Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Diseño de redes de comunicación

Autor: Jorge Gutiérrez de la Asunción

Tutores: Jesús Sáez Aguado
Pedro César Álvarez Esteban

Curso: 2024-2025

Índice general

Resumen

El objetivo principal de este trabajo es el estudio y modelación matemática de diversos problemas de diseño óptimo de redes.
Primero se introducirá un tipo de problema de optimización diferente como son los *problemas de flujo en redes*, que nos servirán de gran ayuda para los problemas de diseño de redes que estudiaremos posteriormente. Se tratará la evolución histórico-científica de las distintas topologías de redes, que se han ido adaptando a las diversas necesidades operativas y tecnológicas de cada momento. Partiremos de un problema central, **el Problema Conector** o también llamado Problema del Mínimo Árbol Expandido (*Minimal Spanning Tree, MST*), que se trata de conectar mediante enlaces los distintos puntos de una red (de ordenadores, telefónica, etc) de manera óptima. A continuación, mostraremos distintas extensiones y especificaciones más restrictivas de este problema: el *MST* con restricciones de capacidades (*CMST*) o restricciones de salto (*HMST*). Estos problemas se presentarán como problemas teóricos de optimización de redes, en los que se mostrarán algoritmos y formulaciones matemáticas basados en programación lineal y entera, que nos permitirán modelar matemáticamente y estudiar cada problema. El trabajo finalizará con un capítulo dedicado a la implementación práctica de las distintas formulaciones y algoritmos, en las que sacaremos algunas observaciones y conclusiones.

ABSTRACT

The main objective of this thesis is the study and mathematical modeling of various optimal network design problems.
We will begin with a historical and scientific overview of the different network topologies, which have evolved over time to meet changing operational and technological needs. The work will be centered around a core problem, the **Connector Problem**, also known as the *Minimal Spanning Tree Problem (MST)*, which consists of optimally connecting the nodes of a network (such as computer or telephone networks) through links.
We will then present several extensions and more restrictive variants of this problem, such as the *Capacitated Minimal Spanning Tree (CMST)* and the *Hop-Constrained Minimal Spanning Tree (HMST)*. These problems will be approached as theoretical network optimization problems, and we will introduce algorithms and mathematical formulations based on linear and integer programming that allow us to model and

analyze each case.
The thesis concludes with a chapter dedicated to the practical implementation of the different formulations and algorithms, from which several observations and conclusions will be drawn.

Introducción

Los problemas de redes aparecen en nuestro día a día en múltiples ámbitos y combinan profundos conocimientos matemáticos con una amplia gama de aplicaciones prácticas. Estos problemas cubren miles de aplicaciones en múltiples campos como en la química, física, múltiples sectores de ingeniería, redes de ordenadores, telecomunicaciones, transporte y también en ámbitos sociales, como programar rutinas o horarios. Por ejemplo, una red eléctrica que quiere brindar servicio a un vecindario, una red de trenes que quiere comunicar una serie de ciudades, etc.

En toda esta cantidad de problemas deseamos transportar o mover una cierta entidad (en nuestros ejemplos la electricidad o los trenes) de un punto a otro de nuestra red de la manera más eficiente posible. En ambos casos, puede ser para brindar un buen servicio a los clientes o también para usar las transmisiones entre punto y punto que nos da nuestra red eficientemente, ya que normalmente el costo del uso de estas transmisiones es caro. En una visión muy general, es de lo que va a tratar este trabajo. Vamos a aprender cómo modelar estos problemas matemáticamente mediante formulaciones basadas en programación lineal y entera y estudiaremos varios algoritmos para tratar de resolverlos.

En la mayor parte de este texto vamos a tratar dos tipos de problemas:

- *Problemas de flujo a mínimo coste:* Si contamos con un coste por unidad transportada de un punto i a un punto j de nuestra red, la pregunta es, ¿Cómo podemos enviar cada unidad con el mínimo coste posible?

Y, sobre todo, nos centraremos en:

- *Problemas de Diseño de Redes*: Como son el Problema del Mínimo árbol expandido (Minimum spanning tree, MST) y varias extensiones de él más restrictivas. Son casos en los que tengamos una red donde desde cada punto i se puede ir a cualquier otro punto j de nuestra red, y la pregunta en este caso es, ¿Cuál es el camino más corto que conecta todos esos puntos?

Un ejemplo que puede ser ilustrativo para entender esta estructura de problemas de flujo en redes es la que presenta el siguiente diagrama representado en la Figura 1, donde tenemos una red de carreteras que conecta las diferentes capitales de provincia de Castilla y León y el número encima de cada conexión es la cantidad en km que hay entre esas dos ciudades.

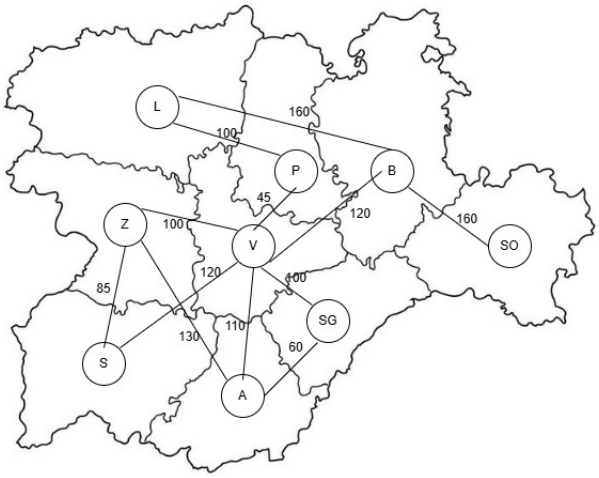


Figura 1: Ejemplo Ilustrativo. Elaboración propia

Un primer problema puede ser el siguiente:

Supongamos que tenemos una empresa de camiones y queremos transportar un producto entre las diferentes ciudades de Castilla y León. Tenemos que transportar desde las ciudades donde tenemos los almacenes (que serán mis puntos de oferta) hasta el resto de ciudades donde tengamos demanda. También, a cada camino entre dos ciudades se le asocia un coste por unidad de producto transportada, que dependerá en este caso de los km recorridos. Entonces, este tipo de problema es un *problema de flujo de coste mínimo* , ya que se trata de ver la cantidad de producto (flujo) transportado en cada camino entre dos ciudades satisfaciendo las demandas a un coste mínimo.

Otro tipo de problema que podemos tener sería:

Supongamos que queremos encontrar una red de carreteras que conecte todas las ciudades de la manera más corta, por cualquier motivo (un repartidor tiene que pasar por todas las ciudades en una semana,...). Esto sería un problema del *mínimo árbol expandido*, ya que necesitamos un camino que conecte todas las ciudades con distancia mínima.
En los capítulos siguientes trataremos estos dos problemas con sus múltiples variantes y adaptaciones y veremos algunos algoritmos y formulaciones para resolverlos.

Capítulo 1

Definiciones previas

Antes de todo necesitamos introducir conceptos y definiciones de la herramienta matemática fundamental que vamos a usar en todos estos problemas, que son los grafos o redes. En nuestro caso no diferenciaremos entre redes y grafos, ya que, para entendernos, los grafos son la herramienta matemática que utilizamos para describir redes de todo tipo (red de trenes, ordenadores, tuberías...). Introducimos las definiciones y conceptos necesarios para los temas que abordaremos en los siguientes capítulos. Para mayores detalles sobre teoría de grafos véase en [19] o [2].
Lo primero de todo vemos la definición general de grafo.

Definición 1.0.1. *Un **grafo** G consiste en un conjunto de puntos y en un conjunto de líneas que unen un par de puntos. Matemáticamente es un par $G = (N, A)$ donde N es el conjunto finito de puntos, llamados nodos o vértices, y A es el conjunto de aristas o arcos que unen dos nodos; estos se representan con el par (i, j) y se dice que esta arista une el nodo i y el nodo j.*

Las aristas de una red pueden estar diseñadas para permitir la transmisión de algún flujo, en nuestro ejemplo, los nodos serían las diferentes ciudades y las aristas las carreteras que conectan las ciudades. En ocasiones, estas aristas pueden estar diseñadas para permitir el flujo solo en una dirección. Cuando una arista solo permite el flujo en una dirección, se dice que la **arista es dirigida** y si el flujo se permite en ambas direcciones, se dice que la **arista no es dirigida**. En estos casos, tenemos la siguiente definición.

Definición 1.0.2. *Un grafo G se dice que es **dirigido**, si todos sus arcos son dirigidos. En caso contrario, es un grafo **no dirigido**. También a estos grafos se les conoce como **dígrafos**.*

Normalmente cuando tenemos un dígrafo a las aristas las llamamos arcos. En el caso de tener un grafo dirigido, cada arco $a = (i, j)$ es un par ordenado, es decir, representa que el flujo a través de ese arco es del nodo i al nodo j. También, toda red

se puede considerar dirigida, ya que en cada arco no dirigido podemos considerar dos arcos dirigidos, uno en cada sentido. En casi todo este trabajo vamos a trabajar con grafos dirigidos.
También vamos a introducir otros tipos de grafos, los grafos con pesos.

Definición 1.0.3. ***Los Grafos con pesos*** *son aquellos a los que a cada arista* (i, j) *se les asocia un valor real* w_{ij} *llamado peso.*

A un grafo $G = (N, A)$ se le puede considerar como un grafo con peso, con peso asociado igual a 1 para cualquier arista.
A continuación vemos más definiciones relativas a grafos y dígrafos.

Definición 1.0.4. *Se dice que dos nodos* i, j *de un grafo* $G = (N, A)$ *son* ***adyacentes*** *si existe el arco* (i, j)*, es decir,* $(i, j) \in A$*. Denotamos por* $i \longleftrightarrow j$ *que "i y j son adyacentes".*
El ***vecindario*** *(entorno) de un nodo* i *es* $N(i) = \{j \in N : (i, j) \in A\}$*, denotamos por* $N_G(i)$ *si se necesita precisar el grafo. Un nodo* i *se dice que es aislado si* $N(i) = \emptyset$*. En el caso de los dígrafos se denota por* $i \to j$ *si* $(i, j) \in A$*. Y entonces podemos definir el* ***vecindario de salida*** *como* $N_G(i)^+$ *como* $\{j \in N : i \to j\}$ *y el* ***vecindario de entrada*** $N_G(i)^-$ *como* $\{j \in N : j \to i\}$*.*

Definición 1.0.5 (Grados de un nodo)**.** *Sea* $G = (N, A)$ *un grafo dirigido, luego denotamos por* ***grado de entrada*** $d(i)^-$ *de un nodo* i *al número de arcos que acaban en el nodo* i*. De la misma manera, denotamos* ***grado de salida*** $d(i)^+$ *de un nodo* i *al número de arcos que salen del nodo* i*. Y por tanto, el grado de un nodo es la suma de estos dos valores.*

Nótese que en un grafo no dirigido no distinguimos entre grado de entrada y de salida, solo consideramos el número de aristas que confluyen en ese nodo.

Definición 1.0.6. *Decimos que* $G' = (N', A')$ *es un* ***subgrafo*** *de* G *si* $N' \subseteq N$ *y* $A' \subseteq A$*. Y decimos que es un* ***subgrafo expandido*** *si* $N' = N$ *y* $A' \subseteq A$*.*

Definición 1.0.7. *Un* ***camino*** *en un grafo* G *es un subgrafo* $G' = (N', A')$ *en el que* N' *es una sucesión de nodos adyacentes. Es decir, si los nodos de* N' *los llamamos como* $i_1, i_2, ..., i_r$ *los arcos del camino son de esta forma* $\{(i_k, i_{k+1})\}_{k=1}^{r-1}$ *donde cada* $(i_k, i_{k+1}) \in A$*. Un* ***camino simple*** *es un camino en el que no se repite ningún nodo. Y un camino se dice que es* ***maximal*** *en* G *si no es subgrafo de ningún otro camino.*

Los caminos los podemos denotar también de la siguiente manera:

$$P = i_1 - i_2 - \cdots\cdots - i_{r-1} - i_r$$

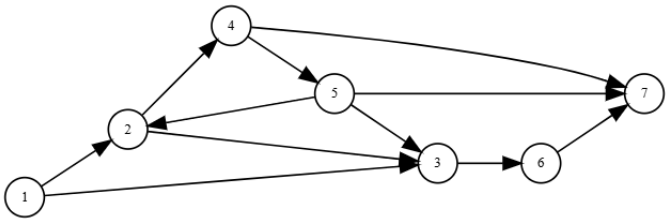


Figura 1.1: Ejemplo caminos y ciclos. Elaboración propia

donde $(i_k, i_{k+1}) \in A$ para $k = 1, ..., r-1$. Además, si consideramos que el grafo es dirigido, se considera siempre que el flujo en el camino va desde el nodo i_k al nodo i_{k+1}, esto es que $i_k \to i_s$ si y solo si $s = k+1$.

Definición 1.0.8. *Un **ciclo** es un camino cerrado, es decir, que el nodo inicial es igual que el nodo final.*

Sería un camino con la forma siguiente $C = i_1 - i_2 - \cdots\cdots - i_{r-1} - i_1$, donde siempre $i_1 = i_r$
Consideramos el grafo $G = (N, A)$ con $N = \{1, 2, 3, 4, 5, 6, 7\}$ y
$A = \{(1,2), (1,3), (2,3), (2,4), (3,6), (4,5), (4,7), (5,2), (5,3), (5,7), (6,7)\}$

Por ejemplo $P_1 = 1 - 2 - 3 - 6 - 7$ representa un camino en el grafo G. Un camino maximal siempre sería el camino al que no podemos añadirle más arcos sin repetir nodos, en particular tendríamos un camino maximal si tenemos una sucesión de arcos que pasan por todos los nodos. Como sería, por ejemplo, el camino $P_2 = 1 - 2 - 4 - 5 - 3 - 6 - 7$. Además, un ejemplo de ciclo en este grafo sería $C = 2 - 4 - 5 - 2$.

Definición 1.0.9. *Se dice que dos nodos $i, j \in N$ de un grafo están **conectados** si existe al menos un camino del nodo i al nodo j. Se dice que el grafo $G = (N, A)$ es **conexo** si cada par de nodos del grafo están conectados por un camino. Y se dice no conexo en caso contrario.*

Entonces si un grafo $G = (N, A)$ es no conexo, los subgrafos maximales conexos de G se llaman **componentes** de G. Quiere decir que las componentes de G son los subgrafos suyos con mayor número de nodos que están conectados entre sí.
Se considera el subgrafo $G-(i,j)$ al grafo generado a partir de $G = (N, A)$ al eliminar de A la arista (i, j). Del mismo modo, se obtiene el subgrafo $G - i$ al eliminar el nodo i.

Definición 1.0.10. *Una **arista de corte o puente** es una arista (i, j) cuya eliminación genera un subgrafo con más componentes.*

Se puede caracterizar las aristas de corte con ciclos.

Proposición 1.0.1. *Una arista es una arista de corte si y solo si no pertenece a ningún ciclo.*

Demostración. Consideramos una arista (i, j) del grafo G. Si i y j están en la misma componente en el grafo G y $G-(i, j)$ es porque G y $G-(i, j)$ tienen el mismo número de componentes, entonces esto significa que (i, j) no es una arista de corte. Por tanto, son equivalentes las siguientes afirmaciones (1) (i, j) es una arista de corte, (2) i y j están en distintas componentes en G y en $G-(i, j)$, (3) no existe un camino entre i y j en $G-(i, j)$ y (4) no existe un ciclo en G que contenga la arista (i, j). La equivalencia entre (1),(2) y (3) es clara, la equivalencia con (4) viene del hecho, que si existiese un ciclo $C = i_0 - i_1 - \cdots - i - j - \cdots - i_0$ que contiene la arista (i, j) podríamos generar un camino entre i y j que no contiene la arista (i, j), que es el siguiente $P = j - \cdots - i_0 - i_1 - \cdots - i$ y de esta manera existe un camino entre i y j en $G-(i, j)$ □

Además, es claro que una arista de corte solo puede disminuir las componentes en 1 unidad.

Definición 1.0.11. *Un **árbol** es un grafo conexo y sin ciclos.*

El concepto de árbol va a ser fundamental en los problemas de *MST* y en los diferentes algoritmos que vamos a introducir. Para ello, vamos a ver unas propiedades que caracterizan los árboles y nos ayudarán a identificar si un grafo es un árbol o no.

Definición 1.0.12. *Un **bosque** es un grafo sin ciclos*

Cada una de las componentes de un bosque son árboles. Pues son subgrafos que son conexos y no contienen ciclos.

Teorema 1.0.1 (Caracterización de los árboles)**.**
Sea un grafo $G = (N, A)$ entonces:

1. *Si G es un árbol con $|N| \geq 2$ este contiene al menos dos nodos hoja, esto es, nodos con grado 1.*
2. *Si G es un árbol con n nodos este contiene exactamente $n-1$ arcos.*
3. *Si G es un árbol cada par de nodos $i, j \in N$ están conectados por un único camino.*
4. *Si G es un árbol y añadimos un arco a A se genera un único ciclo.*

Demostración. 1) Consideramos para el árbol G un camino $P = i_1 - i_2 - \cdots - i_k$ maximal, entonces i_1 y i_k tienen grado 1, ya que como el camino es maximal no hay ningún arco (i_1, i) o (i_k, i) con $i \notin P$ luego si hay otro arco que involucre a i_1 o i_k se

generaría un ciclo, algo que no es posible ya que G es un árbol y por tanto no contiene ciclos.
2) Veámoslo por inducción sobre el número de nodos. Para $n = 1$ como no hay ciclos, el número de arcos es 0 luego es igual a $n - 1$. Ahora supongamos que para $n > 1$ cada árbol con n nodos tiene exactamente $n - 1$ arcos. Consideremos ahora un árbol con $n + 1$ nodos. Como $n > 1$ podemos usar el apartado *1* y sea j un nodo hoja. Veamos que en un árbol G si consideramos el subgrafo $G' = (N', A')$ obtenido al eliminar un nodo hoja y su único arco incidente, este es otro árbol. Bastaría con ver que G' es conexo, ya que si G no tenía ciclos al quitar un nodo sigue sin tenerlos. Sean $i, k \in N'$, existe un camino P que conecta i y k en G. Como los puntos interiores del camino tienen grado 2, estos no pueden ser igual a j, que era el nodo hoja que hemos eliminado. Luego P también es un camino de G'. Por tanto, G' es un árbol con n nodos entonces por la hipótesis de inducción este tiene $n - 1$ arcos, y entonces G claramente tiene exactamente n arcos, que era lo que queríamos probar.
3) Vamos a probarlo por reducción a lo absurdo. Supongamos que para dos nodos distintos $i, j \in N$ existen dos caminos distintos P_1 y P_2. Como estos caminos son distintos tiene que haber un nodo u donde se separen los caminos y un nodo w donde se vuelvan a juntar.

$$P_1 = i - i_1 - i_2 - \cdots - u - u_1 - u_2 - \cdots - w - \cdots - j$$

$$P_2 = i - i_1 - i_2 - \cdots - u - w_1 - w_2 - \cdots - w - \cdots - j$$

Consideramos el subcamino Q_1 en P_1 que va de u hasta j y el subcamino Q_2 en P_2 que va de u hasta j. Entonces juntando Q_1 con el camino inverso de Q_2 tenemos un ciclo, algo que es absurdo porque G era un árbol.
4) Sea (i, j) el arco que hemos añadido con $i, j \in N$. Como G es un árbol, tenemos que existe un único camino entre i y j; luego, añadiéndole a este camino el arco (i, j) que hemos añadido, generamos un ciclo. Además, es único por la unicidad del camino entre i y j.

□

De esta caracterización de los árboles podemos ver una caracterización de los bosques

Corolario 1.0.1. *Un bosque B con n nodos y k componentes tiene exactamente $n-k$ aristas*

Demostración. De la definición de bosque se tiene que cada componente del bosque es un árbol. Sean $n_1, n_2, ..., n_k$ el número de nodos de cada componente. Se tiene obviamente:

$$n_1 + n_2 + \cdots + n_k = n$$

Y que el número de aristas de cada componente, por ser un árbol es $n_i - 1$ con $i = 1, ..., k$. Luego:

$$(n_1 - 1) + (n_2 - 1) + \cdots + (n_k - 1) = n - k$$

□

Terminamos finalmente introduciendo la noción de árbol expandido.

Definición 1.0.13. *Un árbol T es un **árbol expandido** de un grafo G si T es un subgrafo expandido de G.*

Tenemos la situación de que si T es un árbol expandido del grafo $G = (N, A)$ con n el número de nodos. Entonces $T = (N, A')$ y por el Teorema 1.0.1, debe tener $n - 1$ arcos. En un árbol expandido llamamos a los arcos pertenecientes al árbol como los *arcos-árbol* y los demás son los arcos que no pertenecen al árbol.

Proposición 1.0.2. *Cada arista de un árbol es una arista de corte. Luego, en particular, si T es un árbol expandido de un grafo G y eliminamos cualquier arco del árbol se produce un grafo no conexo constituido por dos subárboles T_1 y T_2.*

Demostración. Como en un árbol no hay ciclos, en particular, ninguna arista pertenece a un ciclo, luego por la caracterización de las aristas de corte (Prop.1.0.1) todas las aristas de un árbol son aristas de corte.
Entonces, para un árbol expandido T como cualquier arista es una arista de corte, eliminarla me produce dos componentes porque T al ser un árbol es conexo (tiene una componente). Llamamos T_1 y T_2 a esas componentes, obviamente son conexas por definición y además, no tienen ciclos porque no los tenía T. Por tanto, T_1 y T_2 son árboles.

□

Como hemos visto, la eliminación de una arista me genera dos componentes y llamamos a esto **corte fundamental** de G con respecto al árbol T. Como hemos visto un árbol tiene exactamente $n - 1$ aristas, luego un árbol tiene $n - 1$ cortes fundamentales.
Sea $T = (N, A')$ un árbol expandido del grafo $G = (N, A)$. Consideramos el conjunto de arcos $A - A'$. Por el Teorema 1.0.1(4) sabemos que si al árbol T le añadimos cualquier arco de $A - A'$ se genera un único ciclo.

Definición 1.0.14. *Llamamos a cada ciclo que se genera al añadir cada arco de $A - A'$ en T como **ciclo fundamental** de G con respecto al árbol T.*

Nótese que en esta situación si G tiene $m-n+1$ arcos que no pertenecen al árbol, entonces G tiene $m-n+1$ ciclos fundamentales.
Finalmente, terminamos con este resultado.

Proposición 1.0.3. *Todo grafo conexo contiene un árbol expandido.*

Demostración. Sea $G = (N, A)$ grafo conexo. Como cada nodo por sí mismo es un árbol, entonces todo grafo conexo contiene al menos un árbol. Sea $T = (N', A')$ el árbol maximal (con mayor número de nodos) contenido en G.
Supongamos que $N' \neq N$, sea $v \in N - N'$ y $w \in N'$. Consideramos el camino que sabemos que existe entre v y w. De este camino se deduce que existe un arco a que va de un nodo de $N-N'$ a otro de N, podemos suponer que estos son v y w. Consideramos ahora T' el subgrafo de G obtenido de añadir a T el nodo $v \in N - N'$ y el arco a. Como T ya era conexo y a través de la arista que hemos añadido podemos conectar todos los nodos de T con v, se tiene que T' también es conexo. Además, todo ciclo de T' está también en T, ya que v solo está en un arco. Por tanto, T' no tiene ciclos y es un árbol. Pero esto contradice la hipótesis de que T era un árbol maximal, luego $N' = N$ y T es un árbol expandido. □

Capítulo 2

Problemas de flujo de coste mínimo

Es uno de los problemas principales dentro de los problemas de redes. El problema es sencillo de plantear: queremos minimizar el gasto de transporte de ciertos productos dentro de una red, satisfaciendo unas demandas en ciertos puntos a partir de suministros disponibles en otros puntos de la red.

Ahora presentaremos una formulación matemática de estos problemas y luego veremos algunas especificaciones que se pueden hacer y ejemplos significativos que se pueden formular como problemas de flujo de coste mínimo.

2.1. Problemas de flujo en redes (1 producto)

Sea $G = (N, A)$ un grafo dirigido donde $N = \{1, 2, ..., n\}$ es el conjunto de nodos y A el conjunto de arcos representados con el par (i, j) con $i, j \in N$, considerando $|A| = m$. En la red que describe el grafo G también consideramos los elementos siguientes.

Asociado a cada nodo i se tiene el valor b_i, que representa la demanda u oferta de cierto producto en el nodo i. Consideraremos que si $b_i > 0$, el nodo i es un **nodo oferta** con cantidad de oferta b_i (cantidad de producción, cantidad de inventario,...). Si, en cambio, $b_i < 0$ el nodo i se dice que es un **nodo demanda**, nuevamente con cantidad de demanda b_i. Y si $b_i = 0$ se dice que es un **nodo intermedio o de transbordo**. Consideramos para cada $i \in N$ los conjuntos antes definidos, vecindario de entrada $N(i)^-$ y vecindario de salida $N(i)^+$. Y asociado a cada arco $(i, j) \in A$ se tiene la variable de decisión $x_{ij} \geq 0$ que representa la cantidad de producto o flujo a través del arco (i, j) y el valor c_{ij}, que representa el coste por unidad de flujo/producto transportado por el arco $(i.j)$.

Entonces *el problema de flujo de coste mínimo* consiste en encontrar el valor de las variables de decisión x_{ij} que minimicen el coste, satisfaciendo todas las ofertas y demandas. La formulación matemática de los problemas de este tipo es la siguiente:

$$min \qquad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.1}$$

$$\text{sujeto a} \qquad \sum_{j\in N(i)^+} x_{ij} \quad - \quad \sum_{j\in N(i)^-} x_{ji} = b_i \quad , \qquad \forall i \in N \tag{2.2}$$

$$x_{ij} \geq 0 \qquad\qquad \forall (i,j) \in A \tag{2.3}$$

Este problema, por tanto, tiene una variable para cada arco y una restricción para cada nodo. Las restricciones dadas en (2.2) se denominan como *ecuaciones de conservación, ecuaciones de balance o ecuaciones de Kirchoff*. La expresión $\sum_{j\in N(i)^+} x_{ij} \quad - \sum_{j\in N(i)^-} x_{ji}$ representa el flujo neto de cada nodo i, ya que el primer sumatorio representa el flujo de salida y el segundo el flujo de entrada.
En la formulación que hemos presentado no hemos considerado ninguna cota superior en las variables x_{ij}. Pero en algunos modelos frecuentemente el arco (i,j) tiene una cota superior o capacidad $u_{ij} > 0$, que puede representar el máximo flujo que es capaz de soportar cierta conexión de nuestra red.
A estos modelos se les dice *con capacidades* y basta sustituir (2.3) por las restricciones:

$$0 \leq x_{ij} \leq u_{ij} \qquad\qquad \forall (i,j) \in A \tag{2.3'}$$

Podemos también representar este modelo matricialmente. Si consideramos que m es el número de arcos y n el número de nodos de nuestra red, podemos considerar $\mathbf{x} = (x_{ij})$ el vector de m filas de los flujos, $\mathbf{c} = (c_{ij})$ el vector de m columnas de los costes y $\mathbf{b} = (b_i)$ la demanda/oferta del nodo i. Entonces podemos formularlo como sigue:

$$min \qquad \mathbf{cx} \tag{2.4}$$

$$\text{sujeto a} \qquad M\mathbf{x} = \mathbf{b} \tag{2.5}$$

$$\mathbf{x} \geq 0 \tag{2.6}$$

Donde M es la matriz de coeficientes de las ecuaciones de balance. Para cada arco $(i,j) \in A$ al que le asociamos el flujo x_{ij}, aparece con un 1 para la restricción correspondiente al nodo i y aparece un -1 para la restricción correspondiente al nodo j. Si representamos con m_{ij} el vector columna asociado al arco (i,j) este vector sería

precisamente $a_{ij} = e_i - e_j$, siendo e_i el vector unitario de $\mathbb{R}^n$, que tiene un 1 en la posición i y el resto ceros. Esta matriz M se la conoce como **Matriz de incidencias nodos-arcos** o matriz de incidencias simplemente.

Ejemplo 2.1.1. Consideramos el grafo dirigido $G = (N, A)$ con conjunto de nodos $N = \{1, 2, 3, 4\}$ y de arcos $A = \{(1,2), (2,1), (2,3), (2,4), (3,1), (3,2), (4,1), (4,3)\}$.

Entonces en este caso tenemos las siguientes variables de flujo, los vectores de coste asociado a los arcos del grafo G, y también el vector de demandas/ofertas asociados a los nodos:

$$\boldsymbol{x} = \begin{pmatrix} x_{1,2} \\ x_{2,1} \\ x_{2,3} \\ x_{2,4} \\ x_{3,1} \\ x_{3,2} \\ x_{4,1} \\ x_{4,3} \end{pmatrix} \qquad \boldsymbol{c} = \begin{pmatrix} c_{1,2} \\ c_{2,1} \\ c_{2,3} \\ c_{2,4} \\ c_{3,1} \\ c_{3,2} \\ c_{4,1} \\ c_{4,3} \end{pmatrix}^t \qquad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

Y entonces tenemos el problema de flujo a mínimo coste siguiente:

$$min \qquad \boldsymbol{cx}$$

$$sujeto\ a \qquad M\boldsymbol{x} = \boldsymbol{b}, \quad x_{i,j} \geq 0, \quad (i,j) \in A$$

Donde en este caso la matriz M de incidencias de este grafo es la siguiente*:*

$M =$

	$(1,2)$	$(2,1)$	$(2,3)$	$(2,4)$	$(3,2)$	$(3,1)$	$(4,1)$	$(4,3)$
1	1	−1	0	0	0	−1	−1	0
2	−1	1	1	1	−1	0	0	0
3	0	0	−1	0	1	1	0	−1
4	0	0	0	−1	0	0	1	1

2.1.1. Propiedad de integridad

El modelo de flujo en redes tiene una propiedad importante que puede enunciarse en los siguientes términos: si las ofertas y demandas son números enteros, existe una solución óptima con todas las variables enteras. Esto implica dos cosas:

1. Aunque el problema incluya restricciones de integrabilidad, como $x_{ij} \geq 0$ y $x_{ij} \in \mathbb{Z}$, es posible obtener soluciones enteras resolviendo únicamente el problema lineal $x_{ij} \geq 0$, debido a la estructura del problema. Véase en [2](cap 9)

2. Esta propiedad se traduce en que las versiones especializadas del método simplex para redes, no necesitan efectuar divisiones, y resultan algoritmos mucho más rápidos y eficientes que el método simplex general para PL (Programación Lineal).

2.1.2. Redes balanceadas

Definición 2.1.1. *Dada la red $G = (N, A)$ con vector de ofertas/demandas $\boldsymbol{b} = (b_i)_{i \in N}$, llamamos **oferta total** al valor positivo:*

$$O_T = \sum_{\{i \in A : b_i \geq 0\}} b_i$$

*Y **demanda total** al valor positivo (suma de todas las demandas cambiada de signo) :*

$$D_T = - \sum_{\{i \in A : b_i < 0\}} b_i$$

Definición 2.1.2. *Se dice que una red está **balanceada**, si $O_T = D_T$ o de otra manera si $\sum_{i \in N} b_i = 0$*

Proposición 2.1.1. *Si en un problema de redes de coste mínimo como el descrito en (2.1), (2.2) y (2.3), tenemos una combinación de flujos factible $\boldsymbol{x} = (x_{ij})$, entonces la red está balanceada.*

Demostración. Basta con ver que si $\mathbf{x} = (x_{ij})$ es una combinación factible de mi problema de redes quiere decir que cumple las ecuaciones de balance (2.2) para cada $i \in N$. Entonces, si sumamos todas las ecuaciones de balance tenemos que:

$$0 = \sum_{i \in N} b_i$$

ya que cada variable x_{ij} aparece únicamente dos veces en todas las ecuaciones una con un 1 y otra con un -1. Por tanto la red está balanceada. □

El recíproco no es cierto, ya que puede haber una red que esté balanceada, pero no tenga ninguna solución factible. Esto suele ocurrir cuando hay algún corte de red que no permita el flujo de los nodos oferta a los nodos demanda.
Por ejemplo si consideramos la red de flujos con capacidades $G = (N, A)$:
$N = \{s, t, B, C\}$ y arcos $(s, B), (s, C), (B, t), (C, t)$ con capacidades 5, 5, 3 y 2 respectivamente. Y luego con ofertas/demandas $b_s = +6$ (nodo fuente) $b_t = -6$ (nodo

sumidero) y los demás 0. Se ve claramente que la red es balanceada, pero que no tiene ninguna solución factible, ya que es imposible satisfacer la demanda de 6 unidades en t porque el flujo máximo que le puede llegar es 5: 3 por el arco (C, t) y 2 por el arco (B, t).

Pero en muchos casos tenemos redes no balanceadas, no coincide la demanda total con la oferta total. Por ejemplo, es muy habitual que tengamos $O_T > D_T$, es decir que haya más oferta que demanda. En ese caso las restricciones dadas en (2.2) se pueden sustituir en los nodos oferta por :

$$\sum_{j \in N(i)^+} x_{ij} \quad - \sum_{j \in N(i)^-} x_{ji} \leq b_i$$

Puesto que va a haber cierto flujo ofertado que no va a poder ser distribuido. Sin embargo, a la hora de programar y resolver este problema en el ordenador no resulta práctico distinguir entre las restricciones de los nodos oferta, demanda y los de tránsito. Para solucionar esto, en la práctica lo que se hace es generar un nodo ficticio con arcos con llegada desde cualquier nodo oferta ($b_i > 0$) a este nodo ficticio, con coste asociado nulo y asociándole una demanda $D_T - O_T < 0$. De esta manera tendremos una red balanceada y podremos utilizar la formulación con las ecuaciones de balance que hemos presentado en esta sección. El flujo obtenido a través de los arcos ficticios, será el valor de las ofertas que no han podido ser distribuidas.

Ejemplo 2.1.2. *Consideramos la red $G = (N, A)$ con nodos $N = \{1, 2, 3, 4, 5\}$, arcos $A = \{(1,2), (1,3), (2,4), (2,5), (3,4), (3,5)\}$ y $b_1 = 15$ $b_2 = b_3 = 0$ $b_4 = -6$ $b_5 = 3$. Donde b_i es la oferta/demanda de cada nodo i y también consideramos que c_{ij} son los costes asociados a cada arco $(i, j) \in A$. Por tanto este problema se puede formular como:*

$$min \qquad \sum_{(i,j) \in A} c_{ij} x_{ij}$$

sujeto a $\qquad x_{12} + x_{13} = 15$

$$x_{34} + x_{35} - x_{13} = 0$$

$$x_{24} + x_{25} - x_{12} = 0$$

$$-x_{34} - x_{24} = -6$$

$$-x_{35} - x_{25} = -3$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A$$

Donde x_{ij} es el flujo a través de cada arco $(i, j) \in A$. Pero como hemos dicho para

evitar las restricciones del tipo $\leq$ *podemos introducir un nodo ficticio* 6 *con un único arco* $(1, 6)$*, ya que el nodo* 1 *es el único nodo con oferta* $(b_1 > 0)$*, y así obtenemos una nueva red* $G' = (N', A')$*. A a este arco le asociaremos como demanda la oferta sobrante que en este caso es* $b_6 = -6$ *y costes asociados nulos. Y por tanto la formulación resultante queda:*

$$min \qquad \sum_{(i,j)\in A'} c_{ij}x_{ij}$$

sujeto a

$$x_{12} + x_{13} + x_{16} = 15$$

$$x_{34} + x_{35} - x_{13} = 0$$

$$x_{24} + x_{25} - x_{12} = 0$$

$$-x_{34} - x_{24} = -6$$

$$-x_{35} - x_{25} = -3$$

$$-x_{16} = -6$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A'$$

Ahora en las siguientes secciones veremos dos generalizaciones o especificaciones del problema de flujo a coste mínimo que serán:

- Flujo en redes multiproducto.
- Flujo en redes con costos fijos (uno o varios productos).

2.2. Problemas de flujo multiproducto a coste mínimo

A continuación, se presenta nuevamente un problema de flujo en redes de coste mínimo, cuyo objetivo consiste en minimizar el coste total asociado al transporte de uno o varios flujos a través de los arcos de una red, sujeto a un conjunto de restricciones estructurales.
Este caso particular puede considerarse una generalización del planteado en la primera sección: en lugar de modelar el flujo de un único producto, se contempla el flujo simultáneo de varios productos diferenciados. Esta distinción es necesaria en contextos donde cada producto puede estar asociado a diferentes costes de transporte, capacidades máximas, o condiciones específicas, como ocurre en problemas de transporte con múltiples mercancías. La formulación matemática correspondiente es la siguiente.
Nuevamente consideramos un grafo dirigido $G = (N, A)$ con $N = \{1, ..., n\}$ el conjunto de nodos y el conjunto de arcos A representados por pares (i, j) con $i, j \in N$. En este caso, en vez de un producto, consideramos p productos definiendo el conjunto de los productos como $K = \{1, 2, ..., p\}$, y asociándole el índice $k = 1, ..., p$. Ahora consideramos las siguientes variables o parámetros del problema:

- x_{ijk} representa el flujo del producto k a través del arco (i, j).
- c_{ijk} representa el coste por unidad transportada en el arco (i, j) del producto k.
- b_{ik} oferta o demanda del producto k en el nodo i
- u_{ij} capacidad conjunta del arco (i, j).

Entonces *el problema de flujo multiproducto de coste mínimo* consiste en encontrar los valores de los flujos x_{ijk} que minimicen el coste satisfaciendo las ofertas y demandas de cada producto, sin sobrepasar las capacidades. La formulación de este problema es bastante análoga a la de un producto, y es la siguiente.

$$min \qquad \sum_{k=1}^{p} \sum_{(i,j)\in A} c_{ijk} x_{ijk} \tag{2.7}$$

$$\text{sujeto a} \qquad \sum_{j\in N(i)^+} x_{ijk} - \sum_{j\in N(i)^-} x_{jik} = b_{ik} \quad \forall i \in N \;\; k \in K \tag{2.8}$$

$$\sum_{k=1}^{p} x_{ijk} \leq u_{ij} \qquad \forall (i, j) \in A \tag{2.9}$$

$$x_{ijk} \geq 0 \qquad \forall (i.j) \in A, k \in K \tag{2.10}$$

Las restricciones dadas en (2.8) son las *Ecuaciones de balance* asociadas a las redes multiproducto, y representan el flujo neto (del que sale menos el que entra) de cada

producto k en el nodo i y se requiere que este sea igual a b_{ik}. En las redes multiproducto seguimos teniendo el convenio de que si $b_{ik} > 0$, i es un nodo oferta respecto del producto k y de la misma manera si $b_{ik} < 0$, nodo demanda, o $b_{ik} = 0$, nodo de tránsito. Es importante diferenciar el índice k en las ofertas y demandas ya que puede ser que un nodo sea de demanda para un cierto producto k_1, pero para otro k_2 sea de oferta, aunque puede que esto no sea lo habitual. Y la restricción (2.9) son las restricciones de capacidad conjunta en cada arco (i, j) considerando el flujo total en cada arco, sumando el flujo de todos los k productos que hay en la red.

Algunas consideraciones o comentarios respecto de este problema comparándolo con las redes de un producto son las siguientes:

1. Las restricciones dadas en (2.9) son las que son conjuntas a todos los productos, y son las fundamentales en este tipo de problemas ya que si no estuviesen bastaría con resolver p problemas (nº de productos) de redes para un solo producto independientes, cada uno para cada producto de mi red. Tal cual hemos visto en la anterior sección.

2. En el modelo de redes multiproducto es esencial la diferenciación de cada producto k, al contrario de lo que ocurría en las redes de un producto en las que las unidades de este se mezclaban al llegar al nodo i. Por lo que es muy importante mantener la identificación de cada producto usando el triple índice x_{ijk}, aún en el caso de que los costes de transporte no dependan del producto, c_{ij}.

3. En el caso de redes multiproducto, al estar las ecuaciones dadas como igualdades, se sigue manteniendo la propiedad de que si una solución es factible entonces es necesario que la red esté balanceada para cada producto. Esto es, que $\sum_{i \in N} b_{ik} = 0$ para cada $k \in K$.

4. En este caso los problemas de redes multiproducto no tienen la propiedad de integridad. Es decir, aunque las demandas b_{ik} sean números enteros las solución óptima de los flujos x_{ijk} no tienen porque ser enteras.

Ejemplo red Multiproducto

Distribución de productos en una cadena de suministros:

Imaginemos que tenemos una empresa que se dedica a gestionar una cadena de suministros con múltiples centros de producción y de distribución. La empresa produce diferentes productos electrónicos (móviles, ordenadores, chips, etc), que deben ser transportados desde las fábricas a los centros de almacenaje o distribución para posteriormente llevarlos a las diferentes tiendas minoristas. Cada producto tiene diferentes rutas posibles, costos de transporte y restricciones de capacidad (en los vehículos de transporte). Luego la red descrita en este ejemplo sería tal que así:

- N conjunto de nodos: fábricas, almacenes y tiendas.
- A conjunto de arcos: rutas de transporte entre los nodos.
- K conjunto de productos: móviles, ordenadores,chips,...

Un diagrama más o menos realista de esta situación podría ser el siguiente:

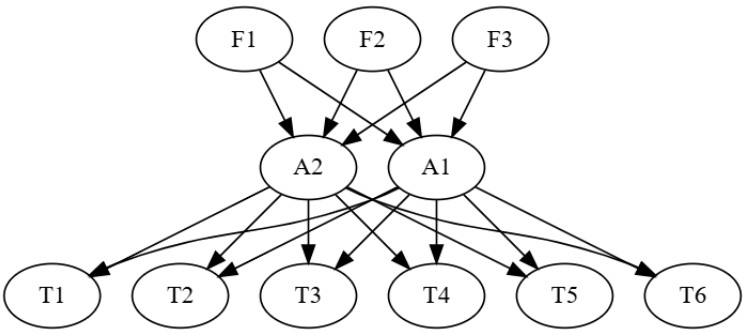


Figura 2.1: F_i conjunto de fábricas, A_i almacenes y T_i las tiendas. Elaboración propia

En este caso tendríamos las variables y parámetros siguientes: c_{ijk}, coste de transportar una unidad del producto k en el arco (i, j); u_{ij} capacidad del arco (i, j), que puede venir por ejemplo, dependiendo de la cantidad de camiones que dispongamos y la cantidad de producto que puedan soportar (por peso o tamaño); b_{ik} demandas u ofertas del producto k en el nodo i, (las fábricas son nodos oferta, los almacenes de tránsito y las tiendas de demanda); x_{ijk} que indican el flujo/cantidad de producto k transportado en el arco (i, j). Este modelo se puede aplicar para ayudar a la empresa a gestionar su logística. Ya que queremos optimizar el coste del transporte, asegurándonos que se satisfaga la demanda en las tiendas respetando las capacidades. Esto es lo que se conoce como transporte en dos etapas, también se podría considerar el caso de que se transporte directamente a las tiendas.

2.3. Problemas de flujo en redes con costes fijos (1 producto)

Nuevamente es otra especificación del problema general de flujo en redes, el objetivo va a seguir siendo el mismo: encontrar la manera óptima de transportar un flujo (gas, productos en un camión, agua, etc.) a través de una red sujeto a restricciones. En los casos anteriores teníamos un coste variable c_{ij} que era proporcional a la cantidad de flujo transportado a través del arco (i, j), pero en este tipo de problemas vamos a tener en cada arco (i, j) un coste fijo f_{ij} asociado al hecho de si utilizamos ese arco o no, que es independiente de la cantidad de flujo que se transporta.
Para ello se introducen unas variables binarias $y_{ij} \in \{0, 1\}$ que tomarán el valor 1 si se utiliza el arco (i, j) y 0 en caso contrario. A estas variables se las conoce como *variables de diseño*, ya que al final darán la configuración o diseño de la red. A ellas es a las que se les asocia el coste fijo f_{ij} (coste de construcción del camino, coste de mantenimiento,...), dependiendo de si se utiliza ese arco o no.

Una formulación matemática clásica para este tipo de problemas es la siguiente.
Nuevamente consideramos una red dirigida $G = (N, A)$ donde $N = \{1, ..., n\}$ es el conjunto de nodos y A es el conjunto de arcos representados por pares (i, j), $i, j \in N$.
Las **variables de decisión** de nuestro problema son

- $x_{ij} \geq 0$: representan la cantidad de flujo transportado a través del arco (i, j).
- $y_{ij} \in \{0, 1\}$ variable binaria que me indica si se utiliza el arco (i, j) o no.

Los **parámetros** de nuestro problema son

- c_{ij} costo variable por unidad de flujo transportada en el arco (i, j).
- f_{ij} costo fijo de utilizar el arco (i, j)
- u_{ij} capacidad del arco $(i, j) \in A$, si es que se establece una capacidad.
- b_i variables de balance, que me indican las ofertas (> 0) y demandas (< 0) en cada nodo i.

Por tanto, *el problema de flujo en redes con costes fijos* consiste en minimizar la función objetivo de los costes:

$$min \qquad \sum_{(i,j)\in A} (c_{ij}x_{ij} \ + f_{ij}y_{ij}) \tag{2.11}$$

Sujeto a las restricciones:

$$\sum_{j \in N(i)^+} x_{ij} \; - \sum_{j \in N(i)^-} x_{ji} = b_i \quad \forall i \in N \tag{2.12}$$

$$x_{ij} \leq u_{ij} y_{ij} \quad \forall (i,j) \in A \tag{2.13}$$

$$x_{ij} \geq 0 \;\; \forall (i,j) \in A \;\; , y_{ij} \in \{0,1\} \;\; \forall (i,j) \in A \tag{2.14}$$

(2.12) son las clásicas ecuaciones de balance, que al igual que en los dos anteriores modelos indican el flujo neto en cada nodo i. Nuevamente para que el problema sea factible, tenemos que tener que la red esté balanceada. Puede ocurrir que esta no lo esté, sino que tengamos más oferta que demanda ($O_T > D_T$), en este caso se puede recurrir al método que vimos en la Sección 2.1.2 (Redes Balanceadas): añadir un nodo ficticio con arcos dirigidos de los nodos oferta hacia él y costo asociado nulo. De esta manera, vimos que se podía obtener una red balanceada y mantener las restricciones con igualdades.

Las restricciones dadas en (2.13) nos dan la relación entre el flujo en el arco (i,j) con su activación. Ya que si $y_{ij} = 0$ quiere decir que no se activa el arco (i,j) y por tanto se tiene que $x_{ij} = 0$, luego no hay ningún flujo en ese arco, como es lógico.

Y en (2.14) nos indica el dominio de las variables de este problema, como siempre las variables de flujo son no negativas y, como hemos dicho, las variables de diseño son binarias. Esto da a este problema una estructura de formulación mixta, mezcla de variables continuas y binarias.

2.4. Problemas de flujo en redes con costes fijos multiproducto

Ahora introducimos una versión especial al problema con costos fijos anterior, en el caso que tengamos varios productos. Consideramos nuevamente una red $G = (N, A)$ dirigida con $|N| = n$ y $(i,j) \in A, \forall i,j \in N$, y además consideramos el conjunto de productos $K = \{1, ..., p\}$. Tendríamos las siguientes variables en el problema:

- x_{ijk} : flujo de i a j para el producto $k \in K$.
- y_{ij}: variable binaria que indica si el arco (i,j) está activo, es decir si se envía flujo de i aj.
- c_{ijk}: coste por unidad de flujo trasportada de i a j para el producto k.

- f_{ij}: coste fijo de la activación del arco (i, j).
- b_{ik}: variables de balance, indican la oferta o demanda en el nodo i del producto k
- u_{ij} capacidad conjunta del arco (i, j).

Nótese que el coste fijo solo tiene doble índice, ya que sería el coste de activar ese arco y solo se activaría si es rentable mandar flujo a través de ese arco para cualquier producto que tenemos. Las variables b_{ik} tienen doble índice, ya que tenemos que diferenciar entre las ofertas y demandas de cada producto. Y como en el caso multiproducto sin costes fijos las capacidades u_{ij} son conjuntas al arco (i, j), contando el flujo total a través de ese arco para todos los productos. Ya que si las capacidades dependiesen del producto bastaría con resolver p problemas de coste fijo diferentes, uno para cada producto. Luego una formulación matemática para este problema podría ser como sigue:

$$Min \quad \sum_{k=1}^{p} \sum_{(i,j)\in A} x_{ijk} c_{ijk} + \sum_{(i,j)\in A} y_{ij} f_{ij} \tag{2.15}$$

Sujeto a:

$$\sum_{j\in N(i)^+} x_{ijk} \quad - \quad \sum_{j\in N(i)^-} x_{jik} = b_{ik} \qquad \forall i \in N, \forall k \in K \tag{2.16}$$

$$\sum_{k=1}^{p} x_{ijk} \leq u_{ij} y_{ij}, \qquad \forall (i, j) \in A \tag{2.17}$$

$$x_{ijk} \geq 0, \quad y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A, \forall k \in K \tag{2.18}$$

Las restricciones de (2.16) son las ecuaciones de balance, y representan nuevamente el flujo neto en cada nodo i y cada producto k. Como en los problemas anteriores para que una solución sea factible debemos tener la red balanceada para cada producto k. Esto es que $\sum_{i=1}^{n} b_{ik} = 0$ para cada $k = 1, .., p$. Las restricciones en (2.17) indican el flujo total del arco (i, j) sumando el flujo para cada producto, y además relaciona las variables de flujo x_{ijk} con las binarias y_{ij}. Puesto que si $y_{ij} = 0$, es decir, no se activa el arco (i, j) hace que $\sum_{k=1}^{p} x_{ijk} = 0$ que implica que no hay flujo por parte de ningún producto a través del arco (i, j). Y (2.18) indica el dominio de las variables, dotando a este problema como en el caso de 1 producto una estructura de programación entera mixta.

Capítulo 3

Problemas de diseño de redes de comunicación

Los problemas de diseño de redes constituyen un área muy amplia de problemas donde el objetivo es encontrar la configuración o diseño óptimo de una red, sujeto a las necesidades o restricciones en cada problema concreto que tengamos. Estas pueden ser: necesidades de servicio, de conectividad entre los puntos, restricciones de capacidad, etc. Este área tiene múltiples aplicaciones prácticas, sobre todo en áreas de redes de comunicaciones (ordenadores, electricidad, tuberías,...), y en redes de logística.
Cuando el objetivo no es el diseño de la red sino la optimización de la misma tenemos alguno de los tipos de problema vistos en el segundo capítulo 2, *problema de flujo de coste mínimo con un producto o multiproducto*. La diferencia con los problemas de diseño, en general, son la presencia de costos fijos, asociados a la activación de esa línea de conexión dentro de mi red de distribución, bien por el costo de construcción, mantenimiento, etc. Dentro de los problemas de diseño además de los costos fijos, podemos tener costos variables proporcionales a la cantidad de flujo transportado. Estos son también unos de los problemas relativos al segundo capítulo, *problemas de flujo de coste mínimo con costos fijos (1 producto o multiproducto)*. Estos aparecen sobre todo en redes logísticas, donde además de tener costos fijos por mantener esas líneas de distribución, se tienen costos variables relativos a la cantidad de productos transportados. Los problemas de diseño sin costo variable son los conocidos como *problemas de diseño puro* o también *problemas topológicos de redes*. Y son aquellos en los que el objetivo es encontrar un diseño óptimo de la red, encontrando qué enlaces (aristas) deben establecerse entre los nodos, sujeto a diversas restricciones. Como el coste de cada enlace es fijo, es decir, independiente del volumen de flujo que soportará, el problema se reduce a una formulación puramente estructural: se trata de decidir qué conexiones físicas o lógicas instalar para cumplir con los requisitos de conectividad y calidad del servicio, minimizando el coste total de implantación. Este enfoque es fundamental para las fases iniciales del diseño de cualquier red.

El diseño de redes es usualmente un proceso de toma de decisiones jerárquica, consistiendo en tres grandes pasos para un planificación y diseño efectivos:

1. Generar un buen conjunto base de buenos diseños topológicos de redes utilizando modelos basados en optimización.

2. Refinar cada uno de estos diseños incluyendo más decisiones cualitativas como requerimientos de capacidad o de calidad de servicio.

3. Evaluar los costes del diseño y su redimiendo usando modelos de simulación analíticos.

En este trabajo nos centraremos sobre todo en el paso 1 y 2, ya que veremos multitud de modelos matemáticos basados en problemas de optimización de programación lineal y entera. En ellos revisaremos algunos de los diseños topológicos más importantes.

3.1. Evolución de las topologías de redes

La evolución del diseño de redes de comunicación, sobre todo en el marco de las telecomunicaciones: redes de ordenadores, telefónicas, etc; ofrece un marco valioso para comprender como han ido variando los diseños topológicos de redes durante el tiempo, adaptándose a nuestras necesidades actuales y al gran crecimiento tecnológico en las últimas décadas. En sus primeras etapas, las redes estaban compuestas por unos pocos terminales conectados directamente a un terminal central. Los usuarios finales estaban conectados cerca de estos terminales centrales, y por tanto los costes totales de estas conexiones eran insignificantes. En este entorno, la topología más común eran las configuraciones de tipo **estrella** (Figura 3.1.a), en la que cada terminal tenía un enlace directo con el nodo central. Este diseño resultaba simple y efectivo mientras el número de usuarios fuera reducido y estuvieran físicamente próximos al terminal central. Sin embargo, a medida que la cantidad de usuarios creció y su dispersión geográfica aumentó, el coste de mantener estos enlaces directos se volvió muy alto.
La mayoría de los terminales usan las líneas de transmisión solo en una fracción pequeña de tiempo. Estos mandaban y recibían mensajes de manera intermitente y las transmisiones duraban apenas unos pocos segundos. Luego, un método para el uso eficiente de las capacidades de las transmisiones consistió en conectar varios terminales a una línea, así varios terminales compartían una misma línea de conexión. Las conexiones estructuradas de esta manera se llaman **redes multidrop** o redes multipunto (3.1.b). Este cambio permitió reducir costes totales, sin sacrificar significativamente la calidad del servicio. Esto significa que la topología óptima para este problema corresponde a un árbol en un grafo $G = (N, A)$ donde todos los nodos de N excepto uno se corresponden con los terminales. El nodo restante se corresponde

con el terminal central y los arcos de A son el posible cableado que conecte los terminales. Cada subárbol, que cuelga del nodo central o raíz, son las distintas líneas multidrop. Este diseño óptimo de la red nos lleva al problema central de este trabajo, **el problema del mínimo árbol expandido (MST)**. Que trata de conectar todos los puntos de una red sin generar ciclos con el mínimo coste o distancia posible, sin ninguna otra restricción. Esta estructura marcó la transición hacia un enfoque más

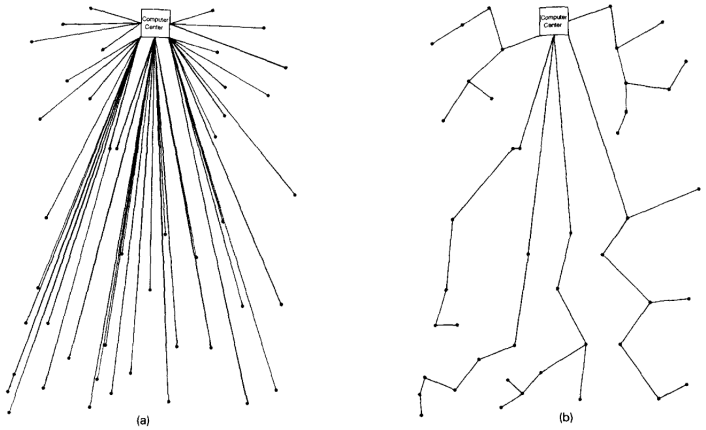


Figura 3.1: a) Diseño de estrella centralizado b) Diseño multidrop centralizado. Imagen sacada de [8]

modular, escalable y reutilizable, en el que la topología se diseña de forma jerárquica. Este proceso histórico ha sido paralelo en otras redes físicas, no solo en las redes de ordenadores. En las redes eléctricas o las logísticas, también se ha pasado de diseños locales y centralizados a estructuras distribuidas y optimizadas mediante puntos o subcentros intermedios.

Usualmente cada puerto de entrada en los distintos terminales podían soportar únicamente, como mucho, una cantidad fija de información. Esto, en términos prácticos, se corresponde a restringir la máxima cantidad de información que fluía a través de cualquier línea de conexión adyacente al puerto central (línea multidrop). Este problema de diseño es conocido en el marco de la optimización como el **problema del mínimo árbol expandido con capacidades (CMST)**. En este trabajo vamos a considerar que cada terminal produce la misma cantidad de tráfico, y lo podemos simplificar a una unidad. En esta situación, las restricciones de capacidad llevan a que cada línea multidrop no puede tener más de un número fijado de terminales. Ya sea porque poner múltiples terminales en una misma línea supera el número máximo de datos que pueden fluir a través de la red o por otros motivos. Entonces este diseño topológico nos lleva a una extensión del MST: queremos encontrar un árbol que minimice el coste, pero que en cada subárbol que cuelga de la raíz (línea multidrop) el número de nodos no supere la capacidad máxima.

Este aumento en las conexiones de telecomunicaciones fue correlativo con la actividad económica de las empresas que se aprovecharon de las múltiples oportunidades que ofrecía el aumento de la conexiones. Debido a esto las redes de conexión y sus aplicaciones aumentaron tanto en tamaño como en alcance, el tiempo de respuesta de las conexiones o la necesidad de un mayor rango en la transferencia de datos, requerían de líneas de conexión con mayor capacidad para satisfacer estos requerimientos. Estas líneas de conexión con mayor capacidad eran más caras comparadas con las de baja capacidad y mucho más sensibles a la pérdida de señal dependiendo de la distancia de las conexiones. Los Concentradores o las técnicas de concentración se han ido desarrollando para evitar estas limitaciones. Los concentradores son dispositivos que recogen el tráfico de múltiples terminales y lo envían, a través de líneas de alta capacidad, a nodos centrales o redes troncales. Esto lleva a redes dispuestas como la Figura 3.2.c (subredes estrella) y la figura 3.2.d (subredes multidrop).

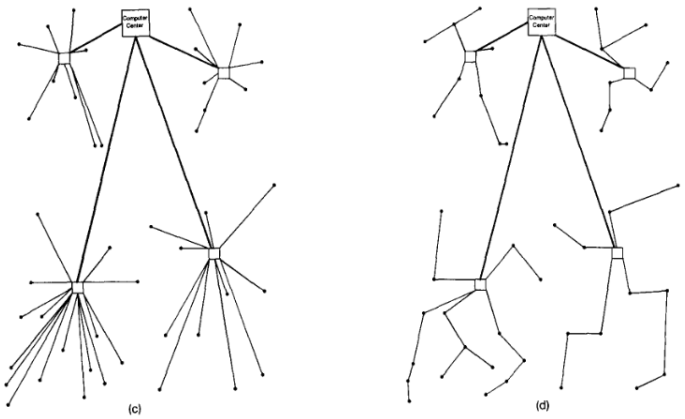


Figura 3.2: c) Red centralizada basada en concentradores con subredes en estrella d) Red centralizada basada en concentradores con subredes multidrop.
Imagen sacada de [8]

Con la proliferación de los ordenadores y las múltiples aplicaciones online, las empresas y organismos gubernamentales instalaron muchas redes centralizadas. Muchas organizaciones grandes tenían una gran cantidad de redes en funcionamiento, cada una de ellas operando como una entidad separada. En muchos casos, cada función principal de la organización había instalado su propia red. La división de marketing tenía una red para apoyar sus actividades, los grupos de manufactura implementaron sus propias redes, las organizaciones de investigación y desarrollo, comercio, contabilidad y las distintas subdivisiones de una misma empresa tenían cada una su propia red dedicada. Las redes, en muchos casos, pasaban por las mismas ubicaciones (ciudades, plantas, oficinas) y, en muchos casos, se solapaban físicamente (ver Figura 3.3.a-b y 3.4.c). Además de ser un derroche económico (debido al alto grado de duplicación y solapamiento), este laberinto de redes ha sido una pesadilla para gestionar y planificar.

Además la llegada de la fibra óptica trajo tremendas mejoras en la posibilidad de

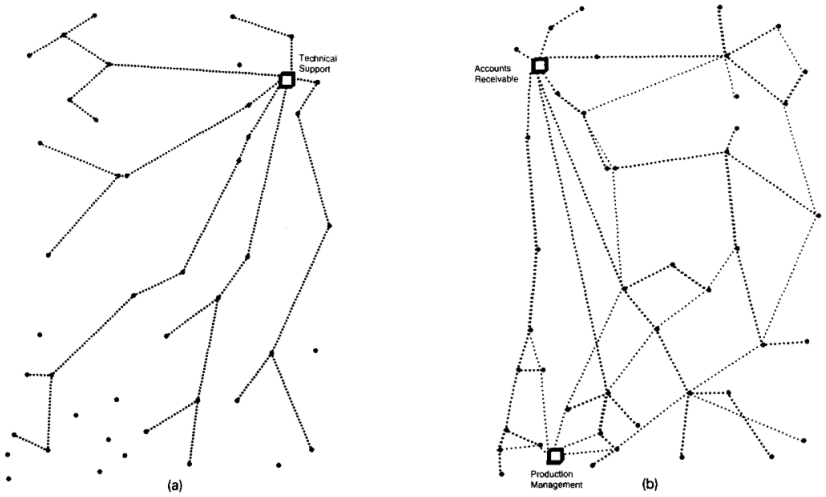


Figura 3.3: a)Red del apoyo técnico b)Redes del centro de gestión de producción y del departamento de contabilidad.
Imagen sacada de [8]

llevar a cabo múltiples servicios online, desde plataformas de vídeo, video conferencias, aprendizaje remoto y en general en las telecomunicaciones. La industria de las Telecomunicaciones invirtió enormes cantidades de dinero ($ 35 billones de dólares en 2003). Dada la magnitud de las inversiones el óptimo diseño de las redes se volvió muy importante. Esto sugirió que las empresas de Telecomunicaciones formaran grandes equipos de investigación y desarrollo en esta área. El objetivo de estos grupos era diseñar las redes de la forma más efectiva posible tanto en términos de costes como en su rendimiento.

Esta visión dio lugar a la red de telecomunicaciones actual. Una estructura típica de una red informática moderna se muestra en la Figura 3.4.d. La mayor parte de la red es la red trocal (*backbone*), que consiste en puntos de acceso (localizaciones NCP: network control point), conectados con líneas de alta capacidad. Puede considerarse como una red de carreteras que conecta los distintos puntos de acceso a una red de autopistas. La otra parte son las redes de acceso local (LAN: local acces network), que canalizan el tráfico desde o hacia la red troncal.
En los siguientes capítulos abordaremos el problema del diseño de la red troncal (*backbone*) bajo restricciones de salto. Es decir, dado un grafo no dirigido en el que se conoce el coste o la distancia asociada a cada enlace, el objetivo es encontrar caminos que conecten todos los nodos con un nodo fuente, minimizando el coste total, y respetando las denominadas *restricciones de salto*. Estas restricciones imponen que la ruta entre el nodo fuente y cualquier otro nodo no exceda un número máximo de conexiones (o saltos), el cual se representa mediante un parámetro de salto. Esta

variante constituye una extensión del problema del mínimo árbol expandido, conocido como el **problema del mínimo árbol expandido con restricciones de salto (HMST)**. El parámetro de salto se define como el número de conexiones que un paquete de datos puede cruzar antes de llegar a su destino. Variando este parámetro podemos generar diseños alternativos que cambiarán el coste y las conexiones. Las topologías comunes de las redes troncales son grafos conexos, que admiten múltiples caminos entre cada par de nodos NCP. Múltiples caminos entre pares de NCPs (de dos a cuatro) aseguran un alto nivel de confiabilidad en caso de fallo de un enlace o nodo. Esto lo podemos hacer reduciendo el parámetro de salto. Sin embargo, al mismo tiempo el coste final del diseño de la red aumenta.

Las redes de acceso local (LANs) consisten en una combinación de árboles, bucles, concentradores y sus subredes conectadas a los NCPs. El tráfico de los usuarios finales puede dirigirse a otros usuarios finales de la red o a centros de cómputo (o *hosts*). Los *hosts* proporcionan a los usuarios servicios de procesamiento de datos y cálculo. Los *hosts* que gestionan grandes volúmenes de tráfico suelen estar conectados directamente a la red troncal mediante enlaces de alta capacidad.

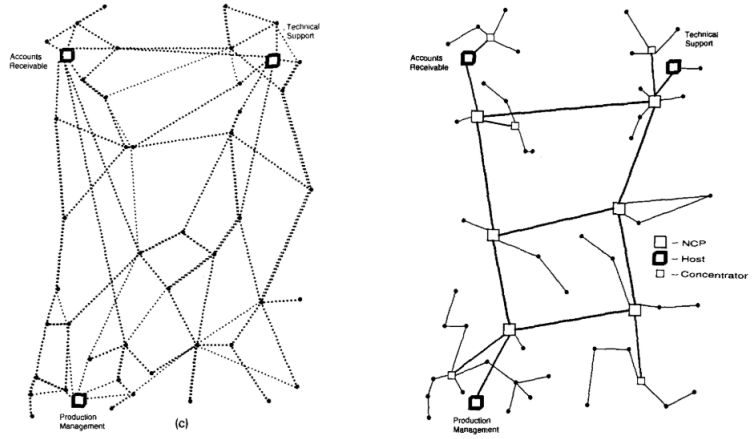


Figura 3.4: c)Superposición de rede multicentralizadas d)Diseño de una red moderna. Imagen sacada de [8]

Capítulo 4

Problema del Mínimo Árbol Expandido

El Problema del Mínimo Árbol Expandido o MST (*Minimal Spanning Tree*) es uno de los problemas más básicos dentro de los problemas de diseño de redes, y consiste en conectar entre sí n puntos distintos con la distancia o costo total mínimo. Recordamos del capítulo de definiciones previas que un árbol expandido T de un grafo G es un subgrafo expandido (mismo conjunto de nodos) conexo y sin ciclos.
Entonces el *MST* lo podemos formular de la siguiente manera, consideramos un grafo con pesos $G = (N, A)$, con $|N| = n$, con $|A| = m$ y con pesos w_{ij} para cada arista $(i, j) \in A$. El problema consiste en encontrar un árbol expandido $T = (N, A')$, *llamado mínimo árbol expandido*, con el mínimo costo/distancia asociado a las aristas que lo constituyen. Esto sería la suma total de los pesos w_{ij} con $(i, j) \in A'$. Antes de todo vamos a considerar dos observaciones:

1. Nótese que en el Problema del Mínimo Árbol Expandido los arcos del grafo son no dirigidos, entonces cuando hablemos de un arco que conecta un nodo i y un nodo j podemos referirnos a ese arco como (i, j) o (j, i). El problema de encontrar un árbol expandido se vuelve mucho más complicado cuando los arcos son dirigidos, de hecho, no hemos introducido la noción de conexión en un grafo dirigido.

2. Este problema tiene sentido cuando la red que genera el grafo $G = (N, A)$ es conexa porque es obvio que, si no, no va a existir ningún árbol expandido de G. Luego normalmente viene implícito que el grafo G considerado es conexo.

El Problema del Mínimo Árbol Expandido aparece en múltiples aplicaciones, tanto de forma independiente o como un subproblema de un problema más complejo. En la mayoría de los casos prácticos de este problema se tiene que los pesos $w_{ij} \geq 0$, bien porque indican distancias o costos de transporte. Por tanto, para el caso que tengamos

un grafo con pesos $G = (N, A)$ conexo, es claro que tenemos una solución en forma de árbol expandido. Pues si un subgrafo expandido conexo tiene algún ciclo, se le puede quitar alguna arista y seguirá siendo conexo y tendrá menor costo asociado que el anterior.
En este tipo de problemas vamos a ver dos condiciones de optimalidad que nos permitirán evaluar si un determinado árbol expandido tiene el mínimo coste/distancia. Una viene de comparar el coste de cada arco del árbol con los otros arcos contenidos en el corte definido de eliminar cada arco del árbol. Y el otro viene de comparar el coste de cualquier arista que no esté en el árbol con los arcos del camino que conecta los dos puntos finales de esa arista. Estas dos condiciones son sencillas de establecer y desarrollar y, naturalmente, van a motivar la creación de algoritmos para resolver el *MST*.
En esta sección vamos a considerar dos algoritmos fundamentales, uno por cada condición de optimalidad: el *algoritmo de Kruskal* y el *algoritmo de Prim*. Estos algoritmos están basados en algoritmos de tipo *greedy*, esto es, una estrategia de resolución de problemas que, en cada paso, elige la opción que parece más prometedora en ese momento, con la esperanza de alcanzar una solución óptima global. Pues estos algoritmos, en cada paso, eligen una arista con menor coste dentro de una lista de candidatos, siempre que añadirla no suponga crear ciclos.
Para probar las condiciones de optimalidad viene bien recordar alguno de los resultados que vimos en el capítulo de Definiciones Previas sobre árboles,

1. Para cualquier arista (i, j) no perteneciente al árbol existe un único camino en el árbol entre i y j, y añadiendo la arista (i, j) creamos un único ciclo.

2. Siendo $T = (N, A')$ un árbol expandido de $G = (N, A)$, sabemos que la eliminación de cualquier arista (i, j) en T, me genera dos subárboles T_1 y T_2, donde sus conjuntos de nodos son una partición de N. Los llamamos S y $S' = N - S$. A cada arista de G que conecta un nodo de S con un nodo de S' decimos que está en el corte $[S, S']$, y todas ellas excepto la que hemos eliminado de T son aristas que no están en el árbol.

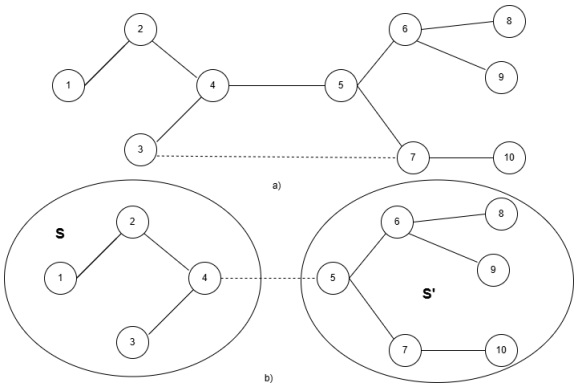


Figura 4.1: a) Añadiendo el arco (3,9) al árbol expandido se crea un ciclo único 3-4-5-9-3; b) eliminando el arco (4,5) creamos el corte $[S, S']$, donde $S = \{1, 2, 3, 4\}$.
Elaboración propia.

Ahora vamos a definir y probar las dos condiciones de optimalidad.

Definición 4.0.1. *Se dice que un árbol T^* verifica* ***la condición de corte óptimo*** *si para cada arco del árbol $(i, j) \in T^*$, se tiene que $w_{ij} \leq w_{kl}$ para cualquier arco (k, l) contenido en el corte $[S, S']$ formado al eliminar (i, j).*

Teorema 4.0.1 (Condiciones de corte óptimo)**.** *Un árbol expandido T^* es un mínimo árbol expandido si y solo si satisface las condiciones de corte óptimo.*

Demostración. Es fácil ver que todo *MST* debe cumplir la condición de corte óptimo, ya que si no, existiría un arco $(i, j) \in T^*$ tal que $w_{ij} > w_{kl}$ para un arco (k, l) del corte formado al eliminar (i, j). Entonces, introduciendo el arco (k, l) en T^* por el arco (i, j) tenemos otro árbol expandido con menos coste que T^*, en contradicción con que era el óptimo.
Por otro lado, vamos a ver que si un árbol expandido T^* cumple la condición de corte óptimo, este tiene que ser óptimo. Suponemos que T' es un *MST* y $T' \neq T^*$. Entonces, T^* contiene un arco (i, j) que no está en T'. Eliminando (i, j) de T^* creamos un corte $[S, S']$. Ahora nótese que añadiendo (i, j) a T' se crea un único ciclo C, que debe contener un arco $(k, l) \neq (i, j)$ donde $k \in S$ y $l \in S'$, luego (k, l) es una arista del corte $[S, S']$. Como T^* satisface la condición de corte óptimo, $w_{ij} \leq w_{kl}$. Además, como T' es un *MST* $w_{ij} \geq w_{kl}$, ya que si no podríamos mejorar el coste remplazando (k, l) por (i, j). Por tanto, $w_{ij} = w_{kl}$. Ahora, si introducimos en T^* el arco (k, l) por (i, j), T^* seguirá siendo un árbol expandido, con el mismo costo que antes y con una arista más en común con T'. Luego, repitiendo este proceso podemos transformar el árbol expandido T^* en el árbol óptimo T'. Esta construcción nos muestra que T^* era también un árbol óptimo.

□

Corolario 4.0.1. *Sea F un subconjunto de aristas en algún árbol expandido de mínimo coste y sea S el conjunto de nodos de F. Supongamos que (i, j) es el arco de mínimo coste en el corte $[S, S']$. Entonces algún mínimo árbol expandido contiene todos los arcos de F y el arco (i, j).*

Demostración. Supongamos que F es un subconjunto de nodos del MST T^*. Si $(i, j) \in T^*$ no hay nada que hacer. Supongamos que $(i, j) \notin T^*$, añadiendo (i, j) a T^* me genera un único ciclo C, y este contiene al menos un $(p, q) \neq (i, j)$ en $[S, S']$. Por hipótesis $w_{ij} \leq w_{pq}$. Además T^* satisface la condición de corte óptimo, luego $w_{ij} \geq w_{pq}$ y entonces $w_{ij} = w_{pq}$. Por tanto añadir el arco (i, j) por el (p, q) en T^* genera un MST que contiene todos los arcos de F e (i, j), que era lo que buscábamos □

Definición 4.0.2. *Se dice que un árbol T^* verifica* ***la condición del camino óptimo****, si para cada arista (k, l) de G que no pertenece al árbol, se tiene que $w_{ij} \leq w_{kl}$ para toda arista (i, j) perteneciente al camino único en T^* que une k y l.*

Teorema 4.0.2 (Condición del camino óptimo)**.** *Un árbol expandido T^* es un mínimo árbol expandido si y solo satisface la condición del camino óptimo.*

Demostración. Nuevamente es sencillo ver que todo mínimo árbol expandido T^* tiene que cumplir la condición del camino óptimo, ya que si no, existiría un arco (k, l) que no está en T^* con $w_{ij} > w_{kl}$ para un (i, j) perteneciente al camino que une k y l en T^*. Luego remplazando (i, j) por (k, l) obtendríamos otro árbol y con menor coste que T^*, en contradicción con que era un mínimo árbol expandido.
Para ver que la condición también es suficiente vamos a usar que la condición de corte óptimo es suficiente. En esta prueba destacamos la equivalencia entre las dos condiciones de optimalidad. Veremos que si un árbol T^* satisface la condición del camino óptimo, también cumple la del corte óptimo; por el Teorema anterior concluiremos que el árbol es óptimo. Sea (i, j) cualquier arco del árbol T^* y sea S y S' los dos conjuntos de nodos conectados producidos de eliminar el arco (i, j) de T^*. Suponemos que $i \in S$ y $j \in S'$ y consideramos $(k, l) \neq (i, j)$ cualquier arco del corte $[S, S']$. Como T^* contiene un único camino entre el nodo k y l y, puesto que, (i, j) es el único arco que une S y S' que está en T^*, el arco (i, j) debe estar en ese camino. La condición del camino óptimo me dice que $w_{ij} \leq w_{kl}$. Como está condición debe ser válida para cualquier arista (k, l) del corte $[S, S']$ formado por eliminar cualquier arco (i, j), T^* satisface la condición de corte óptimo y entonces se concluye que T^* es un mínimo árbol expandido. □

De estas condiciones vamos a derivar y validar los *Algoritmos de Kruskal* y *Prim* para resolver el problema del mínimo árbol expandido.

Algoritmo de Kruskal

La condición del camino óptimo deriva en un algoritmo que construye un árbol óptimo desde cero añadiendo un arco a cada iteración. Este algoritmo es conocido como *el algoritmo de Kruskal* (véase en [15]), el procedimiento es el siguiente. Primero ordenamos la lista de arcos en función de sus pesos de manera no decreciente y definimos un conjunto donde vamos a ir metiendo los arcos elegidos como parte del *MST*. Inicialmente este conjunto está vacío. Entonces en cada paso del algoritmo vamos a ir examinando los arcos uno por uno y verificando si añadir ese arco me genera un ciclo con los arcos que ya tengo añadidos. Si no ocurre añadimos este arco a la lista, en otro caso, lo descartamos. Terminaremos cuando la lista tenga tamaño $n-1$, siendo n el número de nodos del grafo G. Al final la lista de arcos genera un mínimo árbol expandido. Primero es claro que el grafo que genera la lista de arcos final, es un árbol. Pues hemos garantizado que el grafo resultante no tiene ciclos y tiene exactamente $n-1$ arcos, luego es conexo y por tanto un árbol. Y la justificación de porqué efectivamente este algoritmo genera un mínimo árbol expandido viene del hecho que precisamente cada arco (k,l) que no pertenece al árbol lo hemos descartado en algún paso porque generaba un ciclo con los arcos que ya se habían elegido. Luego se observa que el coste del arco (k,l) es mayor o igual que los arcos que forman ese ciclo, porque estamos añadiendo los arcos en orden no decreciente de sus pesos, y efectivamente los arcos con los que forma el ciclo han sido añadidos antes. Por tanto, el árbol T^* verifica la condición del camino óptimo, luego es un mínimo árbol expandido.

Algoritmo de Prim

Tal como la condición del camino óptimo justificaba la validez del *Algoritmo de Kruskal*, la condición del corte óptimo nos va a permitir desarrollar otro algoritmo simple para encontrar árboles óptimos; este algoritmo es conocido como el *Algoritmo de Prim* (véase en [18]. Este algoritmo genera un mínimo árbol expandido desde cero, desplegándose desde un nodo concreto y añadiendo arcos en cada iteración. Mantiene un árbol expandido en un subconjunto S de nodos y añade vecinos cercanos a S. El algoritmo realiza esto identificando un arco (i,j) de mínimo coste del corte $[S,S']$, es decir, encuentra el arco de mínimo coste que une un nodo de S con un nodo de $N-S=S'$. Se añade el arco (i,j) al árbol y j a S, y se vuelve a realizar esta operación. La justificación viene directamente por el Corolario 4.0.1, ya que este resultado implica que cada arco que añadimos al árbol está contenido en algún mínimo árbol expandido con los arcos que habíamos seleccionado en los pasos previos.

Para ilustrar los dos algoritmos vamos a ver un ejemplo sencillo y como lo imple-

mentamos en cada caso. Consideramos la red con pesos ilustrada en las Figuras 4.2 y 4.3.

Mediante el **Algoritmo de Kruskal** ordenamos los arcos en orden no decreciente de pesos de la siguiente manera $(2,4),(3,5),(3,4),(2,3),(4,5),(1,2)$ y $(1,3)$. 1ª Iteración: Añadimos el $(2,4)$ con $w_{24}=10$. 2ª Iteración: Añadimos el $(3,5)$ con $w_{35}=15$. 3ª Iteración: Añadimos el $(3,4)$ con $w_{34}=20$. 4ª Iteración: Los siguientes arcos para examinar serían el $(2,3)$ y el $(4,5)$, pero no los podemos añadir porque generan un ciclo con los arcos ya añadidos, $C_1=2-4-3-2$ y $C_2=4-3-5-4$ respectivamente. Luego, el siguiente es el $(1,2)$ que no genera ningún ciclo, con $w_{12}=35$. Y ya habríamos terminado porque hemos añadido 4 arcos de 5 nodos que teníamos. Hemos obtenido al final un árbol con arcos $\{(2,4),(3,5),(3,4),(1,2)\}$ y coste total $w_{24}+w_{35}+w_{34}+w_{12}=80$.

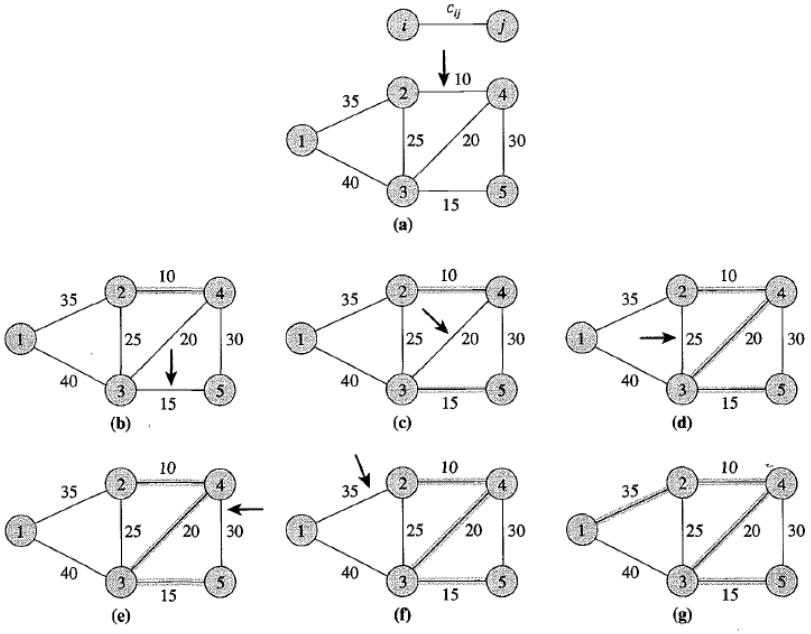


Figura 4.2: Ilustración del Algoritmo de Kruskal.
Imagen sacada de [2].

Mediante el **Algoritmo de Prim** tenemos que elegir un nodo inicial en el que fijarnos, supongamos que $S=\{1\}$. 1ª Iteración: El corte $[S,S']$ contiene los arcos $(1,2)$ y $(1,3)$, añadimos el más barato $(1,2)$ con $w_{12}=35$ y actualizamos $S=\{1,2\}$. 2ª Iteración: El corte $[S,S']$ contiene los arcos $(1,3),(2,3),(2,4)$ y se elije el $(2,4)$ con $w_{24}=10$ y actualizamos $S=\{1,2,4\}$. 3ª Iteración: El corte $[S,S']$ contiene los arcos $(1,3),(2,3),(4,5),(3,4)$ y se añade el $(3,4)$ con $w_{34}=20$. 4ª Iteración; Tenemos finalmente los arcos $(3,5),(4,5)$ y elegimos el $(3,5)$ con $w_{35}=15$. Y habríamos generado nuevamente un mínimo árbol expandido, en este caso con los mismos arcos que el

anterior.

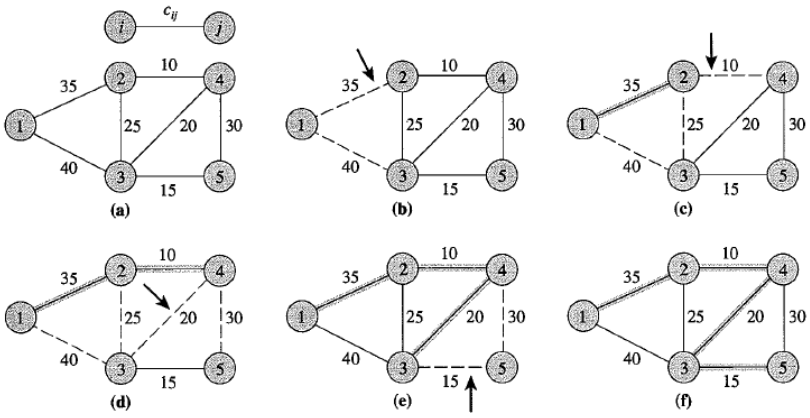


Figura 4.3: Ilustración del Algoritmo de Prim.
Imagen sacada de [2].

En la práctica cuando tenemos redes amplias con muchos arcos, resulta mucho más eficiente usar el algoritmo de Prim. Tanto por el número de operaciones (alrededor de $O(mlog(n))$), como por la implementación en un programa de ordenador. Resulta mucho más sencillo programar las iteraciones que realiza el Algoritmo de Prim, que las de Kruskal; ya que no resulta práctico comprobar en cada iteración si añadir una arista genera un ciclo o no. Sin embargo, ir inspeccionando las aristas entre nodos añadidos al árbol y no añadidos resulta mucho más sencillo de programar. Una especie de *pseudocódigo* para poder implementar el Algoritmo de Prim en un grafo $G = (N, A)$ con $|N| = n$ y pesos w_{ij} sería el siguiente:

Definición de las variables

- $W = (w_{ij})_{i,j=1}^{n}$ matriz de pesos
- *visitado(i)* vector donde pondremos si ya hemos añadido el nodo i o no
- $T = (t_{ij})$ matriz donde $t_{ij} = 1$ si y solo si lo hemos añadido al árbol
- *pred(i)* vector donde vemos el predecesor de cada nodo i
- Z donde iremos variando el peso total del árbol
- m nº de aristas que tiene el árbol

Iteraciones del Algoritmo de Prim:

Initialize:

- visitado(1)=1, empezamos el árbol desde el nodo 1

- $T = 0$ inicializamos la matriz con ceros
- pred(1)=0 el 1 no tiene predecesor
- Z=0
- $m = 0$
- M= 10000

```
while (m < n)
    w_aux = M                                   (peso auxiliar con cota grande)
    for (i, j ∈ N tal que visitado(i) = 1 y visitado(j) = 0)
        if (w_ij ≤ w_aux)
            w_aux = w_ij
            i' = i,    j' = j
        end if
    end for
    visitado(j') = 1
    m = m + 1
    pred(j') = i'
    t_i'j' = 1
    Z = Z + w_i'j'
end while
```

En la última sección vamos a resolver una serie de problemas con datos reales, donde compararemos la eficiencia de este algoritmo con otros modelos para solucionar el *MST*. En la práctica, en algunos casos los datos son una serie de puntos $x_1, x_2, ..., x_n \in \mathbb{R}^2$, que representarán las coordenadas de los distintos nodos de mi red, precisando la localización geográfica de los puntos que se necesita conectar. Estos pueden ser puntos de una ciudad que tengo que abastecer, terminales de ordenadores que hay que conectar, etc. Y los pesos w_{ij} son la distancia euclídea entre estos puntos $w_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2}$. Supondremos que cada punto x_i se puede conectar con cada punto x_j. O, en otros casos simplemente tendremos una red con puntos $x_1, x_2, ..., x_n$ y $w_{ij} \geq 0$ indicará el coste del enlace entre el punto x_i y el x_j, que no tiene porque estar relacionado con la distancia entre ellos. Para un enlace que no es posible, esto es, un arco (x_i, x_j) no esté en mi red por comodidad podemos considerar que si forman parte de la red, pero estableciendo un coste $w_{ij} = M$ con una cota alta para que no sea escogido por el algoritmo.

4.1. Formulaciones para el Problema del MST

Hemos visto en la sección anterior la presentación del problema y dos algoritmos eficientes que me van permitir resolver el problema del *MST* en un tiempo polinómico. A pesar de esto, tener formulaciones matemáticas eficientes es importante, ya que estas pueden aportar en la formulación de problemas más largos que integran otras decisiones, en aplicaciones más complejas. Vamos a ver diferentes formulaciones de Programación entera (PE) y Programación entera mixta (PEM) de este problema, en la mayoría de libros se verán las siglas IP y MIP procedentes del inglés (Véase en [20]). Para ver los conceptos básicos de la Programación Entera véase el Apéndice (A). Los modelos de PE y PEM para resolver el problema del MST, pueden ofrecer ideas teóricas y algorítmicas para resolverlo. Esto se puede extender a problemas mayores donde encontrar un MST es un subproblema.
Las diferentes formulaciones que vamos a ver, se basarán en diferentes variantes de coger las restricciones y de como enfocar el problema, gran parte de ellas se verán como un problema de flujo en redes. De los vistos en el Capítulo 2. En el capítulo 7 resolveremos algunos problemas prácticos con alguna de las formulaciones y los compararemos con los resultados obtenidos a partir del *Algoritmo de Prim*.

Antes de todo consideramos la red $G = (N, A)$ con $|N| = n$ y, para simplificar, podemos considerar que $|A| = \frac{n(n-1)}{2}$, es decir, para todo par de nodos $i, j \in N$ se tiene que $(i, j) \in A$. Además consideramos la matriz de peso $W = (w_{ij})$ con $w_{ij} \geq 0$ para cada arco $(i, j) \in A$, representando la distancia de ese arco. Luego el problema se trata de encontrar un árbol expandido $T \subset G$ con A_T las aristas del árbol, tal que $\sum_{(i,j)\in A_T} w_{ij}$ sea mínima. Sean las variables binarias $y_{ij} \in \{0, 1\}$, donde $y_{ij} = 1$ si el arco $(i, j) \in A$ es seleccionado en el árbol y si no $y_{ij} = 0$. Entonces el árbol óptimo resultante $T = (N, A_T)$ será aquel con $A_T = \{(i, j) \in A : y^*_{ij} = 1\}$, siendo y^* la solución óptima de mi problema de programación lineal.

4.1.1. Modelo de eliminación de ciclos

El primer modelo es una expresión intuitiva directa de la definición del MST. Considerando las variables de decisión binarias y_{ij}, antes descritas. El primer modelo se formula como sigue:

$$Min \qquad \sum_{(i,j)\in A} y_{ij} w_{ij} \tag{4.1}$$

Sujeto a

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{4.2}$$

$$\sum_{(i,j)\in A} y_{ij} = n-1 \tag{4.3}$$

$$\sum_{(i,j)\in A(S)} y_{ij} \leq |S|-1, \qquad \forall S \subsetneq N, |S|>1 \tag{4.4}$$

Donde $A(S)$ representa el subcojunto de arcos $A(S) \subset A$ con puntos finales en $S \subset N$. La restricción (4.3) viene de que un árbol tiene exactamente $n-1$ aristas. Y la restricción (4.4), son exactamente las restricciones de eliminación de ciclos. Ya que si $A(S)$ representa las aristas con nodos en S, la desigualdad impone que no hay más de $|S|-1$ aristas que conecten elementos de S esto es clave porque:

1. Si en un grafo con número de nodos $|S|$ se tiene que hay $|S|$ o más aristas, entonces habría al menos un ciclo.

2. Como la restricción es para todo subconjunto $S \subset N$, prohíbe que cualquier conjunto de nodos forme un ciclo.

Luego conjuntamente (4.3) y (4.4) implican que el grafo que se genera no tiene ciclos y tiene exactamente $n-1$ aristas, entonces este es un árbol. La restricción (4.2) es simplemente la restricción de dominio de las variables. Nótese que como función del número de nodos este modelo tiene un número exponencial de restricciones $(2^n - n - 2)$ lo que lo hace difícil de aplicar en la práctica. Sin embargo, vamos a ver una propiedad muy importante sobre este modelo, que es que la relajación lineal del problema (4.1)-(4.4) también resuelve el problema del mínimo árbol expandido, esto viene del hecho de que el poliedro del modelo lineal que genera las restricciones de ciclo es entero (Véase en el Apéndice A). Para demostrar esta propiedad vamos usar conceptos y propiedades fundamentales de la programación lineal, como es su problema dual y las propiedades de holguras complementarias.

La relajación lineal de (4.1)-(4.4) , se obtiene únicamente modificando las restricciones (4.2) por $0 \leq y_{ij}$, de hecho se tiene que $0 \leq y_{ij} \leq 1$ (Basta con considerar $|S| = 2$ en (4.4)). Para desarrollar su problema dual consideramos un multiplicador $\mu_N \in \mathbb{R}$ asociado a la restricción (4.3) y multiplicadores $\mu_S \geq 0$ dados por las restricciones (4.4). Entonces el problema dual es:

$$max \sum_{\emptyset \subset S \subset N} \mu_S(|S|-1) + \mu_N(n-1) \tag{4.5}$$

Sujeto a

$$\sum_{\{S:(i,j)\in A(S)\}} \mu_S + \mu_N \leq w_{ij} \qquad \forall (i,j) \in A \tag{4.6}$$

Ahora definimos unos costes reducidos w_{ij}^{μ}, de la siguiente manera.

$$w_{ij}^{\mu} = w_{ij} + \sum_{\{S:(i,j)\in A(S)\}} \mu_S + \mu_N \tag{4.7}$$

Que nos van a permitir definir la siguiente condición de optimalidad por holguras complementarias.

Teorema 4.1.1. *Una solución* $\boldsymbol{y}$ *del problema del mínimo árbol expandido es una solución óptima de la relajación lineal del problema (4.1)-(4.4) si y solo si podemos encontrar un conjunto de multiplicadores* $\mu_S \geq 0$ *y* $\mu_N \in \mathbb{R}$ *tal que se cumplan las siguientes condiciones:*

1. $w_{ij}^{\mu} = 0$ *si* $y_{ij} > 0$
2. $w_{ij}^{\mu} \geq 0$ *si* $y_{ij} = 0$

Demostración. De la teoría de la programación lineal un par (y, μ) de soluciones factibles del programa lineal primal-dual son óptimas si y solo si se cumplen las condiciones de holguras complementarias. Que para la relajación lineal del problema (4.1)-(4.4) y su problema dual (4.5)-(4.6) es precisamente que

$$y_{ij}\left[w_{ij} + \sum_{\{S:(i,j)\in A(S)\}} \mu_S + \mu_N\right] = 0 \qquad \forall (i,j) \in A$$

Donde lo de dentro de los corchetes es precisamente los pesos reducidos w_{ij}^{μ}, entonces una solución factible (y, μ) del primal-dual es óptima si y solo si

$$y_{ij} w_{ij}^{\mu} = 0 \qquad \forall (i,j) \in A$$

Por tanto, existe un conjunto de multiplicadores $\mu_S \geq 0$ y $\mu_N \in \mathbb{R}$ tal que si $y_{ij} > 0$ entonces $w_{ij}^{\mu} = 0$ y si $y_{ij} = 0$ entonces $w_{ij}^{\mu} \geq 0$. Que es precisamente lo que había que probar. □

Finalmente, estas condiciones nos van a permitir probar el Teorema Fundamental sobre esta formulación del MST.

Teorema 4.1.2. *El poliedro definido por la relajación lineal del problema es entero.(4.1)-(4.4)*

Demostración. Para demostrar esto bastaría con ver que la solución óptima y generada por el algoritmo de *Kruskal* cumplen las condiciones del Teorema 4.1.1 Si mostramos que para cualquier elección de los pesos $w_{ij} \geq 0$, encontramos al menos una solución entera del programa lineal, entonces todos los vértices del poliedro serán

enteros. Porque para cualquier elección y que sea el vector de incidencias de un árbol T, que obviamente es una solución entera, podríamos encontrar una elección de pesos $w_{ij} \geq 0$ tal que el Algoritmo de Kruskal genere exactamente ese árbol.

Luego sea $G = (N, A)$ un grafo no dirigido con pesos $w_{ij} \geq 0$ para cada arista $(i, j) \in A$. El algoritmo de *Kruskal* selecciona las aristas del mínimo árbol expandido T que genera en orden creciente de peso, evitando ciclos. Denotamos las aristas seleccionadas por $T = \{e_1, e_2, \ldots, e_{n-1}\}$, donde $n = |N|$, y $w_k := w_{i_k j_k}$, con $e_k = (i_k, j_k)$ la arista añadida en el paso k del algoritmo.
Sea $S_1 = \{i_1, j_1\}$ los nodos que forman parte de la primera arista añadida al árbol, ahora construimos S_2 en la siguiente iteración al añadir la arista (i_2, j_2), donde se tiene que al menos uno de los dos nodos es distinto a i_1 o j_1, luego construimos el conjunto de nodos S_2 a partir de S_1 añadiendo i_2 o j_2, o bien ambos; dependiendo de si en la segunda arista añadida al árbol ambos nodos son distintos o solo uno. De forma inductiva se construyen en cada paso k los conjuntos de nodos S_k, añadiendo al menos un nodo de la arista (i_k, j_k) dependiendo de que ambos o solo uno de ellos difieran de los nodos de S_{k-1}, ya que esta arista se ha añadido teniendo en cuenta que no forma ningún ciclo. De esta manera hemos construido los conjuntos S_1, S_2, ...,S_{n-2}, y N, en los que definiremos unos multiplicadores μ_{S_k} que harán que se cumpla el Teorema 4.1.1 y podamos garantizar que la solución y que genera el Algoritmo de *Kruskal* resuelve la relajación lineal. Además, para cualquier otro subconjunto $S \subset N$ distinto de los que hemos construido estableceremos que $\mu_S = 0$.
Para cada arista $e_k = (i_k, j_k)$ añadida en el paso k en *Kruskal*, definimos su multiplicador μ_{S_k} asociado como:

$$\mu_{S_k} := w_{k+1} - w_k, \quad \text{para } k = 1, \ldots, n-2.$$

Y el multiplicador libre:

$$\mu_N := -w_{n-1}.$$

Cada $\mu_{S_k} \geq 0$, ya que en el algoritmo de *Kruskal* se añaden aristas en orden creciente de peso.
Ahora veamos que con la definición de estos multiplicadores se cumplen las condiciones del Teorema anterior.
Aristas del árbol T: Sea $e_k = (i_k, j_k) \in T$, luego $y_{i_k, j_k} = 1$. Los conjuntos S que contienen a ambos extremos de e_k son:

$$S_k, S_{k+1}, \ldots, S_{n-2} \text{ y } N. \qquad k = 1, ..., n-2$$

Entonces:

$$\begin{aligned} w^{\mu}_{i_k j_k} &= w_k + \sum_{m=k}^{n-2}(w_{m+1} - w_m) + (-w_{n-1}) \\ &= w_k + (w_{n-1} - w_k) - w_{n-1} = 0. \end{aligned}$$

Entonces se cumple la primera condición. Si $y_{ij} > 0$ (es una arista del árbol) entonces $w^{\mu}_{ij} = 0$

Aristas fuera del árbol: Sea $f = (i, j) \notin T$ ($y_{ij} = 0$), entonces es una arista que forma un ciclo C al añadirse a T. Sea $e_\ell \in C$ la arista de mayor peso en ese ciclo. Por construcción de *Kruskal*, $w_{ij} \geq w_\ell$.

Los conjuntos S que afectan al coste reducido de f son:

$$S_\ell, S_{\ell+1}, \ldots, S_{n-2} \text{ y } N.$$

Entonces:

$$\begin{aligned} w^{\mu}_{ij} &= w_{ij} + \sum_{m=\ell}^{n-2}(w_{m+1} - w_m) - w_{n-1} \\ &= w_{ij} - w_\ell \geq 0. \end{aligned}$$

Por tanto, hemos construido un conjunto de multiplicadores:

$$\mu_{S_1}, \ldots, \mu_{S_{n-2}} \geq 0, \quad \mu_N \in \mathbb{R}$$

tal que:

- Toda arista del árbol generado por *Kruskal* tiene coste reducido cero.
- Toda arista fuera del árbol tiene coste reducido no negativo.

Por las condiciones de holguras complementarias del Teorema 4.1.1, esto demuestra que la solución generada por *Kruskal* es óptima para la relajación lineal del problema (4.1)-(4.4) , Y como consecuencia, se deduce que el poliedro de dicha relajación lineal tiene puntos extremos enteros. □

Nótese que esta propiedad no es casualidad pues para el problema del MST teníamos dos algoritmos como el de *Kruskal* y *Prim* que nos permitían resolver el algoritmo en tiempo polinómico. Luego, es esperable que podamos encontrar una formulación de programación entera cuya relajación lineal tenga poliedro entero, lo que quiere decir que ese modelo lineal resuelve el problema sin necesidad de imponer que las soluciones sean enteras. Aunque en este caso la relajación lineal tiene un número

exponencial de restricciones lo que lo hace imposible de implementar, a menos que usemos otro tipo de métodos, los cuáles están fuera del objetivo de este texto.

4.1.2. Formulación por cortes

El segundo modelo se basa en conjuntos de corte y en la idea de que un grafo conexo con $n-1$ aristas es un árbol. Para cualquier conjunto no vacío $S \subsetneq N$ el corte $[S, S']$ representa el conjunto de aristas que unen un nodo de S con un nodo de $N - S = S'$. La función objetivo (4.1) y las restricciones (4.2) y (4.3) se mantienen y en este modelo las restricciones de eliminación de ciclos (4.4) se modifican por:

$$\sum_{(i,j)\in[S,S']} y_{ij} \geq 1 \qquad \forall S \subsetneq N, |S| \geq 1 \tag{4.8}$$

Estas son las restricciones de conectividad, estas aseguran que cada corte tiene al menos una arista para cualquier $S \subsetneq N$ no vacío. Efectivamente estas restricciones aseguran que el grafo resultante es conexo, pues no permite la existencia de ningún subconjunto de nodos completamente aislados. Este razonamiento viene de:

- Suponiendo que el grafo no es conexo, existiría al menos una componente conexa $S \subset N$ con $|S| \geq 1$ que desconecta el grafo.
- Lo que haría que $\sum_{(i,j)\in[S,S']} y_{ij} = 0$, ya que no existe ninguna arista entre S y S'. Lo que contradice que las restricciones de conectividad (4.8) , luego el grafo es conexo.

Entonces tenemos un grafo conexo con $n-1$ aristas, que por la caracterización de los árboles, se tiene que el modelo genera un MST. El número de restricciones en (4.8) es también exponencial e igual a $2^{n-1} - 1$, después de eliminar conjuntos de corte duplicados. Para $n > 3$ el número de restricciones de corte son menores que el número de restricciones de eliminación de ciclos (4.4).

3. Formulación por multicortes

El tercer modelo esta basado en definir multicortes. Sea un $1 < K < n$, y sean $S_k \subset N$ para $k = 1, ..., K$, tal que $\cup_{k=1}^{K} S_k = N$ y $S_k \cap S_{k'} = \emptyset$ para $k \neq k'$, es decir, una partición del conjunto de nodos. Entonces un multicorte es el subconjunto de aristas $[S_1, ..., S_K] = \{(i,j) \in A : i \in S_k, j \in S_{k'}, k \neq k'\}$ y $k, k' \in \{1, ..., K\}$. Si eliminamos estas aristas del árbol tendríamos K componentes distintas. Entonces en este modelo se mantienen, nuevamente la función objetivo 4.1 y las restricciones 4.2

y 4.3 e introducimos las restricciones por multicortes:

$$\sum_{(i,j)\in[S_1,...,S_K]} y_{ij} \geq K-1, \qquad \forall S_1,...,S_K \subset N \tag{4.9}$$

Por un razonamiento muy similar al de las restricciones de cortes simples, estas restricciones mantienen el grafo conexo y se obliga nuevamente a que haya $n-1$ aristas, por tanto tenemos un MST. Para $K=2$ el conjunto de restricciones de multicorte se limita a las restricciones de corte (4.8). Este modelo tiene alguna ventaja sobre el modelo con cortes simples, sin embargo no existe ninguna pista del límite superior que puede tomar K para el cual siga habiendo soluciones factibles y se pueda encontrar el MST. De hecho:

Proposición 4.1.1. *Las restricciones (4.9) no previenen la existencia de ciclos cuando* $K = n-1$

Demostración. Se observa que para un conjunto no vacío arbitrario de nodos $\sigma \subset N$, las restricciones (4.9) cuando $K = N-1$ no imposibilitan la siguiente condición:

$$\sum_{i\in\sigma} \sum_{j\in N-\sigma} y_{ij} = 0$$

Por tanto, se podrían formar ciclos. □

Acerca del tamaño de este modelo, el número de restricciones que hay en (4.9) es el número de Stirling [1] de segunda clase, denotado como $\left\{ {n \atop K} \right\}$. Se sabe que $\left\{ {n \atop n-1} \right\} = \frac{n(n-1)}{2}$, que es polinómico pero ya hemos visto que $K = n-1$ no garantiza soluciones factibles por la proposición anterior. Mientras que $\left\{ {n \atop K} \right\}$ es exponencial en n cuando $2 \leq K < n-1$.

4.1.3. Modelos con dígrafos

El problema del mínimo árbol expandido es equivalente a encontrar el conjunto de caminos más cortos que empiezan en un nodo raíz arbitrario, al que conectaremos con los demás nodos. El modelo que encuentra estos caminos con longitud total mínima para todos, encuentra también un MST. Basado en este concepto vamos a poder formular el problema del MST como un problema de flujo en redes.
Para empezar la red $G = (N, A)$ que es no dirigida tenemos que considerarla dirigida,

[1] El número de Stirling de segunda clase cuenta cuántas maneras hay de particionar un conjunto de n elementos en exactamente k subconjuntos no vacíos y disjuntos. Se puede calcular recursivamente como : $\left\{ {n \atop k} \right\} = k \cdot \left\{ {n-1 \atop k} \right\} + \left\{ {n-1 \atop k-1} \right\}$

ya que en un problema de flujo los arcos son siempre dirigidos. Lo hacemos de la siguiente manera: para cada arco no dirigido $(i, j) \in A$ consideramos los dos arcos dirigidos (i, j) y (j, i). Podemos considerar el nuevo conjunto de arcos como D.
De manera bastante análoga se pueden definir modelos como el de las restricciones de eliminación de ciclos y el de las restricciones de cortes, añadiendo la restricción de que si una arista $(i, j) \in A$ es seleccionada en el árbol entonces debe pasar flujo o de i a j o de j a i. Se puede hacer creando las variables de flujo z_{ij}, z_{ji} para cada arco (i, j) y con la restricción:

$$y_{ij} = z_{ij} + z_{ji} \qquad \forall (i,j) \in A \tag{4.10}$$

Nuevamente estos modelos tienen un número exponencial de restricciones.

Formulación de problema de flujo en redes

El problema del *MST* se puede formular como un problema de flujo en redes de 1 producto con costes fijos. Para considerarlo como un problema de flujo añadimos variables de flujo x_{ij} ficticias, ya que los costes variables c_{ij} asociados a la cantidad de flujo transportado de i a j van a ser nulos. Los costes fijos de mi problema serán las distancias w_{ij}. El modelo se basa en elegir un nodo arbitrario que va a ser fijado como nodo fuente y los demás nodos están definidos como nodos demanda. A ese nodo fuente lo denotaremos como la raíz. Entonces tendríamos la siguiente formulación:

$$Min \qquad \sum_{(i,j)\in A} w_{ij} y_{ij} \tag{4.11}$$

Sujeto a:

$$\sum_{(i,j)\in A} y_{ij} = n - 1 \tag{4.12}$$

$$\sum_{(r,j)\in A} x_{rj} \quad - \sum_{(r,j)\in A} x_{jr} = n - 1 \tag{4.13}$$

$$\sum_{(i,j)\in A} x_{ij} \quad - \quad \sum_{(i,j)\in A} x_{ji} = -1 \quad , \forall i \in N, i \neq r \tag{4.14}$$

$$x_{ij} \leq (n-1) y_{ij} \qquad x_{ji} \leq (n-1) y_{ij} \quad \forall (i,j) \in A \tag{4.15}$$

$$x_{ij} \geq 0 \qquad y_{ij} \in \{0, 1\}, \quad \forall (i,j) \in A \tag{4.16}$$

(4.13) y (4.14) son las ecuaciones de balance de mi problema de flujo.(4.13) indica que el nodo raíz envía exactamente $n-1$ unidades de flujo. Y mientras (4.14) indica que todos los demás nodos consumen 1 una unidad de flujo cada una, entre el flujo de entrada y de salida.
(4.15) son las que hacen que el flujo solo pueda pasar por arcos activos, y como cota podemos poner $n-1$. Ya que si el arco (i,j) no es seleccionado en el árbol ($y_{ij}=0$), entonces ambos $x_{ij}=x_{ji}=0$, y si el arco sí que se selecciona ($y_{ij}=1$), las restricciones (4.13) y (4.14) obligan que al menos una de las variables de flujo x_{ij}, x_{ji} tomen un valor estrictamente positivo. Con la función objetivo, las restricciones (4.11)-(4.16) conjuntamente hacen que:

- Para que todos los nodos de la red reciban una unidad de flujo a partir del nodo raíz, la red necesariamente debe estar conectada.
- Como el flujo no puede pasar por arcos no activados, se necesita una red que conecte los n nodos solo por arcos activos,
- Y además, $\sum_{(i,j)\in A} y_{ij} = n-1$.

Por tanto, las restricciones obligan a tener una red conectada y con $n-1$ aristas, entonces se genera un árbol expandido de coste mínimo.
Las 2 anteriores formulaciones tenían un número exponencial de restricciones, en este caso planteando el problema como un problema de flujo tenemos una cantidad polinómica de restricciones.

Formulación de flujo en redes multiproducto

Nuevamente teniendo la red no dirigida $G=(N,A)$ podemos considerar su dígrafo asociado doblando las aristas $(i,j)\in A$ como (i,j) y (j,i), este conjunto de arcos dirigidos lo podemos llamar D y considerar la red dirigida $G=(N,D)$, con $|N|=n$. Como en el caso de la formulación de 1 producto, elegimos un nodo raíz r a partir del cual vamos a mandar el flujo a los demás nodos $K=N-\{r\}$, que será nuestro conjunto de productos. Entonces, podemos considerar las variables de decisión f_{ij}^k, que indican la cantidad de flujo de la raíz r hacia $k\in K$, a través del arco (i,j), y entonces en el marco de un problema de flujo lo podemos entender como la cantidad de flujo del producto k de i hacia j. Las ecuaciones de conservación o ecuaciones de balance, definen parte de las restricciones de estas variables:

$$\sum_{(i,k)\in D} f_{ik}^k \quad - \quad \sum_{(k,j)\in D} f_{kj}^k = 1 \qquad \forall k \in N-\{r\} \tag{4.17}$$

$$\sum_{(r,j)\in D} f_{rj}^k - \sum_{(i,r)\in D} f_{ir}^k = 1 \qquad \forall k \in N - \{r\} \tag{4.18}$$

$$\sum_{(i,v)\in D} f_{iv}^k - \sum_{(v,j)\in D} f_{vj}^k = 0 \qquad \forall v \in N - \{k, r\}, \forall k \in N - \{r\} \tag{4.19}$$

$$f_{ij}^k \geq 0 \quad \forall k \in N - \{r\}, \forall (i,j) \in D \tag{4.20}$$

Por las restricciones (4.17) se entiende que cada nodo $k \in N - \{r\}$ recibe exactamente una unidad del producto k, esto traducido, es que a cada nodo diferente de la raíz le llega exactamente una unidad de flujo neta desde la raíz, entre el flujo que entra (arcos (i,k)) y el flujo que sale (arcos (k,j)). Las restricciones (4.18) indican que de la raíz sale una unidad de flujo hacia cada nodo $k \in N - \{r\}$. Y por (4.19) el flujo a través de otros nodos hacia $k \in N - \{r\}$ se debe consumir, es decir, el que entra menos el que sale es igual. De manera similar al modelo que vimos de un producto debemos relacionar las variables de flujo con las variables binarias, que indican si un arco es escogido o no. Lo hacemos con las siguientes restricciones:

$$f_{ij}^k \leq y_{ij} \qquad f_{ji}^k \leq y_{ij} \quad \forall k \in N - \{r\}, \forall (i,j) \in A \tag{4.21}$$

Ya que si el arco (i,j) no es seleccionado ($y_{ij} = 0$), eso implicaría que $f_{ij}^k = f_{ji}^k = 0$ para todo $k \neq r$, es decir , que no habría ningún flujo de r hacia cualquier $k \neq r$ de i hacia j, o viceversa. Esto ocurre porque no es rentable mandar esa unidad de flujo desde la raíz hacia cualquier otro nodo pasando por el arco (i,j), lo que es equivalente a que este arco no puede estar en el árbol.
Por tanto combinando las restricciones (4.17)-(4.21), con (4.11),(4.12) y con que $y_{ij} \in \{0,1\}$, tenemos un problema de Programación Entera Mixta que resuelve el MST.
Otro modelo a mayores puede ser considerado cambiando las restricciones 4.21 por el conjunto de restricciones:

$$f_{ij}^k + f_{ji}^{k'} \leq y_{ij} \qquad \forall k, k' \in N - \{r\}, \forall (i,j) \in A \tag{4.22}$$

4.1.4. Formulación de Miller-Tucker-Zemlin

Esta formulación está basada en unas restricciones de eliminación de ciclos que fueron introducidas por Miller-Tucker-Zemlin[17] (1960). Consideramos el grafo $G = (N, A)$ y nuevamente consideramos un nodo arbitrario r desde el que empezamos el

árbol, al que llamamos raíz. En esta formulación, además de usar las usuales variables binarias y_{ij}, donde $y_{ij} = 1$ si y solo si (i, j) está en el *MST*, se introducen otro tipo de variables u_i para cada nodo $i \in N - \{r\}$ que indican en que posición del árbol se encuentra ese nodo respecto de la raíz. Esto es, la cantidad de arcos que contiene el único camino entre el nodo i y la raíz r. Luego la formulación para este modelo es la siguiente:

$$Min \qquad \sum_{(i,j)\in A} w_{ij} y_{ij} \tag{4.23}$$

Sujeto a

$$\sum_{(i,j)\in A} y_{ij} = n - 1 \tag{4.24}$$

$$nx_{ij} + u_i \leq u_j + (n-1) \qquad i, j \in N - \{r\} \tag{4.25}$$

$$1 \leq u_i \leq n-1 \qquad y_{ij} \in \{0, 1\} \qquad i \in N - \{r\}, (i, j) \in A \tag{4.26}$$

Donde (4.24) es la restricción de que un árbol debe tener exactamente $n - 1$ arcos. (4.26) son el dominio de las variables, es claro que un nodo que no es la raíz debe tener estar al menos a 1 de distancia de la raíz y como mucho un camino puede tener $n - 1$ arcos que son los que tiene el árbol. Y las restricciones (4.25) son las ya mencionadas restricciones de *Miller-Tucker-Zemlin* y son las que aseguran que la solución no contenga ciclos, veamos esto:

Si $y_{ij} = 0$ se tiene la desigualdad $u_i \leq u_j + (n - 1)$ que se cumple siempre dado el dominio de las variables.
Si $y_{ij} = 1$ la desigualdad se convierte en $u_i + 1 \leq u_j$, es decir, que el nivel del nodo j debe ser al menos uno más que el del nodo i. Veamos porque imposibilita la creación de ciclos.
Supongamos que existiera un ciclo $i_1 - i_2 - \cdots - i_k - i_1$ en el árbol, luego para cada arista (i_l, i_{l+1}) e (i_k, i_1) con $l = 1, ..., k - 1$ se tiene que $y_{i_l i_{l+1}} = 1$ e $y_{i_k i_1} = 1$ luego se cumplen las desigualdades de *MTZ*:

$$u_{i_l} + 1 \leq u_{i_{l+1}} \text{ y } u_{i_k} + 1 \leq u_{i_1}$$

Luego sumando todas las desigualdades se tiene que

$$u_{i_1} + k \leq u_{i_1}$$

Que es imposible pues $k \geq 1$. Por tanto, en la solución no se forman ciclos y además tenemos $n - 1$ aristas. Luego, la solución óptima es un *MST*. Las formulaciones con variables de salto son especialmente importantes para alguna de las extensiones del MST, donde restringiendo el número de saltos obtenemos una formulación válida para

esos problemas.

Capítulo 5

Problema del Mínimo Árbol con capacidades

El Problema del Mínimo Árbol Expandido con Capacidades o *the Capacitated Minimal Spanning Tree (CMST)* en inglés es una de las extensiones más importantes al problema del mínimo árbol expandido. El objetivo es similar, que es encontrar un mínimo árbol expandido desplegado desde un nodo raíz r, aunque en este caso sujeto a unas restricciones de capacidad. Estas restricciones de capacidad limitan el número de nodos que cuelgan del nodo raíz.
Primero, dado un árbol expandido T desplegado desde el nodo raíz r, se considera el bosque formado de eliminar el nodo r y todas las aristas incidentes en él. Nos referimos a cada componente del bosque como *los subárboles fuera del nodo r*. A cada

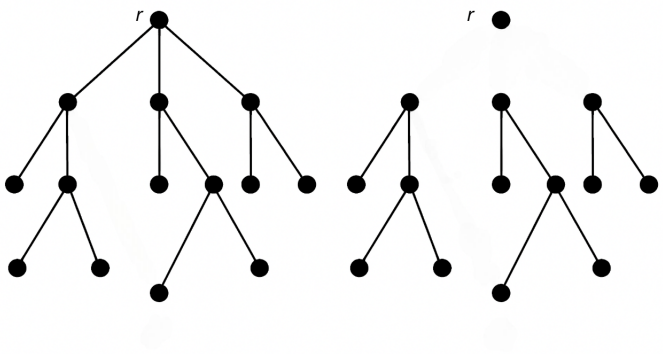


Figura 5.1: Ilustración de un árbol expandido desplegado desde un nodo r comparado con el bosque formado al eliminar todas las aristas incidentes en r.
Elaboración propia.

nodo i se le puede asociar una demanda $d_i > 0$ que debe ser enviada desde el nodo raíz. Y también podemos considerar como D la demanda total de cada subárbol fuera del nodo r, que consiste en sumar las demandas de cada nodo que forman parte de él. Por tanto, las restricciones de capacidad son las que garantizan que la demanda total de cada subárbol fuera de r no supere una cierta capacidad $Q > 0$. Entonces el

CMST puede verse teóricamente como el siguiente problema de optimización de grafos:

Dado un grafo $G = (N, A)$ con $N = \{0, 1, ..., n\}$, donde w_{ij} es el costo o distancia de cada arco $(i, j) \in A$, $d_i > 0$ las demandas de cada nodo i y sea Q un número positivo; *el problema del mínimo árbol expandido con capacidades* consiste en encontrar un árbol expandido desplegado desde r, cumpliendo las restricciones de capacidad, con el mínimo coste.

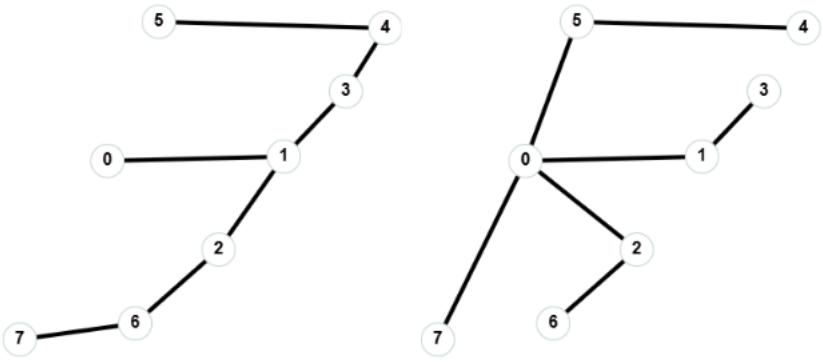


Figura 5.2: Comparativa entre un árbol mínimo sin restricciones y un árbol con restricciones de capacidad ($Q = 2$).

Elaboración propia.

Normalmente para simplificar escogemos que el nodo central o nodo raíz sea el nodo 0, este será el nodo desde el que iremos construyendo las distintas ramas del árbol sin superar la capacidad Q en cada una. Denotaremos a los demás nodos que no son el nodo raíz como los terminales, y consideramos el conjunto $N^+ = N - \{0\}$. Aunque el *CMST* previamente se pueda definir en redes no dirigidas, los modelos que describiremos a continuación se definen sobre grafos dirigidos. Entonces los casos de redes no dirigidas se pueden transformar fácilmente en un modelo con un grafo dirigido, si remplazamos los arcos (i, j) por los arcos dirigidos (i, j) y (j, i) con idéntico coste asociado, si $i, j \in N^+$. Y los arcos no dirigidos $(0, i)$ solo los remplazaremos por los arcos dirigidos $(0, i)$ con $i = 1, ..., n$, es decir, que no consideraremos arcos entrando en la raíz. También supondremos como en los modelos de *MST* que para cada nodo $i \in N$ y cada nodo $j \in N^+$ existe un arco dirigido (i, j) que los conecta.
Un caso particular con especial interés ocurre cuando la demanda de cada nodo i es la misma, este caso es conocido como el de *demanda unitaria*. En este caso, podemos considerar que $d_i = 1$ para todo $i \in N^+$. El caso general donde las demandas no sean iguales, siempre podemos transformarlo en el caso unitario. La transformación se

obtiene remplazando los nodos originales i con $d_i > 1$ con el conjunto de vértices artificiales M con $|M| = d_i$. Además dos tipos de arcos artificiales remplazan los originales que incidían en i. El primer tipo conecta cada par de vértices $m, p \in M$ con $w_{mp} = 0$. El segundo tipo tiene coste $w_{mj} = w_{ij}$ para cada par de vértices $m \in M, j \in N^+$.

A diferencia del problema del mínimo árbol expandido, donde teníamos algoritmos eficientes que resolvían el problema, para el CMST no disponemos de ningún algoritmo polinómico que lo resuelva. En esta sección veremos diferentes formulaciones a este problema como modelos de programación entera. Algunas de ellas como en el caso del MST basados en formulaciones de problemas de flujo con costes fijos. Las formulaciones que veremos están especificadas para el caso de demanda unitaria, a menos que se diga lo contrario.

5.1. Formulaciones para el CMST

Primero vamos a presentar una formulación básica que usa únicamente variables binarias. De manera similar a alguna formulación que presentamos en el caso del MST, están basadas en argumentos combinatorios por enumeración. Más adelante presentaremos formulaciones más sofisticadas, donde se usan variables de flujo u otras técnicas que mejoran en gran medida los resultados en la práctica.

5.1.1. Formulación directa

Es una formulación matemática clásica del *CMST*, donde cada arco sale de la raíz hacia las hojas y el conjunto de restricciones garantizan la conectividad y la capacidad límite. Las variables de decisión de mi problema son $x_{ij} \in \{0, 1\}$, donde $x_{ij} = 1$ si el arco $(i, j) \in A$, con $j \in N^+$, pertenece al árbol solución y $x_{ij} = 0$ en otro caso.
Entonces, tenemos la siguiente formulación para el problema del *CMST*, que denotaremos como **FD**:

Formulación FD

$$min \sum_{i \in N} \sum_{j \in N^+} w_{ij} x_{ij} \tag{5.1}$$

Sujeto a

$$\sum_{i \in N} x_{ij} = 1 \qquad j \in N^+ \tag{5.2}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \lceil \frac{d(S)}{Q} \rceil \qquad S \subseteq N^+, |S| \geq 2 \tag{5.3}$$

$$x_{ij} \in \{0,1\} \qquad i \in N, j \in N^+ \tag{5.4}$$

Con las ecuaciones (5.2) garantizamos que a cada nodo fuera de la raíz, le llegue únicamente una arista. Las restricciones (5.3) son las clásicas restricciones que garantizan la conectividad, como vimos en la sección del *MST*, extendidas para el caso con capacidades. La función $\lceil . \rceil$ es la función que redondea por encima (i.e $\lceil 2{,}2 \rceil = 3$). El sumatorio indica el número de aristas que salen fuera de S y entran en un nodo de S, como ese sumatorio debe ser al menos uno, se garantiza que cada corte $[S', S]$ tiene al menos una arista, que como vimos en la sección anterior implica la conectividad. Recordando que $d(S) = \sum_{j \in S} d_j$ denota la demanda total del conjunto de nodos S. Entonces se tiene que estas restricciones también garantizan que no se excede la capacidad, ya que:

- $\lceil \frac{d(S)}{Q} \rceil$ indica la proporción entre la demanda total de S y la capacidad Q redondeando hacia arriba, ya que si la proporción es 2.5 no podemos llevar esa demanda desde 2.5 arcos, serán necesarios 3.
- Si $d(S) \leq Q$ se tiene que se puede llevar la demanda de S sin problema de sobrepasar la capacidad total. Ya que el número de arcos que lleguen a S puede ser 1 o más.
- Si $d(S) > Q$ debe llegar la demanda de S desde más de una arista (dependiendo de la proporción que haya), porque en caso contrario agotaríamos la capacidad.

Por ejemplo, si la proporción es de 2.5 deberíamos hacer llegar la demanda de S desde al menos 3 caminos diferentes, si fuese por menos habría algún subárbol que excedería la capacidad.
Esta formulación tiene n^2 variables y un número exponencial de restricciones en $|N^+|$. Nótese que hay muchas variables que no se utilizan, como cualquier arco que va desde un nodo terminal hacia la raíz.

5.1.2. Formulación como problema de flujo con 1 producto

Esta formulación como problema de flujo de 1 producto se debe a *Gavish [8][7](1982, 1983)*. Se consideran las variables de diseño binarias x_{ij}, tal que $x_{ij} = 1$ si el arco (i, j) está en el CMST y $x_{ij} = 0$ en otro caso. Además consideramos las variables de

flujo $y_{ij} \geq 0$ que representan la cantidad de flujo producida por la raíz a través del arco (i, j). Y por último, consideramos el vector $d = (d_0, d_1, ..., d_n)$ tal que $d_0 = 0$ y $d_i = 1$ para $i = 1, 2, ..., n$. Entonces tenemos la siguiente formulación para el *CMST*, que denotaremos por **FF**:

Formulación FF

$$min \quad \sum_{i \in N} \sum_{j \in N^+} w_{ij} x_{ij} \tag{5.5}$$

Sujeto a

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in N^+ \tag{5.6}$$

$$\sum_{i \in N} y_{ij} \quad - \quad \sum_{i \in N^+} y_{ji} = 1 \qquad \forall j \in N^+ \tag{5.7}$$

$$x_{ij} \leq y_{ij} \leq (Q - d_i) x_{ij} \qquad \forall i \in N, \quad \forall j, \in N^+ \tag{5.8}$$

$$y_{ij} \geq 0, \quad x_{ij} \in \{0, 1\} \qquad \forall i \in N, \quad \forall j, \in N^+ \tag{5.9}$$

Para facilitar los índices no estamos considerando las variables x_{ii} e y_{ii} para $i \in N^+$. Las restricciones (5.6) aseguran que cada nodo está en la solución, ya que obliga a que para cualquier nodo distinto de la raíz entre solo un arco. Las restricciones (5.7) son las ecuaciones de balance para cada nodo terminal, estas como en el caso del *MST* aseguran la conectividad; pues cada nodo terminal consume exactamente una unidad de flujo, que es mandado desde el nodo raíz. Además, como las restricciones (5.6) implican que haya n arcos se genera un árbol expandido (en este caso consideramos $n+1$ nodos). Finalmente, (5.8) garantiza que únicamente exista flujo a través de aristas activas y que no se sobrepase la capacidad Q. Es claro comprobar que $x_{ij} = 0$ si y solo si $y_{ij} = 0$. También es sencillo ver que la cantidad de flujo y_{ij} a través del arco (i, j) indica el número de nodos que se desconectarían de la raíz si eliminamos el arco (i, j). Por tanto, cada subárbol que cuelga de la raíz no puede superar la capacidad Q, pues $y_{0j} \leq Q$ el flujo que sale desde la raíz no puede superar Q. Y esto es precisamente la definición de árbol expandido con restricción de capacidad que habíamos dado.
Algunas observaciones son las siguientes:

1. Nótese que la ecuación de balance para la raíz:

$$\sum_{i \in N^+} y_{0i} = n,$$

no se pone, pues es redundante.

2. Además el conjunto de restricciones (5.8) se puede remplazar por

$$x_{ij} \leq y_{ij} \leq Qx_{ij} \qquad i \in N,\ j \in N^+ \tag{50'}$$

y se sigue garantizando que $y_{0j} \leq Q$ lo que da una formulación válida para el *CMST*. El motivo por el que se usan las restricciones (5.8) es porque la relajación lineal para las restricciones (50′) es más débil que para la formulación original.

3. Se tiene que en cualquier solución factible para el *CMST*, el flujo en un arco (i, j) en posición t (i.e. el número de arcos/saltos en el único camino desde el nodo raíz hasta el nodo j es igual a t) no puede ser mayor que $Q - t + 1$ pues al menos $t - 1$ unidades de flujo han sido consumidas por los $t - 1$ nodos intermedios del camino entre la raíz y j. Luego, la deficiencia de las restricciones (5.8) son que tienen el mismo límite superior para el flujo en los arcos incidentes en nodos terminales y que están en distintas posiciones del árbol. Y aunque haya $2n^2$ restricciones puede conllevar a programas lineales bastantes grandes, aún con órdenes de n bastante moderados. Entonces esto nos sugiere que debemos mejorar estas cotas, restringiendo el número máximo de flujo cuando nos vamos alejando de la raíz. Así vamos a obtener formulaciones más restrictivas y con relajaciones lineales mejores. En las dos siguiente formulaciones veremos una manera de conseguir esto.

5.1.3. Formulación con 2n restricciones para el CMST

Está formulación fue presentada por *Luis Gouveia (1995)*[10] con el objetivo de presentar una formulación más compacta y con mejor relajación lineal que las conocidas hasta el momento. Veremos en esta formulación que la información que daban las variables x_{ij} e y_{ij} se puede compactar en un conjunto de variables binarias con triple índice: z_{ijq} con $i \in N$, $j \in N^+$ y $q = 1, ..., Q - d_i$, tal que $z_{ijq} = 1$ si existe flujo de valor q a través del arco (i, j) y $z_{ijq} = 0$ en otro caso. Las dos siguientes relaciones muestran como se puede relacionar las variables x_{ij} e y_{ij} de la anterior formulación con las nuevas variables.

$$x_{ij} = \sum_{q=1}^{Q-d_i} z_{ijq} \qquad i \in N, j \in N^+ \tag{5.10}$$

$$y_{ij} = \sum_{q=1}^{Q-d_i} q z_{ijq} \qquad i \in N, j \in N^+ \tag{5.11}$$

Las relaciones (5.10) y (5.11) imponen que para cada arco (i, j) si $x_{ij} = 1$ e $y_{ij} = p$ entonces $z_{ijp} = 1$ y $z_{ijq} = 0$ para $q = 1, ..., Q - d_i$ y $q \neq p$. Claramente también se tiene que $x_{ij} = y_{ij} = 0$ si y solo si $z_{ijq} = 0$ para $q = 1, ..., Q - d_i$. La nueva formulación se obtiene de remplazar en (5.6)-(5.9) las variables x_{ij} e y_{ij} por las relaciones con z_{ijq} de (5.10) y (5.11). Lo más importante de esta transformación es que las restricciones que habíamos impuesto para que se no se excediese la capacidad en (5.8), reescritas en las nuevas variables siempre se cumplen y por tanto podemos omitirlas. Está es la razón por lo que está formulación implica un número de restricciones del orden de $O(n)$ para la formulación del *CMST*. La formulación, que denotaremos por **FQ** es la siguiente:

Formulación FQ

$$min \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{q=1}^{Q-d_i} w_{ij} z_{ijq} \tag{5.12}$$

Sujeto a

$$\sum_{i=0}^{n} \sum_{q=1}^{Q-d_i} z_{ijq} = 1 \qquad j \in N^+ \tag{5.13}$$

$$\sum_{i=0}^{n} \sum_{q=1}^{Q-d_i} q z_{ijq} \quad - \quad \sum_{i=1}^{n} \sum_{q=1}^{Q-1} q z_{ijq} = 1 \qquad j \in N^+ \tag{5.14}$$

$$z_{ijq} \in \{0, 1\} \qquad i \in N \quad j \in N^+ \quad q = 1, ..., Q - d_i \tag{5.15}$$

Como la anterior formulación para simplificar los índices no consideramos las variables z_{iiq} para $i = 1, ..., n$. Nótese que las restricciones de capacidad están garantizadas por como varía el índice q, pues si $z_{ijp} = 1$ tenemos que $x_{ij} = 1$ e $y_{ij} = p$, con $p = 1, ..., Q - d_i$ y nos quedaría

$$p \leq Q - d_i$$

lo cual nunca superaría la capacidad máxima para cualquier i.
Una simple consecuencia de las transformaciones (5.10) y (5.11) es que existe una correspondencia uno a uno entre las soluciones factibles de la formulación **FF** y las de **FQ**. Es por esto que si existe una solución factible para **FF**, también existe una solución factibles para **FQ** y con el mismo coste. Es por esto que las soluciones optimas para ambas formulaciones tienen el mismo valor para la función objetivo.

5.1.4. Formulación con índice de saltos

Está formulación fue presentada por *Gouveia* y *Martins* (1999)[13], y fue mejorada en (2005)[14]. Está formulación también está basada en la formulación **FF** de *Gavish* y la aplicamos también al caso de demanda unitaria. Se basa en el hecho de que cada

arco tiene una relación de *distancia* con la raíz, a la que llamaremos profundidad. Por tanto, decimos que el arco (i, j) tiene profundidad t, si el único camino desde la raíz hasta j tiene distancia (nº de arcos) t. Así, si $t = 1$ significa que i es la raíz ($i = 0$) y el destino j es una subraíz. Cuando $t = 2$, el arco (i, j) tiene que i es un subraíz y j no es ni raíz ni subraíz y ya arcos con profundidades de mayor o igual orden que 3 conectan nodos que ninguno de ellos son raíz ni subraíz. La profundidad de un nodo la podemos definir como la profundidad del arco entrante en él.
Luego, para esta formulación definimos las siguientes variables:

$$u_{ijt} = \begin{cases} 1, & \text{si } (i,j) \text{ con profundidad t está en la solución} \\ 0, & \text{en otro caso} \end{cases}$$

$z_{ijt} \geq 0$: el flujo a través del arco (i, j) cuando el arco tiene profundidad t. Teniendo para ambas $i \in N$, $j \in N^+$ y $t = 1, ..., Q$,
Nótese que el rango de t debe tener valores entre 1 y Q, pues ninguna solución factible del *CMST* tendrá algún arco con profundidad mayor que Q. Por consistencia, los pares de variables (u_{0jt}, z_{0jt}) con $j \in N^+; t = 2, ..., Q$ no las consideramos porque arco que sale desde la raíz debe tener profundidad 1. Lo mismo ocurre con las variables (u_{ij1}, z_{ij1}) con $i, j \in N^+$ porque ningún arco que no parte desde la raíz puede tener profundidad 1. Como estamos considerando que $d_i = 1$ para $i \in N^+$, la máxima capacidad Q para el árbol no indica más que la profundidad máxima que hay en un subárbol. Entonces usando estas variables, la formulación con índice de saltos denotada por **FH** es como sigue:

Formulación FH

$$min \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{t=1}^{Q} w_{ij} u_{ijt} \tag{5.16}$$

$$\sum_{i=0}^{n} \sum_{t=1}^{Q} u_{ijt} = 1 \qquad j \in N^+ \tag{5.17}$$

$$\sum_{i=0}^{n} z_{ijt} \quad - \quad \sum_{i=1}^{n} z_{ji,t+1} = \sum_{i=0}^{n} u_{ijt} \qquad j \in N^+; t = 1, ..., Q-1 \tag{5.18}$$

$$u_{ijt} \leq z_{ijt} \leq (Q - t + 1) u_{ijt} \qquad \forall i \in N \quad \forall j \in N^+ \quad t = 1, ..., Q \tag{5.19}$$

$$u_{ijt} \in \{0, 1\}, z_{ijt} \geq 0 \quad \forall i \in N \quad \forall j \in N^+ \quad t = 1, ..., Q \tag{5.20}$$

Nuevamente para simplificar los índices no consideramos las variables u_{iit} y z_{iit} ($i = 1, ..., n$). Las restricciones (5.18) son la generalización para estas variables de las ecuaciones de balance y se establecen así porque si un nodo j tiene una profundidad de t (i.e. un arco (k, j) tiene profundidad t en la solución), entonces el flujo entrando en ese nodo viene de un arco entrante con profundidad t y el flujo saliendo de él se

hace a través de un arco saliente con profundidad $t+1$. Las restricciones (5.19) son las nuevas restricciones de capacidad y eran las que nos motivaron a introducir esta nueva formulación, tal como vimos en el final de la formulación 5.1.2. La relajaciones lineal de esta formulación se obtienen de cambiar $0 \leq u_{ijt}$ en (5.20) . No es necesario imponer el límite superior $u_{ijt} \leq 1$, pues las restricciones (5.17) ya lo aseguran. Podemos obtener la relación entre las variables de la formulación de flujos **FF** y la de saltos **FH** como:

$$\sum_{t=1}^{Q} u_{ijt} = x_{ij} \quad y \quad \sum_{t=1}^{Q} z_{ijt} = y_{ij} \quad i \in N; j \in N^{+}$$

y como en la formulación anterior obtenemos una correspondencia entre ambas formulaciones. Se puede comprobar que este modelo genera mejores cotas en la relajación lineal que el modelo de flujos y las grandes mejoras se obtienen con el nodo raíz en la esquina.[1].

[1]Como se ve en la sección 9 de [13]

Capítulo 6

Problema del mínimo árbol expandido con restricciones de salto

El Problema del Mínimo Árbol Expandido con restricciones de salto o *the Hop-Constrained Minimum Spanning Tree Poroblem (HMST)* del inglés es otra de las extensiones al problema del *MST*. Que surgió del hecho de querer diseñar redes con mayor densidad de conexiones para mejorar las interconectividad entre los terminales y el nodo central. El *HMST* lo podemos definir como el siguiente problema de optimización con grafos:

Sea $G = (N, A)$ con conjunto de nodos $N = \{0, 1, ..., n\}$ y el conjunto de arcos no dirigidos (i, j) conectando los nodos $i, j \in N$. Además, consideramos el coste o distancia w_{ij} para cada arista $(i, j) \in A$ y un número natural H. El problema consiste en encontrar un árbol expandido T con el mínimo coste posible y tal que cualquier camino empezado en el nodo raíz (nodo 0) no tenga más de H saltos, es decir, que en el camino no actúen más de H arcos. Nótese que el *HMST* es trivial cuando $H = 1$ y las restricciones de salto son redundantes cuando $H \geq n$.

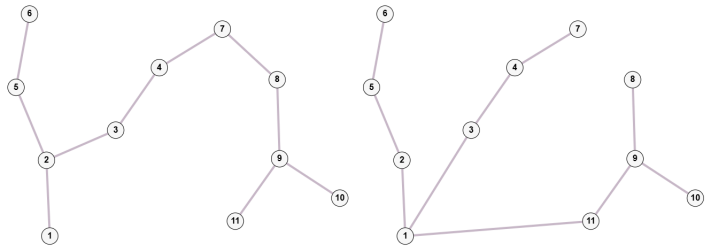


Figura 6.1: Comparativa entre un árbol mínimo sin restricciones y un árbol con restricciones de salto ($H = 3$).

Elaboración propia.

Para el HMST tampoco disponemos de algoritmos polinómicos que lo resuelvan. Luego, nuevamente, presentaremos varias formulaciones para el problema con distintas variables y restricciones. Las formulaciones que presentamos están basadas en grafos dirigidos luego, como de costumbre, para cada arco no dirigido (i, j) con $i, j \in N^+ = \{1, ..., n\}$ consideramos los arcos dirigidos (i, j) y (j, i) y para los arcos $(0, i)$ solo consideramos los que salen de la raíz.

6.1. Formulaciones para el HMST

Primero vamos a presentar un modelo básico e intuitivo para el problema del HMST, que podremos refinar considerando restricciones más fuertes. Además, como en los demás problemas también presentaremos una formulación basada en un modelo de flujo.

6.1.1. Modelo Básico

Primero veremos este modelo básico donde únicamente usaremos variables binarias para cada arista (i, j), y las ya conocidas restricciones de eliminación de ciclo de *MTZ*, que ya mencionamos en la formulación4.1.4 del MST.
Para ello definimos las usuales variables binarias x_{ij} con $i \in N, j \in N^+$, tal que $x_{ij} = 1$ si y solo si el arco (i, j) está en la solución y $x_{ij} = 0$ en otro caso. Consideramos también otras variables u_i con $i \in N^+$ que especifican en que posición del árbol se encuentra el nodo i, es decir, el número de arcos que tiene el único camino entre ese nodo y la raíz. Las soluciones no factibles (aquellas con caminos de más de H arcos) se pueden eliminar sencillamente incluyendo cotas adecuadas a estas variables

u_i. Consideramos, entonces, la siguiente formulación para el *HMST* que denotaremos por **MTZ** por los autores *Miller-Tucker-Zemlin*.

Formulación MTZ

$$min \sum_{i=0}^{n} \sum_{j=1}^{n} w_{ij} x_{ij} \tag{6.1}$$

Sujeto a

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad j \in N^+ \tag{6.2}$$

$$n x_{ij} + u_i \leq u_j + (n-1) \qquad i, j \in N^+ \tag{6.3}$$

$$1 \leq u_i \leq H \quad i \in N^+ \tag{6.4}$$

$$x_{ij} \in \{0, 1\} \qquad i \in N, j \in N^+ \tag{6.5}$$

Para simplificar los índices, no consideramos las variables x_{ii} para $i \in N^+$. Las restricciones (6.3) son las que garantizan que no se forman ciclos (véase en (4.25)). Finalmente para ver que esta formulación es válida para el HMST, habría que ver que cualquier solución factible $\{x_{ij}, u_i\}$ impide que existan caminos desde la raíz con más de H arcos. Esto es garantizado por las restricciones (6.4), para verlo supongamos que la solución contiene un camino desde la raíz (nodo 0) hasta i con más de H arcos. Sea j otro nodo situado en el mismo camino inmediatamente después de la raíz. Entonces, el camino de j hacia i contiene al menos H arcos. Usando repetidas veces las restricciones (6.3) en ese camino llegamos a que $u_j + H \leq u_i$, esto implica que $1 \leq u_j$ y $u_i \leq H$ es falso, lo que contradice la hipótesis.

Nótese que para una solución factible del problema (6.1)-(6.5) el valor dado por un u_i puede que no indique exactamente la posición de ese nodo. De hecho, para un arco (i, j) incluido en la solución, el valor de u_j puede que sea estrictamente mayor que $u_i + 1$. Sin embargo, las restricciones (6.3) indican que el valor de cada variable u_i ($i \in N^+$) es siempre una cota superior de la posición de ese nodo y las restricciones (6.4) que esa cota superior no sea mayor que H. Esto implica que la posición de cada nodo no puede ser mayor que H.

Para mayor clarificación consideramos el ejemplo con $n = 6$ y $H = 3$ de la Figura 6.2. Donde el camino $0 - 1 - 2 - 3$ sí fuerza a que $u_1 = 1, u_2 = 2, u_3 = 3$, pero u_4 puede tener cualquier valor entre $[1, 3]$ y para u_5 y u_6 cualquier valor $1 \leq u_5 \leq x$ y $x + 1 \leq u_6 \leq 3$ con $1 \leq x \leq 2$, es factible para el problema. Luego con este ejemplo

hemos visto que no es necesario saber la posición exacta de los nodos de la solución cuando queremos presentar una formulación válida para el problema del HMST. Lo que es importante es que las variables u_i ($i \in N^+$) satisfagan las desigualdades

$$u_i - u_j \leq (H-1) \qquad i, j \in N^+$$

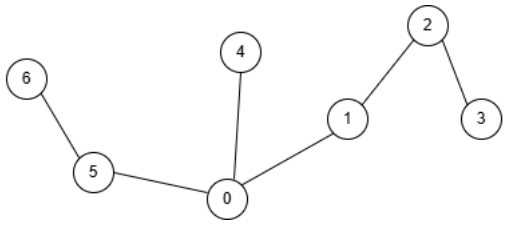


Figura 6.2: Solución factible del HMST, que ilustra algunas variables u_i no están determinadas unívocamente por las restricciones MTZ
Imagen sacada de [11].

Estas restricciones motivan a la siguiente modificación de las restricciones (6.3).

$$Hx_{ij} + u_i \leq u_j + (H-1) \qquad i, j \in N^+ \tag{6.6}$$

Que también garantizan la eliminación de caminos con más de H saltos y hacen que las restricciones (6.4) sean redundantes. Denotamos por **EMTZ** la formulación (6.1)-(6.3) y (6.5-(6.6).
Esta es una formulación también válida pues las restricciones (6.6) implican que $u_i - u_j \leq H - 1$. Anteriormente vimos que si existiese un camino entre 0 e i con más de H arcos, entonces había un j tal que $u_i + H \leq u_j$, que equivale a $H \leq u_j - u_i$ lo que contradice las restricciones (6.6).

6.1.2. Modelo más restrictivos

Ahora veremos una formulación donde podemos restringir todavía más el modelo **EMTZ**. Esta técnica y algunas otras se han usado originalmente para otros problemas como el problema del viajero (Desroches y Laporte (1991)[3]), sin embargo también se pueden utilizar para el problema del HMST. Siguiendo las mejoras de Desroches y Laporte, podemos restringir más aún las restricciones (6.6) de la siguiente manera

$$(H-2)x_{ji} + Hx_{ij} + u_i \leq u_j + (H-1) \qquad i, j \in N^+ \tag{6.7}$$

La validez de (6.7) viene de que no podemos tener $x_{ij} = x_{ji} = 1$
Una característica importante de estas restricciones es que implican $u_i = u_j + 1$ siempre y cuando $x_{ij} = 1$, aplicando de manera simétrica las restricciones (6.7). Luego estas restricciones dan una mayor precisión a las variables u_i, en el sentido de que u_i y u_j deben diferir exactamente en una unidad si existe una conexión entre i y j. Ya que las restricciones (6.7) implican las de (6.6) es claro que la formulación (6.1)-(6.3) y(6.6)- (6.7) es válida para la resolución del HMST. A esta formulación la denotare-

mos como **L1MTZ**.
Claramente, las restricciones (6.7) son al menos tan fuertes como las de (6.6). Por tanto, se espera obtener mejores cotas inferiores en la formulación **L1MTZ** que para **EMTZ**, como se ve en los resultados computacionales mostrados en Luis Gouveia (1995) [11].

6.1.3. Modelo con variables de flujo

Por último, vamos a desarrollar una formulación para el HMST basado en una formulación con variables de flujo multiproducto. Este modelo se basa en redes dirigidas, luego como de costumbre para cada arco $(i, j) \in A$ no dirigido consideramos dos arcos dirigidos (i, j) y (j, i) y con ambos teniendo el mismo coste w_{ij} que el original. Además, asumimos que los arcos son solo dirigidos desde la raíz hacia fuera, es decir, que cualquier arco no dirigido $(0, i)$ con $i \in N^+$ es únicamente remplazado por el arco dirigido $(0, i)$.
Esta formulación usa dos conjuntos de variables, las variables binarias x_{ij}, $(i \in N, j \in N^+, i \neq j)$ que son igual a 1 si y solo si el arco (i, j) está en la solución y las variables de flujo y_{ijk} con $i \in N$ y $j, k \in N^+$, que especificarán si el único camino entre la raíz y el nodo k atraviesa el arco (i, j). Para simplificar los índices, las variables de flujo y_{iik} $(i \in N^+)$ e y_{kik} $(i, k \in N^+, i \neq k)$ no las consideramos en la formulación. Entonces la formulación, que denotaremos por **DMCF**, es como sigue:

$$min \sum_{i=0}^{n} \sum_{j=1}^{n} w_{ij} x_{ij} \tag{6.8}$$

Sujeto a

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad j \in N^+ \tag{6.9}$$

$$\sum_{i=0}^{n} y_{ijk} \quad - \quad \sum_{i=1}^{n} y_{jik} = 0 \qquad j, k \in N^+, j \neq k \tag{6.10}$$

$$\sum_{i=0}^{n} y_{ijj} = 1 \qquad j \in N^+ \tag{6.11}$$

$$\sum_{i=0}^{n} \sum_{j=1}^{n} y_{ijk} \leq H \qquad k \in N^+ \tag{6.12}$$

$$y_{ijk} \leq x_{ij} \qquad i \in N; j, k \in N^+ \tag{6.13}$$

$$y_{ijk} \in \{0,1\} \qquad i \in N; j,k \in N^+ \tag{6.14}$$

$$x_{ij} \in \{0,1\} \qquad i \in N, j \in N^+ \tag{6.15}$$

Las restricciones (6.9) son usuales y las hemos utilizado en muchos otros modelos. Ellas hacen que a cada nodo distinto de la raíz solo le llegue un arco, haciendo que solo podamos coger $|A| - 1$ arcos, lo que es fundamental en un árbol. Luego (6.10) y (6.11) son las restricciones de conservación de flujo e implican conjuntamente que el árbol está conectado, impide que haya nodos aislados. Pues hacen que exista una camino desde el nodo raíz (0) hasta cualquier nodo k. (6.10) hace que se conserven los arcos intermedios del camino de 0 a k, esto es, si se coge un arco (i, j) en ese camino debe haber otro (j, l); (6.11) asegura que efectivamente se llegue a cada nodo k. Por tanto, (6.9)-(6.11) hace que las soluciones factibles sean árboles. Además, (6.12) impide que esos caminos tengan más de H saltos, ya que el sumatorio mide cuantos arcos hay en el camino de 0 a k. Finalmente, (6.13) hace que solo pueda haber flujo por arcos escogidos en el árbol. Por tanto, conjuntamente el modelo (6.9)-(6.15) resuelve el problema del HMST.

Como en los modelos anteriores podemos restringir un poco más las restricciones de salto, para generar formulaciones del HMST más precisas, que en general generarán mejores acotaciones de la solución. Para la formulación **DMCF**, podemos restringir un poco más las restricciones de salto de la siguiente manera:

$$\sum_{i=0}^{n} \sum_{j=1}^{n} y_{ijk} \leq H - x_{kl} \qquad k, l \in N^+, k \neq l \tag{6.16}$$

La validez de estas restricciones vienen del hecho de que el número de arcos en el camino hacia k no puede ser H si algún arco (k, l) $(l \in N^+, l \neq k)$ está en la solución. En textos como los de *Gouveia (1996)[12]* se observan tests computacionales donde se producen significativas mejoras en las cotas inferiores dadas por la relajación lineal. Sin embargo para valores altos de n las restricciones (80) son del orden de $O(n^2)$, frente a las n restricciones de salto originales.

Capítulo 7

Resolución de problemas con datos reales

En esta última sección vamos a resolver una serie de problemas con datos de redes concretas $G = (N, A)$, en los que vamos a poder implementar, resolver y sacar algunas conclusiones de algunos de los diferentes modelos que hemos presentado en las secciones anteriores.
La serie de problemas que vamos a ver son problemas euclídeos donde en las diferentes redes $G = (N, A)$ que vamos a manejar, los nodos se identifican con coordenadas en el plano (x_i, y_i) para cada nodo $i \in N$, los pesos w_{ij} de cada arista (i, j) serán la distancia euclídea entre (x_i, y_i) y (x_j, y_j).
Resolveremos distintos problemas variando el número de nodos $n = 15, 50, 100, 300$ para los 3 problemas principales que hemos visto en este texto: el **problema del mínimo árbol expandido** y las 2 extensiones de él, el **problema del mínimo árbol expandido con capacidades o restricciones de salto**. En ellos resumiremos los datos más relevantes de cada modelo: valor óptimo, nº de variables, restricciones, valor de la relajación lineal, etc. La serie de problemas serán resueltos en *Xpress*, en los que pondremos un límite de tiempo para alcanzar el valor óptimo (60s). Antes de todo, vamos a explicar brevemente como hace Xpress al resolver un problema de programación entera. El *solver* de Xpress utiliza una combinación de técnicas avanzadas para resolver modelos de programación entera. El proceso comienza relajando el modelo, lo que significa que se ignoran temporalmente las restricciones de enteros y se resuelve como un problema de programación lineal utilizando el método simplex clásico. Esta relajación proporciona una solución más sencilla y permite calcular una cota inferior (en problemas de minimización). Si la solución obtenida no es entera, Xpress aplica el método conocido como *branch and bound*. Este es un proceso de exploración de soluciones, cuando se encuentra una solución entera, se guarda como la mejor hasta el momento, y se descartan aquellas que no puedan mejorarla. A lo largo de la exploración, Xpress también puede añadir cortes (*cutting planes*), que son

restricciones adicionales diseñadas para eliminar soluciones e ir restringiendo la zona factible, sin eliminar ninguna solución entera. Además, el *solver* utiliza técnicas de presolución para simplificar el modelo antes de resolverlo, así como heurísticas que le permiten encontrar rápidamente soluciones enteras factibles aunque no sean óptimas, estas darán lugar a cotas superiores del óptimo z^*. Así tras un tiempo de resolución máximo que fijaremos tendremos que:

$$\text{mejor cota} \leq z^* \leq \text{valor actual de la solución entera}$$

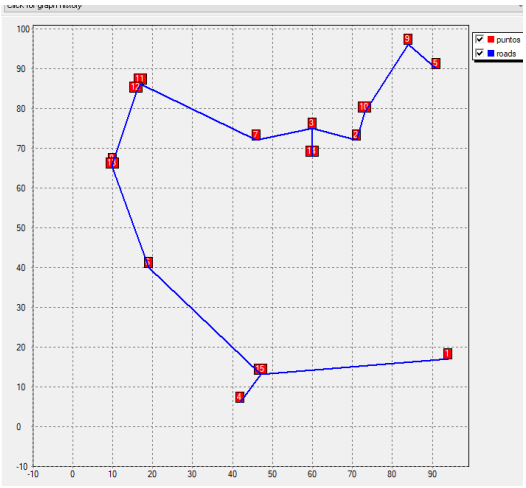
Donde la cota inferior es la mejor cota inferior encontrada y la cota superior es la solución entera encontrada en ese momento. Entonces la solución óptima se obtiene cuando la cota inferior es igual a la superior. En cambio, si no hemos encontrado la solución óptima, la cota superior se tomará como la mejor solución entera encontrada. Así, si en un máximo de tiempo (en nuestro caso 60s), no obtenemos la solución óptima, daremos como solución la cota superior encontrada en ese momento. Y obtendremos, también, una acotación de z^* que también analizaremos. Por tanto, también reflejaremos en los distintos problemas los valores de estas cotas.
Los datos que vamos a usar son coordenadas (x_i, y_i) que han sido generadas aleatoriamente entre 1 y 100.

7.1. Mínimo árbol expandido

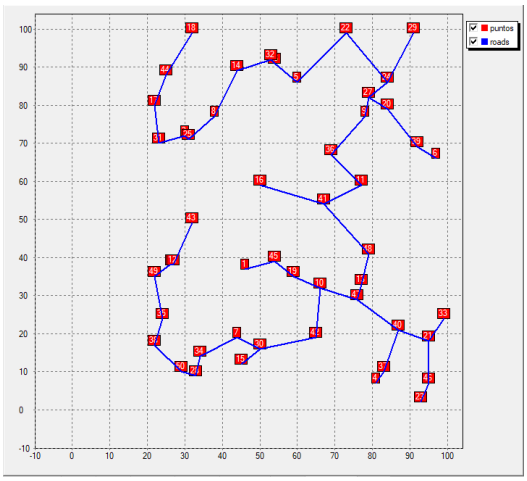
Como hemos mencionado vamos a resolver una serie de problemas concretos con distintos valores en cuanto al número de nodos $n = 15, 50, 100, 300$. Para la resolución del MST, vamos a implementar el *Algoritmo de Prim*, que vimos en el capítulo 4 del MST, en el que ya mencionamos que era un algoritmo muy eficiente para la resolución de este problema. Además, también resolveremos los distintos problemas del MST con la formulación de problema de flujo con 1 producto (4.1.3).
Al resolver los diferentes problemas, obtenemos los datos de la tabla 7.1 y las soluciones que mostramos graficamente en la Figura 7.1. Donde de aquí en adelante, F.obj es el valor de la función objetivo, T tiempo de computación (en s), V.rel.lin el valor óptimo de la relajación lineal y c.inf y c.sup son las cotas inferiores y superiores antes mencionadas.

n	Método	F.obj	T	V.rel.lin	c.inf	c.sup	nº res	nº var
15	Prim	245.1201	0.00032	–	–	–	–	–
15	Flujo	245.1201	0.26668	75.59	245.1201	245.1201	225	420
50	Prim	453.6016	0.00356	0.00605	–	–	–	–
50	Flujo	453.6016	5.8	40.038	453.6016	453.6016	2500	4900
100	Prim	595.9455	0.02042	–	–	–	–	–
100	Flujo	595.9455	17.50966	51.66	595.9455	595.9455	10000	19800
300	Prim	1011.251	0.723	–	–	–	–	–
300	Flujo	–	60	35.65	802	1060.18	90000	179400

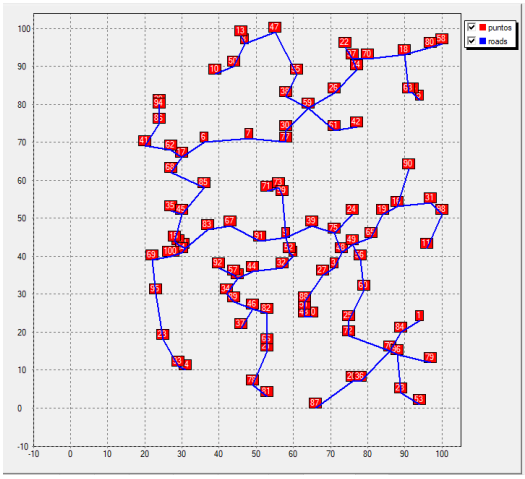
Cuadro 7.1: Datos relevantes para distintos valores de n



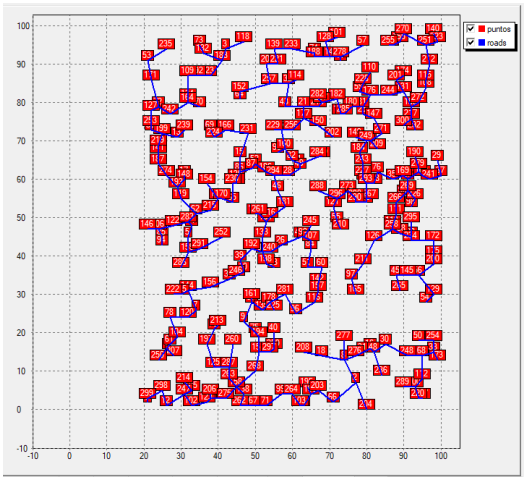
(a) Mínimo árbol expandido con $n = 15$



(b) Mínimo árbol expandido con $n = 50$



(c) Mínimo árbol expandido con $n = 100$



(d) Mínimo árbol expandido con $n = 300$

Figura 7.1: Soluciones obtenidas con los distintos valores de n. Imágenes hechas en XpressIve, basadas en [5]

.

7.1.1. Observaciones

El análisis de los datos obtenidos muestra una ventaja abismal del Algoritmo de Prim frente a la formulación de flujo, y en definitiva a cualquier formulación que exista sobre el problema del mínimo árbol expandido. Ya que el Algoritmo da una solución óptima al problema de una manera casi instantánea, con tiempos 0.00032s, 0.00356s, 0.02042s, 0.723s para $n = 15, 50, 100, 300$ respectivamente. Mientras que la formulación de flujo da tiempos razonables para $n = 15, 50$ con t =0.267, 5.8s, sin embargo para $n = 100$ ya se va a tiempos de $17s$ que es unas 850 veces más que el tiempo obtenido por el Algoritmo de Prim e incluso para $n = 300$ no se obtiene una solución óptima en el tiempo límite de 60s. Obtenemos una acotación de la solución óptima entre [802,1060.18] que tampoco es especialmente buena, sabiendo que la solución es 1011,251. Además se observa que las respectivas relajaciones lineales son realmente malas, ya que por ejemplo para n=50, se tiene

$$\frac{Z^* - Z_L}{Z_L} = \frac{595{,}9455 - 51{,}66}{595{,}9455} \approx 91\,\%$$

que es una brecha muy grande entre el valor de la relajación lineal y el verdadero valor del problema, siendo incluso peor para $n = 300$ ($\approx 96\,\%$).
En conclusión, el Algoritmo de Prim muestra una solución casi inmediata para el problema del mínimo árbol, incluso para ejemplos con muchos nodos, fácilmente aplicable para muchos lenguajes de computación y sin necesidad de tener que incluir variables de flujo ni binarias al problema que le suman en complejidad.

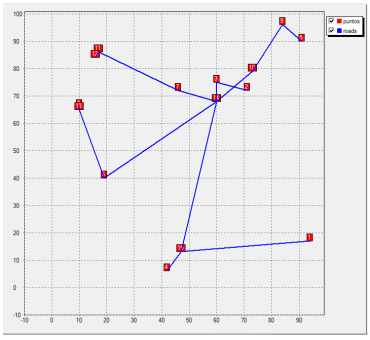
7.2. Mínimo árbol expandido con capacidades

Como en el caso del MST vamos a resolver para el CMST una serie de problemas con número de nodos $n = 15, 50, 100, 300$. A diferencia del MST no disponemos del Algoritmo de Prim para resolver el problema, luego únicamente podremos implementar las distintas formulaciones que vimos en el Capítulo5.1. También mostraremos gráficamente las soluciones obtenidas y varios datos relevantes, tal como la solución óptima (si se llega), valor de la relajación lineal, cota inferior y superior, número de variables y restricciones al resolver los distintos problemas. Para la resolución de los problemas del CMST usaremos la formulación de flujo con 1 producto (Formulación FF 5.1.2) y comentaremos en cada caso los resultados obtenidos. Además, dependiendo del caso fijaremos un valor para Q, la capacidad máxima del árbol. Nótese que para mayor claridad del grafo resultante el nodo raíz se ha escogido en los diferentes casos según lo centrado que está.

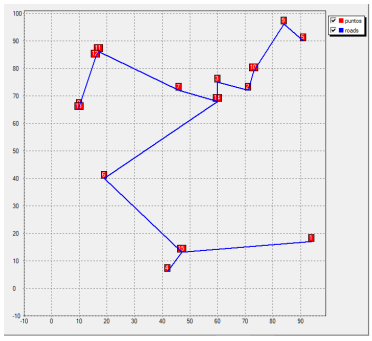
Al resolver los distintos problemas hemos obtenido los datos que resumimos en la tabla 7.2 y los grafos de la Figura 7.2 muestran las diferentes soluciones obtenidas gráficamente.

n	Método	F.obj	T	V.rel.lin	c.inf	c.sup	nº res	nº var
15	FF($Q = 3$)	303.405	0.169	262.291	303.405	303.405	224	406
15	FF($Q = 5$)	268.44	0.125	212.747	268.44	268.44	224	406
50	FF($Q = 5$)	–	60	560.859	621	643.2	2499	4851
50	FF ($Q = 10$)	–	60	433.538	490	504.39	2499	4851
100	FF($Q = 10$)	–	60	650.446	726	801.916	9999	19701
100	FF($Q = 15$)	–	60	568.5608	656	681.816	9999	19701
300	FF ($Q = 15$)	–	60	1163.88	1225	1487.357	89999	179101
300	FF($Q = 20$)	–	60	1047.1703	1111	1356.6	89999	179101

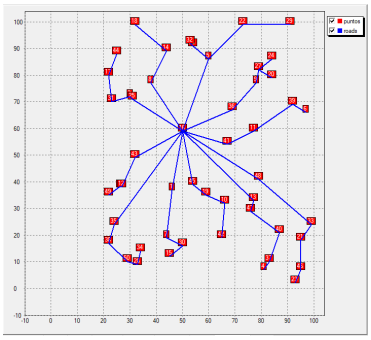
Cuadro 7.2: Datos relevantes del método FF para distintos valores de n y capacidad Q



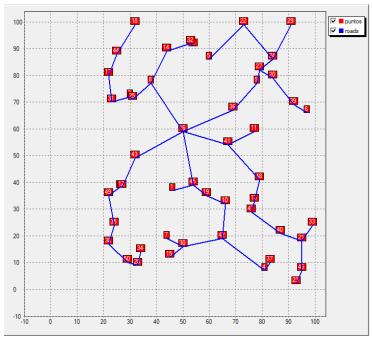
(a) $n = 15$, $Q = 3$



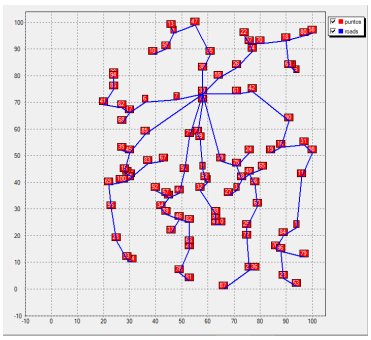
(b) $n = 15$, $Q = 5$



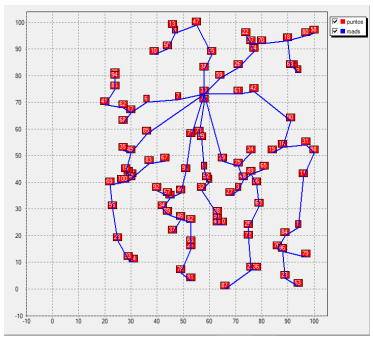
(c) $n = 50$, $Q = 5$



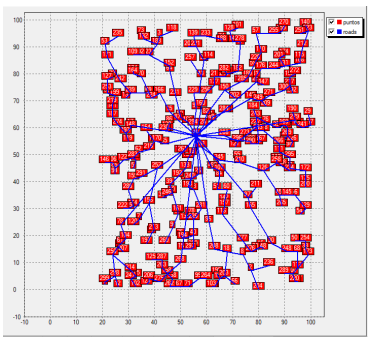
(d) $n = 50$, $Q = 10$



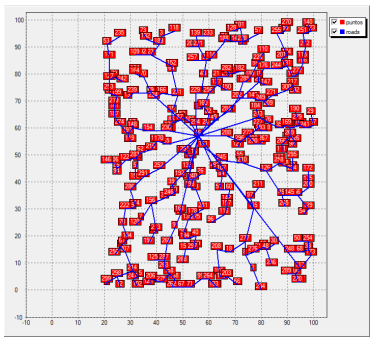
(e) $n = 100$, $Q = 10$



(f) $n = 100$, $Q = 15$



(g) $n = 300$, $Q = 15$



(h) $n = 300$, $Q = 20$

Figura 7.2: Soluciones obtenidas para distintos valores de n y Q. Imágenes hechas en XpressIve, basadas en [5].

7.2.1. Observaciones

Para $n = 15$, ambas versiones (capacidad $Q = 3$ y $Q = 5$) resuelven el problema en tiempos bastante buenos 0.17s,0.12s. Sin embargo, a partir de ahí para $n = 50, 100, 300$ no se ha encontrado el óptimo en el tiempo límite de $60s$ y se ha dado como solución la cota superior. Obteniendo, además, las acotaciones para el valor óptimo siguientes:

- [621,643],[726,801.916] y [1225,1487.357], para $Q = 5, 10, 15$ respectivamente.
- [490,504.39],[656,681.816] y [1111,1356.6], para $Q = 10, 15, 20$ respectivamente.

Estas no son relativamente malas para $n = 50, 100$, sin embargo para $n = 300$ ya es una diferencia bastante mayor.
En cuanto a las relajaciones lineales se puede observar que han mejorado significativamente respecto a las que teníamos en el MST. Teniendo una brecha para $n = 100$ de por ejemplo entre:

$$\frac{726 - 650{,}446}{726} \approx 10\,\% \quad \frac{801{,}916 - 650{,}446}{801{,}916} \approx 19\,\%$$

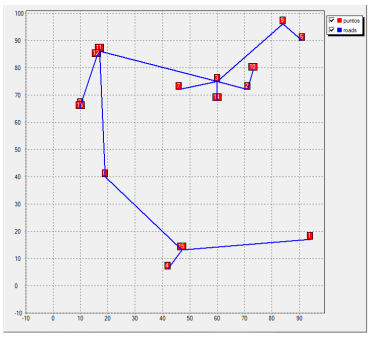
Lejos del más del 90 % que teníamos antes, aunque todavía mejorables.
Otra observación que podemos sacar es que al resolver el mismo problema con distintos valores para la capacidad máxima, el valor óptimo difiere bastante. Cuánto mayor sea Q menor es el coste total del árbol resultante y viceversa. Algo lógico, pues cuanto menor sea Q más restrictiva es la solución que queremos obtener, hay menos capacidad y se necesitan mayor número de ramas saliendo de la raíz.
Como conclusión se puede sacar, que a diferencia del MST en el que disponemos de un algoritmo polinómico que lo resuelve, para el CMST las distintas formulaciones en tiempos razonables de computación y valores medios y altos de n únicamente se obtienen soluciones óptimas aproximadas. En ocasiones con una diferencia bastante amplia entre la cota superior y la cota inferior. Por ello en el campo de la optimización combinatoria se han desarrollado lo que se conoce como heurísticas o metaheurísticas, que son algoritmos o métodos diseñados para encontrar soluciones «suficientemente buenas» en tiempos razonables. Sobre todo se usan en campos como la programación entera mixta, como es el caso del problema del CMST, donde encontrar soluciones óptimas a medida que aumenta el tamaño se vuelve ineficaz debido a la gran complejidad.

7.3. Mínimo árbol expandido con restricciones de salto

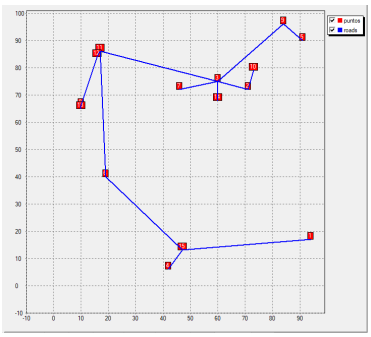
Y finalmente, vamos a resolver para el HMST una serie de problemas con nuevamente número de nodos $n = 15, 50, 100, 300$. En este caso tampoco disponemos de un algoritmo como el de *Prim* para resolver el problema eficientemente, luego resolveremos los distintos problemas únicamente con las distintas formulaciones que resolvían el problema del HMST, mostradas en la sección6.1. También ilustraremos las soluciones obtenidas y los datos relevantes, mostraremos la solución óptima obtenida, tiempo de computación, valor de la relajación lineal, cota superior e inferior, número de variables y restricciones al resolver los distintos problemas. Para la resolución de los problemas del HMST usaremos las formulaciones MTZ (6.1.1) y L1MTZ (6.1.2) comentando los resultados obtenidos. Además para cada caso variaremos el parámetro H y como antes la elección del nodo raíz se ha hecho simplemente por mayor claridad de la imagen.
Al resolver los diferentes problemas obtenemos los datos que resumimos en la Tabla 7.3 y obtenemos los grafos solución en cada caso en la Figura 7.3.

n	Método	F.obj	T	V.rel.lin	c.inf	c.sup	nº res	nº var
15	MTZ	288.42	0.214	165.54	288.42	288.42	224	211
15	L1MTZ	288.42	3.488	225.53	288.42	288.42	196	211
50	MTZ	–	60	336.79	428.84	535.624	2499	2451
50	L1MTZ	–	60	401.15	438.02	555.91	2401	2451
100	MTZ	–	60	436.43	560.08	700.9	9999	9901
100	L1MTZ	–	60	534.98	576.89	739.74	9801	9901
300	MTZ	–	60	768.11	940.48	1504.72	89999	89701
300	L1MTZ	–	60	935.93	977.87	2932.61	89401	89701

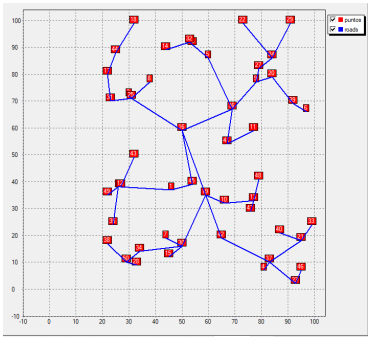
Cuadro 7.3: Datos relevantes para las formulaciones MTZ y L1MTZ para distintos valores de n



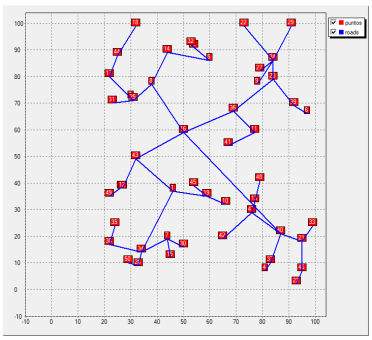
(a) $n = 15$, $H = 3$ (MTZ)



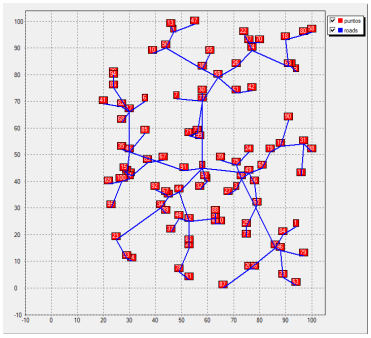
(b) $n = 15$, $H = 3$ (L1MTZ)



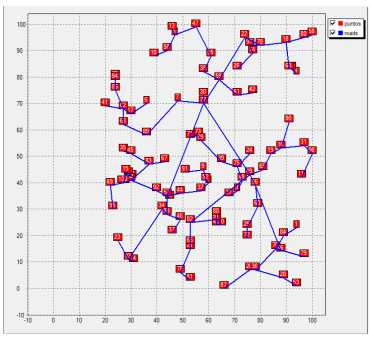
(c) $n = 50$, $H = 5$ (MTZ)



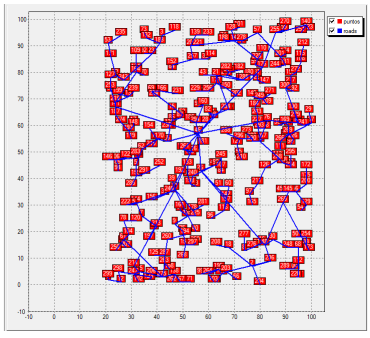
(d) $n = 50$, $H = 5$ (L1MTZ)



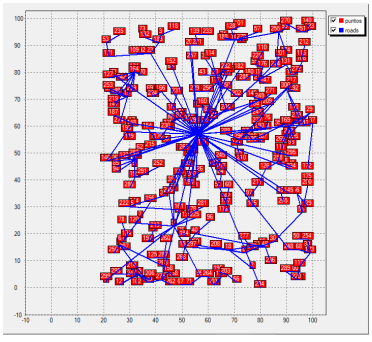
(e) $n = 100$, $H = 7$ (MTZ)



(f) $n = 100$, $H = 7$ (L1MTZ)



(g) $n = 300$, $H = 9$ (MTZ)



(h) $n = 300$, $H = 9$ (L1MTZ)

Figura 7.3: Soluciones obtenidas para distintos valores de n y formulaciones MTZ y L1MTZ Imágenes hechas en XpressIve, basadas en [5].

7.3.1. Observaciones

Como en el caso anterior las dos formulaciones resuelven el problema para $n = 15$, teniendo un tiempo bastante rápido de 0.214s para el MTZ y algo peor para la formulación de L1MTZ de 3.488s. Sin embargo, para los demás casos de $n = 50, 100, 300$ no se ha obtenido una solución óptima en menos del tiempo máximo de $60s$. Se tienen entonces las siguientes acotaciones del valor óptimo:

- Para el MTZ, [428.84,535.624],[560.08,700.9] y [940.48,1504.72] para $n = 50, 100, 300$ respectivamente
- Para el L1MTZ, [438.02,555.91],[576.89,739.74] y [977.87,2932.608] para $n = 50, 100, 300$ respectivamente

Donde las acotaciones obtenidas tras $60s$ van empeorando a medida que aumentamos el n. Especialmente para el L1MTZ con $n = 300$, donde se observa una diferencia alta.
En cuanto al valor de las relajaciones lineales se observa que los valores de la relajación lineal para la formulación L1MTZ están más próximos de la cota inferior que los obtenidos por la de MTZ, lo que parece evidenciar que la formulación L1MTZ sacrifica tiempo de computación por una relajación lineal más fuerte. Como en el caso con $n = 15$ donde tenemos tiempos de 0.214s y 3.488s por una brecha de relajación de aproximadamente

$$\frac{288{,}42 - 165{,}54}{288{,}42} \approx 42\,\% \quad \frac{288{,}42 - 225{,}53}{288{,}42} \approx 21\,\%$$

También se observa diferencia entre el número de restricciones de las dos formulaciones, teniendo hasta casi 600 menos la L1MTZ que la MTZ cuando $n = 300$. Esto es por la sustitución de las restricciones (6.3) y (6.4) en la formulación MTZ, por únicamente las (6.7) que definían la formulación L1MTZ.
Finalmente, se pueden sacar conclusiones parecidas al problema del CMST. Al no disponer de un algoritmo eficiente, hace que no seamos capaces de obtener el valor óptimo en un tiempo de computación moderado, obteniéndose en lugar, valores aproximados de la función objetivo y en ocasiones nada precisos (L1MTZ $n = 300$). Por ello también el concepto de heurísticas y metaheurísticas es importante para el problema del HMST, con el objetivo de encontrar soluciones de buena calidad en tiempos razonables.

Conclusiones

A lo largo de este trabajo se han cumplido los principales objetivos planteados inicialmente, orientados al estudio y modelización matemática de problemas de diseño óptimo de redes.

En primer lugar, se introdujeron los diferentes problemas de flujo en redes, destacando los de costos fijos por su gran aplicabilidad en las formulaciones de los distintos problemas de diseño de redes estudiados.

Posteriormente, se ha presentado un recorrido histórico-científico por la evolución de las topologías de redes, destacando cómo estas han ido adaptándose a las necesidades tecnológicas de cada época. Mostrando la importancia del diseño de redes eficientes, con el objetivo de dar un buen servicio a los usuarios, pero con el mínimo coste posible debido a la gran inversión que supone la implementación de estas redes de comunicación. Esta parte ha servido como contexto fundamental para entender la motivación detrás de los distintos modelos matemáticos desarrollados posteriormente.

A continuación, se ha estudiado en profundidad el Problema del Mínimo Árbol Expandido (MST), presentándolo como un problema base dentro del diseño de redes. Se ha analizado su formulación clásica, así como las propiedades matemáticas que permiten su resolución eficiente mediante algoritmos conocidos como el de *Kruskal* o *Prim*.

Además, se han abordado dos extensiones relevantes del problema: el *MST* con restricciones de capacidad (*CMST*) y el *MST* con restricciones de salto (*HMST*). En ambos casos, se han expuesto formulaciones basadas en programación lineal entera que permiten modelar estas restricciones adicionales. Donde he podido comprobar que implementar estas restricciones adicionales complica mucho más el problema, debido a la no existencia de algoritmos eficientes. Esto ha hecho que a lo largo del tiempo se hayan ido presentando formulaciones mucho más sofisticadas que en el caso del *MST*, con el objetivo de obtener modelos con menos restricciones, mejores relajaciones lineales,...

Uno de los aspectos más destacados del trabajo ha sido la implementación práctica de las formulaciones en el entorno Xpress-Mosel, acompañada de una visualización de las soluciones obtenidas mediante Xpress-IVE. Esta parte me ha permitido observar el comportamiento real de los modelos en ejemplos de distinto tamaño, analizar tiempos de resolución y validar la coherencia de las soluciones. Lo que me ha permitido refutar lo que he mostrado antes sobre la dificultad de añadir restricciones adicionales al problema del *MST*. Ya que para los modelos probados, en la mayoría de ocasiones no hemos podido obtener una solución óptima en un tiempo prudencial.

Como sugerencia de mejora o ampliación futura, sería interesante incorporar o evaluar *heurísticas o metaheurísticas* para resolver problemas de mayor tamaño, dado

que los modelos exactos pueden volverse ineficientes cuando el número de nodos crece significativamente.

En definitiva, este trabajo ha permitido no solo alcanzar los objetivos establecidos, sino también adquirir una visión global y aplicada del campo del diseño óptimo de redes, combinando fundamentos teóricos, modelización matemática y experimentación computacional.

Apéndice A

Elementos básicos de la Programación Entera

En este apéndice veremos una introducción a los elementos básicos de la Programación Entera.
Supongamos que tenemos un problema de programación lineal genérico:

$$min\{cx : Ax \leq b, x \geq 0\}$$

Donde A es una matriz mxn, c un vector fila de tamaño n y b es un vector columna de tamaño m.
Entonces la programación entera surge de añadir las restricciones de que algunas o todas las variables sean enteras.

- Si hay mezcla entre variables continuas y variables enteras: **Programación Entera Mixta (MIP)** *Mixed Integer Program.*
- Si todas las variables son enteras, $x \in \mathbb{Z}^n$: **Programación Entera (IP)** *Integer Program.*
- Si todas las variables son binarias, $x \in \{0,1\}^n$: **Programación Entera Binaria (BIP)** *Binary Integer Program*,

Con este tipo de formulaciones se pueden modelar multitud de problemas: el problema de Asignación, problema de la mochila 0-1, problema recubridor, problema del viajante, entre otros.
En algunos de estos problemas la solución óptima es algún subconjunto de un conjunto finito, luego se podría llegar a la solución óptima por enumeración. Sin embargo el número de subconjuntos de un conjunto finito crece exponencialmente con su tamaño n, lo que hace impracticable resolver estos problemas por combinatoria, es lo que se

conoce como *Explosión Combinatoria*. Por tanto, hay que desarrollar algoritmos más inteligentes que reduzcan bastante las posibilidades.

Formulaciones

Definición A.0.1. *Un* ***poliedro*** P *en* $\mathbb{R}^n$ *se puede definir como el conjunto de puntos que satisfacen un sistema de desigualdades lineales:*

$$P = \{x \in \mathbb{R}^n : Ax \leq b\} \qquad A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^m$$

Definición A.0.2. *Un poliedro* $P \subseteq \mathbb{R}^{n+p}$ *es una* ***formulación*** *para un conjunto* $X \subseteq \mathbb{Z}^n \times \mathbb{R}$ *si y solo si* $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.

Y se puede decir, por tanto:

Definición A.0.3. *Dados* $X \subseteq \mathbb{R}^n$ *y dos formulaciones* P_1, P_2*, se dice que* P_1 ***es mejor*** *que* P_2 *si* $P_1 \subset P_2$.

Relajaciones

Definición A.0.4. *Un problema* $z^R = min\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ *es una relajación del problema de Programación Entera* $z = \{c(x) : x \in X \subseteq \mathbb{R}^n, x \text{ enteros}\}$ *si:*

1. $X \subseteq T$
2. $f(x) \geq c(x)$ *para todo* $x \in X$

La pregunta va a ser como definir relajaciones a un problema de programación entera interesantes. Una de las más útiles y naturales son las relajaciones lineales.

Definición A.0.5. *Para el problema de programación entera* $z = min\{cx : x \in P \cap \mathbb{Z}^n\}$ *con* $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ *una formulación, su* ***relajación lineal*** *es el modelo de programación lineal asociado, es decir,* $z_l = \{cx : x \in P\}$.

Como la función objetivo es la misma en ambos programas y $P \cap \mathbb{Z}^n \subseteq P$, efectivamente se tiene que z_l es una relajación.
Mejores formulaciones generan mejores relajaciones. En particular si $P_1 \subset P_2$ son dos formulaciones de un IP, se tiene que $z_l^1 \leq z_l^2$ siendo z_l^i la respectiva relajación lineal asociada.
Las relajaciones además de darnos cotas para nuestros problemas de programación, también permiten en ocasiones probar la optimalidad.

Proposición A.0.1. *1. Si la relajación de un IP no es factible entonces el IP original tampoco*

2. *Sea x^* una solución óptima de la relajación de un IP. Si $x^* \in X$ y además $f(x^*) = c(x^*)$, entonces x^* es también una solución óptima del IP.*

Demostración. 1. Si la relajación no es factible implica que $T = \emptyset$, entonces $X = \emptyset$ y, por tanto el IP no es factible tampoco.
2. Como $x^* \in X$ $z \leq c(x^*) = f(x^*) = z^R$ y como z^R es un relajación se tiene que $z^R \leq z$, luego $z = z^R$ y x^* es una solución óptima de del IP. □

Se dice que el poliedro $P \subseteq \mathbb{R}^n$ es **entero** si sus vértices pertenecen a $\mathbb{Z}^n$, es decir todas sus coordenadas son enteras. Entonces podemos sacar una conclusión muy importante de la Proposición anterior.

Corolario A.0.1. *Sean z el IP y z_l su relajación lineal asociada. Si P es un poliedro entero entonces la relajación lineal z_l resuelve el IP.*

De la programación lineal se sabe que las soluciones óptimas, si existen, están en los vértices (esquinas) de la región factible. Luego si el poliedro es entero cualquier solución óptima del LP es automáticamente una solución óptima del IP. Esto es fundamental ya que no será necesario imponer que las variables de mi problema sean enteras o binarias. Veremos algunas formulaciones en el caso del MST que tendrán esta propiedad.

Apéndice B

Listado de figuras y tablas

Figuras

Tablas

Bibliografía

[1] Tamer F Abdelmaguid. An efficient mixed integer linear programming model for the minimum spanning tree problem. *Mathematics*, 6(10):183, 2018.

[2] Ravindra K Ahuja, Thomas L Magnanti, James B Orlin, et al. *Network flows: theory, algorithms, and applications*, volume 1. Prentice hall Englewood Cliffs, NJ, 1993.

[3] Martin Desrochers and Gilbert Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.

[4] Neng Fan and Mehdi Golari. Integer programming formulations for minimum spanning forests and connected components in sparse graphs. In *Combinatorial Optimization and Applications: 8th International Conference, COCOA 2014, Wailea, Maui, HI, USA, December 19-21, 2014, Proceedings 8*, pages 613–622. Springer, 2014.

[5] FICO Optimization. *Xpress-IVE and Mosel Language Reference Manual*. FICO, 2020. Section on graphical output and procedures for visualization using IVEdrawline, IVEdrawlabel, etc. Available at `https://www.fico.com/fico-xpress-optimization`.

[6] Bezalel Gavish. Topological design of centralized computer networks—formulations and algorithms. *Networks*, 12(4):355–377, 1982.

[7] Bezalel Gavish. Formulations and algorithms for the capacitated minimal directed tree problem. *Journal of the ACM (JACM)*, 30(1):118–132, 1983.

[8] Bezalel Gavish. Topological design of computer communication networks—the overall design problem. *European Journal of Operational Research*, 58(2):149–172, 1992.

[9] Luis Gouveia. A comparison of directed formulations for the capacitated minimal spanning tree problem. *Telecommunication Systems*, 1:51–76, 1993.

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

[11] Luis Gouveia. Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints. *Computers & Operations Research*, 22(9):959–970, 1995.

[12] Luis Gouveia. Multicommodity flow models for spanning trees with hop constraints. *European Journal of Operational Research*, 95(1):178–190, 1996.

[13] Luis Gouveia and Pedro Martins. The capacitated minimal spanning tree problem: An experiment with a hop-indexedmodel. *Annals of Operations Research*, 86(0):271–294, 1999.

[14] Luis Gouveia and Pedro Martins. The capacitated minimum spanning tree problem: Revisiting hop-indexed formulations. *Computers & Operations Research*, 32(9):2435–2452, 2005.

[15] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[16] Thomas L Magnanti and Laurence A Wolsey. Optimal trees. *Handbooks in operations research and management science*, 7:503–615, 1995.

[17] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulations and traveling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329, 1960.

[18] Robert C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

[19] Douglas B West. *Combinatorial mathematics*. Cambridge University Press, 2021.

[20] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.