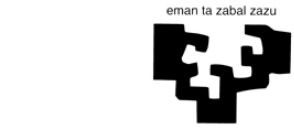


ZTF-FCT
Zientzia eta Teknologia Fakultatea
Facultad de Ciencia y Tecnología

Optimización en redes. El problema del transporte

Trabajo Fin de Grado
Grado en Matemáticas

Christian Carballo Lozano

Trabajo dirigido por
Gloria Pérez Sainz de Rozas

Leioa, 23 de Junio de 2015

Índice general

Agradecimientos

A Gloria por todo el tiempo que ha dedicado a dirigirme el proyecto y guiarme durante el mismo, ayudándome en lo que he necesitado.

A la Universidad del País Vasco, y en especial a aquellos profesores que me han dedicado su tiempo y me han ayudado durante el transcurso del grado.

A mi familia por estar siempre ahí, tanto en los buenos momentos como en los no tan buenos.

Introducción

La *programación lineal* es una rama de las matemáticas que se encuentra dentro de la optimización o programación matemática y que estudia los problemas de la forma

$$\begin{aligned} \text{mín} \quad & c\,x \\ & A\,x \geq b \\ & x \geq 0 \end{aligned}$$

donde $x \in M_{n\times 1}$, $c \in M_{1\times n}$, $A \in M_{m\times n}$ con $Rg(A) = m \leq n$ y $b \in M_{m\times 1}$. Si se sustituye $x \geq 0$ por $x \in \mathbb{Z}^n$ se tiene un problema de *programación entera*.

Dentro de la programación matemática, se encuentra una clase de problemas conocidos como problemas de redes. Hay tres propiedades principales que propician el uso de redes para abordar los problemas de optimización:

- Contenido visual: La posibilidad de representar los problemas de redes mediante un grafo o diagrama conlleva una facilidad a la hora de comprender la estructura del problema y las relaciones entre las variables o constantes que intervienen. Las representaciones utilizadas en los modelos de redes le deben mucho a aquellas de la teoría de grafos, cuyos orígenes se remontan a los años 1700 con el trabajo del matemático suizo Euler.

- Flexibilidad y extensión de los modelos: Una importante característica es la diversidad de problemas que pueden ser modelados mediante redes. Hoy en día es complicado descubir un campo en el que la optimización de redes no esté presente. La parte algorítmica es la que mayor interés despierta en el campo de las matemáticas, mientras que en ciencias físicas e ingenierías las redes tienen aplicación en diseño de circuitos eléctricos, telecomunicaciones, sistemas de transporte, gestión de aguas y procesos químicos entre otras cosas. En el campo de la economía y la empresa se utiliza prácticamente en todos los sectores la optimización de redes, por ejemplo en problemas de producción, distribución, planificación financiera y en selección de proyectos. La buena modelización de un problema puede ser tan importante como el método de resolución y la potencia de la computadora. En efecto, las mejores técnicas matemáticas y la computadora más potente no

hallarán una buena solución ante un problema modelizado de manera pobre.

- Resolubilidad: El primer método de resolución con garantía teórica de optimalidad apareció en 1947 cuando G. B. Dantzig desarrolló el método simplex. Aun así, la habilidad para resolver problemas de redes complejos se mantuvo sin demostrar. En los años 1970 y 1980 hubo un gran avance en los códigos computacionales, obteniendo programas 100-150 veces más rápidos que en la década previa. Los problemas de redes tienen algoritmos particulares, que suelen ser desarrollados a partir de otros algoritmos más generales.

Los problemas de transporte son un caso particular de los problemas de redes, y es uno de los primeros problemas abordados como red, aunque inicialmente fue formulado como un problema de programación lineal. En la realidad, su uso se ha reflejado en ahorros importantes para empresas como puede verse en el siguiente ejemplo real, ver [9].

Procter & Gamble (P & G) produce y comercializa más de 300 marcas de productos a nivel mundial. Esta compañía ha crecido continuamente a través de su larga historia que data desde 1830. Para mantener y acelerar ese crecimiento, se realizó un importante estudio de Investigación de Operaciones para fortalecer la efectividad global de P & G. Antes del estudio, la cadena de suministro de la compañía consistía en cientos de proveedores para las 50 categorías de productos en 60 plantas, 15 centros de distribución y más de 1000 zonas de consumo. Sin embargo, en la medida en que la compañía consideró marcas globales, la administración se percató de que se requería consolidar las plantas a fin de reducir los gastos de manufactura, mejorar la velocidad de entrega al mercado y reducir la inversión de capital. Por lo cual, el estudio se enfocó al rediseño del sistema de producción y distribución de la compañía para sus operaciones en Norteamérica. El resultado fue una reducción del número de plantas en Norteamérica de casi 20 %, ahorrando más de 200 millones de dólares en costos antes de impuestos por año.

Gran parte del estudio consistió en la formulación y solución de problemas de transporte de categorías individuales de producto. Para cada opción referente a mantener abiertas ciertas plantas, la solución del correspondiente problema de transporte para cierta categoría de producto mostró cuál sería el costo de distribución para enviar dicha categoría de producto desde esas plantas hacia los centros de distribución y zonas de consumo.

Resumen

El texto está dividido en tres capítulos. Se explicarán conceptos, teoría y modelos que intervendrán de manera directa en los capítulos posteriores.

En el primer capítulo se abordarán los problemas lineales de redes. Se describe la teoría relativa a redes y con ello se desarrolla el método simplex para redes, una especialización del método simplex. Además se introducen los problemas de flujo de redes a costo mínimo.

En el segundo capítulo se exponen los problemas de transporte y algún caso particular del mismo, para lo cual no será prácticamente necesario el desarrollo de nueva teoría, siendo válido todo lo expuesto en el capítulo previo.

En el tercer capítulo se extiende el concepto de problemas de transporte, mediante modelos más completos que pretenden adecuarse algo más a los modelos de la vida real. A pesar de no ser problemas de transporte, están estrechamente relacionados con ellos y por lo tanto podrá ser explotada su estructura interna de problema de transporte.

Por último, en los apéndices se encuentran los programas utilizados para resolver los problemas y los ejemplos del texto, se explica como resolver el problema de costo mínimo, de transporte o de transbordo computacionalmente y se realizan pruebas computacionales que demuestran la importancia de las propiedades de los problemas de redes.

Objetivos

El objetivo del texto es describir el problema del transporte. Para ello se ha introducido la teoría relativa a la optimización en redes de tipo continuo, es decir, problemas de redes en los que está garantizada la integralidad.

En una segunda parte se explican algunas extensiones del problema del transporte, el problema del transbordo y el problema de transporte con localización. En la última extensión se presentan problemas de redes de tipo discreto que se resolverán utilizando técnicas de programación entera.

Por último, es importante observar que las técnicas de computación y optimización son una parte fundamental para la resolución del tipo de problemas que se explican en el texto.

Capítulo 1

Programación en redes

En este capítulo se introducen los conceptos básicos de grafos y redes, junto con la descripción del método simplex para problemas de redes. Por último, se explican los problemas de flujo de redes a costo mínimo.

1.1. Conceptos básicos de redes

Definición 1.1.1. Un *grafo dirigido* o *digrafo* $\mathcal{G}$ está compuesto por un conjunto no vacío $\mathcal{N}$ de elementos llamados *nudos* y un conjunto finito $\mathcal{A}$ de pares ordenados de nudos distintos llamados *arcos*.

Se denotará $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, con $\mathcal{N} = \{1, 2, \ldots, m\}$ el conjunto de nudos y $\mathcal{A} = \{e_1, e_2, \ldots, e_n\}$ el conjunto de arcos. Se supone que $\mathcal{G}$ no tiene arcos dobles. Para un arco dirigido e_k que va del nudo i al nudo j, se define $F(e_k) = i$ y $T(e_k) = j$, o equivalentemente se dice que i es el *nudo from* y que j es el *nudo to*. Formalmente $e_k = (F(e_k), T(e_k))$. Se dice que e_k es *incidente* en los nudos i y j y que los nudos i y j son *adyacentes*.

Se supondrá, si no se dice lo contrario, que el término grafo se refiere a grafo dirigido. En efecto, si existe un arco no dirigido entre los nudos i y j, se interpretará que dicho arco existe en ambas direcciones, es decir, $(i, j) \in \mathcal{A}$ y $(j, i) \in \mathcal{A}$.

Un grafo se dice que es un *grafo propio* si $|\mathcal{N}| \geq 2$ y $|\mathcal{A}| \geq 1$.

Definición 1.1.2. Una *red* es un grafo en el cual los nudos o arcos tienen asociado algún valor numérico (costo, capacidad,...).

Definición 1.1.3. Un grafo $\hat{\mathcal{G}} = (\hat{\mathcal{N}}, \hat{\mathcal{A}})$ se dice que es un *subgrafo* de un grafo $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ si $\hat{\mathcal{N}} \subset \mathcal{N}$ y $\hat{\mathcal{A}} \subset \mathcal{A}$.

Cuando $\hat{\mathcal{N}} = \mathcal{N}$, se dice que $\hat{\mathcal{G}}$ es un *subgrafo extendido* de $\mathcal{G}$.

Definición 1.1.4. Un *camino* en $\mathcal{G}$ del nudo i_1 al nudo i_{s+1} es una secuencia de nudos y arcos distintos $P = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, i_s, e_{j_s}, i_{s+1}\}$, donde $\forall k$: $1 \leq k \leq s$, se cumple $e_{j_k} = (i_k, i_{k+1})$ o $e_{j_k} = (i_{k+1}, i_k)$.

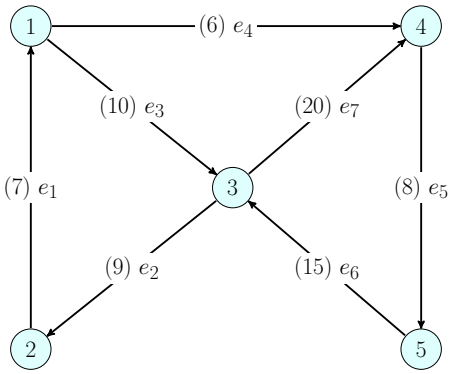


Figura 1.1: Ejemplo de una red propia. $\mathcal{N} = \{1, 2, 3, 4, 5\}$ y $\mathcal{A} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$.

En ocasiones el camino P se escribe indicando únicamente los nudos $P = \{i_1, i_2, \ldots, i_{s+1}\}$ o los arcos $P = \{e_{j_1}, e_{j_2}, \ldots, e_{j_s}\}$.

Definición 1.1.5. Un grafo $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ se dice *conectado* si para cualquier par de nudos $i, j \in \mathcal{N}$ existe un camino de i a j.

Definición 1.1.6. La secuencia $C = \{i_1, e_{j_1}, i_2, e_{j_2}, i_3, \ldots, i_s, e_{j_s}, i_{s+1}\}$ se dice que es un *ciclo* en $\mathcal{G}$ si $\{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, e_{j_{s-1}}, i_s\}$ es un camino en $\mathcal{G}$, $i_1 = i_{s+1}$ y $e_{j_s} = (i_s, i_{s+1})$ o $e_{j_s} = (i_{s+1}, i_s)$.

Definición 1.1.7. Se llama *longitud* de un camino o ciclo al número de arcos que lo componen.

Definición 1.1.8. Se define el *vector orientación* $O(P)$ de un camino o ciclo $P = \{i_1, e_{j_1}, \ldots, e_{j_s}, i_{s+1}\}$ de longitud s en $\mathcal{G}$ como

$$O_{e_{j_k}}(P) = \begin{cases} 1, & \text{si el arco } e_{j_k} = (i_k, i_{k+1}) \\ -1, & \text{si el arco } e_{j_k} = (i_{k+1}, i_k) \end{cases}$$

Definición 1.1.9. Un grafo es *acíclico* si no posee ciclos.

Definición 1.1.10. Un *árbol* es un grafo acíclico y conectado.

Definición 1.1.11. Un *árbol extendido* en $\mathcal{G}$ es un árbol que además es un subgrafo extendido de $\mathcal{G}$.

Teorema 1.1.1. *Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ un grafo con al menos un nudo. Entonces son equivalentes:*

(i) *$\mathcal{T}$ es un árbol.*

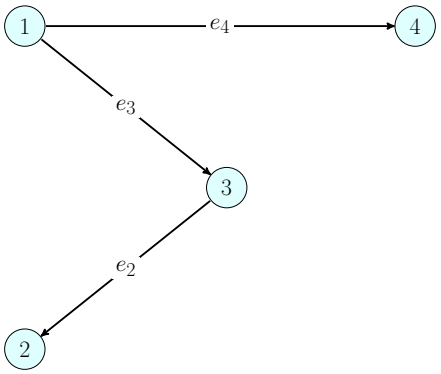


Figura 1.2: Ejemplo de un árbol. Un camino en el árbol es $P = \{2, e_2, 3, e_3, 1, e_4, 4\}$. El vector orientación de P es $O(P) = [-1, -1, 1]$.

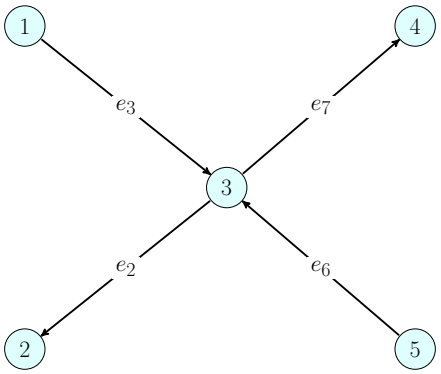


Figura 1.3: Ejemplo de un árbol extendido en la Figura 1.1.

(ii) *Para cada par de nudos $p, q \in \mathcal{N}$, hay un único camino en $\mathcal{T}$ de p a q.*

(iii) *$\mathcal{T}$ tiene un arco menos que nudos y es conectado.*

(iv) *$\mathcal{T}$ tiene un arco menos que nudos y es acíclico.*

Demostración. Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ un grafo con al menos un nudo.

(i)⇒(ii). Se supone que $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ es un árbol y sean $p, q \in \mathcal{N}$, $p \neq q$. Por reducción al absurdo, se suponen dos caminos distintos $P = \{i_1, e_{j_1}, i_2, \ldots, i_s, e_{j_s}, i_{s+1}\}$ y $P' = \{i'_1, e'_{j_1}, i'_2, \ldots, i'_t, e'_{j_t}, i'_{t+1}\}$ de p a q. Sea u el menor entero positivo tal que $i_u = i'_u$ y $i_{u+1} \neq i'_{u+1}$ (existe dicho u porque $i_1 = i'_1$ y si no existiera, los caminos P y P' serían el mismo). Ahora, sea v el menor entero positivo, con $v > u$ y tal que para algún w, $i_v = i'_w$ (existe dicho v ya que al menos $i_{s+1} = i'_{t+1}$). Se puede construir un ciclo

$$C = \{i_u, e_{j_u}, i_{u+1}, e_{j_{u+1}}, \ldots, i_v = i'_w, e'_{j_{w-1}}, i'_{w-1}, e'_{j_{w-2}}, \ldots, i'_u = i_u\}$$

en $\mathcal{T}$. Absurdo por ser $\mathcal{T}$ un árbol. Entonces $P = P'$ para cualquier par de caminos P, P' de p a q.

(ii)⇒(iii). Se supone cierto (ii). Sea $|\mathcal{N}| = m$, se probará (iii) por inducción sobre m. Para $m = 2$, como hay un único camino que va de un nudo al otro, entonces $\mathcal{T}$ tiene un único arco y queda probado. Se supone cierto (iii) hasta un número entero m y se probará que se cumple (iii) para $m+1$.

Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ un árbol con $m+1$ nudos. Sea $e_r \in \mathcal{A}$, se definen

$$\mathcal{M}^1 = \{i : i \in \mathcal{N} \text{ y } e_r \text{ no está en el camino que va de } F(e_r) \text{ a } i\},$$
$$\mathcal{M}^2 = \{i : i \in \mathcal{N} \text{ y } e_r \text{ no está en el camino que va de } T(e_r) \text{ a } i\}.$$

Por tener un camino al menos un arco, y por la definición de los conjuntos $\mathcal{M}^1$ y $\mathcal{M}^2$, se afirma que ni $F(e_r)$ ni $T(e_r)$ están en $\mathcal{M}^1$ o $\mathcal{M}^2$. Se verá que cualquier otro nudo de $\mathcal{N}$ está en $\mathcal{M}^1 \cup \mathcal{M}^2$. Sea $p \in \mathcal{N} - \{F(e_r), T(e_r)\}$. Sea $P = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, e_{j_s}, i_{s+1}\}$ el único (por (ii)) camino que va de $F(e_r)$ a p, de longitud s. Si $e_r \notin P$, entonces $p \in \mathcal{M}^1$. En caso contrario, si $e_r \in P$, necesariamente $e_r = e_{j_1}$ por ser los nudos de un camino distintos, y como los arcos de un camino también son distintos, entonces el camino de $T(e_r)$ a p, $Q = \{i_2, e_{j_2}, i_3, e_{j_3}, \ldots, e_{j_s}, i_{s+1}\}$, no contiene al arco e_r. Por tanto, $p \in \mathcal{M}^2$.

Para ver que $\mathcal{M}^1$ y $\mathcal{M}^2$ son disjuntos, por reducción al absurdo se supone que existe un nudo $p \in \mathcal{M}^1 \cap \mathcal{M}^2$. Entonces existe un camino P de $F(e_r)$ a p y un camino P' de $T(e_r)$ a p en los cuales no está el arco e_r. Sea t la longitud de P', entonces $Q = \{F(e_r), e_r, i'_1, e'_{j_1}, i'_2, e'_{j_2}, \ldots, i'_{t+1}\}$ es un camino de $F(e_r)$ a p distinto a P, ya que $e_r \notin P$. Se ha llegado a un absurdo, ya que contradice (ii).

Por lo tanto, se ha demostrado que $\mathcal{N}^1 = \mathcal{M}^1 \cup \{F(e_r)\}$ y $\mathcal{N}^2 = \mathcal{M}^2 \cup \{T(e_r)\}$ son disjuntos, no vacíos y contienen todos los nudos de $\mathcal{N}$, es decir, $\mathcal{N} = \mathcal{N}^1 \cup \mathcal{N}^2$ (unión disjunta).

Se definen ahora los conjuntos

$$\mathcal{A}^1 = \{e_j : e_j \in \mathcal{A} \text{ y } e_j \text{ está en el camino de } F(e_r) \text{ a } i, \text{ para algún } i \in \mathcal{M}^1\},$$
$$\mathcal{A}^2 = \{e_j : e_j \in \mathcal{A} \text{ y } e_j \text{ está en el camino de } T(e_r) \text{ a } i, \text{ para algún } i \in \mathcal{M}^2\}.$$

El arco $e_r \notin \mathcal{A}^1 \cup \mathcal{A}^2$. Se verá que cualquier otro arco de $\mathcal{A}$ está en $\mathcal{A}^1 \cup \mathcal{A}^2$. Sea un arco e_u tal que $e_u \in \mathcal{A} - \{e_r\}$. Si $F(e_u) = F(e_r)$, entonces $e_u \in \mathcal{A}^1$. En caso contrario, si $F(e_u) \neq F(e_r)$, sea $P = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, e_{j_w}, i_{w+1}\}$ el único camino en $\mathcal{T}$ de $F(e_r)$ a $F(e_u)$ entonces si $e_r \in P$, e_r tiene que ser el primer arco del camino. Se consideran ahora cuatro casos:

1. Si $e_r, e_u \notin P$, entonces $Q = \{i_1, e_{j_1}, \ldots, i_{w+1}, e_u, T(e_u)\}$ es un camino de $F(e_r)$ a $T(e_u)$ tal que $e_r \notin Q$. Por lo tanto, $T(e_u) \in \mathcal{M}^1$ y $e_u \in \mathcal{A}^1$.

2. Si $e_r \notin P$ y $e_u \in P$, entonces $F(e_u) \in \mathcal{M}^1$ y $e_u \in \mathcal{A}^1$.

3. Si $e_r \in P$ y $e_u \notin P$, entonces $Q = \{i_2, e_{j_2}, \ldots, i_{w+1}, e_u, T(e_u)\}$ es un camino de $T(e_r)$ a $T(e_u)$ tal que $e_r \notin Q$. Por lo tanto $T(e_u) \in \mathcal{M}^2$ y $e_u \in \mathcal{A}^2$.

4. Si $e_r, e_u \in P$. Si $T(e_u) = T(e_r)$, entonces $e_u \in \mathcal{A}^2$. En caso contrario, si $T(e_u) \neq T(e_r)$, entonces $Q = \{i_2, e_{j_2}, \ldots, i_{w+1}\}$ el un camino de $T(e_r)$ a $F(e_u)$ tal que $e_r \notin Q$. Por lo tanto, $F(e_u) \in \mathcal{M}^2$ y $e_u \in \mathcal{A}^2$.

Por tanto, se tiene que $e_u \in \mathcal{A}^1 \cup \mathcal{A}^2$.

Ahora se verá que $\mathcal{A}^1$ y $\mathcal{A}^2$ son conjuntos disjuntos. Para ello, por reducción al absurdo se supone que existe un arco e_j tal que $e_j \in \mathcal{A}^1 \cap \mathcal{A}^2$. Entonces existen nudos $p \in \mathcal{M}^1$ y $q \in \mathcal{M}^2$, y caminos $P = \{i_1, e_{j_1}, \ldots, i_t\}$ de $F(e_r)$ a p y $P' = \{i'_1, e'_{j_1}, \ldots, i'_v\}$ de $T(e_r)$ a q, con $e_j \in P$ y $e_j \in P'$. Por consiguiente, existen k y l dos enteros positivos tal que $i_k = i'_l = F(e_j)$. Entonces $Q = \{i_1, e_{j_1}, \ldots, i_k\}$ y $Q' = \{i'_1, e'_{j_1}, \ldots, i'_l\}$ son los caminos de $F(e_r)$ a $F(e_j)$ y de $T(e_r)$ a $F(e_j)$, respectivamente. Por definición de $\mathcal{M}^1$ y $\mathcal{M}^2$, $e_r \notin P$ y $e_r \notin P'$, luego $e_r \notin Q$ y $e_r \notin Q'$. Se tiene entonces que $F(e_j) \in \mathcal{M}^1$ y $F(e_j) \in \mathcal{M}^2$, absurdo porque se ha visto previamente que $\mathcal{M}^1 \cap \mathcal{M}^2 = \emptyset$.

Se ha demostrado entonces que $\mathcal{A}^1 \cap \mathcal{A}^2 = \emptyset$ y que $\mathcal{A} = \mathcal{A}^1 \cup \mathcal{A}^2 \cup \{e_r\}$ (unión disjunta).

Se definen $\mathcal{T}^1 = (\mathcal{N}^1, \mathcal{A}^1)$ y $\mathcal{T}^2 = (\mathcal{N}^2, \mathcal{A}^2)$, que son subgrafos de $\mathcal{T}$ tal que $\mathcal{N}^1 \cap \mathcal{N}^2 = \emptyset$ y $\mathcal{A}^1 \cap \mathcal{A}^2 = \emptyset$. Se demostrará ahora que $\mathcal{T}^1$ y $\mathcal{T}^2$ son grafos conectados. Sean $p, q \in \mathcal{N}^1$ distintos, se consideran tres casos.

1. Si $p = F(e_r)$, entonces $q \in \mathcal{M}^1$ y por definición de $\mathcal{M}^1$ existe un camino de p a q, que necesariamente está en $\mathcal{T}^1$.

2. Si $q = F(e_r)$, entonces $p \in \mathcal{M}^1$ y por definición de $\mathcal{M}^1$ existe un camino de q a p, que necesariamente está en $\mathcal{T}^1$.

3. Si $p \neq F(e_r)$ y $q \neq F(e_r)$, entonces por definición de $\mathcal{M}^1$, existen caminos $P = \{i_1, e_{j_1}, \ldots, i_t\}$ y $P' = \{i'_1, e'_{j_1}, \ldots, i'_v\}$ en $\mathcal{T}$ de $F(e_r)$ a p y de $F(e_r)$ a q, respectivamente. Necesariamente P y P' también son caminos en $\mathcal{T}^1$. Sea k el mayor entero tal que $i_k = i'_k$, dicho entero existe ya que $i_1 = i'_1$, $i_t = p$, $i'_v = q$ y $p \neq q$. Se define la secuencia $Q = \{i_t, e_{j_{t-1}}, i_{t-1}, e_{j_{t-2}}, \ldots, i_k = i'_k, e'_{j_k}, i'_{k+1}, e'_{j_{k+1}}, \ldots, i'_v\}$, que es un camino de p a q y que necesariamente es un camino en $\mathcal{T}^1$.

Entonces, $\mathcal{T}^1$ es un grafo conectado. Análogamente, $\mathcal{T}^2$ es conectado.

Como $\mathcal{T}$ cumple (ii), $\mathcal{T}^1$ y $\mathcal{T}^2$ heredan también (ii). El número de nudos de $\mathcal{T}^1$ y de $\mathcal{T}^2$ es menor que $m + 1$ y cada uno tiene al menos un nudo. Por inducción, como se había supuesto cierto (iii) hasta m nudos, se puede afirmar que $\mathcal{T}^1$ y $\mathcal{T}^2$ tienen un arco menos que nudos. Sean m_1 y m_2 los cardinales de $\mathcal{N}^1$ y $\mathcal{N}^2$, respectivamente, entonces por ser disjuntos y contener todos los nudos de $\mathcal{T}$, se cumple $m_1 + m_2 = m + 1$. Como $\mathcal{A}^1$, $\mathcal{A}^2$ y $\{e_r\}$ son disjuntos dos a dos y contienen todos los arcos de $\mathcal{T}$, entonces el

número de arcos de $\mathcal{T}$ es $(m_1 - 1) + (m_2 - 1) + 1 = m_1 + m_2 - 1 = m$. Se ha demostrado que (iii) se cumple para árboles con $m + 1$ nudos, luego se ha completado la demostración por inducción.

(iii)⇒(iv). Se supone (iii) cierto y sea m el número de nudos en $\mathcal{T} = (\mathcal{N}, \mathcal{A})$, solo queda demostrar que $\mathcal{T}$ es acíclico. Por reducción al absurdo, se supone que hay un ciclo C de longitud s en $\mathcal{T}$. Entonces C tiene s nudos distintos y s arcos distintos. Sean $\mathcal{N}^1$ y $\mathcal{A}^1$ el conjunto de nudos y el conjunto de arcos, respectivamente, que componen el ciclo C, entonces $\mathcal{N} - \mathcal{N}^1 \neq \emptyset$, ya que si fuera $s = m$ entonces no se tendría un arco menos que nudos y contradiría la hipótesis inicial. Sea $\mathcal{N}^2$ el conjunto de nudos de $\mathcal{N} - \mathcal{N}^1$ que están conectados con un nudo de $\mathcal{N}^1$ por algún camino de longitud 1. Por ser $\mathcal{T}$ conectado, $\mathcal{N}^2 \neq \emptyset$. Sea $\mathcal{A}^2$ el conjunto de arcos que forman un camino de longitud 1 entre algún nudo de $\mathcal{N}^2$ y algún nudo de $\mathcal{N}^1$, $\mathcal{A}^2$ tiene al menos tantos arcos como nudos tiene $\mathcal{N}^2$ y además ninguno pertenece a $\mathcal{A}^1$. Mientras que $\mathcal{N} - \bigcup_{j=1,\dots,i} \mathcal{N}^j \neq \emptyset$, se puede continuar con el proceso, definiendo $\mathcal{N}^{i+1}$ como el conjunto de todos los nudos en $\mathcal{N} - \bigcup_{j=1,\dots,i} \mathcal{N}^j$ conectados con un nudo de $\mathcal{N}^i$ por un camino de longitud 1 y definiendo $\mathcal{A}^{i+1}$ como el conjunto de arcos que forman un camino de longitud 1 entre algún nudo de $\mathcal{N}^{i+1}$ y alguno de $\mathcal{N}^i$. Del mismo modo que antes, $\mathcal{A}^{i+1}$ contiene al menos tantos arcos como nudos, y ninguno de los arcos pertenece a $\bigcup_{j=1,\dots,i} \mathcal{A}^j$. Por ser $\mathcal{N}$ finito, el proceso acaba en algún momento. Cuando eso ocurra, se habrán identificado tantos nudos como arcos, lo que contradice la hipótesis de que $\mathcal{T}$ tiene un arco menos que nudos. Se concluye entonces que $\mathcal{T}$ es acíclico y por tanto, que cumple (iv).

(iv)⇒(i). Suponiendo (iv) cierto, solo queda demostrar que $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ es conectado. Sea $\mathcal{M}$ el conjunto de los subgrafos conectados maximales de $\mathcal{T}$ (un subgrafo se dice conectado maximal si no existe otro subgrafo conectado de $\mathcal{T}$ que le contenga), y sea $p \in \mathcal{N}$. Si no existe un arco en $\mathcal{A}$ que sea incidente en p, entonces $\mathcal{T}^1 = (\{p\}, \emptyset) \in \mathcal{M}$. En caso contrario, si e_j es un arco de $\mathcal{A}$ incidente en p, entonces $\mathcal{T}^2 = (\{F(e_j), T(e_j)\}, \{e_j\})$ es un subgrafo de algun grafo de $\mathcal{M}$. Por lo tanto, cualquier nudo de $\mathcal{N}$ estará en algún grafo de $\mathcal{M}$. Con el mismo razonamiento, se afirma que cualquier arco de $\mathcal{A}$ estará en algún grafo de $\mathcal{M}$. Los conjuntos de nudos y arcos de los grafos de $\mathcal{M}$ son disjuntos dos a dos, ya que en caso contrario no serían subgrafos maximales. Por ser $\mathcal{T}$ acíclico, cada componente de $\mathcal{M}$ también es acíclica, luego un árbol. Si $\mathcal{M}$ se compone de al menos dos grafos, entonces el número de arcos en algún grafo de $\mathcal{M}$ necesariamente es menor que la cantidad de nudos del grafo menos uno (ya que $\mathcal{T}$ tiene un arco menos que nudos). Sin embargo, por ser árboles, los grafos de $\mathcal{M}$ tienen que tener un arco menos que nudos. Es decir, necesariamente $\mathcal{M}$ tiene que estar compuesto únicamente por un elemento, el propio $\mathcal{T}$. Se ha demostrado entonces que $\mathcal{T}$ es conectado y por consiguiente, que es un árbol. □

Definición 1.1.12. Se dice *grado* de un nudo i al número de arcos incidentes

en i.

Un nudo de grado 1 se denomina *nudo final*.

Proposición 1.1.2. *La suma de los grados de todos los nudos de un grafo es dos veces el número de arcos del grafo.*

Demostración. Cada arco es incidente en dos nudos, nudo from y nudo to, luego cada arco suma 2 al sumatorio de grados. Entonces la suma de de los grados de todos los nudos es dos veces el número de arcos. □

Teorema 1.1.3. *Un árbol $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ con $|\mathcal{N}| \geq 2$ tiene al menos dos nudos finales.*

Demostración. Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ un árbol con $|\mathcal{N}| = m \geq 2$. Todo nudo de $\mathcal{T}$ es incidente en al menos un arco (en caso contrario, si existiera un nudo que no fuera incidente en ningún arco, el grafo no sería conectado y por lo tanto no sería un árbol). Por el Teorema 1.1.1, $\mathcal{T}$ tiene $m-1$ arcos. Por reducción al absurdo, se supone que $\mathcal{T}$ tiene menos de dos nudos finales. Pueden darse dos casos:

1. $\mathcal{T}$ no tiene nudos finales, es decir, para cada $i \in \mathcal{N}$, $D(i) \geq 2$. Entonces $\sum_{i \in \mathcal{N}} D(i) \geq 2m > 2(m-1)$. Absurdo, ya que por la Proposición 1.1.2, $\sum_{i \in \mathcal{N}} D(i) = 2(m-1)$.

2. $\mathcal{T}$ tiene un nudo final j. Es decir, $D(j) = 1$ y para cada $i \in \mathcal{N} - \{j\}$, $D(i) \geq 2$. Entonces, $\sum_{i \in \mathcal{N}} D(i) = 1 + \sum_{i \in \mathcal{N} - \{j\}} D(i) \geq 1 + 2(m-1)$. Absurdo, ya que por la Proposición 1.1.2, $\sum_{i \in \mathcal{N}} D(i) = 2(m-1) < 1 + 2(m-1)$.

Por lo tanto se concluye que $\mathcal{T}$ tiene al menos dos nudos finales. □

Proposición 1.1.4. *Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ un árbol con $m \geq 2$ nudos. Sea i un nudo final de $\mathcal{T}$ y e_j el único arco incidente sobre i en $\mathcal{T}$. Si $\hat{\mathcal{N}} = \mathcal{N} - \{i\}$ y $\hat{\mathcal{A}} = \mathcal{A} - \{e_j\}$, entonces $\hat{\mathcal{T}} = (\hat{\mathcal{N}}, \hat{\mathcal{A}})$ es un árbol.*

Demostración. $\mathcal{T}$ es un árbol, luego es acíclico por el Teorema 1.1.1. Entonces, $\hat{\mathcal{T}}$ es acíclico por ser un subgrafo de $\mathcal{T}$. Además, como $\mathcal{T}$ tiene un arco menos que nudos y $\hat{\mathcal{T}}$ tiene un arco y un nudo menos que $\mathcal{T}$, $\hat{\mathcal{T}}$ también tiene un arco menos que nudos. Por el Teorema 1.1.1, $\hat{\mathcal{T}}$ es un árbol. □

Teorema 1.1.5. *Todo grafo conectado $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ con al menos un nudo, contiene un subgrafo que es un árbol extendido.*

Demostración. Si $\mathcal{G}$ es un árbol, no hay nada que demostrar. En caso contrario, si $\mathcal{G}$ no es un árbol entonces $\mathcal{G}$ tiene algún ciclo (es conectado, por lo tanto le falta ser acíclico para ser árbol). Sea e_{j_1} un arco que forma el ciclo, se define $\mathcal{G}_1 = (\mathcal{N}, \mathcal{A}_1)$, donde $\mathcal{A}_1 = \mathcal{A} - e_{j_1}$, que es el grafo $\mathcal{G}$ eliminando el arco e_{j_1} (nótese que $\mathcal{G}_1$ sigue siendo conectado). Si $\mathcal{G}_1$ es un árbol, se ha terminado la demostración. En caso contrario $\mathcal{G}_1$ tiene algún ciclo y del mismo modo que en el paso anterior se define $\mathcal{G}_2 = (\mathcal{N}, \mathcal{A}_2)$, con $\mathcal{A}_2 = \mathcal{A}_1 - e_{j_2}$. Se procede con el mismo mecanismo y por ser $\mathcal{G}$ finito existe un k tal que $\mathcal{G}_k$ no contiene ciclos. Como también es conectado (por construcción), $\mathcal{G}_k$ es un árbol. Además, como $\mathcal{G}_k$ es subgrafo extendido de $\mathcal{G}$ (ya que no se han eliminado nudos), entonces se concluye que $\mathcal{G}_k$ es un árbol extendido en $\mathcal{G}$. □

1.2. Matriz de incidencia nudo-arco

La información de un grafo $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, con $\mathcal{N} = \{1, 2, \ldots, m\}$ y $\mathcal{A} = \{e_1, e_2, \ldots, e_n\}$ se puede dar mediante la denominada matriz de incidencia nudo-arco.

Definición 1.2.1. Se define la *matriz de incidencia nudo-arco*, o simplemente *matriz de incidencia* A de un grafo $\mathcal{G}$ como la matriz con una fila asociada a cada nudo y una columna a cada arco ($a^i_{e_j}$ elemento de A asociado al nudo i y arco e_j) definida de la siguiente manera:

$$a^i_{e_j} = \begin{cases} 1, & \text{si } F(e_j) = i \\ -1, & \text{si } T(e_j) = i \\ 0, & \text{en otro caso.} \end{cases}$$

Equivalentemente, la columna de A asociada al arco e_j viene dada por

$$a_{e_j} = e^{F(e_j)} - e^{T(e_j)}$$

donde e^k representa el vector canónico k-ésimo de la base de $\mathbb{R}^m$ (vector de ceros con elemento 1 en la posición k).

La matriz de incidencia nudo-arco asociada al grafo de la Figura 1.1 es:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{bmatrix} -1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix} \end{array}$$

Proposición 1.2.1. *Sea $P = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, i_s, e_{j_s}, i_{s+1}\}$ un camino o un ciclo en un grafo propio $\mathcal{G}$. Sea A la matriz de incidencia de $\mathcal{G}$ y $a_{e_{j_k}}$ la columna de A asociada al arco e_{j_k}. Entonces*

$$\sum_{k=1}^{s} O_{e_{j_k}}(P)\, a_{e_{j_k}} = e^{i_1} - e^{i_{s+1}}$$

Demostración. Sea $k \in \{1, \ldots, s\}$.

1. Si $e_{j_k} = (i_k, i_{k+1})$, $O_{e_{j_k}}(P)\, a_{e_{j_k}} = (+1)(e^{i_k} - e^{i_{k+1}}) = e^{i_k} - e^{i_{k+1}}$.
2. Si $e_{j_k} = (i_{k+1}, i_k)$, $O_{e_{j_k}}(P)\, a_{e_{j_k}} = (-1)(e^{i_{k+1}} - e^{i_k}) = e^{i_k} - e^{i_{k+1}}$.

Por lo tanto, en ambos casos $O_{e_{j_k}}(P)\, a_{e_{j_k}} = e^{i_k} - e^{i_{k+1}}$. Como en un camino cada arco es incidente en los nudos anterior y posterior de la secuencia, se cancelan todos los elementos de la suma menos el primer término y el último, luego se cumple $\sum_{k=1}^{s} O_{e_{j_k}}(P)\, a_{e_{j_k}} = e^{i_1} - e^{i_{s+1}}$. □

Corolario 1.2.2. *Sea $C = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, i_s, e_{j_s}, i_{s+1}\}$ un ciclo en un grafo propio $\mathcal{G}$ con matriz de incidencia A, entonces*

$$\sum_{k=1}^{s} O_{e_{j_k}}(P)\, a_{e_{j_k}} = 0.$$

Demostración. Como $C = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, i_s, e_{j_s}, i_{s+1}\}$ es un ciclo, por la Proposición 1.2.1, $\sum_{k=1}^{s} O_{e_{j_k}}(P)\, a_{e_{j_k}} = e^{i_1} - e^{i_{s+1}}$. Además, por ser C ciclo, se tiene $i_1 = i_{s+1}$, luego $e^{i_1} - e^{i_{s+1}} = 0$ y queda probado el corolario. □

Corolario 1.2.3. *Sea $C = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, i_s, e_{j_s}, i_{s+1}\}$ un ciclo en un grafo propio $\mathcal{G}$ con matriz de incidencia A, entonces los vectores $\{a_{e_{j_k}} : k = 1, \ldots, s\}$ son linealmente dependientes.*

Demostración. Como C es un ciclo, por el Corolario 1.2.2, existe una combinación lineal de columnas $\{a_{e_{j_k}} : k = 1, \ldots, s\}$ con coeficientes no nulos que resulta en el vector nulo, luego $\{a_{e_{j_k}} : k = 1, \ldots, s\}$ forman un sistema linealmente dependiente. □

Teorema 1.2.4. *Sea A la matriz de incidencia de un grafo propio $\mathcal{G}$. Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ un árbol en $\mathcal{G}$ con al menos dos nudos. Entonces $\{a_{e_j} : e_j \in \mathcal{A}\}$ es linealmente independiente.*

Demostración. Sea $|\mathcal{N}| = m$. Si $m = 2$, por el Teorema 1.1.1, $|\mathcal{A}| = m - 1 = 1$. Sea e_j ese arco, como $a_{e_j} = e^{F(e_j)} - e^{T(e_j)} \neq 0$, el teorema se cumple para $m = 2$.

Supóngase ahora $m > 2$. Sea n el mayor número entero tal que para cualquier árbol $\hat{\mathcal{T}} = (\hat{\mathcal{N}}, \hat{\mathcal{A}})$ en $\mathcal{G}$ con k nudos, $2 \leq k \leq n$, el sistema $\{a_{e_j} : e_j \in \hat{\mathcal{A}}\}$ es linealmente independiente, entonces $n \geq 2$, como ya se

ha visto en el párrafo anterior. Por reducción al absurdo se supone $n < m$. Por definición de n, existe un árbol $\hat{\mathcal{T}} = (\hat{\mathcal{N}}, \hat{\mathcal{A}})$ con $n+1$ nudos tal que $\{a_{e_j} : e_j \in \hat{\mathcal{A}}\}$ es linealmente dependiente, es decir, existe un conjunto de constantes $\{c_j : e_j \in \hat{\mathcal{A}}\}$, no todas cero, tal que $\sum_{e_j \in \hat{\mathcal{A}}} c_j a_{e_j} = 0$. Sea p un nudo final de $\hat{\mathcal{T}}$ (existe, por el Teorema 1.1.3) y sea e_w el arco incidente en p. Sea $\bar{\mathcal{T}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$, donde $\bar{\mathcal{N}} = \hat{\mathcal{N}} - \{p\}$ y $\bar{\mathcal{A}} = \hat{\mathcal{A}} - \{e_w\}$, por el Teorema 1.1.4, $\bar{\mathcal{T}}$ es un árbol. Como p solo es incidente en e_w, entonces el elemento $a^p_{e_j} = 0$ para todo $e_j \in \bar{\mathcal{A}}$ ($a^p_{e_j}$ es el elemento de A asociado al nudo p y arco e_j). Entonces

$$\sum_{e_j \in \hat{\mathcal{A}}} c_j a^p_{e_j} = c_w a^p_{e_w} + \sum_{e_j \in \bar{\mathcal{A}}} c_j a^p_{e_j} = c_w a^p_{e_w} = 0.$$

Como $a^p_{e_w} = \pm 1$, $c_w = 0$. Pero entonces $\sum_{e_j \in \mathcal{A}} c_j a_{e_j} = 0$ y $\{c_j : e_j \in \bar{\mathcal{A}}\}$ no son todos cero, luego $\{a_{e_j} : e_j \in \bar{\mathcal{A}}\}$ es linealmente dependiente. Absurdo (por ser $\bar{\mathcal{T}}$ un árbol con n nudos y $\{a_{e_j} : e_j \in \bar{\mathcal{A}}\}$ linealmente dependiente, lo que se contradice con la definición de n). Se concluye entonces $n = m$, luego cualquier árbol de $\mathcal{G}$ con al menos dos nudos cumple el teorema. □

Teorema 1.2.5. *Sea A la matriz de incidencia de un grafo propio conectado $\mathcal{G}$ con m nudos. Entonces el rango de A es $m-1$.*

Demostración. $\mathcal{G}$ contiene un árbol extendido $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ por el Teorema 1.1.5, que tiene m nudos por ser extendido y $m-1$ arcos por el Teorema 1.1.1. Por el Teorema 1.2.4, $\{a_{e_j} : e_j \in \mathcal{A}\}$ es linealmente independiente, luego el rango de A es al menos $m-1$. Como la suma de las m filas de A es el 0, por estar cada columna compuesta de un $+1$, un -1 y el resto ceros, entonces el rango de A es menor que m. Se concluye entonces que el rango de A es $m-1$. □

Proposición 1.2.6. *Sea A la matriz de incidencia de un grafo propio $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ con m nudos. Si $\hat{\mathcal{A}}$ es un subconjunto con $m-1$ arcos de $\mathcal{A}$ tal que $\{a_{e_j} : e_j \in \hat{\mathcal{A}}\}$ es linealmente independiente. Entonces $\mathcal{T} = (\mathcal{N}, \hat{\mathcal{A}})$ es un árbol.*

Demostración. Por el Corolario 1.2.3, como $\{a_{e_j} : e_j \in \hat{\mathcal{A}}\}$ es linealmente independiente, $\mathcal{T}$ es acíclico. Además, $\mathcal{T}$ tiene un arco menos que nudos luego es un árbol por el Teorema 1.1.1. □

1.3. La variable artificial

Sea $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ un grafo propio conectado con $\mathcal{N} = \{1, 2, \ldots, m\}$ el conjunto de nudos, $\mathcal{A} = \{e_1, e_2, \ldots, e_n\}$ el conjunto de arcos y matriz de incidencia nudo-arco A. Para aplicar el método simplex la matriz de condiciones debe

tener rango completo, en este caso m. Sin embargo A es de rango $m-1$, ver Teorema 1.2.5.

Este hecho se soluciona añadiendo una variable artificial, x_a, correspondiente a un arco ficticio con origen en uno de los nudos. A dicho nudo se le denomina *nudo raíz* y al arco artificial, *arco raíz*. Se construye por lo tanto la nueva matriz de incidencia, que es $[A, e^l]$, con l un entero positivo tal que $1 \leq l \leq m$. Véase que lo que se está diciendo es que el arco artificial tiene origen en el nudo l (nudo raíz).

El problema queda formulado de la siguiente forma:

$$\begin{aligned} \text{mín}\, z &= c\, x \\ Ax + e^l x_a &= b \\ 0 \leq x &\leq u \\ 0 \leq x_a &\leq 0 \end{aligned} \tag{1.1}$$

1.4. Caracterización de las bases

Definición 1.4.1. Un *árbol extendido con raíz* es un árbol extendido con un nudo marcado, el nudo raíz, y su correspondiente arco raíz.

Proposición 1.4.1. *Sea A la matriz de incidencia nudo-arco de un grafo propio conectado $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ con $|\mathcal{N}| = m$ y sea $\mathcal{T} = (\mathcal{N}, \hat{\mathcal{A}})$ un árbol extendido en $\mathcal{G}$. Entonces $\Omega = \{a_{e_j} : e_j \in \hat{\mathcal{A}}\} \cup \{e^l\}$ es una base de $\mathbb{R}^m$.*

Demostración. Sea $p \in \{1, \ldots, m\} - \{l\}$, como $\mathcal{T}$ es un árbol extendido, entonces hay un único camino $P = \{i_1, e_{j_1}, i_2, e_{j_2}, \ldots, e_{j_s}, i_{j_{s+1}}\}$ en $\mathcal{T}$ de p a l, ver Teorema 1.1.1. Por la Proposición 1.2.1,

$$\sum_{k=1}^{s} O_{e_{j_k}}(P)\, a_{e_{j_k}} = e^p - e^l.$$

Luego $e^p = e^l + \sum_{k=1}^{s} O_{e_{j_k}}(P)\, a_{e_{j_k}}$. Además, $e^l \in \{a_{e_j} : e_j \in \hat{\mathcal{A}}\} \cup \{e^l\}$. Entonces, los vectores $\{e^p : p = 1, \ldots, m\}$ pueden ser expresados como combinaciónes lineales de vectores de $\{a_{e_j} : e_j \in \hat{\mathcal{A}}\} \cup \{e^l\}$. Se concluye que, como $\{e^p : p = 1, \ldots, m\}$ es un sistema generador de $\mathbb{R}^m$, entonces $\{a_{e_j} : e_j \in \hat{\mathcal{A}}\} \cup \{e^l\}$ es también un sistema generador de $\mathbb{R}^m$. Además es base, ya que $|\{a_{e_j} : e_j \in \hat{\mathcal{A}}\} \cup \{e^l\}| = (m-1)+1 = m$, ver Teorema 1.1.1. □

Nótese en la Proposición 1.4.1 se ve que e^l junto con las columnas de A asociadas a un árbol extendido forman un sistema linealmente independiente de m vectores, luego forman una base para $[A, e^l]$ y esta matriz es de rango completo m.

Proposición 1.4.2. *Sea A la matriz de incidencia de un grafo propio conectado con raíz $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, con nudo raíz l. Si Ω es una base para $[A, e^l]$,*

entonces $e^l \in \Omega$ *y* $\mathcal{T} = (\mathcal{N}, \hat{\mathcal{A}})$*, con* $\hat{\mathcal{A}} = \{e_j : a_{e_j} \in \Omega\}$*, es un árbol extendido en* $\mathcal{G}$*.*

Demostración. Supóngase $|\mathcal{N}| = m$. Sea Ω una base para $[A, e^l]$, como A tiene rango $m-1$, ver Teorema 1.2.5, y $[A, e^l]$ tiene rango completo m, necesariamente $e^l \in \Omega$. Por lo tanto, $\{a_{e_j} \in \Omega\}$ son los $m-1$ vectores linealmente independientes restantes. Sea $\hat{\mathcal{A}} = \{e_j : a_{e_j} \in \Omega\}$, entonces por la Proposición 1.2.6, $\mathcal{T} = (\mathcal{N}, \hat{\mathcal{A}})$ es un árbol extendido en $\mathcal{G}$. □

Teorema 1.4.3. *Sea A la matriz de incidencia de un grafo propio* $\mathcal{G}$ *conectado con nudo raíz l. Las bases para* $[A, e^l]$ *son* e^l *junto con un conjunto de columnas de A correspondientes a un árbol extendido en* $\mathcal{G}$*.*

Demostración. Las condiciones necesaria y suficiente son resultado directo de la Proposición 1.4.1 y la Proposición 1.4.2. □

Definición 1.4.2. Una matriz se dice que es *triangular inferior (superior)* si mediante permutaciones de filas y columnas puede ser escrita de forma que todos los elementos por encima (debajo) de la diagonal principal sean cero.

Proposición 1.4.4. *Sea A la matriz de incidencia de un grafo propio conectado con raíz* $\mathcal{G}$*, con nudo raíz l y sea B una base de* $[A, e^l]$*. Entonces B es triangular.*

Demostración. Sea $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ el árbol extendido con raíz asociado a B y sea $|\mathcal{N}| = m$. Por ser $\mathcal{G}$ un grafo propio, con $m \geq 2$ y por el Teorema 1.1.3, $\mathcal{T}$ tiene al menos dos nudos finales. Sea i_1 un nudo final de $\mathcal{T}$ distinto del nudo raíz y sea e_{j_1} el único arco incidente en i_1. Entonces la fila de B asociada a i_1 tiene un único elemento distinto de cero. Permutando filas y columnas de B de modo que la primera fila y columna de B sean las correspondientes al nudo i_1 y arco e_{j_1}, B queda de la forma

$$\left[\begin{array}{c|c} \pm 1 & 0 \\ \hline \pm e^s & B^1 \end{array}\right].$$

Sea $\mathcal{T}^1 = (\mathcal{N} - \{i_1\}, \mathcal{A} - \{e_{j_1}\})$, entonces $\mathcal{T}^1$ tiene $m-1$ arcos y es un árbol (con raíz) por la Proposición 1.1.4. Si $m-1=1$ entonces B^1 es una matriz 1×1, luego B es triangular y se acaba la demostración. En caso contrario, se tiene $m-1 \geq 2$. Por el Teorema 1.1.3, $\mathcal{T}^1$ tiene al menos dos nudos finales. Sea i_2 un nudo final que no sea el nudo raíz y sea e_{j_2} el único arco incidente en i_2. Entonces la fila de B^1 asociada a i_2 tiene un único elemento distinto de cero. Permutando filas y columnas de B^1 de modo que la primera fila y columna de la matriz resultante correspondan al nudo i_2 y al arco e_{j_2} se obtiene la matriz

$$\left[\begin{array}{c|c|c} \pm 1 & 0 & 0 \\ \hline \pm e^s_{(1)} & \pm 1 & 0 \\ \hline \pm e^s_{(2)} & e^t & B^2 \end{array}\right].$$

Repitiendo el proceso se llega a $\mathcal{T}^{m-1}$ con $m-(m-1)=1$ arcos, luego B^{m-1} es una matriz 1×1. Se concluye entonces que B es triangular. □

La mayor parte de resultados teóricos que aparecen en las secciones 1.1, 1.2, 1.3 y 1.4 han sido extraidas o adaptadas de J. L. Kennignton, R. Helgason, ver [12].

1.5. Unimodularidad total e intregralidad

Se examinarán en esta sección dos importantes características de los problemas de redes. La primera es la unimodularidad total de la matriz de incidencia nudo-arco de un grafo. La segunda es la integralidad de las soluciones de un problema de redes bajo ciertas hipótesis. La base de estos resultados puede encontrarse en Mokhtar S. Bazaraa, ver [4].

Definición 1.5.1. Se dice que una matriz A es *totalmente unimodular* si cualquier submatriz cuadrada de A tiene determinante $+1$, -1 o 0.

Teorema 1.5.1. *La matriz de incidencia A de un grafo $\mathcal{G}$ es totalmente unimodular.*

Demostración. Se demostrará por inducción sobre la dimensión de las submatrices de A. Para matrices 1×1 es trivial, ya que los elementos de A son ± 1 o 0. Se supone cierto para las submatrices de dimensión $(k-1)\times(k-1)$. Sea A_k cualquier submatriz $k\times k$, con $k\geq 2$, se analizarán los tres casos posibles. Si A_k tiene alguna columna completa de ceros, entonces $\det A_k = 0$. Si alguna columna de A_k tiene un único elemento distinto de 0 (tiene que ser ± 1) entonces $\det A_k = \pm \det A_{k-1}$, que es ± 1 o 0, ya que se ha supuesto que $\det A_{k-1}$ es ± 1 o 0. Por último, si todas las columnas contienen los elementos $+1$ y -1, la suma de todas las filas resulta en el vector nulo y por tanto $\det A_k = 0$. Se concluye entonces que A es totalmente unimodular. □

Nótese que si se suprimen filas o columnas de A la matriz resultante sigue siendo totalmente unimodular. Además, por el propio desarrollo de la demostración del Teorema 1.5.1, si se añade a la matriz A un vector (una columna) de ceros excepto un elemento ± 1 la nueva matriz que se obtiene también es totalmente unimodular. Se puede dar entonces el siguiente resultado directo.

Corolario 1.5.2. *La matriz de incidencia $[A, e^l]$ de un grafo propio conectado $\mathcal{G}$ con nudo raíz l es totalmente unimodular.*

Corolario 1.5.3. *Sea $\mathcal{G}$ un grafo propio conectado con nudo raíz l y matriz de incidencia $[A, e^l]$. Sea B una matriz básica de $[A, e^l]$, entonces* $\det B = \pm 1$.

Demostración. Por ser B una submatriz cuadrada de la matriz de incidencia $[A, e^l]$, como $[A, e^l]$ es totalmente unimodular, entonces el determinante de B es 0 o ± 1. Como B es base, por definición el determinante de B es distinto de 0, luego el determinante de B es 1 o -1. □

Proposición 1.5.4. *Sea $[A, e^l]$ la matriz de incidencia de un grafo propio $\mathcal{G}$ conectado con nudo raíz l. Sea B una base de $[A, e^l]$, entonces B^{-1} está comuesta por elementos ± 1 y 0.*

Demostración. $B^{-1} = \frac{(\text{adj } B)^t}{\det B}$. B está compuesta por ± 1 y 0, y cualquier submatriz de B tiene determinante ± 1 o 0 por ser submatrices de $[A, e^l]$, ver Corolario 1.5.2, luego $(\text{adj } B)^t$ está compuesta por ± 1 o 0. Además, por el Corolario 1.5.3, $\det B = \pm 1$, entonces se cumple la proposición. □

Se verán ahora propiedades de la matríz Y, cuyas columnas vienen dadas por la ecuación $Y = B^{-1}N$, donde N está formada por las columnas de la matriz de condiciones $[A, e^l]$ asociadas a las variables no básicas.

Proposición 1.5.5. *La matriz Y está compuesta por elementos 0 y ± 1.*

Demostración. Se tiene que la columna y_{e_j} viene dada por el sistema $By_{e_j} = a_{e_j}$. Por el método de Cramer se puede obtener el elemento $y^i_{e_j}$ como

$$y^i_{e_j} = \frac{\det B^{(i)}}{\det B}$$

donde $B^{(i)}$ se obtiene de sustituir la columna i-ésima de B por a_{e_j}. Se puede afirmar por lo tanto que Y está compuesta por elementos ± 1 y 0. □

Para hallar las soluciones básicas de un problema lineal se tiene

$$x_B = B^{-1}b - B^{-1}Nx_N \tag{1.2}$$

con los valores del vector x_N fijados en su cota inferior 0 o en su cota superior.

Se asume que b y las cotas superiores de las variables son enteras (en caso de haberlas). Se ha visto que B^{-1} y $Y = B^{-1}N$ están compuestas por elementos 0 y ± 1. Entonces cualquier solución básica $x = [x_B; x_N]$, y por consiguiente, la solución óptima, serán enteras. De este modo, para hallar una solución entera de un problema de redes no será necesario el uso de algoritmos específicos de programación entera.

1.6. Método simplex para problemas de redes

Ahora se va a aplicar lo que se ha visto hasta este punto para hacer una especialización del método simplex, ahorrando operaciones y haciéndolas más rápidas para conseguir un algoritmo más eficiente que si se tratase el problema con el método simplex para problemas lineales.

Ya se ha visto que las bases para un problema de redes de la forma (1.1) son triangulares. En el método simplex para determinar la variable no básica que entra en la base es necesario efectuar la operación $c_N - c_B B^{-1} N$. Para calcular $c_B B^{-1} N$, sean $\pi = c_B B^{-1}$ las denominadas *variables duales*, π es la solución al sistema de ecuaciones $\pi B = c_B$. Como B es triangular, π puede obtenerse resolviendo el sistema por sustitución hacia atrás.

Supóngase $\mathcal{T}_B$ el árbol extendido con nudo raíz l. Determinar el valor de las variables duales se reduce a resolver el sistema

$$\begin{cases} \pi_l = 0 \\ \pi_{F(e_j)} - \pi_{T(e_j)}, \quad \text{si } e_j \in \mathcal{T}_B \end{cases} \tag{1.3}$$

el cual determina las variables duales de manera unívoca de forma iterativa, por ser $\mathcal{T}_B$ un grafo conectado y acíclico (el valor de una variable dual determina el valor de otra aún no hallada hasta determinar todas).

Una vez determinado π, para encontrar los candidatos no básicos a entrar a la base, para los $e_j \in N$ (arco no báscio), se tiene

$$(c_N - c_B B^{-1} N)_{e_j} < 0 \Leftrightarrow c_{e_j} - \pi a_{e_j} < 0 \Leftrightarrow c_{e_j} - (\pi_{F(e_j)} - \pi_{T(e_j)}) < 0 \Leftrightarrow$$
$$\Leftrightarrow \pi_{F(e_j)} - \pi_{T(e_j)} - c_{e_j} > 0.$$

Del mismo modo,

$$(c_N - c_B B^{-1} N)_{e_j} > 0 \Leftrightarrow \pi_{F(e_j)} - \pi_{T(e_j)} - c_{e_j} < 0.$$

Es decir, los conjuntos de variables no básicas candidatas a entrar en la base quedan del siguiente modo:

$$\Psi_1 = \{e_j : x_{e_j} = 0 \text{ y } \pi_{F(e_j)} - \pi_{T(e_j)} - c_{e_j} > 0\},$$

$$\Psi_2 = \{e_j : x_{e_j} = u_{e_j} \text{ y } \pi_{F(e_j)} - \pi_{T(e_j)} - c_{e_j} < 0\}.$$

Ahora, sea $e_k \in N$ (arco no básico), se hallará y_{e_k}, que viene dado por $y_{e_k} = B^{-1} a_{e_k}$. Equivalentemente y_{e_k} es solución del sistema $B y_{e_k} = a_{e_k} = e^{F(e_k)} - e^{T(e_k)}$. De nuevo, como B es triangular se puede obtener y_{e_k} sin necesidad de calcular B^{-1}. Para hallar y_{e_k}, sea $P = \{i_1, e_{k_1}, i_2, e_{k_2}, \ldots, e_{k_s}, i_{s+1}\}$ el único camino en $\mathcal{T}_B$ que va de $F(e_k) = i_1$ a $T(e_k) = i_{s+1}$. Por la Proposición 1.2.1, se tiene

$$\sum_{i=1}^{s} O_{e_{k_i}}(P)\, a_{e_{k_i}} = e^{F(e_k)} - e^{T(e_k)}. \tag{1.4}$$

Sean $e_{j_1}, e_{j_2}, \ldots, e_{j_m}$ los arcos correspondientes a las m columnas de B, entonces la columna y_{e_k} queda determinada de la siguiente forma:

$$y^i_{e_k} = \begin{cases} O_{e_{j_i}}(P), & \text{si } e_{j_i} \in P \\ 0, & \text{en otro caso.} \end{cases} \tag{1.5}$$

Supóngase que e_k es el arco no básico que entrará a la base. Se puede especializar el cálculo de Δ_1 y Δ_2, que quedan de forma

$$\Delta_1 = \min_{\sigma(y^i_{e_k})=\delta}\{x_{e_{j_i}}, \infty\} = \min_{O_{e_{j_i}}(P)=\delta}\{x_{e_{j_i}}, \infty\},$$

$$\Delta_2 = \min_{\sigma(y^i_{e_k})=-\delta}\{u_{e_{j_i}} - x_{e_{j_i}}, \infty\} = \min_{O_{e_{j_i}}(P)=-\delta}\{u_{e_{j_i}} - x_{e_{j_i}}, \infty\},$$

$$\Delta = \min\{\Delta_1, \Delta_2, u_{e_k}\}.$$

Una vez hallado Δ, se actualizan las variables de decisión

$$\begin{cases} x_{e_k} = x_{e_k} + \delta\Delta \\ x_{e_{k_i}} = x_{e_{k_i}} - \delta\Delta O_{e_{k_i}}(P), \quad \forall e_{k_i} \in P \end{cases}$$

Para determinar que variable sale de la base, tiene que ser una que se encuentre en una de sus cotas. Al añadir un arco al árbol extendido básico $\mathcal{T}_B$ se forma un ciclo en $\mathcal{T}_B$. En efecto, dicho ciclo es $P \cup \{e_k\}$, por lo tanto tendrá que ser un arco de P el que salga de la base, de modo que la nueva base tambien se corresponda con un árbol extendido.

Ahora se expondra el algoritmo simplex para redes que se ha ido elaborando en el transcurso del capítulo.

Algoritmo simplex para redes

- PASO 0.- Se comienza con una solución básica factible $x = [x_B; x_N]$, que se corresponde con un árbol extendido $\mathcal{T}_B$ y se calcula π resolviendo

$$\begin{cases} \pi_l = 0 \\ \pi_{F(e_j)} - \pi_{T(e_j)}, \quad \text{si } e_j \in \mathcal{T}_B. \end{cases}$$

- PASO 1.- Se determinan los candidatos a entrar en la base:

$$\Psi_1 = \{e_j : x_{e_j} = 0 \text{ y } \pi_{F(e_j)} - \pi_{T(e_j)} - c_{e_j} > 0\},$$
$$\Psi_2 = \{e_j : x_{e_j} = u_{e_j} \text{ y } \pi_{F(e_j)} - \pi_{T(e_j)} - c_{e_j} < 0\}.$$

  Si $\Psi_1 \cup \Psi_2 = \emptyset$, entonces $x = [x_B, x_N]$ es la solución óptima. En caso contrario se escoge $e_k \in \Psi_1 \cup \Psi_2$ (el de mayor $|\pi_{F(e_k)} - \pi_{T(e_k)} - c_{e_k}|$) y se fija

$$\delta = \begin{cases} 1 & \text{si } e_k \in \Psi_1 \\ -1 & \text{si } e_k \in \Psi_2. \end{cases}$$

- PASO 2.- Sea $P = \{i_1, e_{k_1}, i_2, \ldots, e_{k_s}, i_{s+1}\}$ el camino en $\mathcal{T}_B$ que va de $F(e_k)$ a $T(e_k)$. Se fijan

$$\Delta_1 = \min_{O_{e_{k_i}}(P)=\delta} \{x_{e_{k_i}}, \infty\},$$
$$\Delta_2 = \min_{O_{e_{k_i}}(P)=-\delta} \{u_{e_{k_i}} - x_{e_{k_i}}, \infty\},$$
$$\Delta = \min\{\Delta_1, \Delta_2, u_{e_k}\}.$$

- PASO 3.- Se actualizan los valores de las variables:

$$\begin{cases} x_{e_k} = x_{e_k} + \delta\Delta \\ x_{e_{k_i}} = x_{e_{k_i}} - \delta\Delta O_{e_{k_i}}(P), \quad \forall e_{k_i} \in P. \end{cases}$$

  Si $\Delta = u_{e_k}$ (es decir, no cambian los elementos de la base), entonces volver al paso 1.

- PASO 4.- Actualización de la base y de las variables duales. Sean

$$\Psi_3 = \{e_{k_i} : x_{e_{k_i}} = 0 \text{ y } O_{e_{k_i}}(P) = \delta\},$$
$$\Psi_4 = \{e_{k_i} : x_{e_{k_i}} = u_{e_{k_i}} \text{ y } O_{e_{k_i}}(P) = -\delta\}.$$

  Se selecciona un $e_t \in \Psi_3 \cup \Psi_4$ y el nuevo arbol básico es $\hat{\mathcal{T}_B} = (\mathcal{T}_B \cup \{e_k\}) - \{e_t\}$. Se actualizan las variables duales y se vuelve al paso 1.

1.7. Problema de flujo de redes de costo mínimo

Supóngase una red $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, con $\mathcal{N} = \{1, \ldots, m\}$. Cada nudo de la red $i \in \mathcal{N}$ tiene asociado un valor b_i. Se puede interpretar el valor de b_i del siguiente modo.

- Si $b_i > 0$, entonces el nudo i tiene una oferta b_i.
- Si $b_i < 0$, entonces el nudo i tiene una demanda de $-b_i$.
- Si $b_i = 0$, entonces el nudo i no tiene ni oferta ni demanda.

Asociado a cada arco $e_j \in \mathcal{A}$ se define $x_{e_j} \geq 0$ la cantidad de flujo sobre e_j. Además, x_{e_j} puede tener cota superior u_{e_j} que representa la cantidad de flujo máximo que puede soportar el arco e_j. Se define c_{e_j} el costo de envío por unidad por el arco e_j, es decir, del nudo $F(e_j)$ al nudo $T(e_j)$.

Sin pérdida de generalidad, se supone que la demanda total es igual a la oferta total, es decir, $\sum_{i=1}^{m} b_i = 0$. Si $\sum_{i=1}^{m} b_i > 0$, se añade un nudo ficticio $m+1$ con una demanda $b_{m+1} = -\sum_{i=1}^{m} b_i$ y los arcos con costo cero desde

los nudos con oferta hasta el nudo ficticio. De este modo el problema queda convertido a un problema donde la oferta total es igual a la demanda total.

El *problema de flujo de redes de costo mínimo* consiste en transportar la oferta de un producto por la red de modo que se satisfaga la demanda a costo mínimo. Matemáticamente, el problema se puede escribir de la siguiente forma:

$$\begin{aligned} \text{mín } z &= \sum_{e_j \in \mathcal{A}} c_{e_j} x_{e_j} \\ \sum_{e_j:F(e_j)=i} x_{e_j} - \sum_{e_j:T(e_j)=i} x_{e_j} &= b_i, \quad i = 1, \ldots, m \\ 0 \leq x_{e_j} &\leq u_{e_j}, \quad e_j \in \mathcal{A} \end{aligned} \tag{1.6}$$

junto con la condición $\sum_{i=1}^{m} b_i = 0$ (necesaria para la existencia de solución).

Las condiciones

$$\sum_{e_j:F(e_j)=i} x_{e_j} - \sum_{e_j:T(e_j)=i} x_{e_j} = b_i$$

expresan que para cada nudo i, la diferencia entre la cantidad de flujo que sale de i y la cantidad de flujo que llega a i tiene que ser b_i. Por esa razón si $b_i < 0$ el nudo i tiene un requerimiento y llega mas de lo que sale, si $b_i > 0$ el nudo i tiene una oferta y sale más de lo que llega y si $b_i = 0$ lo que llega al nudo i sale de nuevo.

Véase que no se ha considerado el caso $\sum_{i=1}^{m} b_i < 0$, ya que se estaría hablando de una demanda que supera la oferta, luego no sería posible satisfacer toda la demanda.

Nótese que la matriz de coeficientes del problema de redes de costo mínimo (1.6) es la matriz de incidencia nudo-arco A del grafo $\mathcal{G}$ que determina el problema.

Otra formulación del problema

Para los problemas de redes, se puede utilizar una formulación como un problema de programación lineal con variables acotadas.

Como ya se había dicho anteriormente, un arco e_j se puede expresar en la forma $(F(e_j), T(e_j))$. Supóngase $(F(e_j), T(e_j)) = (i, j)$. De este modo, la variable x_{e_j} se puede escribir x_{ij}. Es decir, x_{ij} representa el flujo que pasa por el arco que va del nudo i al nudo j. En efecto, si $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ con

$\mathcal{N} = \{1, \ldots, m\}$, el Problema 1.6 se puede escribir de la siguiente forma:

$$\begin{aligned} \text{mín } z = & \sum_{\substack{i,j=1 \\ (i,j)\in\mathcal{A}}}^{m} c_{ij}\, x_{ij} \\ \sum_{\substack{k=1 \\ (i,k)\in\mathcal{A}}}^{m} x_{ik} - & \sum_{\substack{k=1 \\ (k,i)\in\mathcal{A}}}^{m} x_{ki} = b_i, \quad i = 1, \ldots, m \\ & 0 \leq x_{ij} \leq u_{ij}, \quad (i,j) \in \mathcal{A} \end{aligned} \tag{1.7}$$

junto con la condición $\sum_{i=1}^{m} b_i = 0$ (necesaria para existencia de solución).

Del mismo modo, si se añade al grafo, y por consiguiente al problema, los posibles arcos que no se encuentran en el grafo, es decir los arcos $(i,j) : (i,j) \notin \mathcal{A}$ con costo $c_{ij} = M$, siendo M un número lo suficientemente grande para que la variable asociada al arco x_{ij} se mantenga con valor 0, entonces se puede escribir el Problema 1.7 como

$$\begin{aligned} \text{mín } z = & \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}\, x_{ij} \\ \sum_{k=1}^{m} x_{ik} - & \sum_{k=1}^{m} x_{ki} = b_i, \quad i = 1, \ldots, m \\ & 0 \leq x_{ij} \leq u_{ij}, \quad i,j = 1, \ldots, m \end{aligned} \tag{1.8}$$

junto con la condición $\sum_{i=1}^{m} b_i = 0$.

Ejemplo 1.7.1. Sea la red descrita en la Figura 1.4, entonces el problema de costo mínimo asociado, con formulación (1.7), viene dada de la siguiente forma:

$$\begin{aligned} \text{mín } z = 10\, x_{13} + 6\, x_{14} + 7\, x_{21} + 9\, x_{32} + 20\, x_{34} + 8\, x_{45} + 15\, x_{53} & \\ x_{13} + x_{14} - x_{21} = -17 & \\ x_{21} - x_{32} = 9 & \\ x_{32} + x_{34} - x_{13} - x_{53} = 0 & \\ x_{45} - x_{14} - x_{34} = -11 & \\ x_{53} - x_{45} = 19 & \\ x_{ij} \geq 0 & \end{aligned} \tag{1.9}$$

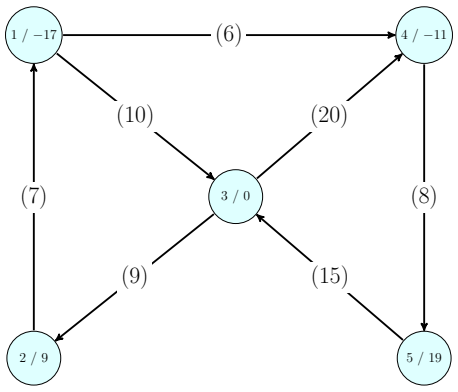


Figura 1.4: Ejemplo de un problema de redes: cada nudo tiene asociado un índice i y un valor numérico b_i, y cada arco un costo.

Se resuelve con el Apéndice A. Se obtiene como solución

x_{ij}	1	2	3	4	5
1					
2	17				
3		8		11	
4					
5			19		

y la función objetivo es $z = 696$. En forma de grafo la solución queda expresada en la Figura 1.5.

1.7.1. Caso particular: problema de la ruta mínima

Se podría pensar en una red donde todos los nudos son de transbordo ($b_i = 0$), excepto dos de ellos, uno con valor $+1$ y otro con -1. En otras palabras, puede decrise que el problema consiste en transportar la unidad desde el nudo que tiene valor $+1$ por los nudos de transbordo hasta el nudo con valor -1. Ahora se formalizará este concepto.

Supóngase $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ con $\mathcal{N} = \{1, \ldots, m\}$. Sea $b_i,\ i = 1, \ldots, m$, el valor asociado al nudo i, se fija para cada $i,\ i = 1, \ldots, m$,

$$b_i = \begin{cases} 1, & \text{si } i = 1 \\ 0, & \text{si } i = 2, 3, \ldots, m-1 \\ -1, & \text{si } i = m. \end{cases}$$

A este caso particular de los problemas de redes se le denomina *problema de*

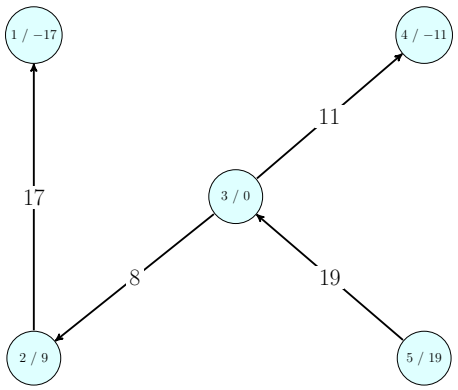


Figura 1.5: Solución gráfica del Problema 1.9. Están representados los arcos de la Figura 1.4 por los cuales atraviesa flujo. Nótese que el grafo asociado a la solución es un árbol extendido.

la ruta mínima y se formula de la siguiente manera:

$$\begin{aligned} \text{mín } z = & \sum_{\substack{i,j=1 \\ (i,j)\in\mathcal{A}}}^{m} c_{ij}\, x_{ij} \\ & \sum_{\substack{k=1 \\ (1,k)\in\mathcal{A}}}^{m} x_{1k} = 1 \\ \sum_{\substack{k=1 \\ (i,k)\in\mathcal{A}}}^{m} x_{ik} - & \sum_{\substack{k=1 \\ (k,i)\in\mathcal{A}}}^{m} x_{ki} = 0, \quad i = 2, \ldots, m-1 \\ & - \sum_{\substack{k=1 \\ (k,m)\in\mathcal{A}}}^{m} x_{km} = -1 \\ & x_{ij} \geq 0, \quad (i,j) \in \mathcal{A} \end{aligned} \tag{1.10}$$

Ejemplo 1.7.2. Ver [12], pág. 17. Un aficionado al fútbol que vive en San Francisco desea ver un partido de fútbol en Dallas, Texas. Este aficionado quiere salir el mismo día del partido de San Francisco y llegar a Dallas no más tarde de las 7:00 PM. Desgraciadamente, el aficionado tiene un presupuesto ajustado y tiene que coger los vuelos más baratos disponibles con Gamma Airlines, incluso si el recorrido o las esperas son largas. Los vuelos disponibles son los expuestos en la Figura 1.6. Para acceder a un vuelo que se encuentre dentro de otro trayecto (entre los 4 disponibles), es necesario estar 1 hora antes de la salida del avión que se desea coger.

Para modelizar el problema se comienza dividiendo el espacio temporal en el que sucede el problema y asignando una etiqueta a cada parte:

Trayecto	Origen	Destino	Hora salida	Hora llegada	Coste
1	San Francisco	Chicago	8:00 AM	1:00 PM	100
	Chicago	Atlanta	2:00 PM	3:00 PM	100
	Atlanta	Dallas	3:40 PM	6:00 PM	250
2	San Francisco	Atlanta	11:00 AM	4:00 PM	250
	Atlanta	Chicago	4:00 PM	5:00 PM	150
	Chicago	Dallas	5:00 PM	7:00 PM	100
3	Atlanta	Miami	4:00 PM	5:00 PM	100
	Miami	Dallas	5:00 PM	7:00 PM	100
4	San Francisco	New York	8:00 AM	2:00 PM	240
	New York	Atlanta	2:00 PM	4:00 PM	50
	Atlanta	Dallas	4:00 PM	6:00 PM	210

Figura 1.6: Tabla de vuelos de Gamma Airlines

Hora	8:00	–	13:00	–	14:00	–	15:40	–	16:00	–	17:00	–	19:00
Intervalo		1		2		3		4		5		6	

Con ello, se definen los nudos del problema de la siguiente manera: se denota la ciudad seguido del intervalo en el que se está. Por ejemplo S1 representa San Francisco en el intervalo 1, es decir, San Francisco entre las 8:00 y las 13:00. Del mismo modo, un arco que sale de C2 y llega a A3 representa salir de Chicago en el intervalo 2 y llegar a Atlanta en el intervalo 3. Con estas nociones el problema de Gamma Airlines se puede modelizar como un problema de ruta mínima descrito en la Figura 1.7.

Escribiendo el problema en forma de tabla se obtiene

c_{ij}	S1	C2	N2	A3	A4a	A4b	A4c	C5	M5	D6	b_i
S1		100	240		250						1
C2				100							0
N2							50				0
A3					0	0	0			250	0
A4a								150			0
A4b									100		0
A4c										210	0
C5										100	0
M5										100	0
D6											-1

Se pasan los datos por el código del Apéndice A y se obtiene como resultado

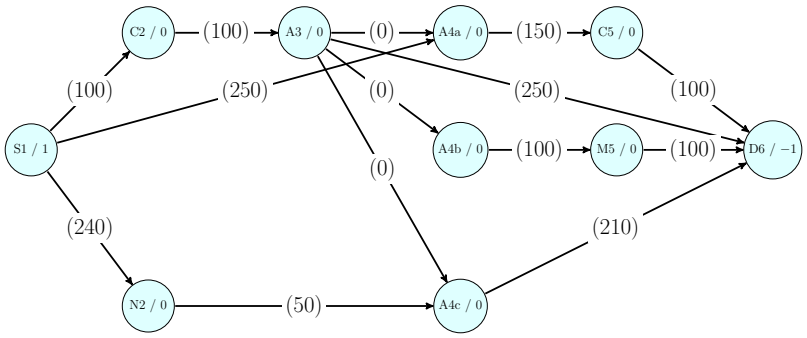


Figura 1.7: Red que describe el problema Gamma Airlines. Están representados únicamente los vuelos permitidos

x_{ij}	S1	C2	N2	A3	A4a	A4b	A4c	C5	M5	D6
S1		1	0		0					
C2				1						
N2							0			
A3					0	1	0			0
A4a								0		
A4b									1	
A4c										0
C5										0
M5										1
D6										

con función objetivo z=400.

Interpretando los datos la solución es S1→C2→A3→A4b→M5→D6. Es decir, el recorrido más barato es el siguiente. Comenzar en el trayecto de vuelos 1 desde San Francisco hasta Atlanta (pasando por Chicago) con llegada a las 3:00 PM. Una vez en Atlanta se cambia del trayecto de vuelos 1 al trayecto de vuelos 3, permitido por haber una hora de diferencia entre la llegada a Atlanta en el trayecto 1 (3:00 PM) y la salida de Atlanta en el trayecto 3 (4:00 PM). Por tanto, se coge el vuelo desde Atlanta hasta Dallas (pasando por Miami) con llegada a las 7:00 PM. El viaje completo tiene un coste total de 400.

1.8. Problemas de redes enteros

Los problemas de redes que incluyen variables de decisión enteras se dice que son *problemas de redes enteros*. Cuando la totalidad de las variables de decisión se requieren enteras se dice que es un problema *entero puro* y en el caso de que no todas las variables se precisen enteras se dice que es un problema *entero mixto*. Cuando el problema es entero y las variables enteras estan acotadas entre 0 y 1, se dice que es un problema *binario* entero, como es el caso de un problema que se verá en el Capítulo 3.

El texto está centrado en las resoluciones mediante algoritmos lineales, por no ser necesarios algoritmos enteros para la búsqueda de soluciones enteras bajo las hipótesis mencionadas anteriormente. Si es necesario un algoritmo entero, se utilizarán algoritmos de programación entera como el algoritmo Branch and Bound.

Capítulo 2

El problema del transporte

2.1. Desarrollo histórico

Los problemas de transporte constituyen uno de los problemas más antiguos estudiados en el campo de la investigación de operaciones. El problema fue formalizado por el matemático Monge en 1781. El matemático Kantorovich realizó importantes avances en este campo durante la Segunda Guerra Mundial. La formulación lineal del problema, conocida como problema de transporte clásico, junto con una solución constructiva fue descrita por Frank L. Hitchcock en el año 1941, ver [10]. Por esa razón también se le conoce en la actualidad como problema de transporte de Hitchcock. De manera independiente, Koopmans también desarrolló avances en los problemas de transporte. En 1951, Dantzig describe un método para la resolución del problema con una algoritmo que es una adaptación del método simplex, ver [5].

El problema del transporte clásico derivó en problemas más complejos. En 1956, A. Orden describe un modelo generalizado del problema de transporte en el cual está permitido el transbordo de los bienes por puntos intermedios conocido como problema de transbordo. Se puede encontrar una colección de variantes de problemas de transporte en Díaz-Parra (2014), ver [6]. Hoy en día se sigue aumentando la complejidad de los problemas, tanto por adición de restricciones como por aumento de tamaño. Por ello, existen librerías de problemas actuales en las que pueden encontrarse, entre otros, problemas de transporte resueltos o pendientes de resolver, como la librería MIPLIB 2010, ver [13].

2.2. El problema del transporte clásico

Se supone que m orígenes (puntos de suministro) $\mathcal{O} = \{O_i : i = 1, \ldots, m\}$ tienen que surtir a n destinos (puntos de demanda) $\mathcal{D} = \{D_j : j = 1, \ldots, n\}$ con un cierto producto. La capacidad de oferta del origen i es $s_i,\ i = 1, \ldots, m$

y la demanda en el centro de consumo j es $d_j,\ j = 1, \ldots, n$. Se supone que c_{ij} es el costo de enviar una unidad de producto del origen i al destino j. El problema consiste en determinar cuantas unidades de producto deben enviarse del origen i al destino j, tal que

- se minimicen los costos totales de distribución,
- no se exceda la capacidad de oferta s_i de cada origen $O_i \in \mathcal{O}$ y
- se satisfaga la demanda d_j de cada destino $D_j \in \mathcal{D}$.

Sean x_{ij} las variables de decisión que determinan cuantas unidades de producto se envían del origen i al destino j, la formulación del problema es

$$\begin{aligned} \text{mín } z &= \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{j=1}^{n} x_{ij} &\leq s_i, \quad i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} &= d_j, \quad j = 1, \ldots, n \\ x_{ij} &\geq 0, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \end{aligned} \tag{2.1}$$

junto con las siguientes hipótesis:

$$\sum_{i=1}^{m} s_i \geq \sum_{j=1}^{n} d_j, \quad s_i \geq 0,\ d_j \geq 0,\ c_{ij} \geq 0, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.$$

con las que se garantiza la factibilidad. El problema (2.1) se denomina *problema de transporte*.

La información que caracteriza un problema de transporte se suele dar en una tabla de la forma

c_{ij}	D_1	D_2	$\ldots$	D_j	$\ldots$	D_n	oferta
O_1	c_{11}	c_{12}	$\ldots$	c_{1j}	$\ldots$	c_{1n}	s_1
O_2	c_{21}	c_{22}	$\ldots$	c_{2j}	$\ldots$	c_{2n}	s_2
$\vdots$	$\vdots$						$\vdots$
O_i	c_{i1}	c_{i2}	$\ldots$	c_{ij}	$\ldots$	c_{in}	s_i
$\vdots$	$\vdots$						$\vdots$
O_m	c_{m1}	c_{m2}	$\ldots$	c_{mj}	$\ldots$	c_{mn}	s_m
demanda	d_1	d_2	$\ldots$	d_j	$\ldots$	d_n	$\sum d_j \leq \sum s_i$

Definición 2.2.1. Se dice que un problema de transporte es *balanceado* si se cumple que la oferta total es igual a la demanda total, es decir:

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$$

Sin pérdida de generalidad, se supondrá que los problemas de transporte son balanceados. En el caso de que el problema de transporte no sea balanceado, es decir, si la oferta total es mayor que la demanda total, se balancea creando un destino ficticio D_{n+1} con

$$d_{n+1} = \sum_{i=1}^{m} s_i - \sum_{j=1}^{n} d_j \quad \text{y} \quad c_{i,n+1} = 0, \ \forall i : i = 1, \ldots, m.$$

La tabla del nuevo problema será la siguiente:

c_{ij}	D_1	D_2	$\ldots$	D_j	$\ldots$	D_n	D_{n+1}	oferta
O_1	c_{11}	c_{12}	$\ldots$	c_{1j}	$\ldots$	c_{1n}	0	s_1
O_2	c_{21}	c_{22}	$\ldots$	c_{2j}	$\ldots$	c_{2n}	0	s_2
$\vdots$	$\vdots$						$\vdots$	$\vdots$
O_i	c_{i1}	c_{i2}	$\ldots$	c_{ij}	$\ldots$	c_{in}	0	s_i
$\vdots$	$\vdots$						$\vdots$	$\vdots$
O_m	c_{m1}	c_{m2}	$\ldots$	c_{mj}	$\ldots$	c_{mn}	0	s_m
demanda	d_1	d_2	$\ldots$	d_j	$\ldots$	d_n	d_{n+1}	$\sum d_j = \sum s_i$

En definitiva, cuando se considera el problema (2.1), junto con la hipótesis de que sea balanceado, el problema se convierte en un problema de programación lineal de la forma

$$\begin{aligned} \text{mín } z &= \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij} \\ \sum_{j=1}^{n} x_{ij} &= s_i, \quad i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} &= d_j, \quad j = 1, \ldots, n \\ x_{ij} \geq 0, \quad i &= 1, \ldots, m, \quad j = 1, \ldots, n. \end{aligned} \tag{2.2}$$

Esta última formulación lineal (2.2) es denominada *problema de transporte clásico.*

Ahora multiplcando por -1 ambos miembros de la ecuación del problema del transporte referente a los d_j, se puede reescribir el problema del transporte de forma

$$\begin{aligned} \text{mín } z &= cx \\ Ax &= b \\ x &\geq 0 \end{aligned}$$

donde cada elemento esta definido de la siguiente forma:

$$x = [x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, x_{mn}]^t$$

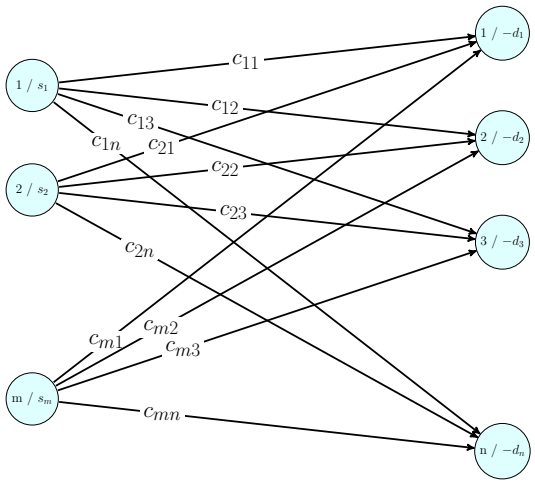


Figura 2.1: El problema del transporte representado en forma de red como problema de flujo de redes a costo mínimo.

$$c = [c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots, c_{2n}, \ldots, c_{mn}]$$

$$b = [s_1, s_2, \ldots, s_m, -d_1, -d_2, \ldots, -d_n]^t$$

$$A = \begin{bmatrix} 1_n & 0 & \ldots & 0 \\ 0 & 1_n & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1_n \\ -I_n & -I_n & \ldots & -I_n \end{bmatrix}$$

siendo 1_n una fila $1 \times n$ de unos.

A es la matriz de incidencia nudo-arco del grafo que describe el problema de transporte, es decir, un grafo con nudos $\mathcal{N} = \{O_1, \ldots O_m, D_1, \ldots, D_n\}$ y arcos dirigidos $\mathcal{A} = \{(O_i, D_j) : i = 1, \ldots m,\ j = 1, \ldots, n\}$. El arco (O_i, D_j) se puede denotar simplemente por (i, j), ya que no da lugar a confusión. Se tiene entonces un problema de redes descrito en el Capítulo 1 como se puede ver en la Figura 2.1, por lo que se pueden adaptar todos los resultados vistos.

- A tiene rango $m+n-1$. Para aplicar el método simplex se le añade al problema la variable artificial (arco raíz) x_a, con nudo raíz uno de los nudos, en general, en el n-ésimo destino, es decir, un arco con columna asociada e^{m+n}. Se denotará $A^0 = [A, e^{m+n}]$.

- Las bases B de A^0 se corresponden con árboles extendidos con raíz en D_n, $\mathcal{T}_B$. Además, B es triangular inferior (o puede conseguirse mediante permutaciones de filas y columnas).

- A^0 es una matriz totalmente unimodular. Luego, cualquier base B de A^0, y su inversa B^{-1} tienen determinante ± 1 y están compuestas por elementos 0 y ± 1.

- Asumiendo que b es entero (se supondrá si no se dice lo contrario), la solución óptima es entera. Es decir, no se requiere tratar el problema como un problema de programación entera.

- Los sistemas de ecuaciones $Bx_B = b$, $\pi B = c_B$ y $By_{ij} = a_{ij}$ para hallar los valores de x_B, π y y_{ij}, respectivamente, pueden ser resueltos de manera eficiente por ser B triangular.

2.3. Búsqueda de una solución inicial

Ya se ha visto que para los problemas de redes, en particular para los problemas de transporte, encontrar una base equivale a encontrar un árbol extendido en el grafo que determina el problema (no se hace alusión a la variable artificial). Para el problema del transporte que se ha descrito, hay que hallar una solución inicial básica factible, que estará compuesta por la variable artificial y $m + n - 1$ variables.

No hay un único método para hallar una solución inicial. Dos métodos conocidos son el método de la esquina noroeste y el método de Vogel. Es más recomendable este último por proporcionar, en general, una solución inicial que es más cercana a la solución óptima. El método de la esquina noroeste es un método intuitivo que puede encontrarse en cualquier texto sobre problemas de transporte. Ambos métodos pueden encontrarse en Bazaraa, ver [4]. Además de los anteriores, se pueden hallar soluciones iniciales con métodos para redes o con otros métodos como el método del costo mínimo, que ofrece una solución mejor que el método de la esquina noroeste pero puede ser necesario un alto número de iteraciones, ver [1].

2.4. El algoritmo del transporte

El algoritmo del transporte es el algoritmo que se obtiene al adaptar el algoritmo para redes a las características del problema del transporte. No es necesario el uso de algoritmos específicos de programación entera, ya que la solución lineal será entera (siempre bajo la hipótesis de que las ofertas, demandas y cotas superiores tengan valores enteros).

Se describirá el algoritmo para el *problema de transporte capacitado*. Este problema es una generalización del problema de transporte clásico y su

formulación es:

$$\begin{aligned} \min z &= \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{j=1}^{n} x_{ij} &= s_i, \quad i = 1,\dots,m \\ \sum_{i=1}^{m} x_{ij} &= d_j, \quad j = 1,\dots,n \\ 0 \le x_{ij} &\le u_{ij}, \quad i = 1,\dots,m,\ j = 1,\dots,n. \end{aligned} \tag{2.3}$$

Es decir, es como un problema de transporte clásico pero con la adición de cotas superiores para las variables de decisión. Tomándolas todas como infinito, el algoritmo es válido para los problemas de transporte no capacitados.

Algoritmo

- PASO 0.- Se parte de una solución inicial básica factible x (conocida o hallada por alguno de los métodos expuestos). Calcular las variables duales $\pi = [u_1,\dots,u_m,v_1,\dots,v_n]$ como solución de $\pi B = c_B$. El sistema queda reducido a determinar la solución de

$$\begin{cases} u_i - v_j = c_{ij}, & (O_i, D_j) \in \mathcal{T}_B \\ v_n = 0. \end{cases}$$

  Como se tiene $v_n = 0$, por la triangularidad de B, se puede sustituir el valor de v_n en otra ecuación y determinar el valor de una variable u, lo cual determinará en otra ecuación el valor de otra variable v. Se procede de este modo hasta determinar π.

- PASO 1.- Las variables no básicas candidatas a entrar en la base son

$$\begin{aligned} \Psi_1 &= \{(O_i, D_j) : x_{ij} = 0 \text{ y } u_i - v_j - c_{ij} > 0\} \\ \Psi_2 &= \{(O_i, D_j) : x_{ij} = u_{ij} \text{ y } u_i - v_j - c_{ij} < 0\} \end{aligned}$$

  Si $\Psi_1 \cup \Psi_2 = \emptyset$, entonces $x = [x_B; x_N]$ es la solución óptima. En caso contrario se escoge $x_{kl} \in \Psi_1 \cup \Psi_2$ (el de mayor $|u_i - v_j - c_{ij}|$) y se fija

$$\delta = \begin{cases} 1, & \text{si } (O_k, D_l) \in \Psi_1 \\ -1, & \text{si } (O_k, D_l) \in \Psi_2. \end{cases}$$

- PASO 2.- Sea x_{kl} la variable seleccionada para entrar en la base, sea P el único camino en $\mathcal{T}_B$ de O_k a D_l, entonces

$$\Delta_1 = \min_{O_{(O_i,D_j)}(P)=\delta}\{x_{ij}, \infty\},$$
$$\Delta_2 = \min_{O_{(O_i,D_j)}(P)=-\delta}\{u_{ij} - x_{ij}, \infty\},$$
$$\Delta = \min\{\Delta_1, \Delta_2, u_{kl}\}.$$

  Saldrá de la base la variable con la que se alcanza el mínimo Δ.

- PASO 3.- Actualización de variables de decisión.

$$\hat{x}_{kl} = x_{kl} + \delta\Delta$$
$$\hat{x}_{ij} = x_{ij} - \delta\Delta O_{(O_i,D_j)}(P), \quad \forall x_{ij} : (O_i, D_j) \in P.$$

  Si $\Delta = u_{kl}$, volver al paso 1.

- PASO 4.- Actualización de la base y de las variables duales. Sean

$$\Psi_3 = \{(O_i, D_j) : x_{ij} = 0 \text{ y } O_{(O_i,D_j)}(P) = \delta\},$$
$$\Psi_4 = \{(O_i, D_j) : x_{ij} = u_{ij} \text{ y } O_{(O_i,D_j)}(P) = -\delta\}.$$

  Se selecciona un $(O_s, D_t) \in \Psi_3 \cup \Psi_4$ y el nuevo arbol básico es $\hat{\mathcal{T}_\mathcal{B}} = (\mathcal{T}_B \cup \{(O_k, D_l)\}) - \{(O_s, D_t)\}$. Se actualizan las variables duales y se vuelve al paso 1.

Ejemplo 2.4.1. Ver [11], pág 324. Siete áreas de irrigación A, B, C, D, E, F, G, requieren de fertilizantes. Supóngase que hay cuatro tipos de fertilizantes X, Y, W, Z. La oferta y el costo de estos fertilizantes (incluido en el costo la tarifa de transporte) es:

Fertilizante	Oferta mensual	Precio por tonelada
X	7000 ton	\$1000/ton
Y	4000 ton	\$4000/ton
W	6000 ton	\$2000/ton
Z	5000 ton	\$5000/ton

Las siete áreas de irrigación requieren (indistintamente) de lo siguiente:

Area	Tipos de fertilizante factibles	Demanda total de fertilizante
A	X, Z	2000 ton
B	W, X, Y	3000 ton
C	Y, W	1000 ton
D	Z, X	2000 ton
E	X, Y	3000 ton
F	Y, X, Z	2000 ton
G	W, Z	1000 ton

Si se quiere programar la entrega de fertilizantes a zonas de irrigación a un costo mínimo, se puede formular el problema como un problema de transporte. Los orígenes se corresponden con los tipos de fertilizante y los destinos con las áreas que los precisan. Además, que cierto fertilizante no sea factible para cierta zona significa que no hay envío posible de ese orígen a ese destino, luego en la tabla de transporte dicho costo será M. Con todo ello, la tabla del problema de transporte que se quiere resolver es

costos	A	B	C	D	E	F	G	oferta
X	1000	1000	M	1000	1000	1000	M	7000
Y	M	4000	4000	M	4000	4000	M	4000
W	M	2000	2000	M	M	M	2000	6000
Z	5000	M	M	5000	M	5000	5000	5000
demanda	2000	3000	1000	2000	3000	2000	1000	

Como la oferta total es mayor que la demanda total, se añade un destino ficticio con costos igual a cero y con una demanda de $22000 - 14000 = 8000$ de modo que la tabla de transporte balanceada resultante es

costos	A	B	C	D	E	F	G	fic	oferta
X	1000	1000	M	1000	1000	1000	M	0	7000
Y	M	4000	4000	M	4000	4000	M	0	4000
W	M	2000	2000	M	M	M	2000	0	6000
Z	5000	M	M	5000	M	5000	5000	0	5000
demanda	2000	3000	1000	2000	3000	2000	1000	8000	

Al resolver el problema se obtiene como resultado

x_{ij}	A	B	C	D	E	F	G	fic
X	2000	0	0	2000	1000	2000	0	0
Y	0	0	0	0	2000	0	0	2000
W	0	3000	1000	0	0	0	1000	1000
Z	0	0	0	0	0	0	0	5000

y función objetivo $z = \$25000000$.

Capítulo 3

Extensiones del problema del transporte

3.1. Problema de transbordo

Se considerará a continuación un modelo de problema de transbordo que se adapta a la realidad en cuanto a la estructura de cadenas de suministro se refiere, que clasifica los nudos según sus propiedades y que no permite envíos entre cualquier par de nudos.

En el problema de transbordo (transshipment problem) los nudos del grafo que describe el problema se pueden clasificar en tres conjuntos disjuntos: orígenes puros, destinos puros y transbordos.

- *Orígenes puros.* Solo pueden enviar bienes, a destinos puros o a transbordos. Equivalentemente, de los orígenes puros solo salen arcos, los cuales tienen como destino un transbordo o un destino puro. Cada origen puro tiene asociado una oferta de bienes. Sea $\mathcal{O}$ el conjunto de orígenes puros, cada $i \in \mathcal{O}$ tiene asociada la oferta a_i.

- *Destinos puros.* Solo pueden recibir bienes, de orígenes puros o de transbordos. Equivalentemente, a los destinos puros solo llegan arcos, los cuales tienen su origen en un origen puro o en un transbordo. Cada destino puro tiene asociada una demanda. Sea $\mathcal{D}$ el conjunto de destinos puros, cada $j \in \mathcal{D}$ tiene asociada una demanda b_j.

- *Transbordos.* Pueden enviar y/o recibir bienes. Tienen permitida la circulación de bienes entre ellos, y como se ha dicho antes, pueden recibir bienes de orígenes puros y enviar a destinos puros. Sea $\mathcal{T}$ el conjunto de transbordos, cada $k \in \mathcal{T}$ tiene asociada una producción p'_k y una necesidad q'_k. Entonces la oferta p_k y la demanda q_k del transbordo k vienen dadas por las fórmulas $p_k = p'_k - \text{mín}\{p'_k, q'_k\}$ y $q_k = q'_k - \text{mín}\{p'_k, q'_k\}$. Nótese que al menos uno de los dos, p_k o q_k, tendrá valor 0.

Se van a definir ahora las variables que intervienen en el modelo que se propondrá a continuación para problemas de transbordo:

- c_{ij}: costo de envío por unidad del origen puro i al destino puro j.
- d_{ik}: costo de envío por unidad del origen puro i al transbordo k.
- e_{kl}: costo de envío por unidad del transbordo k al transbordo l, con $k \neq l$.
- f_{kj}: costo de envío por unidad del transbordo k al destino puro j.
- $a_i,\ i \in \mathcal{O}$: oferta del origen puro i.
- $p_k,\ k \in \mathcal{T}$: oferta del transbordo k.
- $q_k,\ k \in \mathcal{T}$: demanda del transbordo k.
- $b_j,\ j \in \mathcal{D}$: demanda del destino puro j.
- $x_{ij},\ i \neq j$: variable de decisión que indica la cantidad de bienes enviados del nudo i al nudo j.
- $x_{kk},\ k \in \mathcal{T}$: variable de decisión que indica la cantidad de los denominados bienes de transbordo en k, es decir, los bienes que recibe k que no necesita (los bienes que recibe para enviar).

Con estas variables, se puede definir un modelo lineal que garantice que satisfagan todas las demandas de los nudos sin exceder las ofertas y de modo que se minimice el costo total de envíos. El modelo propuesto es el siguiente:

$$\begin{aligned}
\text{mín} \sum_{i\in\mathcal{O}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} + \sum_{i\in\mathcal{O}}\sum_{k\in\mathcal{T}} d_{ik}x_{ik} + \sum_{k\in\mathcal{T}}\sum_{l\in\mathcal{T}} e_{kl}x_{kl} + \sum_{k\in\mathcal{T}}\sum_{j\in\mathcal{D}} f_{kj}x_{kj} & \\
\sum_{k\in\mathcal{T}} x_{ik} + \sum_{j\in\mathcal{D}} x_{ij} \leq a_i, \quad \forall i \in \mathcal{O} & \\
\sum_{l\in\mathcal{T}} x_{kl} + \sum_{j\in\mathcal{D}} x_{kj} - x_{kk} \leq p_k, \quad \forall k \in \mathcal{T} & \\
\sum_{i\in\mathcal{O}} x_{ik} + \sum_{l\in\mathcal{T}} x_{lk} - x_{kk} = q_k, \quad \forall k \in \mathcal{T} & \\
\sum_{i\in\mathcal{O}} x_{ij} + \sum_{k\in\mathcal{T}} x_{kj} = b_j, \quad \forall j \in \mathcal{D} & \\
x_{ij} \geq 0, \quad \forall\ i \in \mathcal{O}\cup\mathcal{T},\ j \in \mathcal{T}\cup\mathcal{D} &
\end{aligned} \tag{3.1}$$

satisfaciendo las condiciones

$$c_{ij},\ d_{ij},\ e_{ij},\ f_{ij} > 0,\ \text{si } i \neq j \quad \text{y} \quad \sum_i a_i + \sum_k p_k \geq \sum_j b_j + \sum_k q_k.$$

El problema (3.1) junto con las condiciones adicionales, se denominará *problema de transbordo.*

Como en los problemas de transporte, se supondrán los problemas de transbordo balanceados, es decir, problemas de transbordo tal que

$$\sum_{i\in\mathcal{O}} a_i + \sum_{k\in\mathcal{T}} p_k = \sum_{j\in\mathcal{D}} b_j + \sum_{k\in\mathcal{T}} q_k.$$

En caso contrario, si $\sum\limits_i a_i + \sum\limits_k p_k > \sum\limits_j b_j + \sum\limits_k q_k$, el problema se balancea añadiendo un destino ficticio f a $\mathcal{D}$ con demanda $b_f = \sum\limits_i a_i + \sum\limits_k p_k - (\sum\limits_j b_j + \sum\limits_k q_k)$ y costos $c_{if} = 0,\ \forall i \in \mathcal{O}\cup\mathcal{T}$. De este modo, el Problema (3.1) se escribe

$$\begin{aligned}
\text{mín} \sum_{i\in\mathcal{O}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} + \sum_{i\in\mathcal{O}}\sum_{k\in\mathcal{T}} d_{ik}x_{ik} + \sum_{k\in\mathcal{T}}\sum_{l\in\mathcal{T}} e_{kl}x_{kl} + \sum_{k\in\mathcal{T}}\sum_{j\in\mathcal{D}} f_{kj}x_{kj} & \\
\sum_{k\in\mathcal{T}} x_{ik} + \sum_{j\in\mathcal{D}} x_{ij} = a_i, \quad \forall i \in \mathcal{O} & \\
\sum_{l\in\mathcal{T}} x_{kl} + \sum_{j\in\mathcal{D}} x_{kj} - x_{kk} = p_k, \quad \forall k \in \mathcal{T} & \\
\sum_{i\in\mathcal{O}} x_{ik} + \sum_{l\in\mathcal{T}} x_{lk} - x_{kk} = q_k, \quad \forall k \in \mathcal{T} & \\
\sum_{i\in\mathcal{O}} x_{ij} + \sum_{k\in\mathcal{T}} x_{kj} = b_j, \quad \forall j \in \mathcal{D} & \\
x_{ij} \geq 0, \quad \forall\ i \in \mathcal{O}\cup\mathcal{T},\ j \in \mathcal{T}\cup\mathcal{D} &
\end{aligned} \tag{3.2}$$

3.1.1. Resolución del problema

Hay varias formas de abordar un problema de transbordo, entre ellas la resolución del problema diréctamente tratándolo como un problema de redes o la conversión del problema a un problema de transporte clásico visto anteriormente para su resolución con las técnicas de dicho modelo.

Transbordo como problema de redes

Esta sería la forma directa de resolverlo, viéndolo como un problema de flujo de redes de costo mínimo. El grafo asociado al problema es el que viene dado por todos los nudos, junto con los arcos dirigidos permitidos. El valor numérico asociado a cada nudo es a_i si $i \in \mathcal{O}$, $-b_j$ si $j \in \mathcal{D}$ y para los transbordos $k \in \mathcal{T}$, si $p_k = 0$ el valor asociado es $-q_k$ y si $q_k = 0$ el valor asociado es p_k.

Tratando el problema de este modo se pueden usar las técnicas y propiedades vistas para redes y resolver el problema mediante el método simplex para redes.

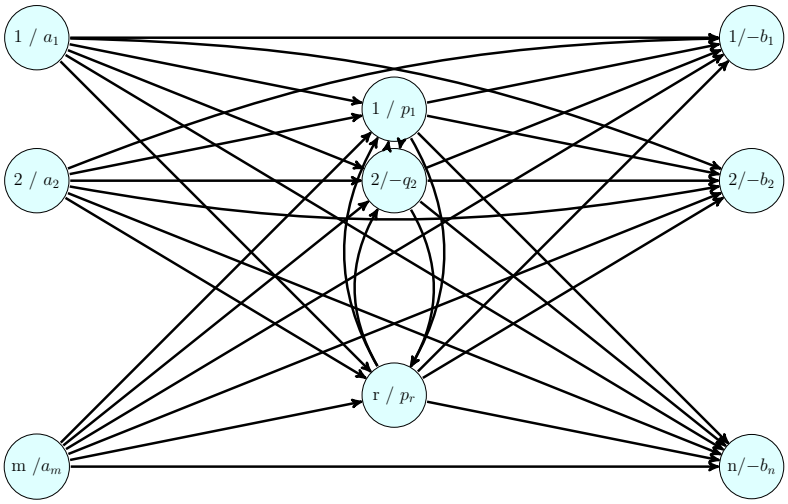


Figura 3.1: Red asociada a un problema de transbordo. De izquierda a derecha: m orígenes puros, r transbordos y n destinos puros.

Reducción del problema de transbordo a un problema de transporte

A continuación, se verá como puede reducirse un problema de transbordo a un problema de transporte como los que fueron descritos en el Capítulo 2. De este modo, se podrá tratar como un problema de transporte, haciendo posible el uso de los resultados que ya se han explicado para este tipo de problemas. También es posible el proceso inverso, es decir, la formulación de un problema de transporte como un problema de transbordo, produciéndose una equivalencia entre los problemas de transporte y los problemas de transbordo (incluso en los problemas en los que no hay distinción entre nudos, ver [5]).

Si se tiene que los nudos del problema de transbordo a resolver son m orígenes puros $O_i,\ i = 1, \ldots, m$, n destinos puros $D_j,\ j = 1, \ldots, n$ y r transbordos $T_k,\ k = 1, \ldots, r$. Sea

$$L = \sum_{i=1}^{m} a_i + \sum_{k=1}^{r} p_k = \sum_{j=1}^{n} b_i + \sum_{k=1}^{r} q_k, \tag{3.3}$$

entonces en cualquier solución óptima las variables x_{kk} están acotadas. En efecto,

$$x_{kk} \leq L, \quad \forall k \in \mathcal{T}. \tag{3.4}$$

Si para algún $k \in \mathcal{T}$ fuera $x_{kk} > L$, entonces hay bienes que pasan por el transbordo k más de una vez y por lo tanto no puede ser solución óptima.

Definiendo una nueva variable x'_{kk} como

$$x'_{kk} = L - x_{kk} \tag{3.5}$$

se puede reducir el problema del transbordo al problema de transporte que determina la tabla

costos	D_1	D_2	$\dots$	D_n	T_1	T_2	$\dots$	T_r	oferta
O_1	c_{11}	c_{12}	$\dots$	c_{1n}	d_{11}	d_{12}	$\dots$	d_{1r}	a_1
O_2	c_{21}	c_{22}	$\dots$	c_{2n}	d_{21}	d_{22}	$\dots$	d_{2r}	a_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
O_m	c_{m1}	c_{m2}	$\dots$	c_{mn}	d_{m1}	d_{m2}	$\dots$	d_{mr}	a_m
T_1	f_{11}	f_{12}	$\dots$	f_{1n}	0	e_{12}	$\dots$	e_{1r}	$p_1 + L$
T_2	f_{21}	f_{22}	$\dots$	f_{2n}	e_{21}	0	$\dots$	e_{2r}	$p_2 + L$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_r	f_{r1}	f_{r2}	$\dots$	f_{rn}	e_{r1}	e_{r2}	$\dots$	0	$p_r + L$
demanda	b_1	b_2	$\dots$	b_n	$q_1 + L$	$q_2 + L$	$\dots$	$q_r + L$	

siendo x'_{kk} la variable de decisión que determina el flujo que va del origen T_k al destino T_k.

3.1.2. Red de suministro eléctrica

Supóngase una empresa eléctrica que suministra energía a una gran zona, la cual quiere hacer la planificación anual de envíos. La empresa posee plantas de producción de energía en zonas aisladas. Además, en las grandes ciudades también posee puntos de generación de energía, que dependiendo de la ciudad puede crear más de lo que necesitan o pueden tener un déficit, en cuyo caso necesitarán energía adicional. Por otra parte, se encuentran los pueblos o pequeñas ciudades, las cuales tienen una necesidad de energía concreta y no poseen ningún modo de generación de energía.

Gracias a las instalaciones de la empresa, la energía puede ser enviada desde las plantas de producción a las grandes ciudades y a los pueblos. Además, las grandes ciudades pueden enviarse energía entre ellas y también a los pueblos, siendo estos últimos los únicos que solo pueden recibir energía de los anteriores, sin posibilidad de enviar a ningún punto.

Por último, hay algunas instalaciones de transporte de energía que tienen una capacidad limitada de transporte.

La empresa eléctrica quiere satisfacer todas las necesidades energéticas de las ciudades y pueblos, de modo que el costo total de los envíos le resulte lo mas económico posible. Para ello se tiene la siguiente información (los datos se ofrecen en millones MWh y la unidad de tiempo es un año).

- La empresa cuenta con 7 plantas: O_1, O_2, O_3, O_4, O_5, O_6 y O_7.
- Hay 5 grandes ciudades, o ciudades en las que la empresa produce energía: T_1, T_2, T_3, T_4 y T_5.
- Hay 6 pueblos, es decir, que únicamente pueden recibir energía: D_1, D_2, D_3, D_4, D_5 y D_6.

Se tiene la siguiente información:

Planta	O_1	O_2	O_3	O_4	O_5	O_6	O_7	Total
Producción (a_i)	33	26	21	37	15	28	31	191

Pueblo	D_1	D_2	D_3	D_4	D_5	D_6	Total
Consumo (b_j)	13	16	27	15	21	20	112

Ciudad	T_1	T_2	T_3	T_4	T_5
Producción	25	30	15	23	17
Consumo	28	12	26	23	32
Diferencia	-3	18	-11	0	-5

De la última tabla se extraen las ofertas y demandas netas, que son

Ciudad	T_1	T_2	T_3	T_4	T_5	Total
p_k	0	18	0	0	0	18
q_k	3	0	11	0	5	19

costos	D_1	D_2	D_3	D_4	D_5	D_6	T_1	T_2	T_3	T_4	T_5
O_1	26	23	–	–	–	11	19	16	–	14	11
O_2	31	–	–	–	27	–	14	12	16	–	–
O_3	–	25	30	–	34	–	11	7	19	16	12
O_4	35	–	–	–	–	–	–	19	–	15	17
O_5	–	–	32	–	–	–	31	14	19	16	12
O_6	–	30	–	–	–	27	15	11	15	13	19
O_7	29	31	–	–	–	–	11	15	13	14	15
T_1	15	–	13	12	9	11	–	5	5	8	3
T_2	14	16	12	11	–	–	9	–	7	4	8
T_3	–	19	11	14	13	–	6	–	–	9	9
T_4	–	–	–	–	–	–	5	3	6	–	3
T_5	15	13	16	–	11	7	6	–	4	5	–

El problema de transbordo no es balanceado. Para balancearlo se añade al problema un destino puro ficticio D_f con demanda $b_f = \sum_{i=1}^{7} a_i + \sum_{k=1}^{5} p_k -$

$\sum_{j=1}^{6} b_j - \sum_{k=1}^{5} q_k = 191 + 18 - 112 - 19 = 78$. También se calcula $L = \sum_{i=1}^{7} a_i + \sum_{k=1}^{5} p_k = 209$. Por lo tanto, este problema de transbordo puede ser convertido al problema de transporte cuya tabla es:

costos	D_1	D_2	D_3	D_4	D_5	D_6	D_f	T_1	T_2	T_3	T_4	T_5	oferta
O_1	26	23	–	–	–	11	0	19	16	–	14	11	33
O_2	31	–	–	–	27	–	0	14	12	16	–	–	26
O_3	–	25	30	–	34	–	0	11	7	19	16	12	21
O_4	35	–	–	–	–	–	0	–	19	–	15	17	37
O_5	–	–	32	–	–	–	0	31	14	19	16	12	15
O_6	–	30	–	–	–	27	0	15	11	15	13	19	28
O_7	29	31	–	–	–	–	0	11	15	13	14	15	31
T_1	15	–	13	12	9	11	0	0	5	5	8	3	209
T_2	14	16	12	11	–	–	0	9	0	7	4	8	227
T_3	–	19	11	14	13	–	0	6	–	0	9	9	209
T_4	–	–	–	–	–	–	0	5	3	6	0	3	209
T_5	15	13	16	–	11	7	0	6	–	4	5	0	209
demanda	13	16	27	15	21	20	78	212	209	220	209	214	

Además, las cotas superiores son:

cot. sup.	D_1	D_2	D_3	D_4	D_5	D_6	D_f	T_1	T_2	T_3	T_4	T_5
O_1		10										5
O_2												
O_3		5							20			
O_4												
O_5			20						15			
O_6												
O_7	10									15		
T_1					15							15
T_2												15
T_3			5								10	
T_4									10			10
T_5	10	15								15		

Pasando esta información por el programa para problemas de transporte

del Apéndice A se obtiene como resultado:

x_{ij}	D_1	D_2	D_3	D_4	D_5	D_6	D_f	T_1	T_2	T_3	T_4	T_5
O_1		10				20						3
O_2							26					
O_3									20			1
O_4							37					
O_5							2					13
O_6							11		17			
O_7							2	18		11		
T_1					15			194				
T_2	13		27	15					172			
T_3										209		
T_4											209	
T_5		6			6							197

y el valor de la función objetivo es $z = 2269$.

Ahora se interpreta la solución obtenida. Los envíos de origenes a destinos, de orígenes a transbordos, de transbordo a transbordo distinto y de transbordo a destino se corresponden con los números que aparecen en la tabla. Los envíos con destino D_f representan elementos que no han salido de sus respectivos orígenes. Los envíos entre dos puntos de transbordo iguales se corresponden con $x'_{T_k,T_k} = L - x_{T_k,T_k}$. Por ejemplo $x'_{T_1,T_1} = 194$ indica que $x_{T_1,T_1} = 209 - 194 = 15$ lo cual quiere decir que el transbordo T_1 ha recibido 15 unidades que no necesitaba, es decir, que ha enviado a otro punto. Observando en la tabla de la solución se comprueba que efectivamente T_1 recibe 18 unidades de O_7, de las cuales solo precisa 3 unidades y las 15 unidades restantes las reenvía a D_5.

3.2. Transporte con localización y coste de producción

Cuando se habla de los problemas de transporte clásicos, no se tiene en cuenta el coste de producción de los bienes en los orígenes o puntos de suministro. Por otro lado, puede ocurrir que exista un límite de orígenes operativos y además mantener un orígen operativo, tal como una fábrica, también influye en el coste de la operación.

Teniendo en cuenta estos factores, se va a proponer un modelo más extenso que el de un problema de transporte clásico. Estas modificaciones, al contrario de lo que ocurría con los problemas de transbordo, van a requerir la resolución del problema con algoritmos de programación entera para hallar una solución entera.

3.2.1. Modelización del problema

Se suponen m orígenes, n destinos y r posibles orígenes operativos. Además, se tiene:

- d_{ij}: costo de envío unitario del origen i al destino j. Se considerará $d_{ij} = M$ (siendo M un número muy grande) si no hay envío posible entre i y j.
- e_i: costo de hacer operativo el origen i
- d'_i: costo de producción unitario en el origen i
- a_i: capadicad máxima de producción del origen i
- b_j: demanda en el destino j
- x_{ij}: unidades a enviar del origen i al destino j
- $y_i = \begin{cases} 1, & \text{si está operativo el origen } i \\ 0, & \text{si no está operativo el origen } i \end{cases}$

Se define $c_{ij} = d_{ij} + d'_i$, que es el costo unitario transporte del origen i al destino j (incluye coste de producción y coste de envío). Entonces, el modelo que describe problema propuesto queda de la siguiente forma:

$$\begin{aligned} \text{mín} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij} + \sum_{i=1}^{m} e_i\, y_i \\ \sum_{j=1}^{n} x_{ij} \le a_i\, y_i, \quad i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, n \\ \sum_{i=1}^{m} y_i \le r \\ x_{ij} \ge 0,\ i = 1, \ldots, m,\ j = 1, \ldots, n \\ y_i \in \{0, 1\},\ i = 1, \ldots, m \end{aligned} \tag{3.6}$$

Para cada $i,\ i = 1, 2, \ldots, m$, se tiene que

$$\sum_{j=1}^{n} x_{ij} \le a_i\, y_i = \begin{cases} \displaystyle\sum_{j=1}^{n} x_{ij} \le a_i, & \text{si } y_i = 1 \\ x_{ij} = 0,\ j = 1, 2 \ldots, n, & \text{si } y_i = 0. \end{cases}$$

Las variables que únicamente pueden tomar valor 0 o 1, como las variables y_i de este modelo, se dicen *variables binarias*. Nótese que fijadas las

variables $y_i,\ i = 1, \ldots, m$ de modo que el problema que quede tenga solución factible, el problema que hay que resolver es un problema de transporte. Con ello, cabe decir que suponiendo las ofertas y las demandas enteras, no es necesaria exigir la integralidad de las variables x_{ij} para obtener una solución entera.

3.2.2. Expansión estratégica de una empresa

Una empresa que fabrica piezas está asentada en la Península Ibérica. Debido a su tecnología puntera, muchos países demandan dicha pieza. Las zonas que requieren sus servicios son:

Zona	Demanda (b_j)	Zona	Demanda (b_j)
O-Península Ibérica	1500	D-Asia	4500
A-Europa Central	6300	E-Canadá	900
B-Reino Unido	500	F-Estados Unidos	7800
C-Rusia	6500	G-Sudamérica	3200

Esa demanda, ha hecho que la empresa se plantee una expansión estratégica, ya que los costos de envío pueden encarecer mucho el proceso de entrega, reduciendo las ganancias de la propia empresa de manera notable. Por ello, la empresa ha observado que podría abrir fábricas en las siguientes zonas:

Disponibles	Establecerse (e_i)	Producción (a_i)	Precio fabricación(d_i'')
P. Ibérica	Establecido	30000	2500
Rusia	5 mill.	10000	2000
Asia	3 mill.	15000	1500
Estados Unidos	7 mill.	12000	2500
Sudamérica	2 mill.	6000	1500
H-África	1 mill.	8000	200

Por la dificultad de gestionar varias fábricas al mismo tiempo, la empresa ha decidido que no abrirá más de tres fábricas nuevas. Según su información, que se supone fiable, el costo de envío por cargamento entre las distintas zonas (depende de distancia, políticas locales,...) es el que se refleja en la siguiente tabla:

Origen\Destino (d_{ij})	O (1)	A (2)	B (3)	C (4)	D (5)	E (6)	F (7)	G (8)
O (1)	15	800	900	1400	1900	-	2300	950
C (2)	1600	1500	-	35	800	1300	-	1900
D (3)	600	700	500	600	20	1300	1500	1300
F (4)	1500	1600	1500	-	1700	800	25	-
G (5)	400	700	800	1500	900	500	-	10
H (6)	800	1000	1200	2200	2000	2400	2300	2100

$c_{ij} = d_{ij} + d'_i$	O	A	B	C	D	E	F	G
O	2515	3300	3400	3900	4400	-	4800	3450
C	3600	3500	-	2035	2800	3300	-	3900
D	2100	2200	2000	2100	1520	2800	3000	2800
F	4000	4100	4000	-	4200	3300	2525	-
G	1900	2200	2300	3000	2400	2000	-	1510
H	1000	1200	1400	2400	2200	2600	2500	2300

El problema que hay que resolver para minimizar el costo total de las operaciones es

$$\begin{aligned} \text{mín} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij} + \sum_{i=1}^{m} e_i\, y_i & \\ \sum_{j=1}^{n} x_{1j} \leq a_1 & \\ \sum_{j=1}^{n} x_{ij} \leq a_i\, y_i, \quad i = 2, \ldots, 6 & \\ \sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, 8 & \\ \sum_{i=2}^{m} y_i \leq 3 & \\ x_{ij} \in \mathbb{Z}^+,\ i = 1, \ldots, 6,\ j = 1, \ldots, 8 & \\ y_i \in \{0, 1\},\ i = 2, \ldots, 6 & \end{aligned} \tag{3.7}$$

que es un problema de los descritos en esta sección, con la particularidad de que la variable y_1 viene fijada a 1.

Resolviendo el problema mediante el programa del Apéndice C se obtiene la solución

x_{ij}	O	A	B	C	D	E	F	G	y_i
O	1500	700	0	0	0	0	0	0	-
C	0	0	0	0	0	0	0	0	0
D	0	0	0	6500	4500	0	4000	0	1
F	0	0	0	0	0	0	0	0	0
G	0	1400	500	0	0	900	0	3200	1
H	0	4200	0	0	0	0	3800	0	1

que indica que la mejor opción es instalarse, además de en la Península Ibérica, en Asia, Sudamérica y África, y el coste total de la expansión, la fabricación y el transporte mínimo es de $z = 69974500$.

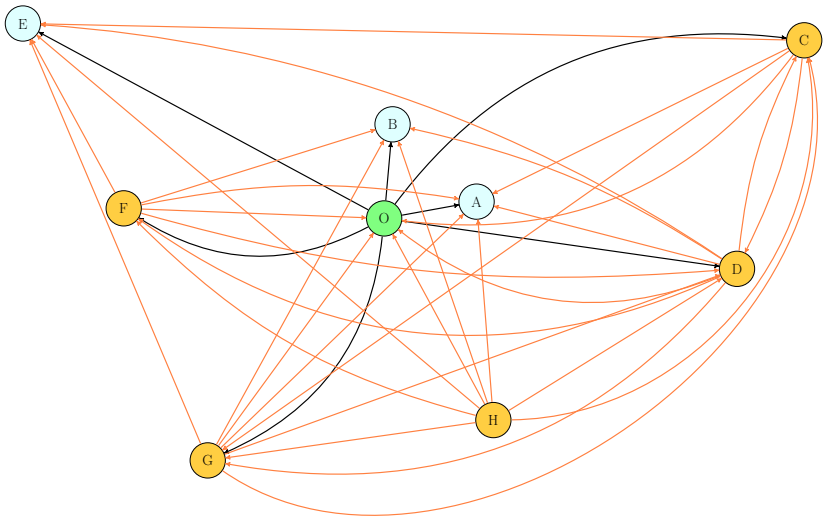


Figura 3.2: Representación del Problema (3.7) como red. En verde el origen existente y en amarillo los posibles nuevos orígenes. Los arcos amarillos representan un posible envío que depende de la construcción de plantas.

Apéndice A

Resolución de problemas en entorno COIN-OR con solver de CPLEX

Será necesario tener descargado e instalado en el ordenador:

- Compilador Visual Express C++ 2013 para implementar los códigos de resolución.
- Software de optimización COIN-OR 1.6 (libre disposición) para establecer la estructura del problema y utilizar las funciones de optimización.
- Software de optimización CPLEX 12.5 (versión académica de 8 procesadores) para utilizar las funciones de optimización.
- Hoja de cálculo Microsoft Office Excel 2010 y Visual Basic (VBA) para generar un entorno amigable de introducción de datos.

Además de instalarlos hay que enlazarlos entre si para el funcionamiento conjunto. La información detallada de la instalación y de enlazamiento se puede consultar en Gloria Pérez, María Araceli Garín, ver [17].

Ahora se procederá a explicar como se pueden introducir los datos en una hoja de cálculo, donde los datos son más amigables para el usuario. Los datos se volcarán en un archivo que sea leido por el compilador Visual C++ para después llamar al solver adecuado para hallar la solución óptima del problema.

Para ello, cada apartado se ilustrará mediante la resolución de un problema del tipo indicado que se haya propuesto en su respectivo capítulo.

Los algoritmos han sido implementados en una computadora con Intel Core i5 3.20GHz y 4GB de memoria.

A.1. Resolución de problemas de flujo de redes a costo mínimo

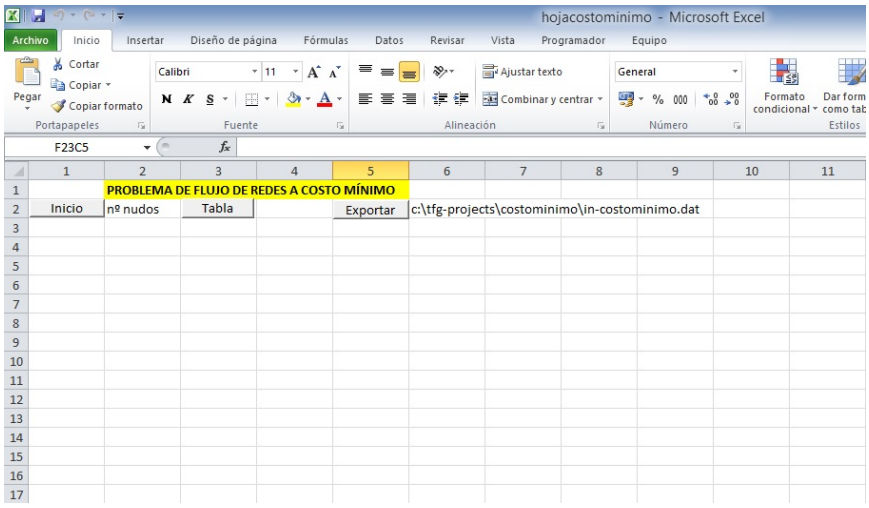
Se va a explicar como resolver computacionalmente el problema de flujo de redes a costo mínimo descrito en la Sección 1.7. Se trabajará con la formulación (1.8).

El problema que se va a tomar como ejemplo es el que viene dado por la red de la Figura 1.4 que se explicó en la página 20. En forma de tabla, el problema queda expresado de este modo:

c_{ij}	1	2	3	4	5	b_i
1			10	6		-17
2	7					9
3		9		20		0
4					8	-11
5			15			19

Para resolver el problema se procederá del siguiente modo.

1. Abrir la hoja de cálculo preparada para resolver los problemas.



2. Introducir el número de nudos en la casilla que se solicita y pulsar el botón TABLA para que se creen las tablas a rellenar con los costos y los valores de los nudos, y con las cotas superiores.

	1	2	3	4	5	6	7	8	9	10
1		PROBLEMA DE FLUJO DE REDES A COSTO MÍNIMO								
2	Inicio	5	Tabla		Exportar	c:\tfg-projects\costominimo\in-costominimo.dat				
3										
4	c_ij	1	2	3	4	5	valor nudo			
5	1									
6	2									
7	3									
8	4									
9	5									
10	u_ij	1	2	3	4	5				
11	1									
12	2									
13	3									
14	4									
15	5									
16										
17										

3. Rellenar la tabla superior con los costos sobre los arcos existentes y la última columna con el valor de cada nudo. Las casillas para las que no exista arco se dejan en blanco. En este caso las cotas superiores quedan en blanco porque es un problema no capacitado.

	1	2	3	4	5	6	7	8	9	10
1		PROBLEMA DE FLUJO DE REDES A COSTO MÍNIMO								
2	Inicio	5	Tabla		Exportar	c:\tfg-projects\costominimo\in-costominimo.dat				
3										
4	c_ij	1	2	3	4	5	valor nudo			
5	1			10	6		-17			
6	2	7					9			
7	3		9		20		0			
8	4					8	-11			
9	5			15			19			
10	u_ij	1	2	3	4	5				
11	1									
12	2									
13	3									
14	4									
15	5									
16										
17										

4. Seleccionar la ruta de destino para el archivo con la información que se utilizará como datos de entrada para el programa de Visual C++. Se escogerá como destino el directorio de trabajo de Visual C++, en este caso, c:\tfg-projects\costominimo\in-costominimo.dat. Pulsar el botón EXPORTAR para generar el archivo en la ruta especificada.

5. En la el directorio de trabajo c:\tfg-projects\costominimo está el archivo \in-costominimo.dat, que tiene la forma: número de nudos, bloque de costos, fila de valores de los nudos y por último bloque de cotas superiores.

```
 5
1e+7 1e+7   10   6 1e+7
 7 1e+7 1e+7 1e+7 1e+7
1e+7   9 1e+7   20 1e+7
1e+7 1e+7 1e+7 1e+7   8
1e+7 1e+7   15 1e+7 1e+7
```

```
-17  9  0 -11  19

1e+7 1e+7 1e+7 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7
```

6. Ejecutar el código .cpp de Visual C++, que es el siguiente:

```
#include "pm.h"

//Funcion para la escritura del fichero de salida
void escribir(double a, int b, int c, const double *x, int nudos,
        double t)
{
        ofstream salida("out-costominimo.dat", ios::out | ios::app);
        int i, j;
        salida << "\nTiempo de CPU es:" << t;
        salida << "\nFuncion objetivo:" << a;
        salida << "\nNumero de variables:" << b;
        salida << "\nNumero de condiciones:" << c;
        salida << "\nLa solucion es:\n";
        for (j = 0; j<nudos; j++)
        {
                for (i = 0; i<nudos; i++)
                {
                        salida << x[i + j*nudos] << " ";
                }
                salida << "\n";
        }
}

//Programa principal
int main()
{
        ifstream entrada("in-costominimo.dat", ios::in);//Entrada
        ofstream salida("out-costominimo.dat");//Salida
        double *dels, *drowlo, *drowup, *dcollo, *dcolup, *costos;
        int *mcolindx, *mrowindx; //indices
        int ncols, nelements, nrows, i, j, m1, k;
        entrada >> m1;
        ncols = m1*m1; nrows = m1; nelements = 2 * ncols;
        //Reserva de memoria
        dels = new double[nelements];
        mcolindx = new int[nelements]; mrowindx = new int[nelements];
        drowlo = new double[nrows]; drowup = new double[nrows];
        dcollo = new double[ncols]; dcolup = new double[ncols];
        costos = new double[ncols];

        //Matriz de condiciones por indices
        k = 0; //posicion en que toca insertar elemento
        for (i = 0; i < m1; i++){
```

```
                for (j = 0; j < m1; j++)
                {
                        dels[k] = 1;
                        mrowindx[k] = i;
                        mcolindx[k] = m1*i + j;
                        k++;
                        dels[k] = -1;
                        mrowindx[k] = i;
                        mcolindx[k] = m1*j + i;
                        k++;
                }
        }

        //Lectura de costos
        for (i = 0; i<ncols; i++) entrada >> costos[i];

        //Lectura de condiciones
        for (i = 0; i < nrows; i++) {
                entrada >> drowup[i];
                drowlo[i] = drowup[i];
        }

        //Cotas inferiores siempre 0
        for (i = 0; i < ncols; i++) dcollo[i] = 0;
        //Lectura de cotas superiores
        for (i = 0; i < ncols; i++) entrada >> dcolup[i];

        //Solver de COIN o de CPLEX
        OsiClpSolverInterface sol1;
        //OsiCpxSolverInterface sol1;

        CoinPackedMatrix A(true, mrowindx, mcolindx, dels, nelements);
        sol1.loadProblem(A, dcollo, dcolup, costos, drowlo, drowup);
        sol1.setObjSense(1);//minimizar

        //Solucion lineal
        sol1.initialSolve();
        double time1 = CoinCpuTime();

        //Escribir en fichero de salida
        escribir(sol1.getObjValue(), sol1.getNumCols(),
                sol1.getNumRows(), sol1.getColSolution(), m1,time1);
        entrada.close(); salida.close();
        return 0;
}
```

7. Se ha creado en el directorio de trabajo el archivo out-costominimo.dat que contiene la información de la solución del problema

```
Tiempo de CPU es:0
Funcion objetivo:696
Numero de variables:25
Numero de condiciones:5
```

```
La solucion es:
0 0 0 0 0
17 0 0 0 0
0 8 0 11 0
0 0 0 0 0
0 0 19 0 0
```

8. Se interpreta la solución. La función objetivo es $z = 696$ y el valor de las variables es

x_{ij}	1	2	3	4	5
1					
2	17				
3		8		11	
4					
5			19		

9. Para limpiar la hoja de cálculo y comenzar otro problema pulsar el botón INICIO.

Macros asociados los botones

Cada botón que se ha utilizado en la hoja Excel tiene asociado una Macro escrita en Visual Basic. Los códigos son los siguientes:

Inicio

```
Sub InicioPCM()

Dim i1, j1, c As Integer

' Limpiar lo anterior '
i1 = 4
j1 = 1
c = 0 'Contador'
While Cells(i1 + c, j1) <> ""
    c = c + 1
Wend
Range(Cells(i1, j1), Cells(i1 + c, j1 + c + 1)).Clear

Cells(2, 2) = "n nudos"
If Cells(2, 6) = "" Then
Cells(2, 6) = "Ruta destino"
End If

End Sub
```

Tabla

```
Sub TablaPCM()

Dim i1, j1, m, k As Integer
```

```
i1 = 4
j1 = 1
m = Cells(2, 2)

'Tabla de costos'
Cells(i1, j1) = "c_ij"
Cells(i1, j1).Interior.ColorIndex = 19
Cells(i1, j1).Interior.ColorIndex = 19
Range(Cells(i1, j1), Cells(i1 + m, j1 + m + 1)).HorizontalAlignment = xlCenter
Range(Cells(i1, j1), Cells(i1 + m, j1 + m + 1)).Borders.LineStyle = xlContinuous

For k = 1 To m
Cells(k + i1, j1) = k
Cells(k + i1, j1).Interior.ColorIndex = 19
Cells(i1, j1 + k) = k
Cells(i1, j1 + k).Interior.ColorIndex = 19
Next k

Cells(i1, m + j1 + 1) = "valor nudo"
Cells(i1, m + j1 + 1).Interior.ColorIndex = 19

'Tabla de cotas superiores'
i1 = i1 + m + 1
Cells(i1, j1) = "u_ij"
Cells(i1, j1).Interior.ColorIndex = 19
Cells(i1, j1).Interior.ColorIndex = 19
Range(Cells(i1, j1), Cells(i1 + m, j1 + m)).HorizontalAlignment = xlCenter
Range(Cells(i1, j1), Cells(i1 + m, j1 + m)).Borders.LineStyle = xlContinuous

For k = 1 To m
Cells(k + i1, j1) = k
Cells(k + i1, j1).Interior.ColorIndex = 19
Cells(i1, j1 + k) = k
Cells(i1, j1 + k).Interior.ColorIndex = 19
Next k

End Sub
```

Exportar

```
Sub ExportarPCM()
Dim cij, uij, bi, salida As String
Dim m As Integer
Dim i1, j1, i As Integer
i1 = 5 'primera fila'
j1 = 2 'primera columna'
salida = Cells(2, 6)
m = Cells(2, 2)
cij = ""
bi = ""
uij = ""

For i = i1 To i1 + m - 1
```

```
    For j = j1 To j1 + m - 1
        If Cells(i, j) = "" Then
            cij = cij & "1e+7" & " "
        Else
            cij = cij & Str(Cells(i, j)) & " "
        End If
    Next j
    cij = cij & Chr(13) + Chr(10)
Next i

For i = i1 To i1 + m - 1
bi = bi & Str(Cells(i, j1 + m)) & " "
Next i
bi = bi & Chr(13) + Chr(10)

i1 = i1 + m + 1
For i = i1 To i1 + m - 1
    For j = j1 To j1 + m - 1
        If Cells(i, j) = "" Then
            uij = uij & "1e+7" & " "
        Else
            uij = uij & Str(Cells(i, j)) & " "
        End If
    Next j
    uij = uij & Chr(13) + Chr(10)
Next i

Open salida For Output As #1
Print #1, Str(m)
Print #1, cij
Print #1, bi
Print #1, uij
Close #1
End Sub
```

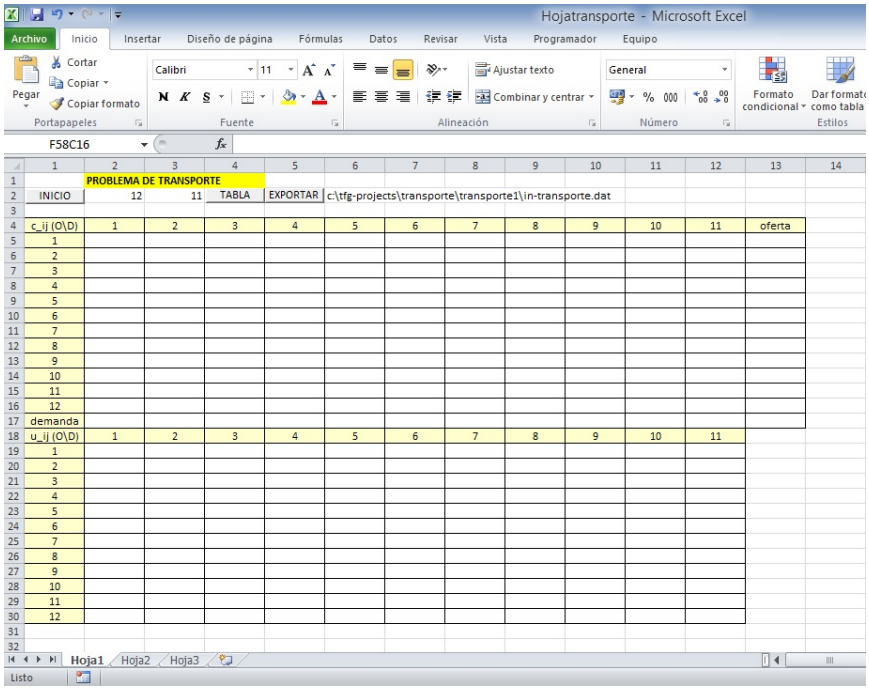
A.2. Resolución de problemas de transporte y transbordo

En esta sección se hará lo análogo a los problemas de redes a costo mínimo, esta vez con problemas de transporte y transbordo. En particular, los códigos están diseñados para la resolución de problemas de transporte con cotas superiores. Sin embargo en el Capítulo 3 se explica como transformar un problema de transbordo en un problema de transporte, por tanto es válido para ambos problemas. Se tomará como ejemplo de referencia el problema de la compañía eléctrica descrito en la Sección 3.1.2. Los pasos a seguir son los mismos que en la sección anterior:

1. Abrir la hoja de cálculo preparada para resolver los problemas de transporte.

2. Introducir el número de orígenes y destinos y pulsar el botón TABLA

para que se creen las tablas a rellenar con los costos, ofertas y demandas, y con las cotas superiores.



3. Rellenar ambas tablas con los datos que se requieren, dejando en blanco las posiciones donde no hay envío posible o donde no hay cota superior.

c_ij (O\D)	1	2	3	4	5	6	7	8	9	10	11	oferta
1	26	23				11	19	16		14	11	33
2	31				27		14	12	16			26
3		25	30		34		11	7	19	16	12	21
4	35							19		15	17	37
5			32				31	14	19	16	12	15
6		30				27	15	11	15	13	19	28
7	29	31					11	15	13	14	15	31
8	15		13	12	9	11	0	5	5	8	3	209
9	14	16	12	11			9	0	7	4	8	227
10		19	11	14	13		6		0	9	9	209
11							5	3	6	0	3	209
12	15	13	16		11	7	6		4	5	0	209
demanda	13	16	27	15	21	20	212	209	220	209	214	

u_ij (O\D)	1	2	3	4	5	6	7	8	9	10	11
1		10									5
2											
3		5						20			
4											
5			20					15			
6											
7	10								15		
8					15						15
9											15
10			5							10	
11								10			10
12	10	15							15		

4. Seleccionar la ruta de destino para el archivo con la información que se utilizará como datos de entrada para el programa de Visual C++. Se escogerá como destino el directorio de trabajo de Visual C++, en este caso, c:\tfg-projects\transporte\in-transporte.dat. Pulsar el botón EXPORTAR para generar el archivo en la ruta especificada.

5. En el directorio de trabajo c:\tfg-projects\transporte está el archivo in-transporte.dat, que tiene la forma: número de orígenes y destinos, bloque de costos, fila de ofertas, fila de demandas y por último bloque de cotas superiores.

```
 12 11
 26  23 1e+7 1e+7 1e+7  11  19  16 1e+7  14  11
 31 1e+7 1e+7 1e+7  27 1e+7  14  12  16 1e+7 1e+7
1e+7  25  30 1e+7  34 1e+7  11  7  19  16  12
 35 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  19 1e+7  15  17
1e+7 1e+7  32 1e+7 1e+7 1e+7  31  14  19  16  12
1e+7  30 1e+7 1e+7 1e+7  27  15  11  15  13  19
 29  31 1e+7 1e+7 1e+7 1e+7  11  15  13  14  15
 15 1e+7  13  12  9  11  0  5  5  8  3
 14  16  12  11 1e+7 1e+7  9  0  7  4  8
1e+7  19  11  14  13 1e+7  6 1e+7  0  9  9
1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  5  3  6  0  3
 15  13  16 1e+7  11  7  6 1e+7  4  5  0

 33 26 21 37 15 28 31 209 227 209 209 209
 13 16 27 15 21 20 212 209 220 209 214

1e+7  10 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  5
1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7
1e+7  5 1e+7 1e+7 1e+7 1e+7 1e+7  20 1e+7 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7
1e+7 1e+7  20 1e+7 1e+7 1e+7 1e+7  15 1e+7 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7
 10 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  15 1e+7 1e+7
1e+7 1e+7 1e+7 1e+7  15 1e+7 1e+7 1e+7 1e+7 1e+7  15
1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  15
1e+7 1e+7  5 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  10 1e+7
1e+7 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  10 1e+7 1e+7  10
 10  15 1e+7 1e+7 1e+7 1e+7 1e+7 1e+7  15 1e+7 1e+7
```

6. Ejecutar el código .cpp de Visual C++, que es el siguiente:

```
#include "pm.h"

//Funcion para la escritura del fichero de salida
void escribir(double a, int b, int c, const double *x,
        int origenes, int destinos,double t )
{
        ofstream salida("out-transporte.dat", ios::out | ios::app);
        int i, j;
        salida << "\nTiempo de CPU es:" << t;
        salida << "\nFuncion objetivo:" << a;
        salida << "\nNumero de variables:" << b;
        salida << "\nNumero de condiciones:" << c;
        salida << "\nLa solucion es:\n";
        for (j = 0; j<origenes; j++)
        {
                for (i = 0; i<destinos; i++)
                {
                        salida << x[i + j*destinos] << " ";
```

```
                }
                salida << "\n";
        }
}

//Programa principal
int main()
{
        ifstream entrada("in-transporte.dat", ios::in);//Entrada
        ofstream salida("out-transporte.dat");//Salida
        double *dels, *drowlo, *drowup, *dcollo, *dcolup, *costos;
        int *mcolindx, *mrowindx; //indices
        int ncols, nelements, nrows, i, j, m1, n1, k;
        entrada >> m1; entrada >> n1;
        ncols = m1*n1; nrows = m1 + n1; nelements = 2*ncols;
        //Reserva de memoria
        dels = new double[nelements];
        mcolindx = new int[nelements]; mrowindx = new int[nelements];
        drowlo = new double[nrows]; drowup = new double[nrows];
        dcollo = new double[ncols]; dcolup = new double[ncols];
        costos = new double[ncols];

        //Matriz de condiciones por indices
        k = 0; //posicion en que toca insertar elemento
        for (j = 0; j<m1; j++)
        {
                for (i = 0; i<n1; i++)
                { //Cada columna tiene dos 1-s.
                        dels[k] = 1;
                        mcolindx[k] = j*n1 + i;
                        mrowindx[k] = j;
                        k++;
                        dels[k] = 1;
                        mcolindx[k] = j*n1 + i;
                        mrowindx[k] = m1 + i;
                        k++;
                }
        }

        //Lectura de ofertas, demandas y costos
        for (i = 0; i<ncols; i++) entrada >> costos[i];
        for (i = 0; i < m1; i++) drowlo[i] = 0;
        for (i = 0; i<m1; i++) entrada >> drowup[i];
        for (i = m1; i < nrows; i++) {
                entrada >> drowlo[i];
                drowup[i] = drowlo[i];
        }
        //Cotas de las variables
        for (i = 0; i<ncols; i++) dcollo[i] = 0;
        for (i = 0; i < ncols; i++) entrada >> dcolup[i];

        //Solver de COIN o de CPLEX
        //OsiClpSolverInterface sol1;
        OsiCpxSolverInterface sol1;
```

```
        CoinPackedMatrix A(true, mrowindx, mcolindx, dels, nelements);
        sol1.loadProblem(A, dcollo, dcolup, costos, drowlo, drowup);
        sol1.setObjSense(1);

        //Solucion lineal
        sol1.initialSolve();
        double time1 = CoinCpuTime();

        escribir(sol1.getObjValue(), sol1.getNumCols(), sol1.getNumRows(),
                 sol1.getColSolution(), m1, n1, time1);
        entrada.close(); salida.close();
        return 0;
}
```

7. Se ha creado en el directorio de trabajo el archivo out-transporte.dat:

```
Tiempo de CPU es:0.046
Funcion objetivo:2269
Numero de variables:132
Numero de condiciones:23
La solucion es:
0 10 0 0 0 20 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 20 0 0 1
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 13
0 0 0 0 0 0 0 17 0 0 0
0 0 0 0 0 0 18 0 11 0 0
0 0 0 0 15 0 194 0 0 0 0
13 0 27 15 0 0 0 172 0 0 0
0 0 0 0 0 0 0 0 209 0 0
0 0 0 0 0 0 0 0 0 209 0
0 6 0 0 6 0 0 0 0 0 197
```

8. Se extrae la solución. La función objetivo es $z = 2269$ y el valor de las variables es

x_{ij}	D_1	D_2	D_3	D_4	D_5	D_6	T_1	T_2	T_3	T_4	T_5
O_1		10				20					3
O_2											
O_3								20			1
O_4											
O_5											13
O_6								17			
O_7							18		11		
T_1					15		194				
T_2	13		27	15				172			
T_3									209		
T_4										209	
T_5		6			6						197

Apéndice B

Experiencia computacional. Generación aleatoria de casos en Matlab

La teoría relativa a la programación en redes, que incluye los problemas de transporte y de transbordo, la cual se se estudió en el Capítulo 1, estaba orientada a demostrar que las particularidades que tienen los problemas de este tipo influyen notablemente en la velocidad con la que pueden ser resueltos comparados con problemas lineales generales.

Se desarrollaba un algoritmo basado en el método simplex, pero además se había demostrado que una característica de los problemas de redes era la integralidad de estos. Se sabe que la solución a un problema de redes donde las ofertas, demandas y cotas superiores de las variables vienen dadas por números enteros es entera. En otras palabras, para resolver un problema de redes entero en el cual la oferta, la demanda y las cotas superiores tienen valores enteros puede ser utilizado un algoritmo de programación lineal, no es necesario el uso de algoritmos específicos de programación entera.

Generación de datos y algoritmos de resolución

Para comprobar el efecto que tiene la integralidad en problemas de redes se ha tomado como referencia los problemas de transporte. Las pruebas realizadas consisten en la resolución de dos tipos de problemas de transporte. En ambos casos se precisa una solución entera. Por una parte se resolverán problemas de transporte con datos reales (dos decimales), para los cuales es necesario el uso de un algoritmo de resolución entero, se utilizará el algoritmo Branch and Bound. Por otra parte se resolverán problemas de transporte con ofertas, demandas y cotas superiores enteras, para los cuales se obtiene solución entera mediante algoritmos lineales. Ambos problemas serán resueltos con los solvers de COIN-OR y de CPLEX. Los problemas han sido

generados de forma aleatoria con Matlab mediante la siguiente función:

```
function []=datawrite(k,m,n,destino)
%k=0: 0 decimales, k=-2: 2 decimales ,...
dim=[m n];
costos=roundn(99*rand(m,n)+1,-2);
o=0; d=1;
while o<d
    demanda=roundn(30+30*rand(1,n),k);
    oferta=roundn(50+50*rand(1,m),k);
    o=sum(oferta); d=sum(demanda);
end
upper=roundn(30+50*rand(m,n),k);
fich=fopen(destino,'w');

for i=1:length(dim)
    fprintf(fich,'%i',dim(i));
    fprintf(fich,' ');
end
fprintf(fich,'\n\n');

for i=1:m
    for j=1:n
        fprintf(fich,'%g',costos(i,j));
        fprintf(fich,' ');
    end
    fprintf(fich,'\n');
end
fprintf(fich,'\n');

for i=1:m
    fprintf(fich,'%g',oferta(i));
    fprintf(fich,' ');
end
fprintf(fich,'\n');

for j=1:n
    fprintf(fich,'%g',demanda(j));
    fprintf(fich,' ');
end
fprintf(fich,'\n\n');

for i=1:m
    for j=1:n
        fprintf(fich,'%g',upper(i,j));
        fprintf(fich,' ');
    end
    fprintf(fich,'\n');
end
fclose('all');
end
```

La resolución lineal ha sido efectuada con el código del Apéndice A. Para los problemas enteros, se ha implementado el algoritmo Branch and Bound sustituyendo en el código mencionado el comando

```
sol1.initialSolve();
```

por la secuencia de comandos

```
//Establecer las variables enteras
for (i = 0; i < ncols; i++) enteras[i] = i;
for (i = 0; i < ncols; i++) sol1.setInteger(enteras[i]);

//Solucion entera
sol1.branchAndBound();
```

Resultados de las pruebas

Los algoritmos han sido implementados en una computadora con Intel Core i5 3.20GHz y 4GB de memoria.

Las pruebas se han realizado sobre problemas con distintas dimensiones $m \times n$, donde m es el número de orígenes y n el número de destinos. Los problemas de transporte de dimensión $m \times n$ tienen mn variables de decisión. El programa se detendrá en caso de no hallar solución óptima pasados los 3600 segundos (1 hora).

En la tabla de algoritmo lineal se resuelven los problemas de transporte con datos enteros, cuya resolución entera se puede efectuar con algoritmos lineales. En la tabla de algoritmo entero se resuelven los problemas de transporte con datos reales, cuya resolución entera hay que efectuar con algoritmos de programación entera (Branch and Bound). Se han realizado varias pruebas para cada tipo de problema y se ha hallado la media de los tiempos para cada uno. Los resultados obtenidos son los siguientes:

ALGORITMO LINEAL				
Dimensión	20×30	150×200	500×750	1200×1800
Tiempo COIN-OR (s)	0.00	0.23	2.74	15.20
Tiempo CPLEX (s)	0.00	0.23	3.44	24.41

ALGORITMO ENTERO					
Dimensión	3×5	5×8	15×20	500×750	800×1200
Tiempo COIN-OR (s)	52.82	326.02	>3600	>3600	>3600
Tiempo CPLEX (s)	0.00	0.00	0.01	11.75	30.24

Conclusiones

Observando los datos que se han obtenido mediante las pruebas, se puede afirmar que la resolución de problemas enteros es más costosa que la de problemas lineales.

Respecto a los algoritmos lineales, COIN-OR y CPLEX no tienen grandes diferencias en los tiempos de resolución de los problemas expuestos. Es

destacable el hecho de que la resolución de un problema de transporte lineal de dimensión 1200×1800, o lo que es lo mismo, un problema con 2160000 variables de decisión, no supera en ningún caso los 25 segundos.

Las diferencias más reseñables entre la resolución lineal y entera se dan al resolver los problemas con COIN-OR, ya que el algoritmo entero Branch and Bound no resuelve un problema de dimensión 15×20, mientras que un problema lineal de tamaño 20×30 es resuelto en un tiempo cercano a cero segundos.

El tiempo de resolución de problemas enteros de CPLEX crece de manera más lenta que con COIN-OR, ya que CPLEX utiliza la programación en paralelo para la resolución de dichos problemas.

En conclusión, con estas pruebas se ha demostrado que la integralidad de los problemas de transporte (y de redes en general) es un factor de gran importancia para su resolución, siendo más notables las diferencias con COIN-OR, pero destacables también para CPLEX.

Apéndice C

Resolución del problema de la expansión de una empresa

La resolución del problema de la expansión de una determinada empresa explicado en el Sección 3.2.2 se ha efectuado con el siguiente código:

```
#include "pm.h"
int main()
{
        ifstream entrada("in-expansion.txt", ios::in);
        ofstream salida("out-expansion.txt");//salida

        double *dels, *drowlo, *drowup, *dcollo, *dcolup, *costos,
                *a, *b;
        int *mcolindx, *mrowindx, *enteras;
        int ncols, nelements, nrows, i, j, m1, n1, k, r1;
        entrada >> m1; entrada >> n1; entrada >> r1;
        ncols = m1*n1 + m1 - 1; nrows = m1 + n1 + 1;
        nelements = 2*m1*n1 + 2 * (m1 - 1);
        dels = new double[nelements];
        mcolindx = new int[nelements]; mrowindx = new int[nelements];
        drowlo = new double[nrows]; drowup = new double[nrows];
        dcollo = new double[ncols]; dcolup = new double[ncols];
        costos = new double[ncols];
        a = new double[m1]; b = new double[n1]; enteras = new int[ncols];

        //Matriz de coeficientes por indices
        k = 0;
        //La parte de las variables x
        for (i = 0; i < m1; i++)
        {
                for (j = 0; j < n1; j++)
                {
                        dels[k] = 1;
                        mrowindx[k] = i;
                        mcolindx[k] = i*n1 + j;
                        k++;
                        dels[k] = 1;
```

```
                    mrowindx[k] = m1 + j;
                    mcolindx[k] = i*n1 + j;
                    k++;
            }
    }

    for (i = 0; i < ncols; i++) entrada >> costos[i];
    for (i = 0; i < m1; i++) entrada >> a[i];

    //La parte de las variables y
    for (i = 0; i < m1 - 1; i++){
            dels[k] = -a[i+1];
            mrowindx[k] = i+1;
            mcolindx[k] = m1*n1 + i;
            k++;
    }
    for (i = 0; i < m1 - 1; i++){
            dels[k] = 1;
            mrowindx[k] = m1 + n1;
            mcolindx[k] = m1*n1 + i;
            k++;
    }

    //Cotas de restricciones y variables
    drowlo[0] = -1e+31; drowup[0] = a[0];
    for (i = 1; i < m1; i++){
            drowlo[i] = -1e+31; drowup[i] = 0;
    }
    for (i = 0; i < n1; i++) entrada >> b[i];
    for (i = m1; i < nrows-1; i++){
            drowlo[i] = b[i - m1]; drowup[i] = drowlo[i];
    }
    drowlo[nrows-1] = 0; drowup[nrows-1] = 3;
    for (i = 0; i < m1*n1; i++){
            dcollo[i] = 0; dcolup[i]=1e+31;
    }
    for (i = m1*n1; i < ncols; i++){
            dcollo[i] = 0; dcolup[i] = 1;
    }
    //Solver de COIN o de CPLEX
    OsiClpSolverInterface sol1;
    //OsiCpxSolverInterface sol1;
    CoinPackedMatrix A(true, mrowindx, mcolindx, dels, nelements);
    sol1.loadProblem(A, dcollo, dcolup, costos, drowlo, drowup);
    sol1.setObjSense(1);

    //Solucion entera
    for (i = m1*n1; i < ncols; i++) enteras[i] = i;
    for (i = m1*n1; i < ncols; i++) sol1.setInteger(enteras[i]);
    sol1.branchAndBound();
    double time1 = CoinCpuTime();

    //Escribir
    salida << "\nTiempo de CPU es:" << time1;
```

```
        salida << "\nFuncion objetivo:" << sol1.getObjValue();
        salida << "\nNumero de variables:" << sol1.getNumCols();
        salida << "\nNumero de condiciones:" << sol1.getNumRows();
        salida << "\nLa solucion es:\n x_{ij}:\n";
        for (j = 0; j<m1; j++)
        {
                for (i = 0; i<n1; i++)
                {
                        salida << sol1.getColSolution()[i + j*n1] << " ";
                }
                salida << "\n";
        }
        salida << " y_i:\n";
        for (i = 0; i < m1 - 1; i++){
                salida << sol1.getColSolution()[m1*n1 + i] << " ";
        }
        entrada.close(); salida.close();
        return 0;
}
```

El fichero de entrada in-expansion.txt es

```
6 8 3

2515 3300 3400 3900 4400 1e+31 4800 3450
3600 3500 1e+31 2035 2800 3300 1e+31 3900
2100 2200 2000 2100 1520 2800 3000 2800
4000 4100 4000 1e+31 4200 3300 2525 1e+31
1900 2200 2300 3000 2400 2000 1e+31 1510
1000 1200 1400 2400 2200 2600 2500 2300

5e+6 3e+6 7e+6 2e+6 1e+6

30000 10000 15000 12000 6000 8000

1500 6300 500 6500 4500 900 7800 3200
```

y el fichero out-expansion.txt que devuelve el problema con información y la solución óptima es el siguiente:

```
Tiempo de CPU es:0
Funcion objetivo:6.99745e+007
Numero de variables:53
Numero de condiciones:15
La solucion es:
 x_{ij}:
1500 700 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 6500 4500 0 4000 0
0 0 0 0 0 0 0 0
0 1400 500 0 0 900 0 3200
0 4200 0 0 0 0 3800 0
 y_i:
0 1 0 1 1
```

Bibliografía

[1] Adhikari, Purnima, Gyan Badhur Thapa, A Note on Feasibility and Optimality of Transportation Problem, Journal of the Institute of Engineering, Volume **10**, No. 1 (2014), pp. 59-68.

[2] Ahrens, J.H., G. Finke, Primal Transportation and Transshipment Algorithms, Zeitschrift fr Operations Research February 1980, Volume **24**, Issue 1, pp. 1-32.

[3] Bapat, R. B., *Graphs and Matrices*, Springer, Hindustan Book Agency, London, New Delhi, 2010.

[4] Bazaraa, Mokhtar S., *Linear programming and network flows*, 4ª ed., John Wiley & Sons, Hoboken, New Jersey, 2010.

[5] Dantzig, George B., *Linear Programming and Extensions*. Princeton University, Princeton, New Jersey, 1963.

[6] Díaz-Parra, O., J. A. Ruiz-Vanoye, B. B. Loranca, A. Fuentes-Penna, R. A. Barrera-Cámara, A Survey of Transportation Problems, Journal of Applied Mathematics, 2014, pp. 1-17.

[7] Glover, Fred, Darwin Klingman, Nancy V. Phillips, *Network Models in Optimization and Their Applications in Practice*, John Wiley & Sons, New-York, 1992.

[8] Hartsfield, N., G. Ringel, *Pearls in Graph Theory*, Dover, New York, 1994.

[9] Hillier, Frederick S., Gerald J. Lieberman, *Introducción a la investigación de operaciones*, 9ª ed., McGraw-Hill, México, 2010.

[10] Hitchcock, F.L., The distribution of a product from several sources to numerous localities, Journal of Mathematics and Physics, **20** (1941), pp. 224-230.

[11] Jauffred Mercado, Francisco J., *Métodos de optimización: Programación lineal-gráficas*. Representaciones y Servicios de Ingeniería, México, 1971.

[12] Kennington, J. L., R. Helgason. *Algorithms for network programming*, John Wiley & Sons, 1980.

[13] Koch, T., T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, K. Wolter, MIPLIB 2010, Mathematical Programming Computation, **3**, Issue 2, pp. 103-163.

[14] Koopmans, Tjalling C., *Activity Analysis of Production and Allocation*, John Wiley & Sons, Inc., New York Chapman & Hall, Limited, London, 1951.

[15] Luenberger, David E., *Programación lineal y no lineal*, Addison-Wesley Iberoamericana, Buenos Aires, 1989.

[16] Pérez, Gloria, *Programación Matemática*, Servicio Editorial UPV-EHU, 2000.

[17] Pérez, Gloria, María Araceli Garín, On Downloading and Using CPLEX within COIN-OR for Solving Linear/Integer Optimization Problems, Biltoki, 2010. https://addi.ehu.es/handle/10810/5504.

[18] Prawda, Juan, *Métodos y modelos de investigación de operaciones*, Editorial Limusa, México, 1976.