para el diseñador especialista.

En este capítulo se desarrolla una metodología de diseño automatizado de redes complejas basada en la descomposición de problemas de flujo multiservicio en flujos secuenciales separables evitando los problemas de constructibilidad basados en algoritmos de aprendizaje automático. La herramienta proporcionada puede integrarse en las plataformas de los diseñadores y acelerar la generación de los datos de entrada, para poder generar los escenarios y los diagramas funcionales inteligentes de forma rápida. Además proporciona la opción de modificarlos para absorber los cambios que se producen en las condiciones de contorno del diseño en un proyecto real.

5.2 El problema de enrutamiento en el contexto del diseño naval.

El objetivo del capítulo es modelar y resolver el problema de enrutar en un barco diferentes conductos que compite por un espacio reducido teniendo en cuenta ciertos requisitos técnicos. A continuación, se describen los principales elementos del problema.

Supóngase que la región donde se enrutarán las tuberías está representada por un poliedro $\mathcal{P}_0$ acotado en $\mathbb{R}^3$. En situaciones reales, el poliedro suele ser un ortoedro en la forma $\mathcal{P}_0 = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$, y representa una cabina del barco. Hay también un conjunto finito de prismas $O_1, \dots, O_S \subseteq \mathcal{P}_0$ que representan obstáculos que no pueden ser atravesados por las tuberías. En favor de una mayor simplicidad, estos obstáculos también serán ortoedros. Por tanto, la región tridimensional donde se permite trazar las tuberías es $\mathcal{P} = \mathcal{P}_0 \setminus U_{s=1}^{S} O_s$. Además, se tiene un conjunto finito de servicios (tuberías cilíndricas) cada uno de ellos representado por las coordenadas de un punto de origen y uno de destino (ambos en $\mathcal{P}$), un radio (de la tubería cilíndrica)

y una distancia mínima de seguridad permitida respecto a otros servicios. Esta metodología se puede adaptar fácilmente a otras formas de tuberías, como paralelepípedos (con secciones rectangulares).

La principal decisión a tomar en este problema es determinar el conjunto de (dimensiones) caminos en $\mathcal{P}$ que debe seguir cada uno de los servicios. Una ruta factible para un servicio es una unión continua de segmentos en $\mathcal{P}$ comenzando en su punto inicial y terminando en su punto final. Además, debe garantizarse una distancia mínima entre dos puntos de ruptura consecutivos (codos) por razones de constructibilidad. Además, una vez que se traza una ruta factible para cada servicio, hay que asegurar que, una vez que se traza una tubería (cilíndrica) alrededor de la ruta, se respeta una mínima distancia entre los diferentes servicios. En particular, no se permite que los caminos se crucen o superpongan en una solución.

La calidad de una solución adecuada del problema, que es un conjunto de caminos factibles, se evalúa respecto a diferentes criterios. Además de la longitud, que es proporcional a los costes, también se buscan caminos con un número reducido de codos, que pasen cerca del techo de la región o atravesando zonas específicas que son de preferencia.

A continuación, se describe la representación matemática de los elementos involucrados en el problema de enrutamiento de la tubería.

Para discretizar el espacio de trabajo, primero, se construye una cuadrícula 3D ortogonal en el paralelepide $\mathcal{P}_0$. Esta cuadrícula define un grafo base no dirigido al que se le eliminan los nodos y aristas que atraviesan los obstáculos. Sin embargo, al trazar un camino en un grafo de este tipo, no se puede detectar ni penalizar los codos que aparecen en el trazado. Los codos en una tubería deben identificarse adecuadamente, tanto por la constructibilidad como porque se prefieren las rutas con un menor número de codos. El objetivo es obtener un modelo de programación matemática para el problema que evite las no linealidades. Para considerar codos en una función objetivo lineal, se modifica el grafo inicial explotando los nodos del grafo de la siguiente manera:

1. Cada nodo físico v en el grafo inicial es reemplazado por un nodo explotado, es decir, un conjunto de tres nodos virtuales, v_X, v_Y, v_Z con las mismas coordenadas 3D que v, una para cada dirección de la base canónica de $\mathbb{R}^3$ (X, Y y Z).
2. Cada arista física en el grafo inicial es reemplazada por otra arista con la misma longitud que la arista física, pero en lugar de unir dos nodos físicos, conecta

dos nodos virtuales (asociados a los mismos nodos físicos de la arista original). Específicamente, cada arista está vinculada al nodo virtual identificado con una dirección. De esta manera, las aristas paralelas al eje X vinculan los nodos virtuales X, las aristas paralelas al eje Y vinculan los nodos virtuales Y y las aristas paralelas al eje Z vincula los nodos virtuales Z.

3. Los nodos virtuales asociados a un mismo nodo físico están vinculados a través de aristas virtuales. Sus longitudes son cero (ya que enlazan nodos con las mismas coordenadas), pero tendrán un coste positivo que representa el uso de un codo en la ruta.

En la Figura 5.1 se muestra una ilustración de la explosión de nodos. En la imagen de la izquierda se muestra un nodo v de un cruce, que está unido a otros seis nodos. En la imagen de la derecha se muestra los tres nodos explotados, v_X, v_Y y v_Z , del nodo v. El nodo v_X (respectivamente, v_Y, v_Z) está unido sólo a los dos nodos adyacentes mediante aristas paralelas al eje X (respectivamente, eje Y, eje Z) y con los otros dos nodos virtuales asociados al mismo nodo físico. Como puede verse, las tres aristas virtuales que unen v_X con v_Y, v_X con v_Z y v_Y con v_Z están dibujados con líneas discontinuas en esta imagen. En la Figura 5.2 se ilustra el uso de este grafo para modelar diferentes tipos de giros en una ruta. En la imagen de la izquierda, para pasar de a a b no se utilizan codos y se utiliza un único nodo explotado (v_Z). Por otro lado, en la imagen central (respectivamente, derecha), girar desde una arista paralela al eje Z hacia una arista paralela al eje Y (respectivamente, Eje X) requiere atravesar la arista virtual (v_Y; v_Z) (respectivamente, (v_Z; v_X)) agregando un coste por usar ese codo. El resultado es un grafo que representa el espacio factible donde se enrutan las tuberías.

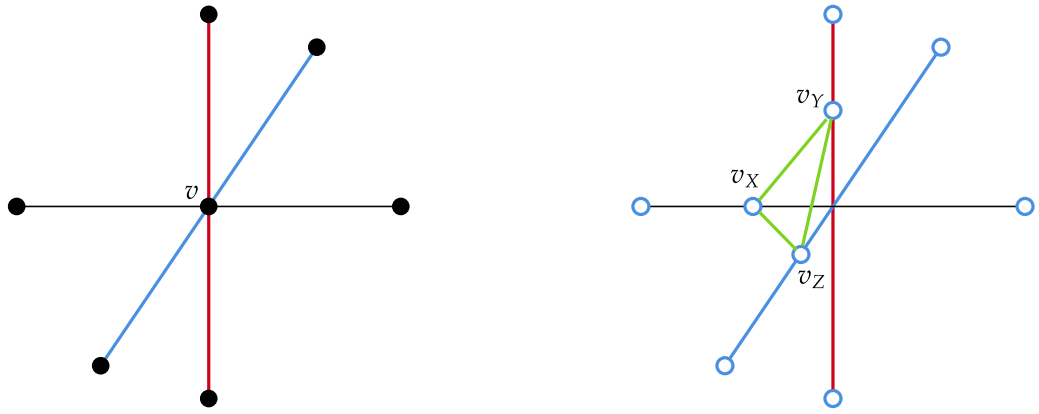

Figura 5.1: Ilustración de nodos explotados y aristas virtuales.

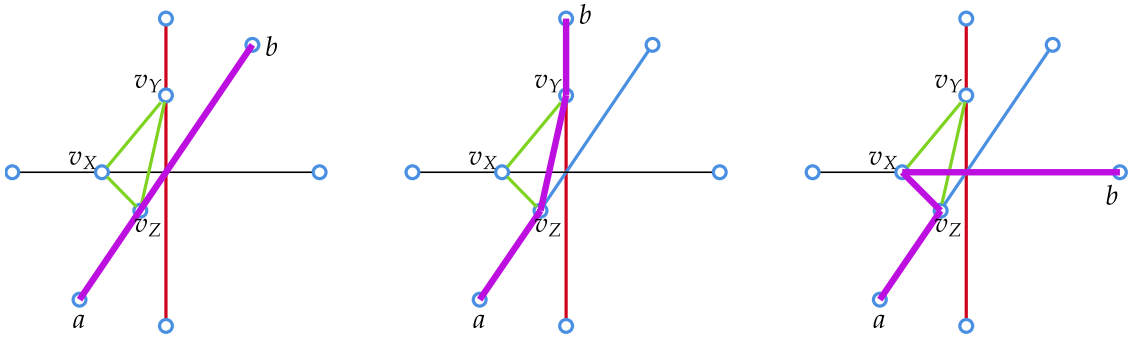

Figura 5.2: Ilustración del uso de nodos explotados y aristas virtuales para modelar diferentes tipos de giros en un camino del nodo a al nodo b.

Se denota por $\widetilde{G} = (V; E)$ el grafo construido como se ha descrito anteriormente, donde V es el conjunto de nodos (virtuales) y E el conjunto de aristas. El conjunto E incluye el conjunto de aristas virtuales, denotadas por E^v, y las réplicas de las aristas físicas iniciales. Dado que $\widetilde{G}$ está enmarcado en $\mathbb{R}^3$, cada nodo $v \in V$ también se identifica con sus coordenadas 3D, $(x_v, y_v, z_v) \in \mathbb{R}^3$. Se denota por $d_{vv'} = d((x_v, y_v, z_v), (x_{v'}, y_{v'}, z_{v'}))$ la distancia euclidia entre los nodos $v, v' \in V$ y por $d_{ee'}$ la distancia euclidia entre las aristas $e, e' \in E$ (que representa tanto un punto en $\mathbb{R}^3$ si $e \in E^v$ o un segmento en $\mathbb{R}^3$ si $e \in E \setminus E^v$). Se supone que se tiene un conjunto finito de servicios, K, y cada uno de ellos requiere su propia tubería para determinar el diseño de su ruta. Cada servicio $k \in K$ se define mediante un par (s^k, t^k) donde $s_k \in V$ es la fuente del servicio y $t_k \in V$ es el destino del servicio. Dependiendo del servicio $k \in K$, se define una estructura de costes, $c_e^k \geq 0$ para cada arista del grafo $e \in E$. Estos costes dependen de muchos factores como, por ejemplo, la longitud de las aristas, el coste de un codo, las preferencias del diseñador al enrutar las tuberías, etc. En la Sección 5.4.2 se describirá una estructura de costes detallada para el caso de estudio que trata este trabajo.

5.3 Un modelo basado en el flujo de red multiservicio (Faltan referencias y dibujos)

En esta sección se describe el modelo de programación matemática propuesto para resolver el problema de enrutamiento de tuberías que impone requisitos técnicos de diseño naval. En una primera aproximación, se puede modelar el problema como un

modelo de flujo de red multiservicio de mínimo coste (PFRMC, para abreviar). Se denota con $G = (V, A)$ (donde $A = \{(i, j) \cup (j, i) : e = \{i, j\} \in E\}$ es el conjunto de arcos) es la versión dirigida del grafo $\widetilde{G}$ descrita en el Sección 5.2 previa. Similar a lo que ocurre en la versión no dirigida, se asume que el conjunto A incluye el conjunto de arcos virtuales, denotado por A^v, y las réplicas de los arcos físicos y se denota con $d_{aa'}$ la distancia euclidea entre los arcos $a, a' \in A$ (obsérvese que $d_{aa'} = 0$ si a y a' son los arcos correspondientes a las dos direcciones opuestas de la misma arista). Además, supóngase que, para cada servicio $k \in K$, el coste de cada arco en A es $c^k_{ij} = c^k_{ji} = c^k_e$. El objetivo del PFRMC es enrutar conjuntamente todos los servicios de K desde sus respectivas fuentes, s_k, a sus destinos, t_k, a través de la red G a un mínimo coste.

Ejemplo 5.1. En la Figura 5.3, a la izquierda, se muestra un escenario que bien podría ser parte de un barco. El escenario mide tiene 3 servicios y un obstáculo de dimensiones que podría ser una viga. También en la Figura 1, a la derecha, se muestra la discretización del escenario en un mallado. Este mallado representa una versión simplificado del grafo $G = (V, A)$ que realmente se usa en la resolución del problema de flujo multiservicio puesto que en el mallado los nodos no están explotados. Sea i un nodo genérico del mallado, entonces i_x, i_y, i_z son sus correspondientes nodos virtuales y sea n el número de nodos en el mallado. De manera que $V = \{i_x, i_y, i_z\}$ con $i = \{1, \dots, n\}$. A su vez, de acuerdo con la descripción de grafo dirigido G, $E^v = \{(v_{i_x}, v_{i_y}), (v_{i_x}, v_{i_z}), (v_{i_y}, v_{i_z})\}$ son las aristas virtuales resultantes de explotar los nodos y $E \setminus E^v = \{(v_{i_x}, v_{i+1_x}), (v_{i_y}, v_{i+1_y}), (v_{i_z}, v_{i+1_z})\}$ las aristas reales que aparecen en el mallado.

Para describir el modelo de programación matemática para el PFRMC, se usa el siguiente conjunto de variables de decisión binarias:

$$x^k_{ij} = \begin{cases} 1 & \text{si la ruta del servicio } k \text{ utiliza el arco } (i, j), \\ 0 & \text{otro caso } n > 0. \end{cases} \tag{5.1}$$

Con este conjunto de variables, el coste total de enrutar los servicios a través de la red se puede expresar como:

$$\sum_{(i,j)\in A} \sum_{k\in K} c^k_{ij} x^k_{ij}. \tag{5.2}$$

El siguiente conjunto de restricciones lineales asegura la representación correcta del

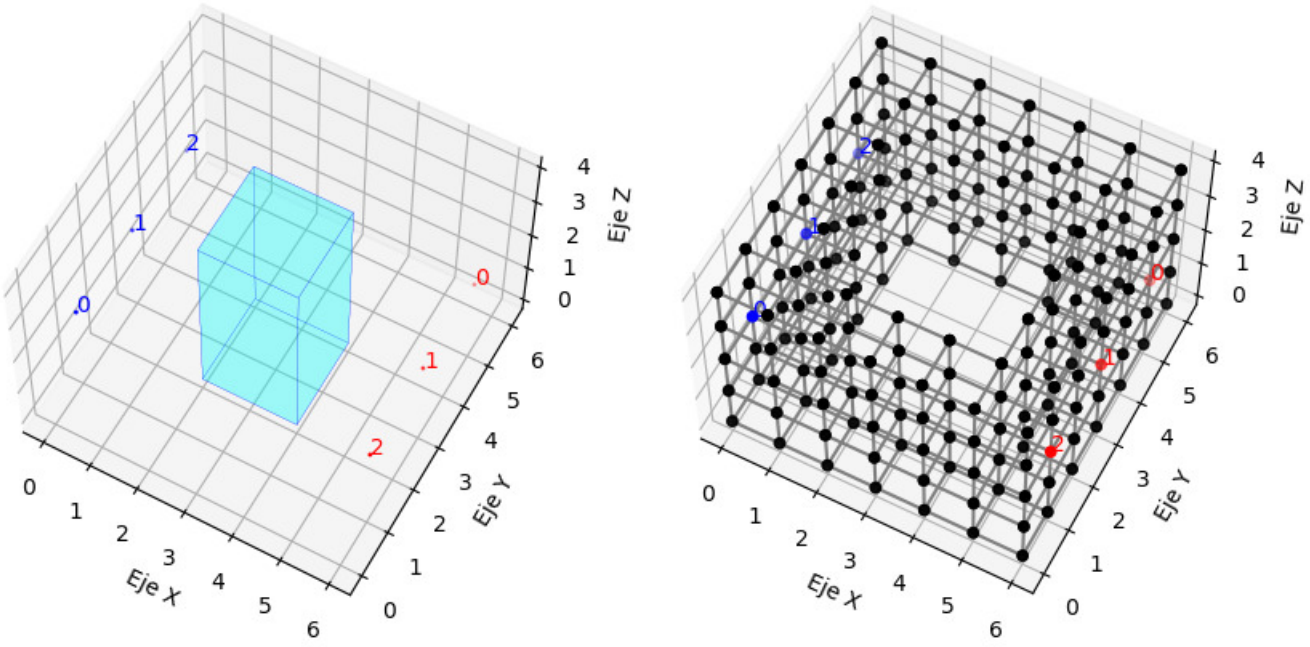

Figura 5.3: A la izquierda, escenario con tres fuentes en azul, tres sumideros en rojo y un obstáculo que se interpone entre ellos. A la derecha, mallado resultante de discretizar el espacio del escenario a la izquierda.

problema:

$$\sum_{k\in K}\sum_{j\in V} x_{ij}^k \leq 1, \qquad \forall i \in V, \tag{5.3}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0, \qquad \forall k \in K, i \in V(i \neq s^k, t^k), \tag{5.4}$$

$$\sum_{j\in V} x_{s^k,j}^k = 1, \qquad \forall k \in K, \tag{5.5}$$

$$\sum_{i\in V} x_{i,t^k}^k = 1, \qquad \forall k \in K, \tag{5.6}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K. \tag{5.7}$$

donde (5.3) establece que por un arco sólo pase un servicio, (5.4) impone que de un nodo sale tanto flujo como entra en él, (5.5) establece que el flujo que sale de una fuente debe ocupar una arista adyacente, (5.6) impone que una sólo por una arista

adyacente a un sumidero debe pasar flujo y (5.7) limita las variables del modelo a valores binarios.

La solución del PFRMC es un conjunto de rutas no superpuestas (una para cada servicio) que conectan las fuentes con los sumideros. Se sabe que este problema es NP-duro, incluso para el caso de dos servicios (ver [2]). Además, en el enrutamiento de tuberías hay dos requisitos técnicos que deben considerarse para construir diseños factibles: una mínima distancia entre las tuberías y una mínima distancia entre dos codos consecutivos de una tubería. El primer requisito permite separar adecuadamente los diferentes servicios para ser utilizados por las tuberías así como para mantener espacio suficiente para la manipulación de maquinaria en los camarotes de los barcos. El segundo, es una limitación física del trazado de la tubería, ya que no hay suficiente espacio para que una tubería gire dos veces en caso de que los codos estén colocados demasiado cerca. Las soluciones constan de segmentos en $\mathbb{R}^3$ sin dimensionalidad y, por ello, estas consideraciones no se tienen en cuenta en el PFRMC. Por lo tanto, para imponer esos requisitos técnicos, se deben agregar al problema las siguientes restricciones para modelar adecuadamente el problema de trazado de tuberías:

1. *Distancia entre servicios:*
   Dado una arista real (no virtual) usado por el servicio k, se requiere una mínima distancia de separación respecto a otros servicios. Sea R^k el radio del cilindro de la tubería $k \in K$ y sea Δ^k la mínima distancia de seguridad desde la superficie de la tubería k hasta cualquier otro elemento. La mínima distancia permitida entre los servicios k y k, es entonces $R^{kk'} = R^k + R^{k'} + \max\{\Delta^k \Delta^{k'}\}$.
   Se asegura el cumplimiento de este requisito imponiendo el siguiente conjunto de restricciones:

$$\sum_{k' \in K : k' \neq k} \sum_{\substack{a' \in A \setminus A^v \\ d_{aa'} < R^{kk'}}} x_{a'}^{k'} \leq M(1 - x_a^k), \qquad \forall a \in A \setminus A^v, k \in K, \tag{5.8}$$

   es decir, si a es un arco en la ruta del servicio k, entonces, no puede haber ningún arco en la ruta de cualquier otro servicio $k' \neq k$ a una distancia menor que la mínima requerida. Aquí, M es una constante suficientemente grande.
2. *Test de codos:*
   Se requiere de un camino factible del servicio para verificar una distancia mínima , $D^k \geq 0$, entre codos consecutivos. Dado que los codos en el grafo no dirigido se identifican con aristas virtuales y luego, en el grafo dirigido con ar-

cos virtuales, este requisito se puede incorporar al modelo como sigue:

$$x_a^k + x_{a'}^k \leq 1, \qquad \forall a, a' \in A^v : d_{aa'} \leq D^k, a \neq a', k \in K, \tag{5.9}$$

es decir, si dos arcos virtuales (codos) diferentes del mismo servicio $k \in K$ están a una distancia menor o igual que D^k, entonces solo uno de ellos puede estar activo.

Con las consideraciones anteriores, el problema de trazado de tuberías en un barco (PTTB) se puede formular como:

$$\text{mín} \sum_{(i,j)\in A} \sum_{k\in K} c_{ij}^k x_{ij}^k. \tag{5.10}$$

$$\text{s. a} \sum_{k\in K} \sum_{j\in V} x_{ij}^k \leq 1, \qquad \forall i \in V, \tag{5.11}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0, \qquad \forall k \in K, i \in V(i \neq s^k, t^k), \tag{5.12}$$

$$\sum_{j\in V} x_{s^k,j}^k = 1, \qquad \forall k \in K, \tag{5.13}$$

$$\sum_{i\in V} x_{i,t^k}^k = 1, \qquad \forall k \in K, \tag{5.14}$$

$$\sum_{k'\in K : k'\neq k} \sum_{\substack{a'\in A\setminus A^v \\ d_{aa'} < R^{kk'}}} x_{a'}^{k'} \leq M(1 - x_a^k), \qquad \forall a \in A \setminus A^v, k \in K, \tag{5.15}$$

$$x_a^k + x_{a'}^k \leq 1, \qquad \forall a, a' \in A^v : d_{aa'} \leq D^k, a \neq a', k \in K, \tag{5.16}$$

$$x_{ij}^k \in \{0, 1\}, \forall (i, j) \in A, k \in K. \tag{5.17}$$

Ejemplo 5.2. Continuando con el ejemplo 5.1 anterior, se resuelve este en la Figura 5.3. A la izquierda se resuelve el problema de flujo multiservicio sin introducir en el modelo las restricciones correspondientes al test de codos y al test de revestimiento. A la derecha aparece la solución del modelo una vez introducidas las restricciones correspondientes al test de codos y revestimiento.

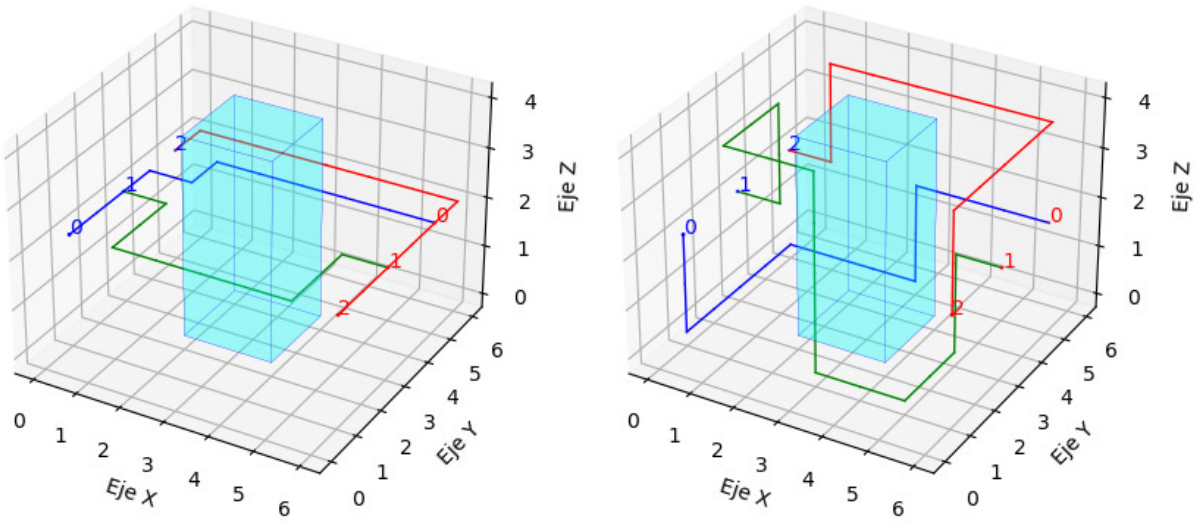

Figura 5.4: Solucion sin aplicar test de codos o test de revestimientos a la izquierda y aplicándolos a la derecha.

El (PTTB) es un problema de flujo de red multiservicio de mínimo coste con restricciones adicionales que imponen el cumplimiento de los requisitos técnicos de las tuberías en la solución. Aunque el problema se modela como un problema de programación lineal de números enteros, en situaciones del mundo real tiene un gran número de variables ($\mathcal{O}(|A||K|)$) y un gran número de restricciones. En particular, las restricciones (5.15) y (5.16) imponen una carga considerable sobre el modelo ($\mathcal{O}(|A|^2|K|)$).

5.3.1 Un método de solución exacta para el (PTTB)

Las grandes dimensiones del modelo hacen inviable el uso de un solucionador de programación entera mixta incluso para instancias de pequeño tamaño. Este inconveniente se supera utilizando un método de ramificación y corte (sección 2.1) que incorpora las restricciones que se requieren, en lugar de considerar inicialmente todas.

Esta estrategia es una forma eficiente de alcanzar la solución exacta del problema al

resolver una formulación incompleta con sólo algunas de las restricciones del modelo, mientras que las restricciones restantes, se incorporan en el modelo según sea necesario. Aunque en el peor de los casos, el procedimiento puede necesitar todas las restricciones que no se incluyeron inicialmente, en la práctica, solo se agrega una pequeña cantidad de ellas, reduciendo considerablemente el tamaño del problema.

Específicamente, considerando al principio el siguiente problema maestro relajado:

$$\text{mín} \sum_{(i,j)\in A} \sum_{k\in K} c_{ij}^k x_{ij}^k. \tag{5.18}$$

$$\text{s. a} \sum_{k\in K} \sum_{j\in V} x_{ij}^k \leq 1, \qquad \forall i \in V, \tag{5.19}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0, \qquad \forall k \in K, i \in V(i \neq s^k, t^k), \tag{5.20}$$

$$\sum_{j\in V} x_{s^k,j}^k = 1, \qquad \forall k \in K, \tag{5.21}$$

$$\sum_{i\in V} x_{i,t^k}^k = 1, \qquad \forall k \in K, \tag{5.22}$$

$$\tag{5.23}$$

Este problema no es más que un modelo de flujo de red múltiservicios que no considera las dos familias de restricciones técnicas que se requieren para una solución factible. Así, al resolver 5.18-, pueden obtenerse soluciones que verifican algunas de las restricciones (5.8) y/o (5.9). Para separar las restricciones violadas, se aplica un procedimiento enumerativo. En caso de que una de las restricciones no se verifique, se agrega al grupo de restricciones y el problema se resuelve nuevamente (eliminando la solución obtenida previamente).

Más concretamente, para una solución dada del problema (5.18)-(5.3.1), $\overline{x}$ se verifica si viola (5.8) y/o (5.9) de la siguiente manera:

1. La violación de las restricciones (5.8) se comprueba examinando la solución, si una arista de un servicio se encuentra demasiado cerca de la arista de otro servicio, se añade una nueva restricción en el grupo de restricciones del problema principal.
2. Para verificar la violación de cualquiera de las restricciones (5.9) se examina la solución servicio a servicio. En caso de encontrar dos aristas virtuales demasiado cerca entre sí dentro de un mismo servicio, se añade una nueva restricción en el grupo de restricciones del problema principal.

5.3.2 Heurística

El modelo de trazado de tuberías (PTTB) presentado en la Sección 5.3 requiere resolver un problema de programación de enteros, que en situaciones del mundo real, incluso el método de ramificación y corte no es capaz de cargar y resolver debido a la gran cantidad de variables y restricciones que tiene. En esta sección se proporciona un algoritmos basado en reducir la dimensión del modelo de programación entera.

Una instancia del problema de trazado de tuberías es un ortoedro donde se han eliminado algunos ortoedros (obstáculos) más pequeños. Hay zonas de dicha región que rara vez son ocupadas por las rutas de las soluciones (por todos o algunos de los servicios). Se plantea un modelo que hace uso de esto para reducir su número de variables. Se propone una estrategia de recorte para las instancias.

El procedimiento propuesto es iterativo y se describe en el pseudocódigo del algoritmo 7. El algoritmo comienza resolviendo el problema de la ruta más corta para cada servicio y determina la mejor solución para cada servicio independientemente de los servicios restantes. Luego, en lugar de resolver (PTTB) en todo el grafo, se resuelve el problema en la unión de tubos de sección cuadrada creada al agrandar las trayectorias obtenidas en una distancia desde la arista hasta el punto medio de la sección cuadrada inicial dada δ_k (no declarando las variables que indican arcos fuera de las regiones cilíndricas). En caso de que el problema sea factible, proporciona una solución de (PTTB). De lo contrario, se aumenta δ_k y se repite el procedimiento hasta que se obtiene la viabilidad. Aunque en el peor de los casos el algoritmo requiere resolver la instancia original del problema (el grafo completo), en la práctica, permite resolver el problema con un número considerablemente menor de variables.

distancia inicial para trazar los servicios alrededor de la solución del problema de su ruta más corta. $\delta_k \leftarrow R^k + \Delta^k + \delta$ para $k \in K$
$it \leftarrow 1$
$stop \leftarrow 0$
para $k \in K$ **hacer**
  Resuelva el problema del camino más corto para el servicio k en todo el grafo G : $\{camino_k\}_{k \in K}$
fin
mientras $stop = 0$ **hacer**
  Resuelve el (PTTB)(it), donde todas las variables x_{ij}^k para un servicio k dado que están fuera del tubo alrededor de la ruta k y el radio k se fijan a cero.
  si *(PTTB)(it) es factible* **entonces**
    $stop = 1$
  en otro caso
    Aumentar δ_k para todo $k \in K$
  fin
  $it = it + 1$
fin

Algoritmo 7: Reducción del número de variables.

5.4 Experiencia computacional (provisional, habría que cambiarlo)

A continuación, se exponen los resultados de algunos experimentos computacionales realizados, con el fin de comparar empíricamente los métodos exacto y heurístico propuestos.

5.4.1 Instancias aleatorias

Las instancias del grafo $G = (V, E)$ se generan de la siguiente manera. Se genera un cubo de longitud de arista igual a 128 unidades que contenga un mallado de 17×17×17 nodos reales ($d = 17$). Se generan cinco grupos diferentes de 15 servicios cada uno con un mismo radio de 4 unidades y estableciendo una distancia de seguridad entre servicios de 1 unidad. Para cada servicio, se ha generado aleatoriamente un origen y un destino en los nodos de la cuadrícula con $d = 17$ pertenecientes a los planos $z = 0$

(para origen) y $z = 17$ (para destino). De esta forma, los servicios atraviesan el cubo iendo de una cara a la opuesta. De forma análoga, se generan cinco grupos diferentes de 15 obstáculos cada uno con forma de cubo y longitud de borde $l = 10$ unidades. Cada obstáculo se coloca dentro de un cubo interior de longitud de borde igual a 115 unidades de longitud situado en el centro de la instancia. Se deja una zona ”exterior” sin obstáculos que evita que se bloqueen orígenes y destinos y asegura la viabilidad de los caminos. Se denota por (d, s, o, g) la instancia de densidad $d \in \{16, 32\}$, servicios $\{1, 2, \dots, s\}$ con $s \in \{5, 8, 12\}$, obstáculos $\{1, 2, \dots, o\}$ con $o \in \{5, 10, 15\}$ y tomando tanto los servicios como los obstáculos del grupo $g \in \{1, 2, 3, 4, 5\}$. Por lo tanto, se generan 90 instancias diferentes.

Todas las instancias se resolvieron con el solver Gurobi 7.7, en un entorno Windows 10 en un procesador Intel(R) Core(TM) i7 CPU de 2,93 GHz y 16 GB de RAM. Los valores predeterminados se utilizaron inicialmente para todos los parámetros del solucionador de Gurobi y se estableció un límite de tiempo de CPU de 7200 segundos. También se han probado diferentes combinaciones de parámetros para la estrategia de corte del solucionador y la intensidad de la heurística pero, a menos que se especifique, los mejores resultados se obtuvieron con los parámetros del solucionador establecidos con los valores predeterminados.

En la tabla, se reportan resultados correspondientes a grupos de 5 instancias con el mismo triplete (d, s, o) de parámetros así como resultados promedio para cada grupo.

La tabla está agrupada en bloques. Las tres primeras columnas de cada bloque contienen los valores de los parámetros de las instancias. Cada bloque tiene unas 9 columnas. Las columnas de cada bloque son las siguientes:

1. La columna s indican el número de servicios.
2. La columna o indican el número de obstáculos.
3. La columna $Vars$ indican el número de variables.
4. La columna $Cons$ indican el número de restricciones.
5. La columna $Solved$ indican el número de instancias resueltas.
6. La columna GAP indica la distancia relativa entre la mejor cota superior conocida obj_U y el valor óptimo de la relajación lineal en el nodo raíz obj_R. Se calcula como $100\frac{obj_U - obj_R}{obj_U}$.
7. La columna $Time$ indica el tiempo de resolución.
8. La columna $GAP - Heur$ indica la diferencia de GAP entre la resolución del modelo heurístico y el modelo exacto.

s	o	Vars	Cons	Solved	GAP	Time	GAP-Heur
5	5	228232,0	73572,0	5	0	11,15	0,10
	10	227420,8	73386,0	5	0	11,51	0,32
	15	226736,0	73230,0	5	0	12,21	1,70
5		227462,9	73396,0	15	0	11,62	0,63
8	5	399406,0	117760,8	5	0	400,17	0,31
	10	397986,4	117463,2	5	0	339,85	1,96
	15	396788,0	117213,6	5	0	347,41	5,86
8		398060,1	117479,2	15	0	362,48	2,49
12	5	627638,0	176735,2	5	0	4216,11	0,18
	10	625407,2	176288,8	5	0	3367,94	3,02
	15	623524,0	175914,4	5	0	2596,96	6,76
12		625523,1	176312,8	15	0	3450,58	3,29
		417015,4	122396,0	45	0	1225,45	2,02

Cuadro 5.1: Resltado comptacional de las instancias aleatorias.

Un primer análisis de los resultados muestra que el algoritmo exacto resuelve todas las instancias con 5 y 8 servicios con cualquier número de obstáculos, mientras que para $s = 12$ servicios hay 4 instancias que no se pudieron resolver dentro del límite de tiempo.

Los resultados correspondientes al método heurístico son bastante similares. Sin embargo, hay 4 instancias donde el algoritmo exacto no encuentra soluciones factibles pero la heurística sí y al revés, en 5 instancias el algoritmo heurístico no encuentra soluciones pero el exacta sí. La distancia entre las soluciones exacta y heurística $(100 \cdot (HeuristicoExacto)/Exacto)$ es muy pequeña: para instancias con $s = 5$ servicios siempre está por debajo del 0,7 % , con $s = 8$ servicios es menos del 2,5 % y para aquellas instancias que se resuelven con $s = 12$ servicios, esta brecha es inferior al 3.3 %.

5.4.2 Caso de estudio

La Figura 5.6 muestra un ejemplo realista de un problema de trazado de canalizaciones. Se trata de la cubierta de un buque, cuyos datos fueron facilitados por la empresa GHENOVA. Consiste en diferentes compartimentos en los que se sitúan las fuentes y sumideros de 10 servicios, separados entre sí por paredes y comunicados por agujeros

en estas paredes. Las dimensiones del escenarios son las de un prisma rectangular con 5732 unidades de largo (eje X), 2836 de alto (eje Y) y 2013 de ancho (eje Z). Describe una cabina de barco donde se deben trazar la traectoria de 10 servicios diferentes desde sus orígenes a los destinos, considerando una serie de obstáculos dados por 5 paredes que pueden atravesarse sólo a través de 8 de ventanas (agujeros en las paredes). Las paredes son planchas de metal de 10 unidades de grosor. La distancia mínima entre codos para todos los servicios es de 50 unidades, los radios de revestimientos de los servicios son de 50, 75 y 100 unidades; y la distancia de seguridad entre servicios es de 50 unidades (ver figura 5.6). El coste asociado a cualquier arista (i, j) del grafo viene dado por la siguiente expresión, en la cual, los parámetros tienen el mismo valor para todos los servicios:

$$C_{ij} = (\alpha_1 + \alpha_5)d_{ij} + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_6 + \alpha_7 \tag{5.24}$$

donde:

- d_{ij} distancia entre el nodo i y el nodo j (longitud de la arista).
- α_1 peso asociado a la longitud de la arista (ponderación relativa de la longitud).
- α_2 coste de un codo de 90º.
- α_3 penalización para alejar el trazado del techo (su valor será mayor cuanto más lejos esté del techo).
- α_4 penalización por cambiar el trazado en altura.
- α_5 bonificación por trazar en zona preferente ($\alpha_5 < 0;\ \alpha_1 + \alpha_5 \leq 0$).
- α_6 coste por entrar en una zona penetrable.
- α_7 penalización por poner un codo cerca de un punto de entrada o salida.

Los valores de estos parámetros aparecen en la siguiente tabla:

α_1	α_2	α_3	α_4	α_5	α_6	α_7
1	2800	700	200	-0.7	4000	3000

Las zonas preferentes en las que se aplican α_5 son los prismas con pares de vértices

$$(154241, 535, 24249; 158844, 1419, 25221) \tag{5.25}$$

y

$$(153013, 535, 24249; 154241, 3371, 25221). \tag{5.26}$$

Para ilustrar cómo la función de costes (5.24) tasa las aristas del grafo se recurre al subgrafo de la Figura 5.5, supóngase las siguientes hipótesis:

- El plano 1 está más cerca del techo que el plano 2.
- El plano 1 pasa por una zona preferente y el 2 no.
- El nodo situado en el plano 2 se encuentra cerca de un punto de entrada o salida (el del plano 1 no).
- La arista negra del plano 2 atraviesa una zona penetrable.

Bajo estas hipótesis el coste asociado a las diferentes aristas es el siguiente:

- Coste de las aristas dentro de los nodos explotados:
  - Las del nodo explotado del plano 1 tienen coste α_2.
  - Las del nodo explotado del plano 2 tienen coste $\alpha_2 + \alpha_7$.
- Coste de las aristas fuera de los nodos explotados:
  1. Las aristas de color azul tienen coste $a_1 d_{ij} + \alpha_4$.
  2. Las aristas rojas del plano 1 tienen coste $(\alpha_1 + \alpha_5) d_{ij} + \alpha_3$.
  3. Las aristas rojas del plano 2 tienen coste $\alpha_1 d_{ij} + \alpha_3^{'}$ con $\alpha_3^{'} > \alpha_3$.
  4. La arista negra del plano 2 tiene coste $\alpha_1 d_{ij} + \alpha_3^{'} + \alpha_6$.

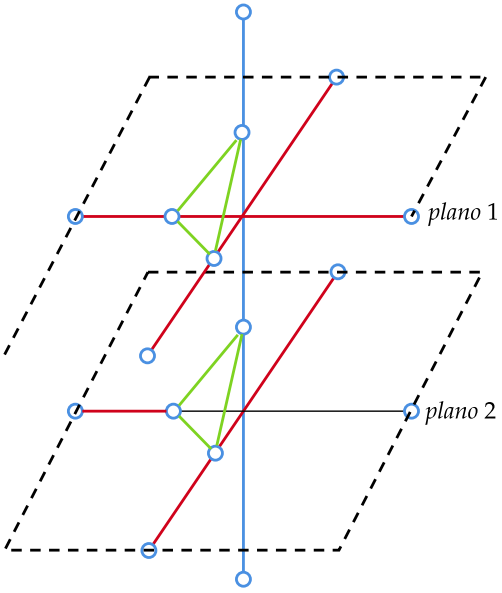

Figura 5.5: Grafo con nodos explotados.

Este escenario 5.6 fue resuelto con el algoritmo heurístico descrito en la Sección 5.3.2 en 9617.91 segundos de tiempo de CPU, la solución obtenida tiene un valor de

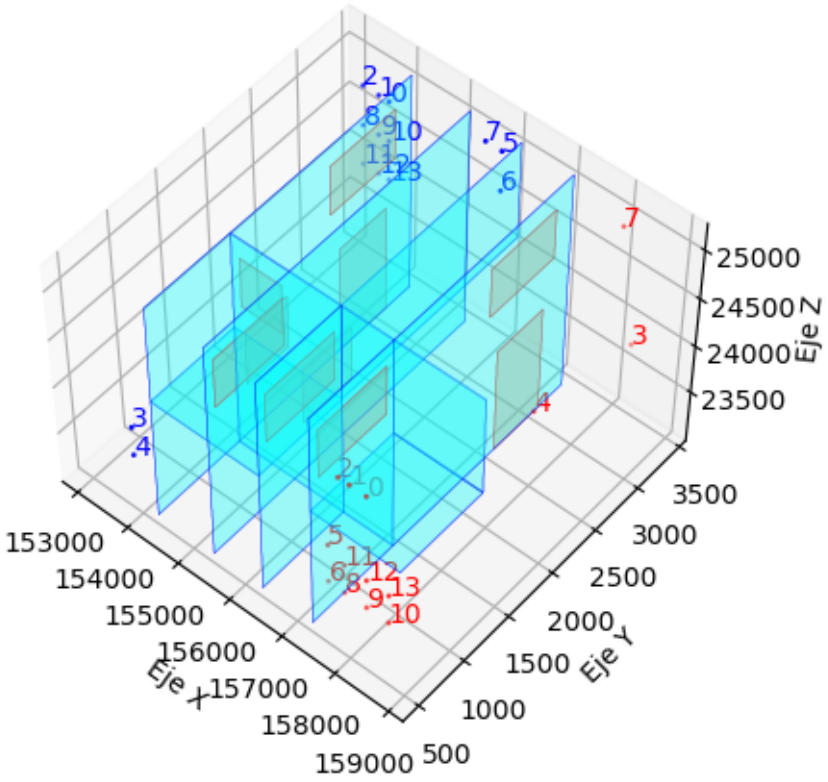

Figura 5.6: Escenario del caso de estudio.

454872.50 y se muestra en la Figura 5.7 (derecha). El solver exploró 9569 nodos y en el algoritmo branch-and-cut se utilizaron 19 restricciones perezosas de test de codos, 35422 restricciones perezosas de distancia entre servicios. La misma solución integrada dentro de los obstáculos que se muestra en la Figura 5.7 (izquierda). La Figura 5.8 muestra la misma solución con tuberías reales con sus diámetros realistas. Como antes, la figura de la derecha muestra tuberías sin obstáculos y a la izquierda con los obstáculos.

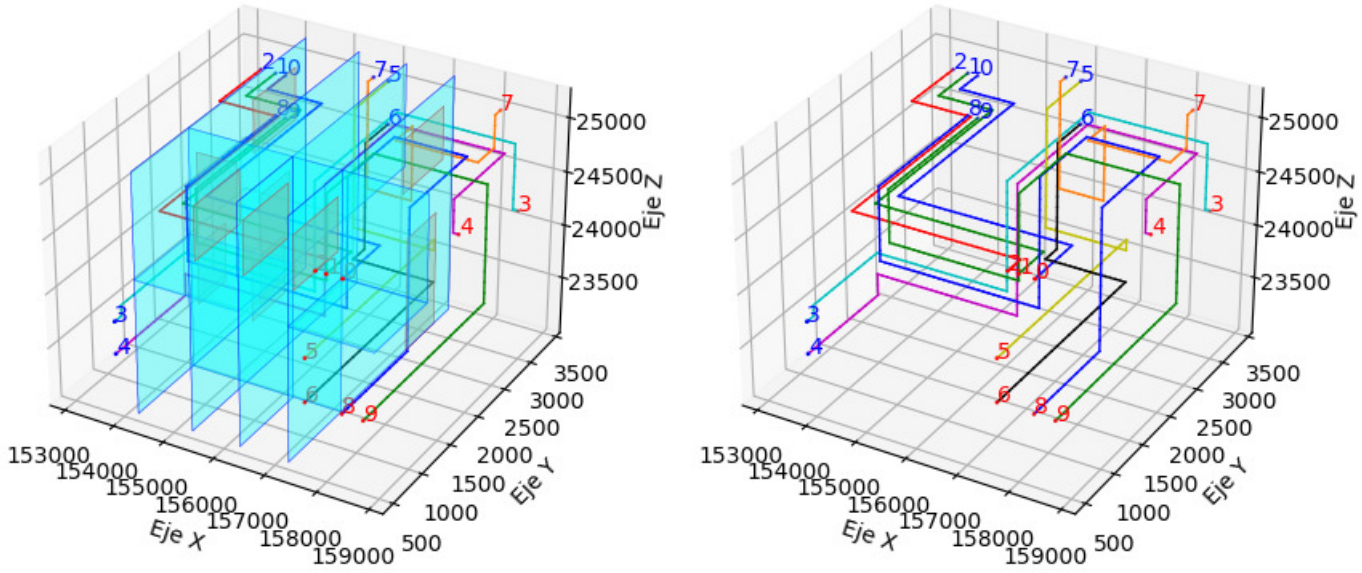

Figura 5.7: Representación gráfica de la solución para el escenario 5.4 de Ghenova con 10 servicios. A la derecha se muestra la solución sin mostrar los obstáculos. La figura de la izquierda muestra la misma solución integrada dentro del conjunto de obstáculos del escenario.

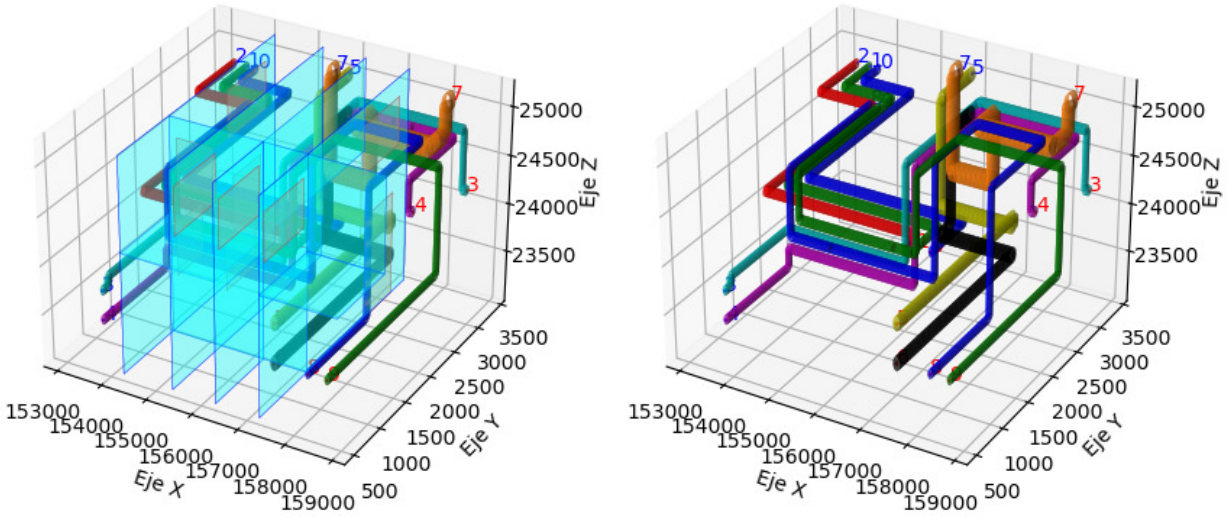

Figura 5.8: Representación gráfica de las tuberías resultantes al aplicar la solución de la Figura 5.7 para el escenario Ghenova con 10 servicios. A la derecha se muestra la solución sin mostrar los obstáculos. La figura de la izquierda muestra la misma solución integrada dentro del conjunto de obstáculos del escenario.

6 | Conclusión

En este trabajo se ha sintetizado en la medida de lo posible alguna de las principales herramientas disponibles para enfrentar los problemas de flujo multiservicio. También se han aplicado algunas de ellas para modelizar y resolver el trazado de tuberías y cables en barcos minimizando costes. Para ello se ha dispuesto de información de escenarios reales, esto es, partes de barcos; gracias a la colaboración de la empresa GHENOVA. Otras herramienas, sin embargo, se han estudiado en vistas a posteriores trabajos.

En primera instancia, se ha partido tratando los algoritmos de ramificación y corte (Sección 2.1), pues es, herramienta ineludible para poder introducir los requisitos de diseño propios del diseño naval. Estos requisitos son, el test de codos (5.9) y la distancia entre servicios y obstáculos o test de revestimiento (5.8). Debido a las enormes dimensiones de los escenarios con los que irremediablemente se trabaja en este tipo de problemas, existen una cantidad colosal de estas restricciones y la forma de obtener una solución que cumpla estas restricciones es mediante la ramificación y corte.

Seguidamente se ha considerado la relajación lagrangiana (Sección 2.2) y la generación de columnas (Sección 2.3). Se considera la relajación lagrangiana como una forma de relajar los problemas eliminando alguna de sus restricciones y aplicando un peaje en la función objetivo. Y la generación de columnas como una metodología para lidiar con problemas con un enorme número de variables, en la que se relajan sucesivos problemas y se compara su resultado con el de la relajación lagrangiana. Ambas dos herramientas son fundamentales en el estudio de los problemas de flujo multiservicio (Capítulo 4).

Tras esto, se lleva a cabo un estudio del problema de flujo de mínimo coste mediante el algoritmo simplex en redes (capítulo 3). El problema de flujo de mínimo coste tiene siempre al menos una solución óptima de árbol de expansión y el algoritmo simplex en redes explota esta propiedad del problema recorriendo soluciones de ár-

bol de expansión, sustituyendo en cada paso un arco del árbol por otro que no lo es. La comprensión de los problemas de flujo de mínimo coste es necesario para, en el siguiente capítulo, trabajar en el problema de flujo multiservicio.

En conclusión de una parte más teórica de este trabajo, se estudia el problema de flujo multiservicio en el que varios servicios comparten una misma red y compiten por el espacio en ella para minimizar el coste total de los servicios (capítulo 4). Aquí hace uso de las herramientas anteriormente consideradas en el trabajo.

Como aplicación práctica, en el siguiente capítulo (el 5) se modeliza el trazado de servicios con mínimo coste en el contexto del diseño naval. Se elabora un modelo que resuelve a optimalidad el problema y un heurístico necesario para escenarios grandes (todo escenario real). Se hace un estudio computacional de los resultados del modelo exacto y el algoritmo heurístico, así como se resuelve mediante el heurístico desarrollado un escenario real proporcionado por la empresa GHENOVA.

En posteriores trabajos, se desea aplicar los resultados teóricos del Capítulo 4 en la resolución del problema descrito en el Capítulo 5. También se desea resolver otros problemas en diseño naval relacionados con este en colaboración con GHENOVA.

Bibliografía

[1] C. A. Floudas, P. M. P. E. *Encyclopedia of Optimization.* Springer, 2009.

[2] Even, S., I. A., and Shamir, A. On the complexity of time table and multi-commodity flow problems. *16th Annual Symposium on Foundations of Computer Science* (1975), 184–193.

[3] R. K. Ahuja, T. L. Magnanti, J. B. O. *Network Flows: Theory, Algorithms and applications.* Prentice hall, 1993.

[4] Wolsey, L. A. *Integer Programming.* Wiley-Interscience, 1998.