Programación Lineal y Entera

Balbina Virginia Casas Méndez

Apuntes de la asignatura. Segunda parte.

MÁSTER EN TÉCNICAS ESTADÍSTICAS

Curso 2009/10

Tema 3

- El problema del transporte
- Los problemas de asignación
- El problema del camino más corto
- Problema del árbol de expansión mínima
- Problema del flujo máximo
- Planificación de horarios
- Problema del viajante de comercio
- El problema de la mochila
- Planificación y control de proyectos

Tema 3

- El problema del transporte
- Los problemas de asignación
- El problema del camino más corto
- Problema del árbol de expansión mínima
- Problema del flujo máximo
- Planificación de horarios
- Problema del viajante de comercio
- El problema de la mochila
- Planificación y control de proyectos

Tema 4

- Introducción a la heurística
- Introducción a la programación dinámica

Tema 3

- El problema del transporte
- Los problemas de asignación
- El problema del camino más corto
- Problema del árbol de expansión mínima
- Problema del flujo máximo
- Planificación de horarios
- Problema del viajante de comercio
- El problema de la mochila
- Planificación y control de proyectos

Tema 4

- Introducción a la heurística
- Introducción a la programación dinámica

Ejercicios

Optimización en redes

Vamos a considerar problemas de Programación Matemática que se pueden plantear sobre un *grafo*: un conjunto de *vértices* o nodos conectados con *arcos*, dado que muchos problemas de Investigación Operativa se pueden modelar y resolver de esta forma:

- Diseñar el trazado de una red de fibra óptica de manera que se cubran ciertos puntos de la manera más económica posible. *Árbol generador de coste mínimo.*
- Determinar la ruta más corta que recorre las parcelas que tiene que procesar una cosechadora de forraje. *Camino más corto.*
- Determinar la cantidad máxima de electricidad que se puede enviar a través de una red eléctrica. *Problema de flujo máximo*.
- Decidir el calendario de fechas en el que deben iniciarse y terminarse una serie de tareas para llevar a cabo un proyecto. *Camino crítico*.

A través de la teoría de grafos, en algunos casos, se pueden desarrollar algoritmos más eficientes que los vistos hasta ahora.

En ocasiones resultan más difíciles de resolver de forma exacta y se necesitan técnicas heurísticas para encontrar buenas soluciones.

Definiciones básicas

Un *grafo* consta de un conjunto de *vértices* (nodos) y de un conjunto de *arcos* que unen los vértices. El grafo puede tener:

- Todos los *arcos dirigidos*.
- Todos los *arcos no dirigidos*.

En otro caso se trata de un *grafo mixto*.

Se denomina *flujo* a cualquier “bien" (tangible o no) que circule por las conexiones del grafo (o red) (electricidad, una cosechadora, mensaje, tiempo).

Una *ruta* es una secuencia de arcos, dirigidos o no, que unen dos vértices.

Un *ciclo* es una ruta que une un vértice consigo mismo.

Un *grafo es conexo* si cualquier pareja de vértices puede unirse con una ruta sobre el grafo.

El problema del transporte: introducción

El problema del transporte surge frecuentmente en la distribución óptima de bienes desde varios puntos de suministro (por ejemplo, factorías, plantas) a puntos de demanda (por ejemplo, almacenes, clientes, puntos de venta).

El objetivo es normalmente encontrar el mejor plan de distribución para minimizar los costes totales de envío desde los puntos de suministro a los puntos de demanda.

Un problema más general involucra la decisión acerca de la misma *localización* de los orígenes, por ejemplo, servicios de ambulancias que tienen que atender las necesidades de los pacientes de ciertos hospitales.

Un ejemplo de problema del transporte: resolución informática

Ejemplo. Gastos de envío de esquíes. Para ilustrar este modelo vamos a considerar el problema de SunSno, una empresa multinacional con tres factorías en (1) Jasper, Canadá, (2) Seoul, Korea, and (3) Toronto, Canadá. SunSno transporta esquíes a cuatro empresas propietarias de almacenes en (1) Frankfurt, Alemania, (2) New York, USA, (3) París, Francia, y (4) Yokohama, Japón.

Las capacidades de producción semanales, a_i, $i = 1,\ 2,\ 3$ de las $m = 3$ factorías y las demandas semanales, b_j, $j = 1,\ 2,\ 3,\ 4$ de los $n = 4$ almacenes están dados en la siguiente tabla donde también indicamos los costes por unidad de transporte c_{ij}, $i = 1,\ 2,\ 3$ y $j = 1,\ 2,\ 3,\ 4$.

Desde$\downarrow$ \ A$\rightarrow$	Frankfurt	NY	París	Yoko-hama	SUMI-NISTRO
Jasper	19	7	13	8	100
Seoul	15	21	18	6	300
Toronto	11	3	12	20	200
DEMANDA	150	100	200	150	600

Para resolver este problema como uno de programación lineal, definimos las variables X_{ij} como el número de unidades transportadas desde la factoría i al almacén j para las factorías $i = 1, \cdots, m$ y los almacenes $j = 1, \cdots, n$.

De esta forma, el modelo de programación lineal del problema del transporte toma la siguiente forma:

$$\text{Min} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij}$$

sujeto a

$$\sum_{j=1}^{n} X_{ij} = a_i \text{ para } i = 1, \cdots, m$$

$$\sum_{i=1}^{m} X_{ij} = b_j \text{ para } j = 1, \cdots, n$$

con $X_{ij} \geq 0$, $i = 1, \cdots, m$ y $j = 1, \cdots, n$.

Particularizando para el problema de decisión de la empresa SunSno, obtenemos la siguiente formulación:

$$\text{Min } Z = 19X_{11} + 7X_{12} + 13X_{13} + 8X_{14} + 15X_{21} + 21X_{22} + 18X_{23} + 6X_{24} + 11X_{31} + 3X_{32} + 12X_{33} + 20X_{34}$$

sujeto a

$$X_{11} + X_{12} + X_{13} + X_{14} = 100 \text{ (Capacidad de Jasper)}$$
$$X_{21} + X_{22} + X_{23} + X_{24} = 300 \text{ (Capacidad de Seoul)}$$
$$X_{31} + X_{32} + X_{33} + X_{34} = 200 \text{ (Capacidad de Toronto)}$$
$$X_{11} + X_{21} + X_{31} = 150 \text{ (Demanda de Frankfurt)}$$
$$X_{12} + X_{22} + X_{32} = 100 \text{ (Demanda de New York)}$$
$$X_{13} + X_{23} + X_{33} = 200 \text{ (Demanda de París)}$$
$$X_{14} + X_{24} + X_{34} = 150 \text{ (Demanda de Yokohoma)}$$

Todo $X_{ij} \geq 0$.

Notemos que, en general, formular el problema del transporte como un problema de programación lineal resulta en $m \times n$ variables de decisión y $m + n$ restricciones. Ahora bien, aunque hay un total de $m + n$ restricciones en los problemas de transporte, el número total de variables básicas nunca es de más de $(m + n - 1)$ debido a la redundancia en las restricciones: si se encuentra una solución que satisface las primeras $m + n - 1$ restricciones, debe automáticamente satisfacer la última ya que los suministros totales igualan las demandas totales.

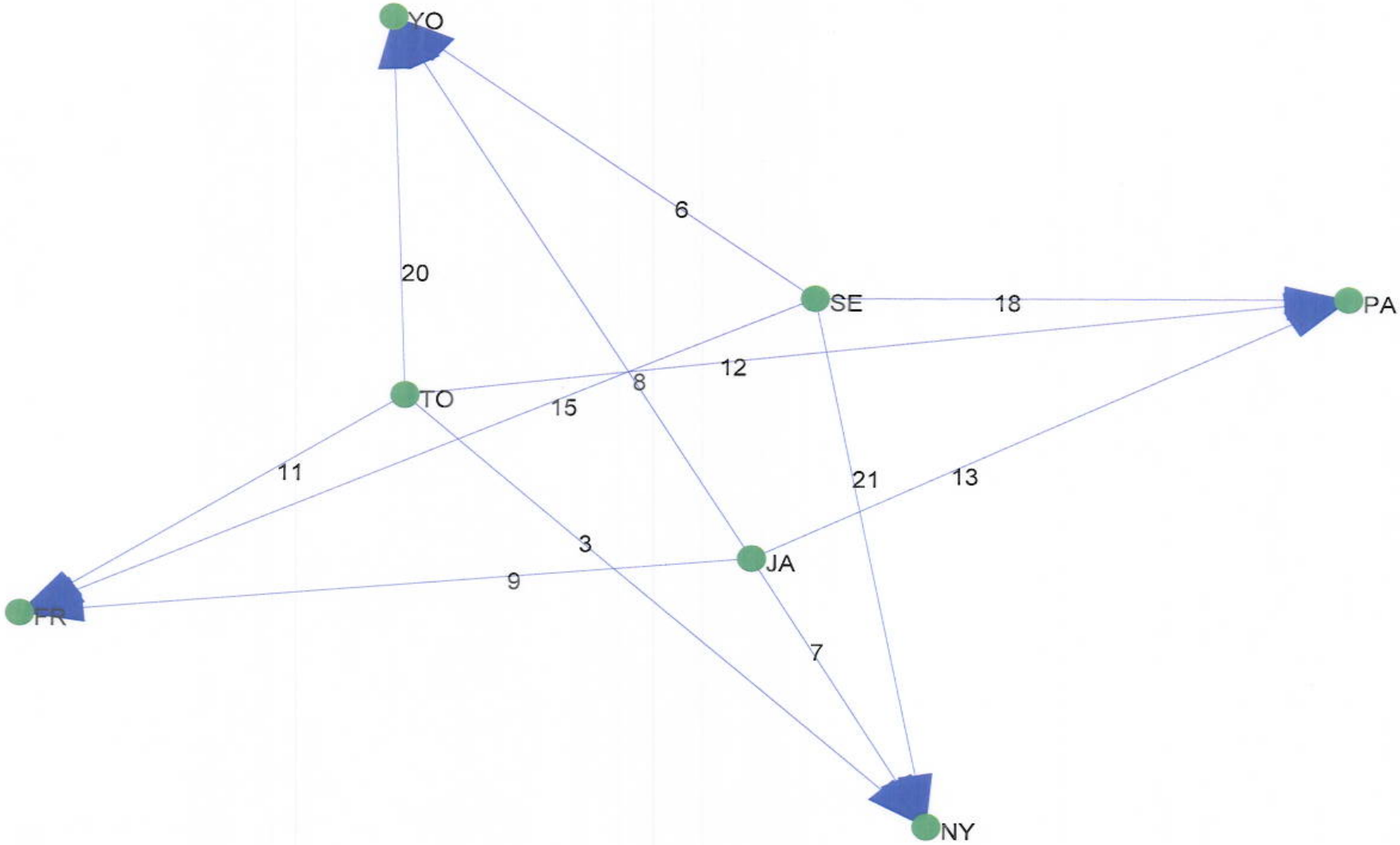
En este problema tenemos $3 \times 4 = 12$ variables y $3+4=7$ restricciones. También, ya que el suministro total es 600 unidades e igual a la demanda total, las factorías transportarán todo lo que producen y las demandas de los almacenes van a satisfacerse completamente. Los problemas del transporte donde $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ se denominan *equilibrados*.

Dibujo del grafo del transporte con Maxima

```
(%i1) load (graphs)$
(%i2)g:create_graph([[1,"JA"],[2,"SE"],[3,"TO"],
[4,"FR"],[5,"NY"],[6,"PA"],[7,"YO"]],[[[1,4],9],
[[1,5],7],[[1,6],13],[[1,7],8],[[2,4],15],[[2,5],21],
[[2,6],18],[[2,7],6],[[3,4],11],[[3,5],3],[[3,6],12],
[[3,7],20]],'directed = true)$
(%i3) draw_graph(g,show_label=true,show_weight=true,
edge_color=blue,vertex_color=green);
```

La resolución se puede hacer con Maxima o Solver (Excel/OpenOffice).

El grafo del problema del transporte



Problema del transporte en forma estándar

Vamos a describir diferentes heurísticos para encontrar soluciones básicas factibles iniciales y un procedimiento para encontrar una solución óptima.

Aunque este tipo especial de problemas de programación lineal se usa para encontrar la planificación óptima de un transporte también se puede aplicar a otras áreas, entre otras, problemas de inventario, problemas de horarios y de asignación de tareas, entre otros.

Un producto homogéneo debe ser transportado de m orígenes, s_i, $i = 1, \cdots, m$ a n destinos, d_j, $j = 1, \cdots, n$. Se tienen unas disponibilidades, a_i, en cada origen s_i, $i = 1, \cdots, m$ y unas demandas b_j, en cada destino d_j, $j = 1, \cdots, n$. Además, c_{ij} es el coste de transporte unitario del origen s_i al destino d_j.

Matriz de costes $m \times n$

Con $m = 3$ y $n = 3$ y x_{ij} denotando el número de unidades transportadas desde el origen s_i hasta el destino d_j, resulta:

Desde$\downarrow$ \ A$\rightarrow$	d_1	d_2	d_3	SUMI-NISTRO $\downarrow$
s_1	$X_{11}/[c_{11}]$	$X_{12}/[c_{12}]$	$X_{13}/[c_{13}]$	a_1
s_2	$X_{21}/[c_{21}]$	$X_{22}/[c_{22}]$	$X_{23}/[c_{23}]$	a_2
s_3	$X_{31}/[c_{31}]$	$X_{32}/[c_{32}]$	$X_{33}/[c_{33}]$	a_3
DEMANDA$\rightarrow$	b_1	b_2	b_3	

Se quiere determinar el número de unidades a transportar desde cada origen a cada destino de forma que se minimice el coste de transporte.

Modelo matemático del problema del transporte

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij}$$

sujeto a

$$\sum_{j=1}^{n} X_{ij} \leq a_i \text{ para } i = 1, \cdots, m \text{ (suministro)}$$

$$\sum_{i=1}^{m} X_{ij} \geq b_j \text{ para } j = 1, \cdots, n \text{ (demanda)}$$

con $X_{ij} \geq 0$, $i = 1, \cdots, m$ y $j = 1, \cdots, n$.

PROPOSICIÓN En un problema del transporte equilibrado con m orígenes y n destinos, sólo $m + n - 1$ restricciones son linealmente independientes.

Demostración El conjunto de restricciones de un problema del transporte equilibrado con m orígenes y n destinos está dado por:

$$\sum_{j=1}^{n} X_{ij} = a_i, \ i = 1, \cdots, m \text{ restricciones fila}$$

$$\sum_{i=1}^{m} X_{ij} = b_j, \ j = 1, \cdots, n \text{ restricciones columna}$$

Para ver que $m+n-1$ restricciones son linealmente independientes, basta ver que una cualquiera de las $m+n$ restricciones se puede escribir como combinación lineal de las otras.

Sumamos todas las restricciones fila y $n-1$ restricciones columna:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{m} a_i \quad \text{y} \quad \sum_{j=1}^{n-1}\sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n-1} b_j$$

Restando la última igualdad de la penúltima:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n-1}\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j + \sum_{i=1}^{m} x_{in} = \sum_{i=1}^{m} x_{in}$$

Así, la n restricción columna se ha escrito como la diferencia de la suma de todas las restricciones fila y la suma de $n-1$ restricciones columna. $\square$

Por la proposición anterior, tendremos $m+n-1$ variables básicas y las restantes $(m-1)(n-1)$ variables no básicas. En consecuencia, como mucho se tendrán:

$$\frac{(mn)!}{(m+n-1)!(mn-(m+n-1))!}$$

soluciones factibles básicas.

En muchas aplicaciones, suministros, demandas y valores de las variables deben de tomar valores enteros. A causa de la especial estructura del problema, si el modelo tiene una solución factible, siempre va a tener una solución óptima con valores enteros.

Soluciones básicas factibles iniciales: regla de la esquina noroeste

En este método, se asigna $\min(a_i, b_j)$, donde a_i es la disponibilidad del origen i y b_j la demanda del destino j en la esquina noroeste de la matriz de costes. La fila o columna que lo verifica deja de considerarse en las restantes submatrices. Encontrar la esquina noroeste de la submatriz restante y asignar de la manera anterior. Se sigue con el proceso hasta cumplir restricciones de filas y columnas.

La siguiente tabla ilustra el método.

15/[5] /[6] /[2]	15/[2] 5/[4] /[3]	/[4] 35/[9] 5/[8]	/[3] /[5] 50/[1]	~~30~~ ~~40~~ ~~55~~	~~15~~ ~~35~~ ~~50~~
~~15~~ ~~5~~	~~20~~ ~~5~~	~~40~~ ~~5~~	~~50~~		

Así la solución inicial es:

$x_{11} = 15,\ x_{12} = 15,\ x_{22} = 5,\ x_{23} = 35,\ x_{33} = 5,\ x_{34} = 50,\ z = 530.$

Este método no considera los costes de distribución y por eso, aunque es muy sencillo, es menos preferido que otros.

Método de mínimo coste

Se asigna $\min(a_i, b_j)$ a la celda con menor coste en la matriz de costes. Se deja de considerar la fila o columna que proporciona ese mínimo en las siguiente iteraciones. Efectuar las asignaciones de la misma manera para las restantes submatrices. En caso de empate para la celda con menor coste, escoger la celda que permita acomodar una mayor asignación y en otro caso seleccionar arbitrariamente.

/[5]	20/[2]	10/[4]	/[3]	~~30~~	~~10~~
10/[6]	/[4]	30/[9]	/[5]	~~40~~	~~30~~
5/[2]	/[3]	/[8]	50/[1]	~~55~~	~~5~~
~~15~~	~~20~~	~~40~~	~~50~~		
~~10~~		~~30~~			

Así la solución inicial es:

$$x_{12} = 20,\ x_{13} = 10,\ x_{21} = 10,\ x_{23} = 30,\ x_{31} = 5,\ x_{34} = 50,\ z = 470.$$

Método de aproximación de Vogel

- **Paso I**. Considerar la primera fila y escoger el coste menor y restarlo del siguiente al coste menor. Se escribe enfrente de la fila y constituye el penalty para esa fila. Proceder análogamente con todas las filas y con todas las columnas, escribiendo en este caso los penaltys bajo las correspondientes columnas.
- **Paso II**. Seleccionar el penalty más alto y en la fila o columna correspondiente seleccionar el coste más bajo. En esa casilla asignar $\min(a_i, b_j)$ e ignorar en lo sucesivo la fila o columna que proporciona ese número.
- **Paso III**. Recalcular los penaltys con la submatriz restante, según el primer paso y efectuar una asignación según el segundo paso. Se continúa hasta que queda sólo una fila o columna por asignar. La asignación de ésta se realiza por el método de mínimo coste.

En caso de empates para el penalty más alto se escoge la celda con menor coste en todas las filas y columnas empatadas. Si nuevamente se tiene un empate, seleccionar para la asignación la celda que proporciona un menor $c_{ij}x_{ij}$.

La intuición del método radica en que si no se considera la celda con menor coste en una fila o columna con el mayor penalty, otra asignación va a producir un incremento del penalty por cada unidad.

En el ejemplo que venimos considerando, se obtiene, al aplicar este método, la solución:

$$x_{13} = 30,\ x_{21} = 10,\ x_{22} = 20,\ x_{23} = 10,\ x_{31} = 5,\ x_{34} = 50,\ z = 410.$$

/[5]	/[2]	30/[4]	/[3]	~~30~~		1	- -
10/[6]	20/[4]	10/[9]	/[5]	~~40~~		1	1 2
5/[2]	/[3]	/[8]	50/[1]	~~55~~	~~5~~	1	1 1
~~15~~	~~20~~	~~40~~	~~50~~				
~~10~~		~~10~~					
3	1	4	2				
4	1	1	4				
4	1	1	-				

Soluciones básicas factibles degeneradas

Si en cualquier método de obtención de una solución factible básica inicial (excepto en la última asignación), al efectuar una asignación el mínimo se alcanza en una fila y en una columna, se considera que se alcanza en una de ellas y se escribe un cero en la otra.

20/[2] /[4] /[3]	/[3] 50/[1] /[4]	30/[7] 0/[5] /20[7]	/[1] /[8] 60/[1]	~~50~~ ~~50~~ ~~80~~	~~30~~ ~~0~~ ~~20~~
~~20~~	~~50~~	~~50~~ ~~20~~	~~60~~		

La solución básica factible degenerada es:

$$x_{11} = 20,\ x_{13} = 30,\ x_{22} = 50,\ x_{23} = 0,\ x_{33} = 20,\ x_{34} = 60,\ z = 500.$$

Comentarios

- El método de la esquina noroeste proporciona peores soluciones factibles básicas iniciales y obliga a un mayor número de iteraciones. El método de mínimo coste da soluciones factibles básicas iniciales bastante buenas. El método de Vogel da las más prźimas al óptimo.
- Condición necesaria y suficiente de existencia de una solución inicial básica factible es que el problema sea equilibrado.
- Un problema del transporte equilibrado siempre tiene solución óptima.

Determinación de soluciones óptimas

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij}$$

sujeto a

$$\sum_{j=1}^{n} X_{ij} = a_i \text{ para } i = 1, \cdots, m$$

$$\sum_{i=1}^{m} X_{ij} = b_j \text{ para } j = 1, \cdots, n$$

con $X_{ij} \geq 0$, $i = 1, \cdots, m$ y $j = 1, \cdots, n$.

Sea u_i la variable dual asociada con la restricción $i = 1, \cdots, m$ y v_j la variable dual asociada con la columna $j = 1, \cdots, n$. El dual del problema anterior es:

$$\text{Max} \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j$$

sujeto a

$u_i + v_j \leq c_{ij}$ para $i = 1, \cdots, m,\ j = 1, \cdots, n$

u_i, v_j no restringidas, para $i = 1, \cdots, m,\ j = 1, \cdots, n$.

En la proposición anterior, probamos que en una solución factible básica del primal, $m + n - 1$ variables son básicas y las restantes son no básicas. Supongamos que las variables duales se seleccionan de tal manera que:

$$u_i + v_j = c_{ij} \text{ en } m + n - 1 \text{ celdas base}$$

$$u_i + v_j \leq c_{ij} \text{ en las restantes celdas}$$

Tal elección de las variables duales implica que las variables de holgura en el dual verifiquen $r_{ij} = 0$ en las celdas básicas y que $r_{ij} \geq 0$ en las celdas no básicas. De esta forma, además, para las celdas básicas:

$$x_{ij} \geq 0,\ r_{ij} x_{ij} = 0.$$

Para las restantes celdas,

$$x_{ij} = 0,\ r_{ij} x_{ij} = 0.$$

Semejante elección de las variables duales satisface la propiedad de holguras complementarias y así conduce a una solución óptima del problema del transporte. De esta forma, ya podemos esquematizar el algoritmo:

Método símplex del transporte de obtención de soluciones óptimas

- **Paso I**. Se introducen las variables duales u_i y v_j correspondientes a cada fila i y a cada columna j, respectivamente. Escribiremos u_i en frente de cada fila i y v_j en la parte superior de cada columna. Tomaremos algún u_i o algún v_j como cero.
- **Paso II**. Para celdas básicas (en las que se han realizado asignaciones), $u_i + v_j = c_{ij}$. Esta relación asigna valores a todas las variables u_i y v_j.
- **Paso III**. Para las celdas no básicas (que no tienen asignación), calcular las holguras $u_i + v_j - c_{ij}$ y se van a anotar entre paréntesis en la celda correspondiente.

- **Paso IV**. Teniendo en cuenta que las holguras $u_i + v_j - c_{ij}$ se corresponden con los **costes reducidos** $z_j - c_j$ para las variables del problema original en el método del símplex (nulas, como sabemos, para las variables básicas), si todas las entradas de las celdas entre paréntesis son ≤ 0, se concluye que la solución factible básica que estamos chequeando es óptima. Si al menos una de esas entradas es positiva, la solución factible básica que estamos chequeando no es óptima. En este caso, buscamos la más positiva para decidir la variable de entrada.
- **Paso V**. Se asigna una cantidad θ en la celda con la holgura más positiva y hacer un bucle como se explica a continuación.

Regla para hacer el bucle. Comenzando con la celda donde está θ desplazarse horizontal y verticalmente a la celda básica más próxima con la restricción de que la esquina del bucle no coincida con una celda no básica (excepto aquélla en la que anotamos θ). De esta forma, volver a la celda con θ para completar el bucle.
Se suma o se resta θ en las esquinas del bucle manteniendo la factibilidad, y el valor de θ se fija como el mínimo de las entradas de las que θ se debe de restar. Insertando el valor fijado para θ, se obtiene la nueva solución factible básica que mejora el coste de transporte inicial. Una celda va a pasar a tener valor cero y no hay que anotarlo porque se corresponde con la variable que deja la base, es decir, es una celda que se convierte en no básica. De esta forma, concluye una iteración con una nueva solución básica factible y se repiten los pasos desde el primero al quinto. El algoritmo termina cuando todas las holguras correspondientes a celdas no básicas son iguales o menores que cero.

Comentarios a la realización de los bucles

- Un bucle puede cruzar sobre sí mismo, en particular cruzar por una celda no básica.
- Un bucle puede pasar sobre celdas básicas o sobre celdas no básicas en una fila o en una columna. Ahora bien, las líneas sucesivas deben de ser perpendiculares.
- Siempre existe un único bucle empezando desde una celda no básica dada.

Un ejemplo de problema del transporte: resolución manual

Vamos a considerar el siguiente problema del transporte, cuya solución básica factible inicial se obtuvo por el método del mínimo coste y vamos a determinar su solución óptima.

15/[1] /[7] /[4]	15/[2] /[6] 15/[3]	/[3] 25/[2] /[2]	/[4] 25/[5] 20/[7]	30 50 35
15	30	25	45	

Introduciremos las variables duales u_i y v_j. Fijaremos a cero alguna de estas variables y como criterio lo haremos con la variable asociada a la fila con más celdas básicas y arbitrariamente, en caso de empate.

	$v_1 = 1$	$v_2 = 2$	$v_3 = 3$	$v_4 = 6$
$u_1 = 0$	15/[1]	15/[2]	(0) /[3]	(2)/[4]
$u_2 = -1$	(-7)/[7]	(-5) /[6]	25/[2]	25/[5]
$u_3 = 1$	(-2)/[4]	15/[3]	(2) /[2]	20/[7]

Asignamos $u_1 = 0$ y calculamos los valores de las otras variables duales usando la relación $u_i + v_j = c_{ij}$ para celdas básicas:

$$u_1 + v_1 = 1,\ u_1 + v_2 = 2,\ u_2 + v_3 = 2,$$

$$u_2 + v_4 = 5,\ u_3 + v_2 = 3,\ u_3 + v_4 = 7.$$

De donde,
$u_1 = 0, v_1 = 1, v_2 = 2, u_3 = 1, v_4 = 6, u_2 = -1, v_3 = 3.$

	$v_1 = 1$	$v_2 = 2$	$v_3 = 3$	$v_4 = 6$
$u_1 = 0$	15/[1]	15/[2]	(0) /[3]	(2)/[4]
$u_2 = -1$	(-7)/[7]	(-5) /[6]	$25-\theta$ $\rightarrow$/[2]	$25+\theta$ $\downarrow$/[5]
$u_3 = 1$	(-2)/[4]	15/[3]	(2) θ $\uparrow$/[2]	$\leftarrow 20-\theta$/[7]

$$\theta = \min\{20, 25\} = 20.$$

	$v_1 = 1$	$v_2 = 2$	$v_3 = 3$	$v_4 = 6$
$u_1 = 0$	15/[1]	15/[2]	(0) /[3]	(2)/[4]
$u_2 = -1$	(-7)/[7]	(-5) /[6]	5/[2]	45/[5]
$u_3 = 1$	(-2)/[4]	15/[3]	(2) 20/[2]	/[7]

$$u_1 + v_1 = 1,\ u_1 + v_2 = 2,\ u_2 + v_3 = 2,$$

$$u_2 + v_4 = 5,\ u_3 + v_2 = 3,\ u_3 + v_3 = 2.$$

De donde, $u_1 = 0, v_1 = 1, v_2 = 2, u_3 = 1, v_4 = 4, u_2 = 1, v_3 = 1$. Y sobre la misma tabla determinamos las holguras correspondientes a las casillas no básicas. Vemos que todas son menores o iguales que cero, con lo que la actual solución básica factible es óptima:

$x_{11} = 15, x_{12} = 15, x_{23} = 5, x_{24} = 45, x_{32} = 15, x_{33} = 20, z = 365.$

	$v_1 = 1$	$v_2 = 2$	$v_3 = 1$	$v_4 = 4$
$u_1 = 0$	15/[1]	15/[2]	(-2) /[3]	(0)/[4]
$u_2 = 1$	(-5)/[7]	(-3) /[6]	5/[2]	45/[5]
$u_3 = 1$	(-2)/[4]	15/[3]	20/[2]	(-2)/[7]

Los problemas de asignación: introducción

Son un caso particular del problema del transporte que destacan por sus aplicaciones y desde un punto de vista teórico son de sencilla formulación y disponen de un algoritmo eficiente para su resolución (el denominado **método húngaro**).

Permiten asignar eficientemente un conjunto de personas a un conjunto de trabajos, máquinas a tareas, coches de policía a sectores de una ciudad, vendedores a zonas, etc.

El objetivo es, según el problema, minimizar los costes, tiempos de desplazamiento, o maximizar la efectividad.

Es un modelo muy frecuente como submodelo en otros más complejos.

Supongamos que se dispone de n trabajadores y de n tareas a realizar de forma que cada trabajador sólo es capaz de realizar una única tarea y cada tarea debe ser realizada por un único trabajador. Supongamos conocido el “coste" c_{ij} de que sea el trabajador i el que realice la tarea j. Entonces, el *problema de asignación* consiste en determinar qué trabajador debe realizar cada tarea de manera que el coste total resulte lo menor posible.

El modelo matemático

Se introducen las variables de decisión:

$$X_{ij} = \begin{cases} 1 & \text{si el trabajador } i \text{ realiza la tarea } j \\ \\ 0 & \text{en otro caso} \end{cases},$$

resultando el siguiente modelo:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} X_{ij}$$

sujeto a:

$$\sum_{j=1}^{n} X_{ij} = 1 \text{ para todo } i = 1, \cdots, n$$

$$\sum_{i=1}^{n} X_{ij} = 1 \text{ para todo } j = 1, \cdots, n$$

$$X_{ij} \geq 0 \text{ para todo } i = 1, \cdots, n, j = 1, \cdots, n.$$

Las variables X_{ij} serán enteras en una solución extrema óptima, sin necesidad de imponerlo explícitamente en el modelo matemático.

Resolución informática de un problema de asignación

Un supercomputador con 4 ordenadores diferentes (O_j) ha de procesar 4 tareas (T_i). Todas se pueden realizar en cualquier ordenador, pero se deben completar en el ordenador en que se inician. Además, se quieren distribuir de forma que cada ordenador va a ser utilizado para realizar una única tarea[1]. Los tiempos estimados de procesamiento son los siguientes:

Tarea/Ordenador	O_1	O_2	O_3	O_4
T_1	18	16	12	10
T_2	14	21	19	12
T_3	23	27	33	28
T_4	16	24	23	32

Decídase a qué ordenador mandar cada tarea para minimizar el tiempo total de procesamiento. Lo resolvemos con Solver o con Maxima.

[1] Otro problema es suponer que un ordenador puede realizar varias tareas sin superar un tiempo total y/o que las tareas se realizan en paralelo en varios procesadores

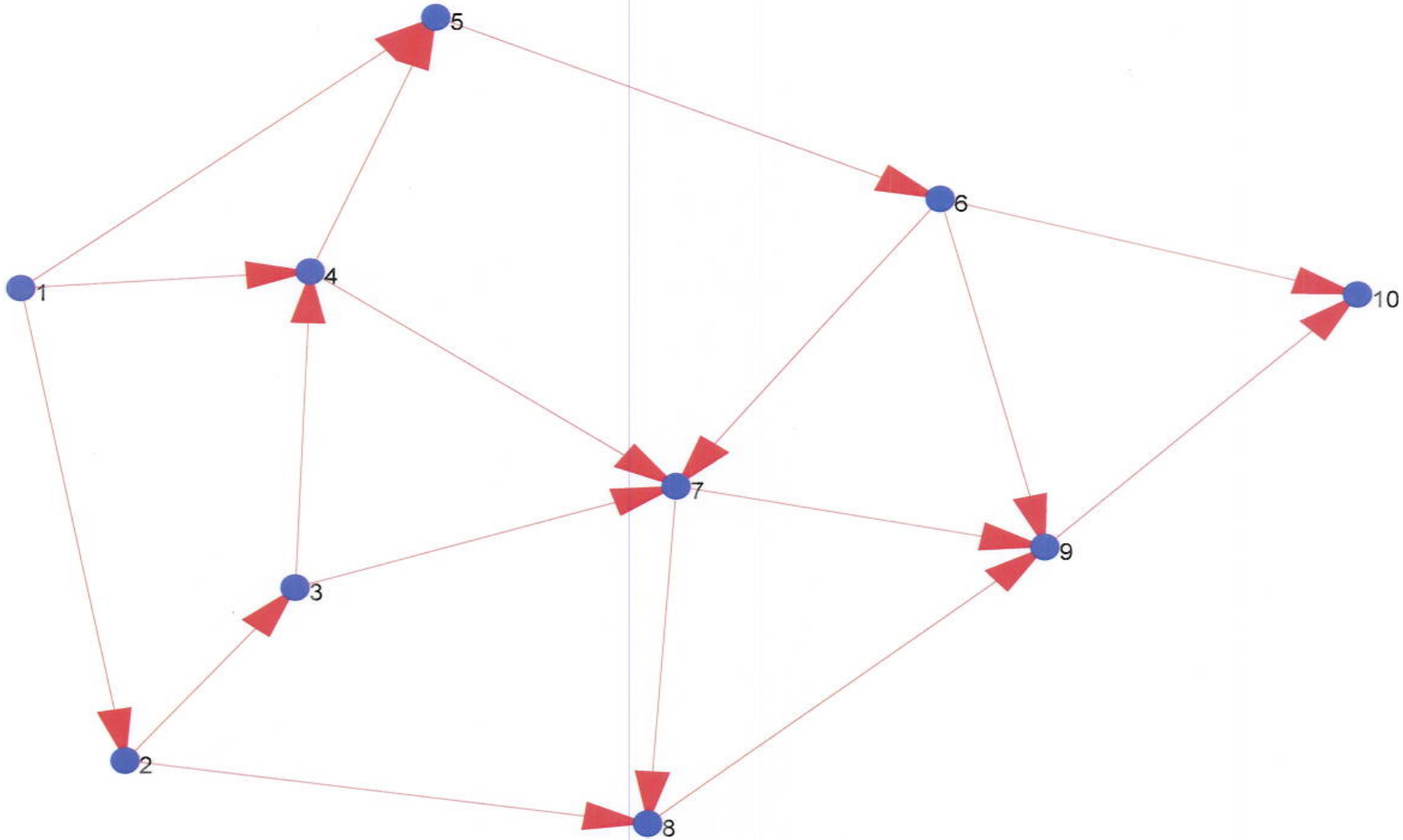
Problemas de flujo en redes con mínimo coste (Ferris, Mangasarian y Wright, 2007[2])

Los problemas de redes implican la optimización de un modelo de flujo en una red y son importantes por su aplicabilidad en diferentes problemas de la vida real.

Consideramos inicialmente una clase particular de problema en redes denominado *red de flujo de coste mínimo* en el que el "flujo" consiste en el movimiento de un cierto bien por los arcos de la red, desde nodos en los que el bien es producido a los nodos donde se consume. Si el coste de transporte del bien a lo largo del arco es un múltiplo fijo de la cantidad de bien, entonces el problema de minimizar el coste total puede ser formulado como un problema de programación lineal. Redes, como la representada en la siguiente figura, consisten en un conjunto N de nodos y un conjunto de arcos A tales que el arco (i, j) conecta el nodo origen i con el nodo destino j.

[2] Trae muchas aplicaciones interesantes de la programación matemática a la estadística: problemas de clasificación, de ajuste de un conjunto de observaciones a una curva, etc.

Nodos y arcos en una red



Asociado con cada nodo i hay una divergencia b_i que representa la cantidad de producto producido o consumido por el nodo i. Cuando $b_i > 0$, el nodo i es un *nodo de suministro*, mientras que cuando $b_i < 0$, el nodo i es un *nodo de demanda*. Asociado con cada arco (i, j) se tienen una cota inferior l_{ij} y una cota superior u_{ij} de la cantidad de bien que puede circular por el arco. Cada variable X_{ij} en el problema representa la cantidad de bien que se mueve a lo largo del arco (i, j). El coste de mover una unidad de flujo por el arco (i, j) es c_{ij}. Se quiere minimizar el coste total de mover el bien desde los nodos de suministro a los nodos de demanda. Con todos estos elementos se puede formular el problema de flujo en redes con coste mínimo como sigue:

$$\min_X z = \sum_{(i,j)\in A} c_{ij} X_{ij}$$

$$\text{sujeto a } \sum_{j:\,(i,j)\in A} X_{ij} - \sum_{j:\,(j,i)\in A} X_{ji} = b_i,\ \forall i \in N$$

$$l_{ij} \leq X_{ij} \leq u_{ij},\ \forall (i,j) \in A.$$

La primera restricción establece que el flujo neto a través de cada nodo debe coincider con su divergencia. La primera suma representa el flujo total que sale del nodo i, extendida a todos los arcos que tienen ese nodo como origen. La segunda suma representa el flujo total que entra en el nodo i, extendida a todos los arcos que tienen ese nodo como destino. La diferencia entre flujo entrante y saliente está restringido a la divergencia.

El problema del camino más corto

La estructura especial del modelo se puede usar para construir versiones especiales del símplex más eficientes. Nótese que los coeficientes de la matriz de restricciones contiene solamente los números 0, 1 y -1. Si todos los datos del problema son enteros, se puede probar que la solución X sólo tiene componentes enteros.

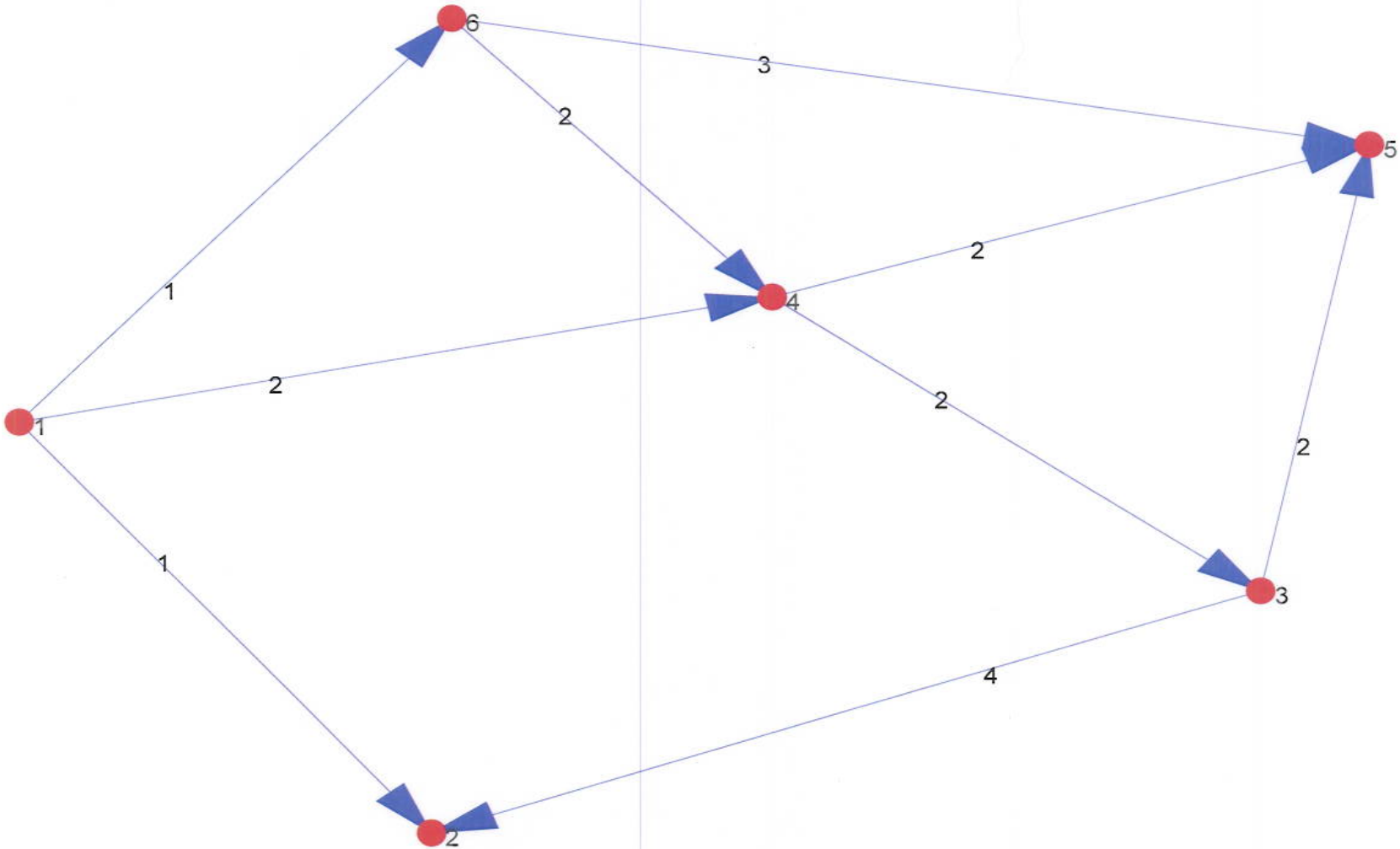
El problema de encontrar *el camino más corto* entre dos nodos dados de una red se formula fácilmente como un problema de flujo en redes a coste mínimo estableciendo los costes de los arcos como las distancias entre los nodos que conectan los arcos. Si se necesita conocer el camino más corto entre los nodos s y t, establecemos las divergencias como sigue:

$$b_s = 1, \quad b_t = -1, \quad b_i = 0, \quad \forall i \in N \backslash \{s, t\}.$$

Las cotas inferiores l_{ij} de los flujos se establecen a cero mientras que las cotas superiores u_{ij} pueden ser infinito.

En el ejemplo de la figura siguiente, donde las distancias c_{ij} se anotan al lado de cada arco, el camino más corto desde el nodo $s = 1$ hasta el nodo $t = 5$ es 1, 6, 5 con un coste de 4. Otra solución con la misma distancia es 1, 4, 5. Si se aplica el método del símplex a esta formulación del problema del camino más corto, y la solución es única, la solución se puede reconocer fácilmente como la consistente en aquellos arcos en los cuales el flujo óptimo es 1.

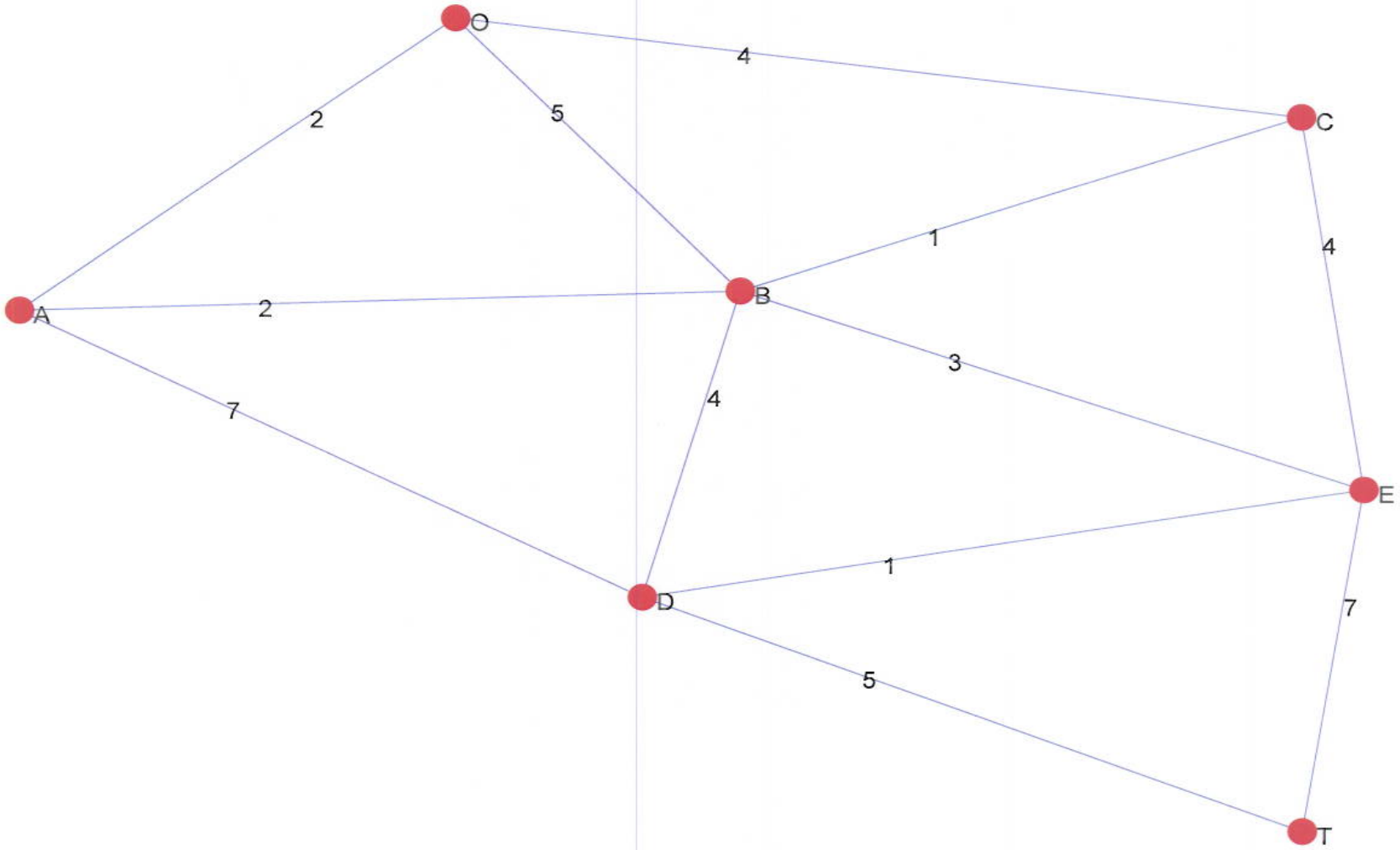
Red con longitudes de arcos



Ejemplo de problema de camino más corto: resolución informática (Hillier y Lieberman, 2005)

Se reservó el área de SEERVADA PARK para paseos y campamentos. No se permite la entrada de automóviles, pero existe un sistema de caminos angostos y sinuosos para camionetas y “jeeps" conducidos por los guardabosques. En la siguiente figura se muestra este sistema de caminos (sin las curvas), en donde O es la entrada del parque y las otras letras representan la localización de las casetas de los guardabosques y otras instalaciones de servicio. Los números son las distancias en millas de estos caminos. El parque tiene un mirador a un paisaje en la estación T. Unas cuantas camionetas transportan a los visitantes desde la entrada a la estación T y viceversa.

Sistema de caminos del Seervada Park



Uno de los problemas a los que se enfrenta la administración del parque consiste en determinar qué ruta, desde la entrada del parque a la estación T, es la que tiene la *distancia total más corta* para la operación de las camionetas. Lo resolvemos con Solver.

El problema del árbol de expansión mínima

Supongamos que tenemos una red (N, A) con N el conjunto de nodos y A el conjunto de arcos, y unos costes c_{ij} asociados a cada arco (i, j) de A. Se quiere determinar el conjunto T de arcos de la red que une todos los nodos, de manera que la suma de las longitudes de los arcos sea mínima.

Algunas aplicaciones:

1. Diseño de redes de telecomunicaciones: de fibra óptica, de ordenadores, telefónicas, de televisión por cable, etc.
2. Diseño de redes de transporte: vías ferroviarias, carreteras, etc.
3. Diseño de una red de líneas de transmisión de energía eléctrica de alto voltaje.
4. Diseño de una red de tuberías para conectar varias localidades.

Consideremos un *grafo no orientado* (N, A). Un grafo (N', A') se dice *subgrafo* de (N, A) cuando $N' \subseteq N$ y $A' \subseteq A$. Dos nodos $i, j \in N$ se dicen *conectados* si y sólo si existe un camino en el grafo que los tiene como extremos. Dado un nodo $i \in N$, define una *componente conexa* del grafo si consideramos el conjunto de nodos conectados con el nodo i. El grafo se dice *conexo* si contiene una única componente conexa. Un subgrafo (N', A') se dice que es un árbol cuando $\mid A' \mid = \mid N' \mid - 1$ y es conexo. El árbol se dice *generador* de (N, A) cuando es un árbol de (N, A) y no existe otro árbol de (N'', A'') con $N' \subseteq N''$ y $A' \subseteq A''$.

Si en el grafo no orientado (N, A) todos los arcos tienen asociados un coste, el *problema del árbol de expansión mínima* consiste en encontrar un árbol generador (N, T) en (N, A) tal que la suma de los costes de los arcos de T sea mínima.

PROPOSICIÓN Las siguientes afirmaciones son equivalentes:

1. (N, T) es un árbol generador.
2. Cada par de nodos en (N, T) están unidos por un único camino.
3. (N, T) es conexo y $\mid T \mid = n - 1$.
4. (N, T) no tiene ciclos y $\mid T \mid = n - 1$.
5. (N, T) es conexo pero $(N, T \backslash \{(i, j)\})$ no lo es, para cualquier $(i, j) \in T$.
6. (N, T) no tiene ciclos pero $(N, T \cup \{(i, j)\})$ tiene un ciclo para cualquier $(i, j) \notin T$.

Un modelo matemático para el problema del árbol de expansión mínima

Se introducen las variables de decisión:

$$X_{ij} = \begin{cases} 1 & \text{si el arco } (i,j) \in T \\ \\ 0 & \text{en otro caso} \end{cases} .$$

Y el modelo de programación lineal se define como:

$$\min_X z = \sum_{(i,j)\in A} c_{ij} X_{ij}$$

$$\text{sujeto a } \sum_{(i,j)\in A} X_{ij} = n - 1$$

$$\sum_{(i,j)\in A(S)} X_{ij} \leq \mid S \mid -1, \ \forall S \subset N.$$

La primera ecuación, en las restricciones anteriores, impone que el número de arcos elegidos debe de ser minimal, mientas que la familia de desigualdades imponen que los arcos elegidos no pueden formar ningún ciclo.

A pesar de que el problema presenta un número exponencial de restricciones cuenta con algoritmos eficientes que lo resuelven con una complejidad de tipo polinomial. Esto muestra que la dificultad intrínseca del problema puede no depender del número de desigualdades que aparezcan en su modelo.

La Proposición 8.4.2 en Salazar González (2000) proporciona una caracterización de la optimalidad para problemas del árbol de expansión mínima. A partir de ese resultado se pueden diseñar diferentes algoritmos que encuentran un arbol de expansión mínima. Uno de ellos es el siguiente algoritmo.

PROPOSICIÓN Sea (N, A) un grafo no dirigido conexo. Consideremos $(N_1, T_1), (N_2, T_2), \cdots, (N_k, T_k)$ árboles en (N, A) con $N = \cup_{i=1}^{k} N_i$ y $N_i \cap N_j = \emptyset$ para todo $i \neq j$. Si consideramos e^* un arco que conecta un nodo de S con un nodo de $N \backslash S$ para S unión de algunos N_i y tal que el coste de e^* es el mínimo de los costes de todos los arcos que unen un nodo de S con un nodo de $N \backslash S$, entonces existe un árbol generador (N, T^*) de coste mínimo en (N, A) tal que $\cup_{i=1}^{k} T_i \subset T^*$ y $e^* \in T^*$.

Demostración Supongamos por reducción al absurdo que no es así, y que por tanto existe un árbol (N, T') tal que:

- $\cup_{i=1}^{k} T_i \subset T'$,
- $e^* \notin T'$,
- el coste de (N, T') es menor que el coste de cualquier árbol (N, T) con $\cup_{i=1}^{k} T_i \subset T$ y $e^* \in T$.

Entonces $(N, T' \cup \{e^*\})$ tiene un único ciclo (N, C), quien a su vez contiene un arco $e' \in T'$ distinto de e^*, y por tanto el coste de e' es mayor o igual que el coste de e^*. Consecuentemente $(N, T' \setminus \{e'\} \cup \{e^*\})$ contradice la hipótesis. $\square$

Algoritmo de Kruskal

Paso 1

Elegimos arbitrariamente un nodo i de la red.

Sea $X = \{i\}$, $\overline{X} = N - X$.

Elegir el nodo $j \in \overline{X}$ “más próximo" a i.

Unir estos nodos.

Hacer $X = X \cup \{j\}$, $\overline{X} = N - X$

(El arco (i, j) estará en el árbol de expansión mínima.)

Paso 2

Elegir el nodo $l \in \overline{X}$ “más próximo" a X:

Sea c_{lk} el mínimo de las distancias del nodo l a los nodos de X.

Entonces unir los nodos l y k y hacer $X = X \cup \{l\}$, $\overline{X} = N - X$.

(El arco (l, k) estará en el árbol de expansión mínima.)

Repetir el paso 2 hasta que todos los nodos estén unidos ($\overline{X} = \emptyset$).

EJEMPLO

En un campus universitario hay cinco ordenadores en edificios diferentes y el objetivo es conectarlos mediante un cable subterráneo. En la gráfica se dan las distancias entre cada par de ordenadores.

Se quiere determinar cuál es la mínima longitud de cable que se necesita.

Lo que tenemos que hacer es obtener el árbol de expansión mínima.

Para empezar se escoge de manera arbitraria el nodo 1.

$X = \{1\}$

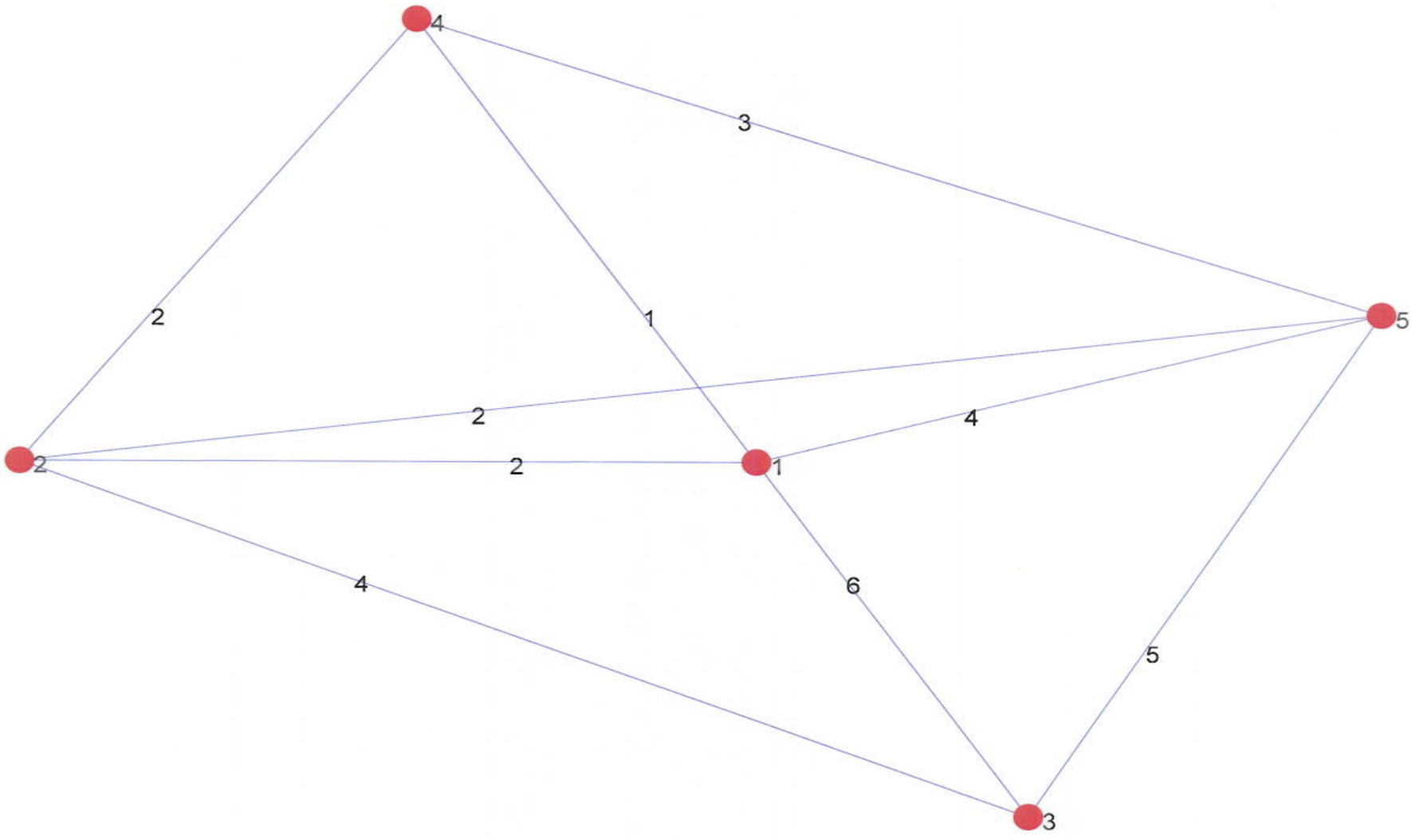
$\overline{X} = \{2, 3, 4, 5\}$

mín $\{c_{ij}/j \in \overline{X},\ (1, j) \in A\} = min\{c_{14}, c_{15}, c_{12}, c_{13}\} = 1 = c_{14}$,

es decir, el nodo 4 es el nodo más próximo al nodo 1, entonces el arco (1,4) estará en el árbol de expansión mínima, y además:

$X = \{1, 4\},\ \ \overline{X} = \{2, 3, 5\}$

Grafo de las distancias entre ordenadores



mín$\{c_{ij}/i \in X, j \in \overline{X}, (i,j) \in A\} = min\{c_{15}, c_{12}, c_{13}, c_{42}, c_{45}\} = 2 = c_{42}$

entonces el arco (4,2) estará en el árbol de expansión mínima y, además:

$X = \{1,2,4\}$, $\overline{X} = \{3,5\}$.

mín$\{c_{ij}/i \in X, j \in \overline{X}, (i,j) \in A\} = min\{c_{13}, c_{15}, c_{23}, c_{25}, c_{45}\} = 2 = c_{25}$

entonces se incluye el arco (2,5) en el árbol de expansión mínima y, además:

$X = \{1,2,4,5\}$, $\overline{X} = \{3\}$.

mín$\{c_{ij}/i \in X, j \in \overline{X}, (i,j) \in A\} = min\{c_{13}, c_{23}, c_{53}\} = 4 = c_{23}$

entonces se añade el arco (2,3) al árbol de expansión mínima y, además:

$X = \{1,2,3,4,5\}$, $\overline{X} = \{\emptyset\} \quad \Longrightarrow$ FIN

Ya hemos obtenido el árbol de expansión mínima:

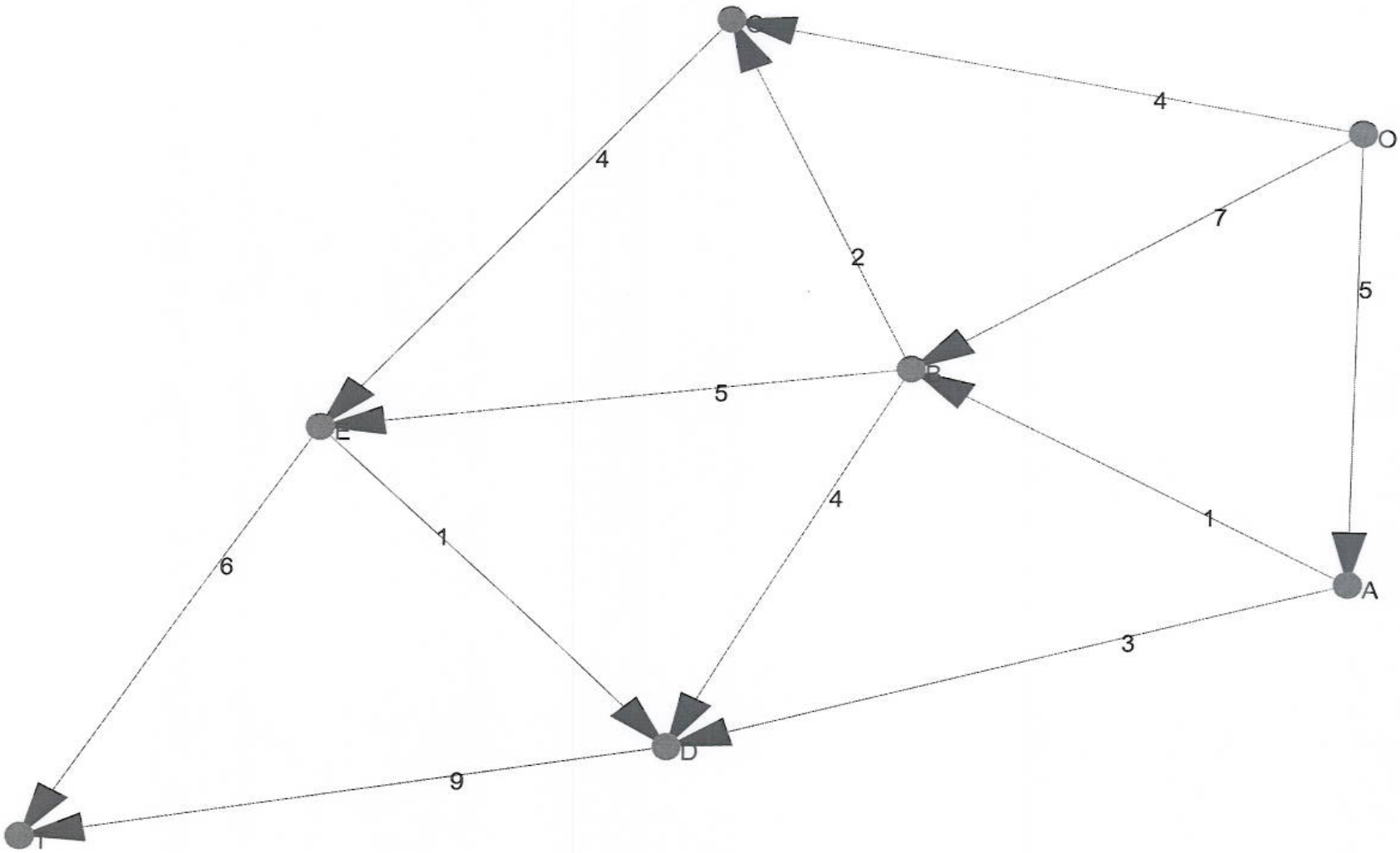
(1,4), (4,2), (2,5), (2,3)

Y la longitud (mínima) es 1+2+2+4=9.

Un ejemplo de problema de flujo máximo para resolver informáticamente con Solver

Retomamos el caso del Seervada Park. Uno de sus problemas se refiere a que, durante la temporada alta, hay más personas que quieren tomar la camioneta a la estación T que aquéllas a las que se les puede dar servicio. Para evitar la perturbación indebida de la ecología y de la vida silvestre de la región, se ha impuesto un racionamiento estricto al número de viajes al día que pueden hacer las camionetas en cada camino, límites que difieren entre los caminos y que podemos ver en la siguiente figura. De esta forma, durante la temporada alta, se pueden seguir varias rutas, sin tener en cuenta la distancia, para aumentar el número de viajes de camionetas diarios. La cuestión es cómo planificar las rutas para los distintos viajes, de manera que se *maximice* el número total de viajes que se pueden hacer al día, sin superar los límites impuestos sobre cada camino.

Sistema de caminos del Seervada Park y capacidades



Algunas aplicaciones del problema del flujo máximo

- Maximizar el flujo a través de la red de distribución de una empresa desde sus fábricas hasta sus clientes.
- Maximizar el flujo a través de la red de distribución de una empresa desde sus proveedores hasta sus fábricas.
- Maximizar el flujo de petróleo por un sistema de tuberías.
- Maximizar el flujo de vehículos por una red de transporte.

El modelo de programación lineal para el problema de flujo máximo desde s hasta t

Tenemos una red de comunicaciones y deseamos enviar la mayor cantidad de bien desde un nodo "fuente" a otro nodo "sumidero" utilizando las conexiones de la red pero sin exceder ciertos límites de tráfico que tienen tales conexiones. Se considera entonces una variable asociada a cada arco, X_{ij}, representando el flujo asociado a dicho arco y una única variable, Z, que representa la capacidad de dicho flujo.

Problema primal:

maximizar Z

$$\text{sujeto a } \sum_{j=1}^{n} X_{ij} - \sum_{k=1}^{n} X_{ki} = \begin{cases} Z & \text{si} \quad i = s \\ 0 & \text{si} \quad i \neq s \text{ ó } t \\ -Z & \text{si} \quad i = t \end{cases}$$

$$l_{ij} \leq X_{ij} \leq u_{ij}, \ \forall (i,j)$$

En el modelo anterior, las restricciones de igualdad se conocen con el nombre de

“ecuaciones de conservación de flujo".

Las restricciones de desigualdad se conocen con el nombre de

“restricciones de capacidad".

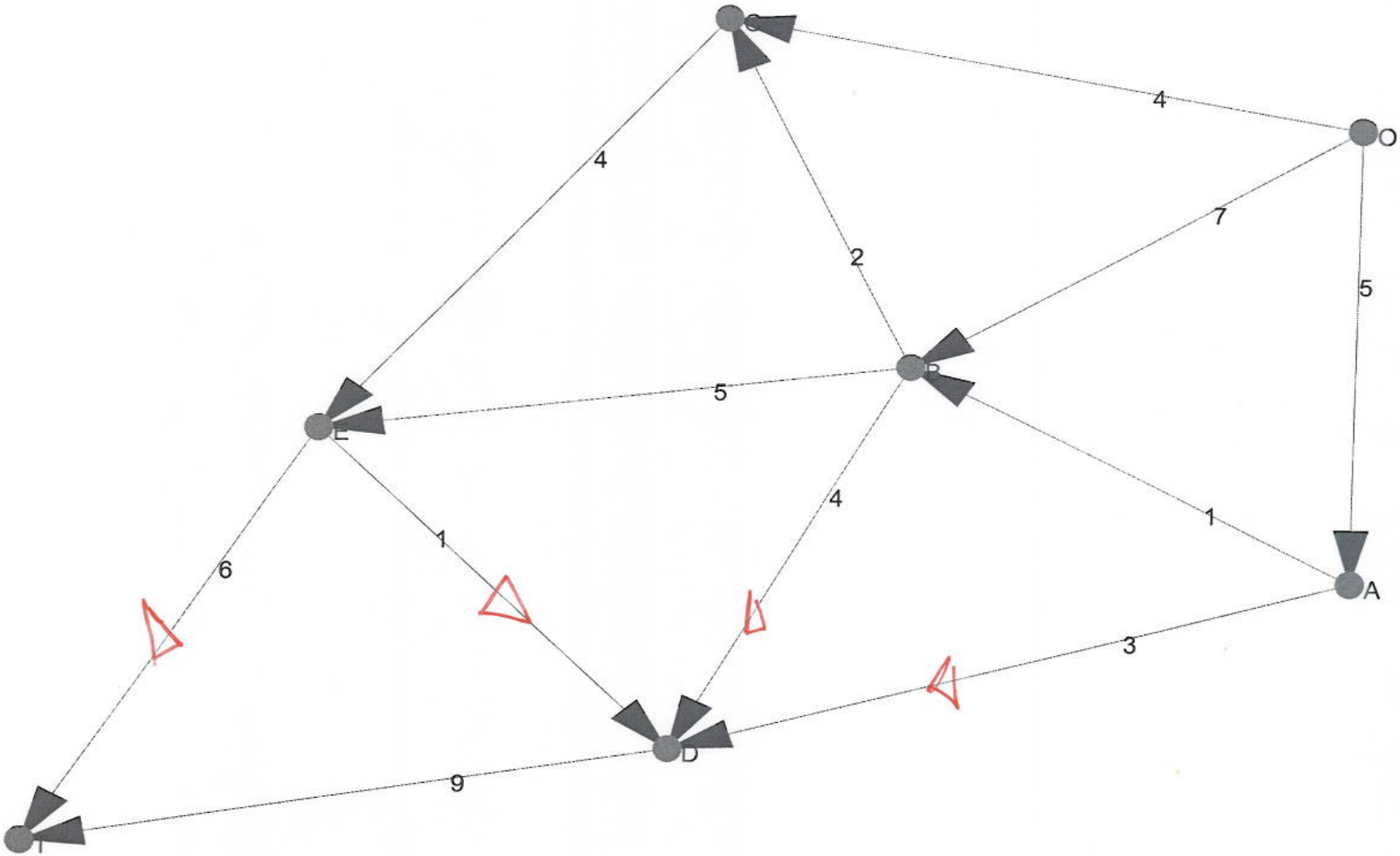
Una **trayectoria dirigida** del nodo *i* al nodo *j* es una sucesión de arcos cuya dirección (si la tienen) es hacia el nodo *j*, de manera que el flujo del nodo *i* al nodo *j* a través de esta trayectoria es factible.

Un **corte** se puede definir como cualquier conjunto de arcos dirigidos que contienen al menos un arco de cada trayectoria dirigida que va del nodo origen al nodo destino. En general, hay muchas formas de dividir una red para formar un corte que ayude a analizarla. El **valor del corte** es la suma de las capacidades de los arcos (en la dirección especificada) del corte.

TEOREMA DE FLUJO-MÁXIMO CORTE-MÍNIMO Para cualquier red con un solo nodo origen y un solo nodo destino, cualquier flujo factible del origen al destino es igual o menor al valor de cualquiera de los cortes de la red.

En el ejemplo de Seervada Park, consideramos el corte reflejado en la siguiente figura. Su valor es, que coincide con el flujo asignado en la última iteración del algoritmo. Por tanto, se trata de un corte mínimo y 14 es el flujo máximo factible.

Ejemplo de corte en la red de Seervada Park



El arco más crítico de una red

Una cuestión interesante en el contexto del problema del flujo máximo se refiere al arco más crítico de una red, esto es, aquél que, al ser suprimido, se reduce el flujo máximo que puede atravesar la red resultante en la cantidad mayor.

MODO DE ENCONTRARLO Una vez que hemos obtenido Z', *flujo máximo*, tendremos también $(X, \overline{X})$, *conjunto de corte de capacidad mínima*.

Entonces buscamos el arco $k_0 = (i_0, j_0) \in (X, \overline{X})$ tal que el flujo que pasa por él, z_{k_0}, sea el máximo de los flujos que pasan por los arcos $k = (i, j) \in (X, \overline{X})$: $z_{k_0} = \max\{z_k / k \in (X, \overline{X})\}$.

Ahora buscamos los arcos $k = (i, j) \notin (X, \overline{X})$ de la red (N, A) por los que pasa un flujo mayor que z_{k_0}.

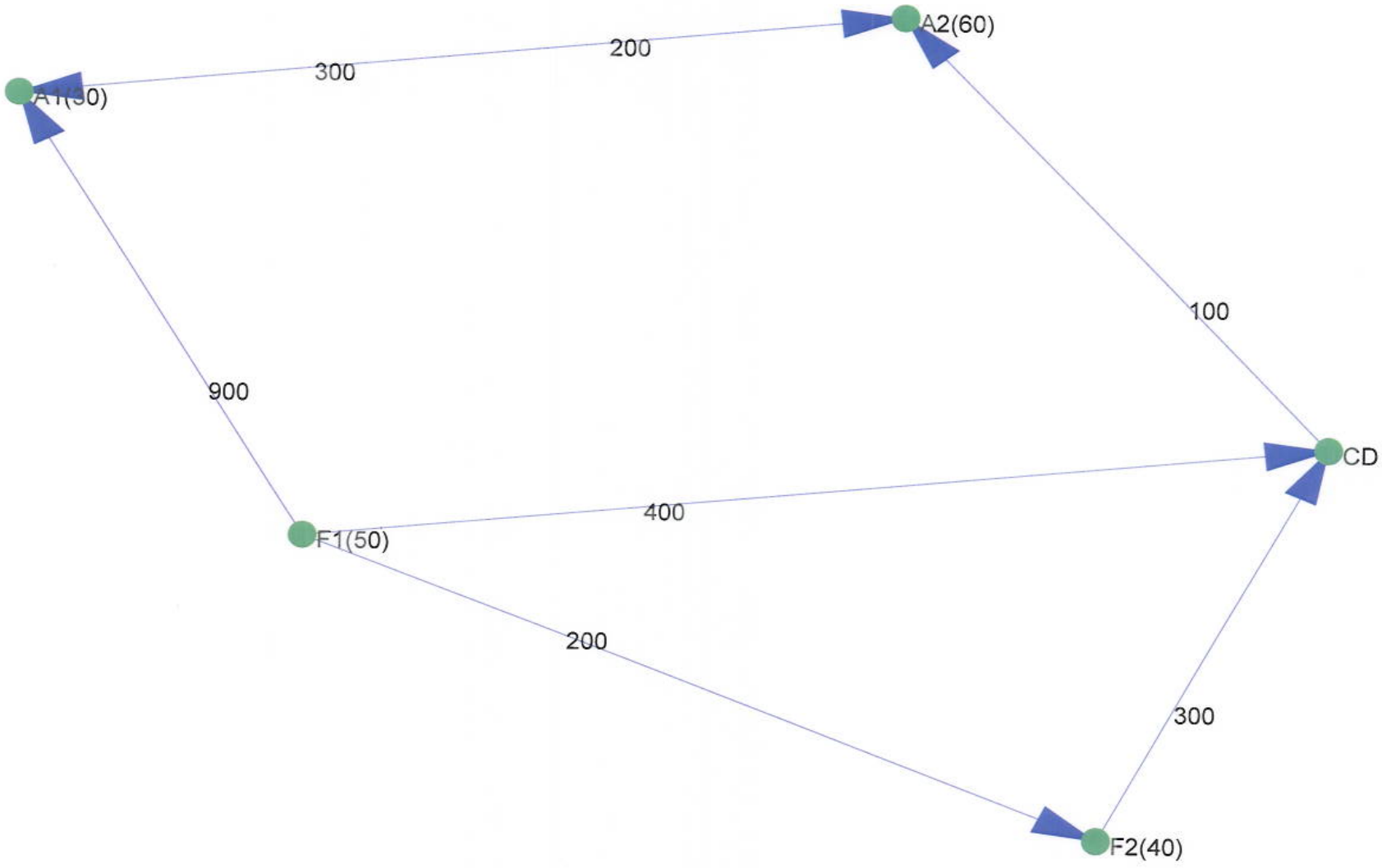
A continuación, calcularemos el flujo máximo que pasa por la red obtenida al eliminar dichos arcos k=(i,j) (uno de cada vez, es decir, repetiremos el algoritmo tantas veces como arcos haya en esta condiciones).

El arco más crítico será el que, al eliminarlo, el flujo se reduce en la cantidad mayor.

Un ejemplo de problema de flujo de coste mínimo para resolver informáticamente con Solver

La Distribution Unlimited Co. fabricará el mismo nuevo producto en dos plantas distintas y después tendrá que enviarlo a dos almacenes de distribución, donde cualquiera de las dos fábricas puede abastecer a cualquiera de los dos almacenes. La red de distribución disponible para el envío de este producto se muestra en la siguiente figura, donde $F1$ y $F2$ son las dos fábricas, $A1$ y $A2$ son los dos almacenes y CD es el centro de distribución. Se muestran los números de unidades que deben enviarse desde $F1$ y $F2$ y los que deben recibirse en $A1$ y $A2$. Cada flecha representa un canal factible de envío, y se proporciona el coste en dolares por unidad enviada. Se quiere saber qué cantidades enviar a través de cada canal con el objetivo de minimizar el coste total de envío. Los canales $F1 \rightarrow F2$ y $CD \rightarrow A2$ tienen unas capacidades máximas de 10 y 80 unidades respectivamente. Formular como un problema de programación lineal.

Red de distribución de Distribution Unlimited Co.



Planificación de horarios

El objetivo de este problema es planificar la asignación de aulas y horarios de asignaturas en un centro académico de forma tal que el horario sea lo más compacto posible.

Se dispone de n_c aulas y de n_h horas para impartir n_s asignaturas. Estas asignaturas se distribuyen en n_b cursos y por profesores, donde n_i denota el número de asignaturas que imparte el profesor i.

Se denota por Ω_i al conjunto de asignaturas que imparte el profesor i y por Λ_b al conjunto de asignaturas del curso b.

Se define la variable $X_{sch} = 1$ si la asignatura s se imparte en la clase c y a la hora h, y en caso contrario $X_{sch} = 0$.

El objetivo es:

Minimizar $\sum_s \sum_c \sum_h (c + h) X_{sch}$

Restricciones del problema de planificación de horarios

1. $\sum_{s \in \Omega_i} \sum_{c} \sum_{h} X_{sch} = n_i \ \forall i$
2. $\sum_{s \in \Omega_i} \sum_{c} X_{sch} \leq 1 \ \forall h \text{ y } \forall i$
3. $\sum_{c} \sum_{h} X_{sch} = 1 \ \forall s$
4. $\sum_{s} X_{sch} \leq 1 \ \forall c \text{ y } \forall h$
5. $\sum_{s \in \Lambda_b} \sum_{c} X_{sch} \leq 1 \ \forall h \text{ y } \forall b$
6. $X_{sch} \in \{0, 1\} \ \forall s, \ \forall c \text{ y } \forall h.$

Interpretación del objetivo y las restricciones del problema de planificación de horarios

- El objetivo penaliza el que las variables X_{sch} tomen el valor 1 para valores elevados de c y h, por lo que se compacta el horario. Pueden existir otros objetivos.
- La restricción 1 obliga a que cada profesor imparta todas sus asignaturas.
- La restricción 2 no permite que un profesor imparta más de una asignatura por hora.
- La restricción 3 no permite que una asignatura se imparta más de una vez.
- La restricción 4 obliga a que en cada clase y hora se imparta como mucho una sola asignatura.
- La restricción 5 indica que, en cada hora, se imparta como mucho una asignatura de cada curso.

Optimización Combinatoria

La Optimización Combinatoria es una parte de la Programación Matemática que estudia problemas del tipo

$$\min\{f(x) \,:\, x \in S\},$$

siendo $|S| < \infty$, es decir, la Optimización Combinatoria trata de desarrollar algoritmos para afrontar problemas de optimización caracterizados por tener un número finito de soluciones factibles.

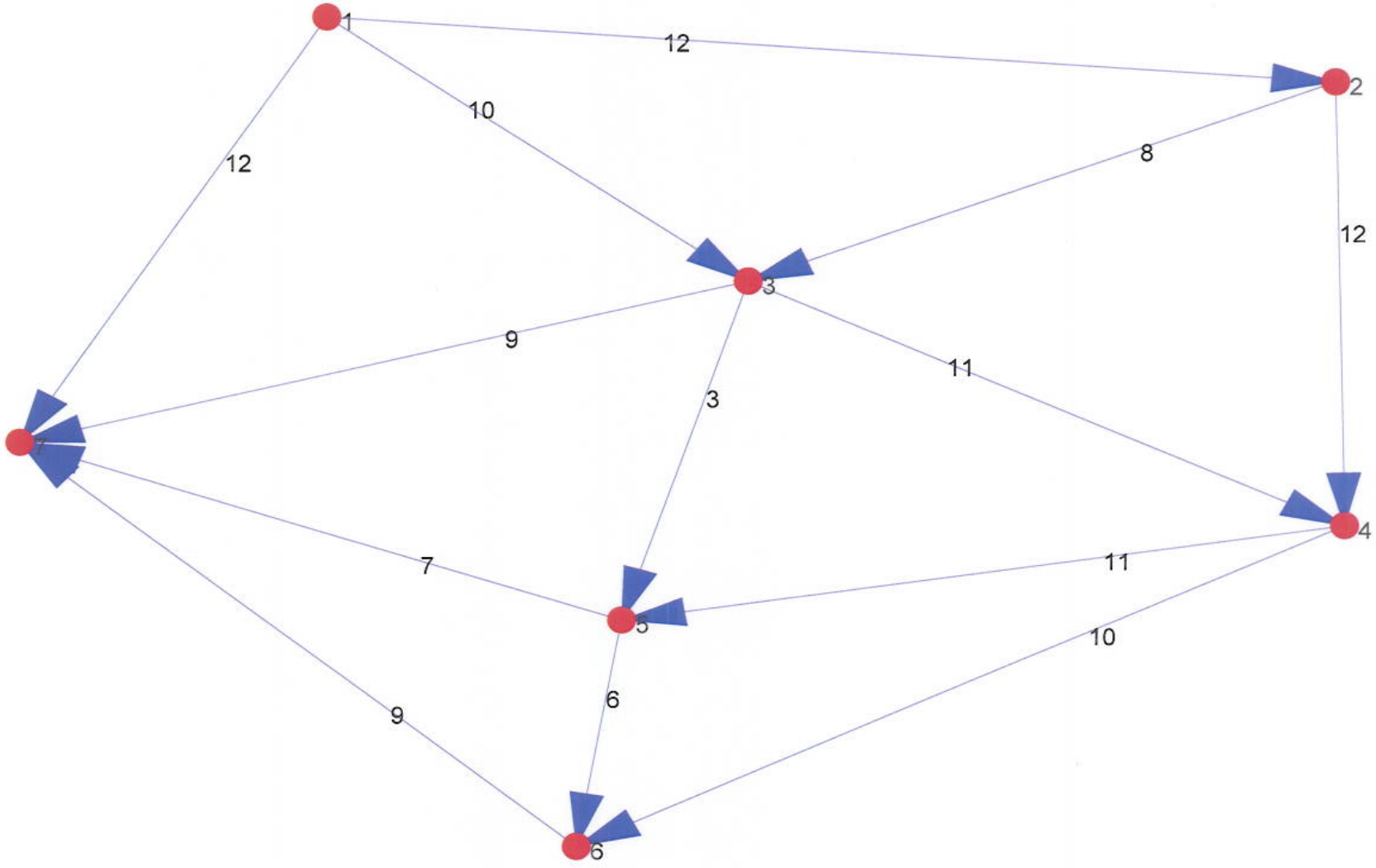
Aunque por definición puede parecer que la “simple" enumeración de las soluciones en X es suficiente para resolverlos, esta idea no es realizable en la práctica ya que el número $|S|$ puede ser exageradamente grande. Por otra parte, particularidades del conjunto S permiten en ocasionas construir algoritmos efectivos que afrontan dichos problemas.

Problema del viajante de comercio

Algunos problemas de Optimización Combinatoria han sido tratados anteriormente, como el del árbol de expansión mínima. Pero tal vez el problema clásico de optimización combinatoria más famoso es conocido como el *problema del viajante de comercio* (TSP en la literatura, esto es, *traveling salesman problem*). Recibió este nombre porque puede describirse en términos de un agente de ventas que debe visitar cierta cantidad de ciudades en un solo viaje. Si comienza desde su ciudad de residencia, el agente debe determinar qué ruta debe seguir para visitar cada ciudad exactamente una vez antes de regresar a su casa de manera que se minimice la longitud total del viaje. Esto es, debe construir un *ciclo Hamiltoniano* de coste mínimo.

La siguiente figura muestra un problema del agente viajero con siete ciudades. La ciudad 1 es el lugar de residencia del agente.

Ejemplo del problema del viajante de comercio



Por lo tanto, si comienza desde su ciudad, el agente debe elegir una ruta para visitar cada una de las otras ciudades exactamente una vez antes de regresar a su punto de partida. El número colocado junto a la ligadura entre cada par de ciudades representa la distancia (el costo o el tiempo) entre estas ciudades. Se supone que la distancia es la misma en cualquier dirección (problema *simétrico*). Aunque por lo general existe una ligadura directa entre cada par de ciudades, en este caso el ejemplo se simplifica al suponer que las únicas ligaduras directas son las que muestra la figura.

El objetivo es determinar qué ruta minimizará la distancia total que el agente viajero debe recorrer.

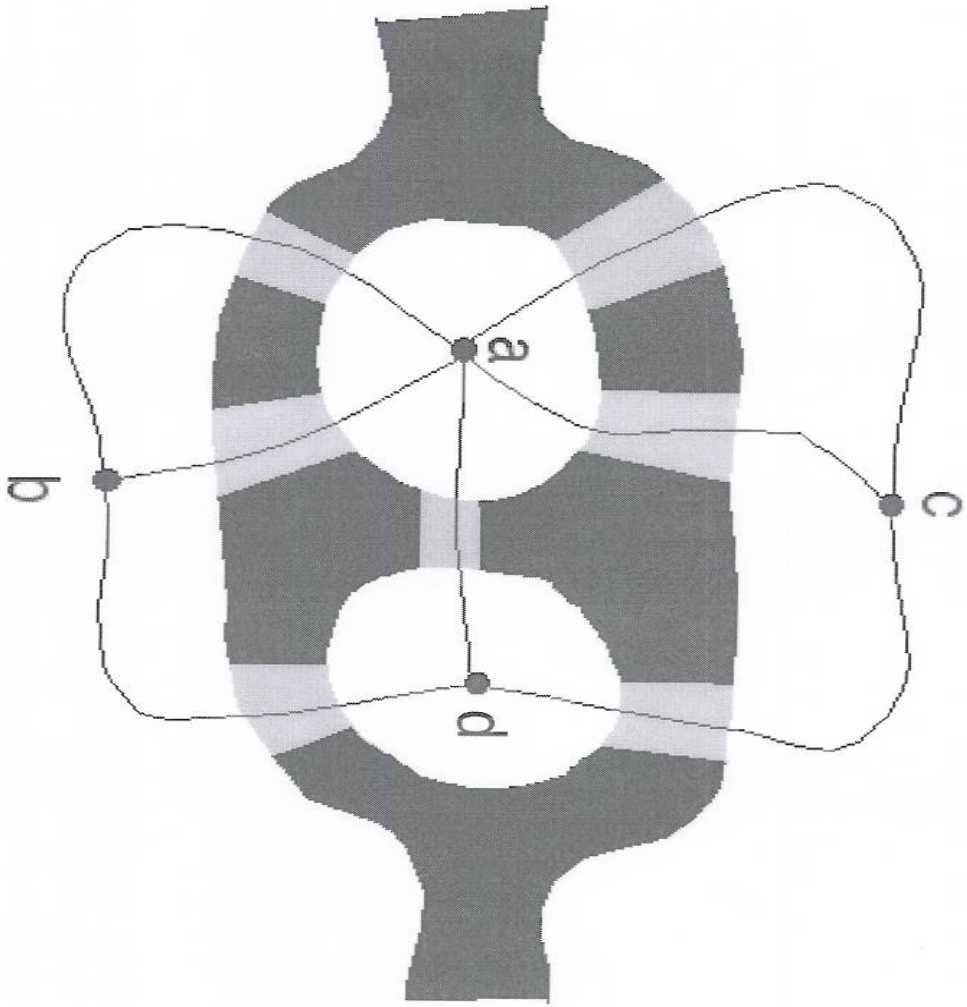
Existen distintas aplicaciones del problema del agente viajero a problemas que no tienen nada que ver con un agente de ventas.

Por ejemplo, cuando un camión sale de un centro de distribución para entregar bienes a cierta cantidad de ubicaciones, el problema de determinar la ruta más corta es un problema del agente viajero. Otro ejemplo es el de la fabricación de tableros de circuitos impresos, semiconductores cableados y otros componentes. Cuando es necesario perforar muchos orificios en un tablero de circuitos impresos, el problema de encontrar la secuencia de perforaciones más eficiente constituye un problema del agente viajero.

La complejidad del problema del agente viajero se incrementa con rapidez a medida que aumenta el número de ciudades.

El problema de determinar un camino Hamiltoniano (que conecta todos los nodos una vez y sólo una) difiere del de la construcción de un camino Euleriano: que contiene una y sólo una vez todos los arcos del grafo. Euler (s. XVIII) estudió y caracterizó los grafos que contienen un ciclo Euleriano motivado por el *problema de los 7 puentes de Königsberg*: conociendo que hay dos islas en el río que atraviesa la ciudad, que hay un puente entre ellas, uno que conecta la isla menor con cada orilla, y dos que conectan la isla mayor con cada orilla, se quiere saber si hay alguna manera de organizar un paseo por todos los puentes sin que se repita alguno. Probó que el problema no tiene solución.

El problema de los 7 puentes de Königsberg



En el caso de un problema con n ciudades y una ligadura entre cada par de ciudades, el número de rutas factibles que debe considerarse es $(n-1)!/2$ puesto que hay $(n-1)$ posibilidades para la primera ciudad después de la ciudad de residencia del agente, $(n-2)$ posibilidades para la siguiente ciudad y así sucesivamente. El denominador 2 surge porque cada ruta tiene una ruta inversa equivalente con la misma distancia. En consecuencia, mientras un problema del agente viajero con 10 ciudades tiene menos de 200.000 soluciones factibles que deben ser consideradas, un problema con 20 ciudades tiene alrededor de 10^{16} soluciones factibles, mientras que un problema con 50 ciudades tiene alrededor de 10^{62}.

Algunos algoritmos basados en el enfoque de ramificación y acotamiento como los vistos al estudiar programación entera han tenido éxito en resolver ciertos problemas del agente viajero con cientos o incluso miles de ciudades.

Modelo matemático del viajante de comercio

Consideremos una variable decisional X_{ij}, asociada a cada posible arco (i, j), representando:

$$X_{ij} := \begin{cases} 1 & \text{si el arco } (i,j) \text{ forma parte del circuito Hamiltoniano} \\ 0 & \text{en otro caso.} \end{cases}$$

Entonces un modelo de Programación Lineal Entera del TSP es :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} X_{ij}$$

sujeto a:

- $\sum_{j=1}^{n} X_{ij} = 1$ para todo $i = 1, \cdots, n$
- $\sum_{j=1}^{n} X_{ji} = 1$ para todo $i = 1, \cdots, n$
- $\sum_{(i,j) \in A(S)} X_{ij} \leq |S| - 1$ para todo $S \subseteq N$ tal que $1 \notin S$
- $0 \leq X_{ij} \leq 1$ para todo $(i, j) \in A$
- $X_{ij} \in \mathbb{Z}$ para todo $(i, j) \in A$

Las dos primeras familias de restricciones imponen que todo nodo es incidente a dos arcos con variable correspondiente de valor 1. La tercera familia de restricciones impide soluciones que consistan en uniones de ciclos disjuntos.

Definición

Consideremos dado un conjunto de objetos $\{1, \cdots, n\}$, cada uno j caracterizado por un peso p_j y un valor v_j, siendo tales números enteros positivos. Consideremos dado también un contenedor (por ejemplo, una mochila) caracterizada por poder contener objetos cuyo peso total no exceda de p_0 unidades.

Entonces se llama *Problema de la Mochila* (del inglés, *Knapsack Problem*: KP) al problema de determinar qué objetos conviene introducir en el contenedor de manera que el peso total de los objetos contenidos no supere el límite impuesto por p_0, y que al tiempo el valor total de lo contenido sea máximo.

Obviamente, se suele asumir que $p_j \leq p_0$ para todo j (ya que de lo contrario conviene no introducir j en el contenedor) y que se verifica $\sum_{j=1}^{n} p_j > p_0$ (ya que de lo contrario conviene introducir a todos los objetos en el contenedor).

Un modelo matemático

La motivación para el estudio de este problema es porque aparece en numerosas aplicaciones donde hay que seleccionar un conjunto de tareas en presencia de recursos limitados.

Consideremos una variable decisional X_j, asociada a cada objeto j, representando:

$$X_j := \begin{cases} 1 & \text{si el objeto } j \text{ debe introducirse} \\ 0 & \text{en otro caso.} \end{cases}$$

Entonces un modelo de Programación Lineal Entera para KP es :

$$\max \sum_{j=1}^{n} v_j X_j$$

sujeto a:

Un algoritmo

$$\sum_{j=1}^{n} p_j X_j \leq p_0$$

$$0 \leq X_j \leq 1 \qquad \text{para todo } j \in \{1, \cdots, n\}$$

$$X_j \in \mathbb{Z} \qquad \text{para todo } j \in \{1, \cdots, n\}$$

Una primera técnica es usar ramificación y acotación aprovechando el modelo de Programación Entera anterior. Un problema relajado obvio para obtener las cotas (superiores, al ser el objetivo de maximizar) proviene de la relajación lineal del modelo, esto es, de resolver el modelo anterior eliminando la última familia de restricciones. Observemos que este último problema lineal es fácilmente resoluble ordenando los objetos en orden decreciente de los cocientes valores entre pesos, esto es:

$$\frac{v_1}{p_1} \geq \cdots \geq \frac{v_n}{p_n},$$

y llamemos *objeto crítico* a aquel j' tal que

$$\sum_{j=1}^{j'-1} p_j < p_0 \leq \sum_{j=1}^{j'} p_j.$$

Entonces una solución óptima de la relajación lineal continua viene dada por:

$$X_j := \begin{cases} 1 & \text{si } j < j' \\ (p_0 - \sum_{k=1}^{j'-1} p_k)/p_{j'} & \text{si } j = j' \\ 0 & \text{si } j > j' \end{cases}$$

Ejemplo

En efecto, la solución de la relajación lineal elige los objetos como divididos en trozos, y considera aquellos trozos con mayor valor por unidad de peso. Esto muestra que no hace falta usar un algoritmo propiamente para problemas lineales, sino usar una técnica de ordenación. Por lo tanto, este mismo objeto crítico sirve para forzar la ramificación.

Supongamos $p_0 = 8$,

$$\begin{array}{lccccc} p_i: & 4 & 7 & 5 & 1 & 3 \\ v_i: & 5 & 9 & 6 & 2 & 5 \end{array}$$

Se tiene que:

$$\frac{v_1}{p_1} = 1'25,\ \frac{v_2}{p_2} = 1'28,\ \frac{v_3}{p_3} = 1'2,\ \frac{v_4}{p_4} = 2,\ \frac{v_5}{p_5} = 1'66.$$

Por tanto, la solución del problema relajado es:

$X_1 = 0, X_2 = (8 - 1 - 3)/7, X_3 = 0, X_4 = 1, X_5 = 1$ con un valor del objetivo $9 \times 4/7 + 2 + 5 = 12'1428$

La solución no es entera, por lo que consideramos dos subproblemas:

Si $X_2 = 1$, la solución del problema relajado resulta $X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 1, X_5 = 0$ con un valor del objetivo $9 + 2 = 11$. Esta solución es entera.

Si $X_2 = 0$, la solución del problema relajado resulta $X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 1$ con un valor del objetivo $5 + 2 + 5 = 12$. Esta solución es entera y además es, obviamente, óptima.

Otros algoritmos

- Se ha estudiado la estructura poliédrica de este problema combinatorio. Ello ha permitido determinar familias de cortes, encaminadas a la aplicación del algoritmo de planos de corte y con el fin de obtener mejores relajaciones que las explicadas con anterioridad. El tipo de restricciones obtenidas son útiles para problemas que combinan este problema combinatorio con otros.
- Siempre que p_0 sea un entero no excesivamente grande, es bastante conocido otro algoritmo que sigue las ideas de la denominada *Programación Dinámica*.

Problemas relacionados

-Una primera extensión del KP aparece cuando cada objeto está disponible en cantidad ilimitada de unidades. Otra extensión surge cuando se considera alguna otra característica de cada objeto tal como su volumen. Otra variante pretende contemplar relaciones de precedencia entre los objetos.

-Un caso particular importante por sus aplicaciones y dificultad de resolución es cuando $v_j = p_j$, para todo objeto j. Este problema se conoce como *Subset-sum Problem*.

-Otro problema relacionado es el llamado *Problema de Empaquetamiento* (del inglés *Bin Packing Problem*): dados n objetos cada uno j con un valor v_j y un peso p_j, y dado un número ilimitado de contenedores idénticos capaces de cargar un peso máximo de p_0, encontrar el número mínimo de contenedores necesario para cargar todos los objetos.

Técnicas de planificación y control de proyectos

Ayudan en la planificación de proyectos con un gran número de actividades:

- Detectan “cuellos de botella" en el desarrollo de las actividades.
- Evalúan posibilidades de cumplimiento de plazos de entrega.
- Estudian los efectos de cambios en el programa.

Concretamente, se ocupan de determinar el denominado *camino crítico* de un proyecto. Por último consideran el problema de asignación de recursos.

Red de actividades de un proyecto

Un proyecto se representa mediante una red que visualiza gráficamente las relaciones de precedencia en la realización de las actividades.

Actividad (tarea)	Arco
Duración de la actividad	Longitud del arco
Evento (fin de las tareas que llegan al nodo)	Nodo
Secuencia	Sentido del arco
Comienzo y fin de las actividades	Nodo inicial y final

Propiedades de la red de un proyecto

1. Dos nodos no pueden estar conectados directamente por más de un arco.
2. Cada actividad se representa por un solo arco.

Actividad ficticia

- Se utiliza para establecer relaciones de precedencia.
- No tiene duración.
- Se utiliza para evitar violar las propiedades anteriores.

Cálculo de los *instantes más tempranos*, t_i, para la ejecución de las actividades

CÁLCULO HACIA DELANTE:

1. Etiquetar el comienzo del proyecto con tiempo 0.
2. El instante más temprano de cada nodo es el tiempo más temprano del nodo inmediatamente anterior (si sólo tiene uno) más la duración de la actividad (arco) que los une.
3. Si existe más de una actividad que llega a un nodo el tiempo de dicho nodo es el máximo para cada actividad de la suma del tiempo del antecesor más la duración de la actividad.
4. Realizar los pasos 2 y 3 hasta el fin del proyecto.

Cálculo de los *instantes más tardíos*, T_i, para la ejecución de las actividades

CÁLCULO HACIA ATRÁS:

1. El instante más temprano del final del proyecto=instante más tardío del final del proyecto.
2. El instante más tardío de cada nodo es el tiempo más tardío del nodo inmediatamente posterior (si sólo tiene uno) menos la duración de la actividad (arco) que los une.
3. Si existe más de un nodo posterior se toma el mínimo de las diferencias previas.

Definiciones

- **Holgura de un evento (nodo):** Diferencia entre su instante más tardío (T_i) y su instante más temprano (t_i).
- **Holgura total de una actividad (arco) de i a j:** $TF_{ij} = T_j - t_i - d_{ij}$. Es la diferencia entre el instante más tardío de j y la suma del instante más temprano de i y la duración de la actividad de i a j. Se puede interpretar como el máximo retraso en su punto de comienzo o el máximo incremento en su duración que no retrasa el proyecto.
- **Holgura libre de una actividad (arco) de i a j:** $FF_{ij} = t_j - t_i - d_{ij}$. Es la diferencia entre el instante más temprano de j y la suma del instante más temprano de i y la duración de la actividad de i a j. Ídem que no retrasa el inicio de una actividad posterior. Se verifica que: $FF_{ij} \leq TF_{ij}$.

Propiedades de los caminos críticos

Un camino a través de la red donde todas las actividades son críticas (tienen holgura 0) se llama crítico.

1. Una red de un proyecto siempre tiene al menos un camino crítico.
2. Toda actividad con holgura 0 tiene que estar en un camino crítico. Ninguna actividad con holgura > 0 puede estar en un camino crítico.
3. Todo evento con holgura 0 tiene que estar en un camino crítico. Ningún evento con holgura > 0 puede estar en un camino crítico.
4. En un camino crítico todos sus eventos y todas sus actividades tienen holgura cero.

Un ejemplo de proyecto

El siguiente gráfico muestra una red de proyecto con 6 nodos.

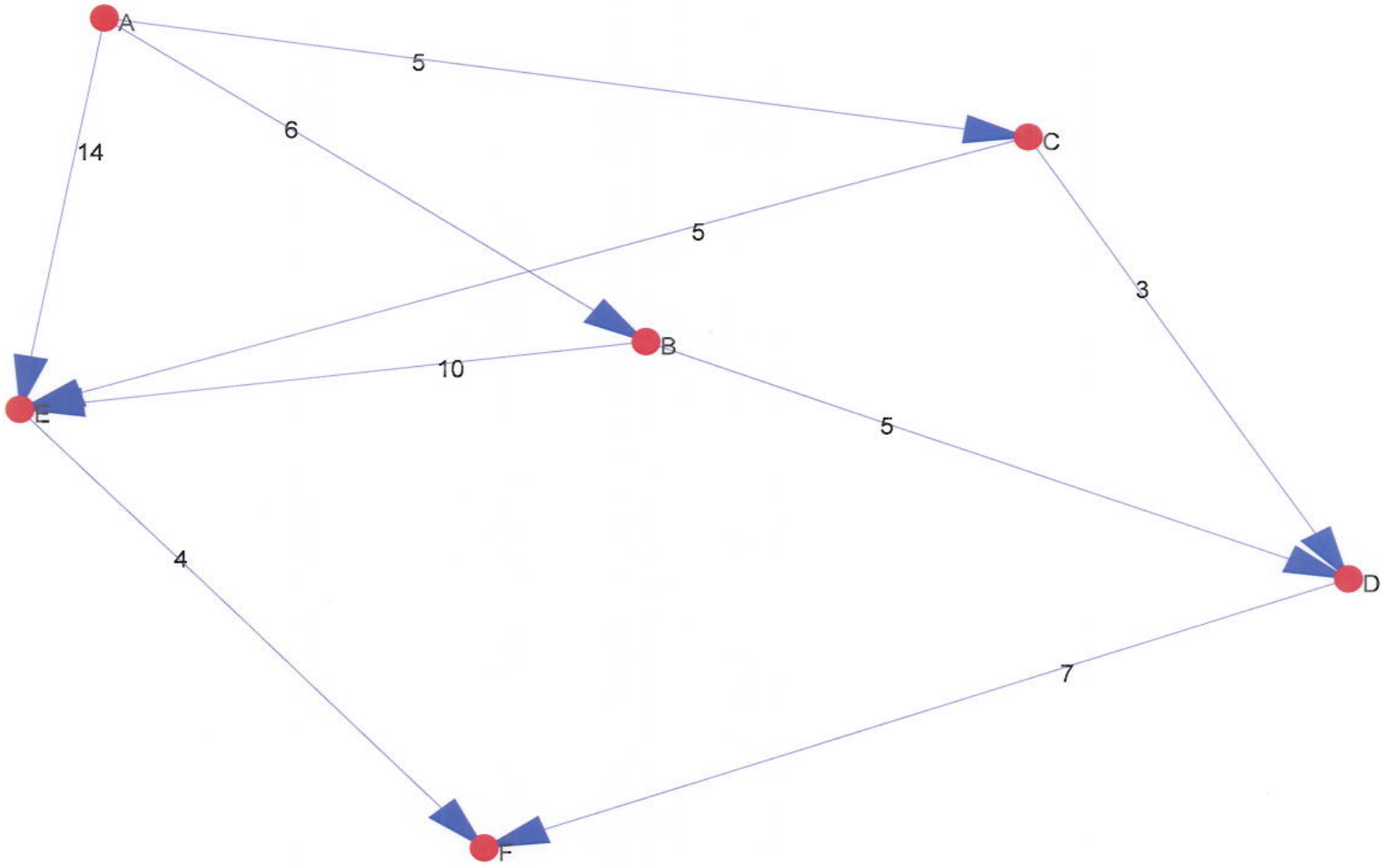
Se muestran las duraciones máximas, $\overline{D}_{ij}$, de cada actividad.

Es fácil ver que el camino crítico es $A \to B \to E \to F$.

La duración total del proyecto es $T = 20$.

Además, para cada nodo, el tiempo mínimo de comienzo de las actividades que parten del nodo es 0, 6, 5, 11, 16 y 20, respectivamente.

La red del proyecto del ejemplo



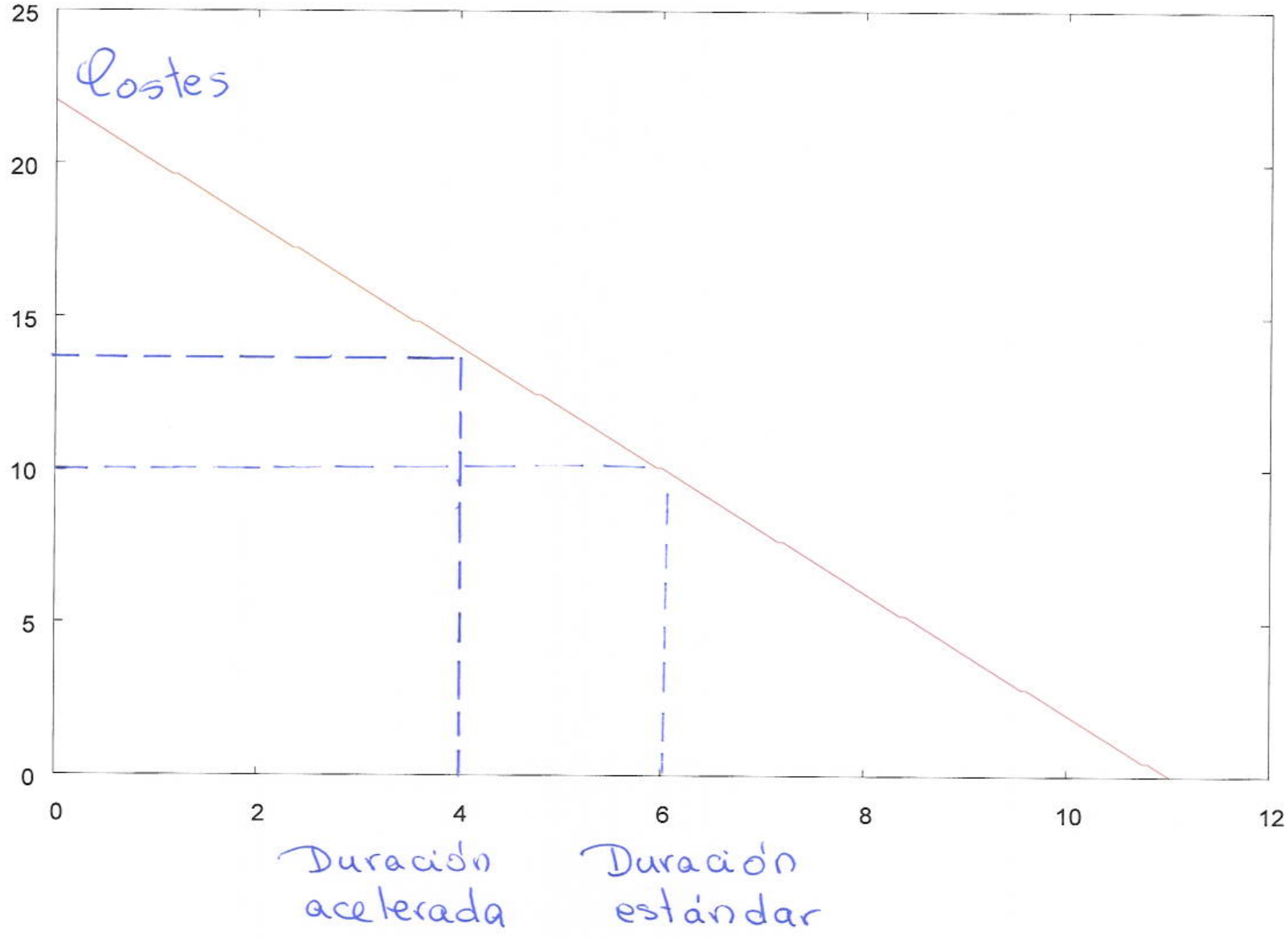
Acti-vidad	Duración Estándar Semanas	Coste Previsto M. euros	Duración Aceleración Semanas	Coste Acelerado M. euros
AB	6	10	4	14
AC	5	8	3	14
AE	14	34	10	54
BD	5	6	3	14
BE	10	20	7	29
CD	3	10	2	14
CE	5	12	3	18
DF	7	16	6	23
EF	4	10	2	30

En la tabla vemos además de las duraciones máximas, las duraciones mínimas, $\underline{D}_{ij}$ y los costes para las duraciones máximas y mínimas, $\overline{C}_{ij}$ y $\underline{C}_{ij}$, respectivamente. La cuestión es determinar el tiempo mínimo de realización del proyecto de forma acelerada sin variar las actividades críticas.

Se define el coste de aceleración de cada actividad como el incremento del coste de la actividad por una reducción unitaria de tiempo, es decir, $|A_{ij}| = \frac{\overline{C}_{ij} - \underline{C}_{ij}}{\overline{D}_{ij} - \underline{D}_{ij}}$.

Otro objetivo puede ser determinar el coste mínimo para finalizar un proyecto en una fecha prefijada. En este caso, se considera que el coste de una actividad se obtiene de la recta con valores factibles entre los límites máximo y mínimo de duración de la actividad.

Gráfica del coste de una actividad



Programación lineal y camino crítico con costes

Coste	$\sum_i \sum_j (A_{ij} X_{ij} + B_{ij})$
Duraciones	$\underline{D}_{ij} \leq X_{ij} \leq \overline{D}_{ij}$
Instantes t_j	$y_i + X_{ij} \leq y_j$
	$y_1 = 0$ comienzo del proyecto
	$y_n \leq T$ final del proyecto de duración T prefijada

En el ejemplo anterior deseamos completar el proyecto en 13 unidades de tiempo.

Los algoritmos heurísticos

Sin embargo, los métodos heurísticos son una forma popular de enfrentar ciertos problemas.

Anteriormente, hemos descrito algoritmos que pueden usarse para obtener una solución óptima para varios tipos de modelos de Investigación Operativa, incluyendo ciertas clases de modelos de programación lineal y programación entera. Sin embargo, hay muchos problemas y sus modelos correspondientes que pueden ser muy complicados y en consecuencia no es posible resolverlos para encontrar una solución óptima. En tales situaciones, puede ser importante encontrar una buena solución factible que al menos esté razonablemente cerca del óptimo. Por lo general, para buscar esa solución se utilizan métodos heurísticos.

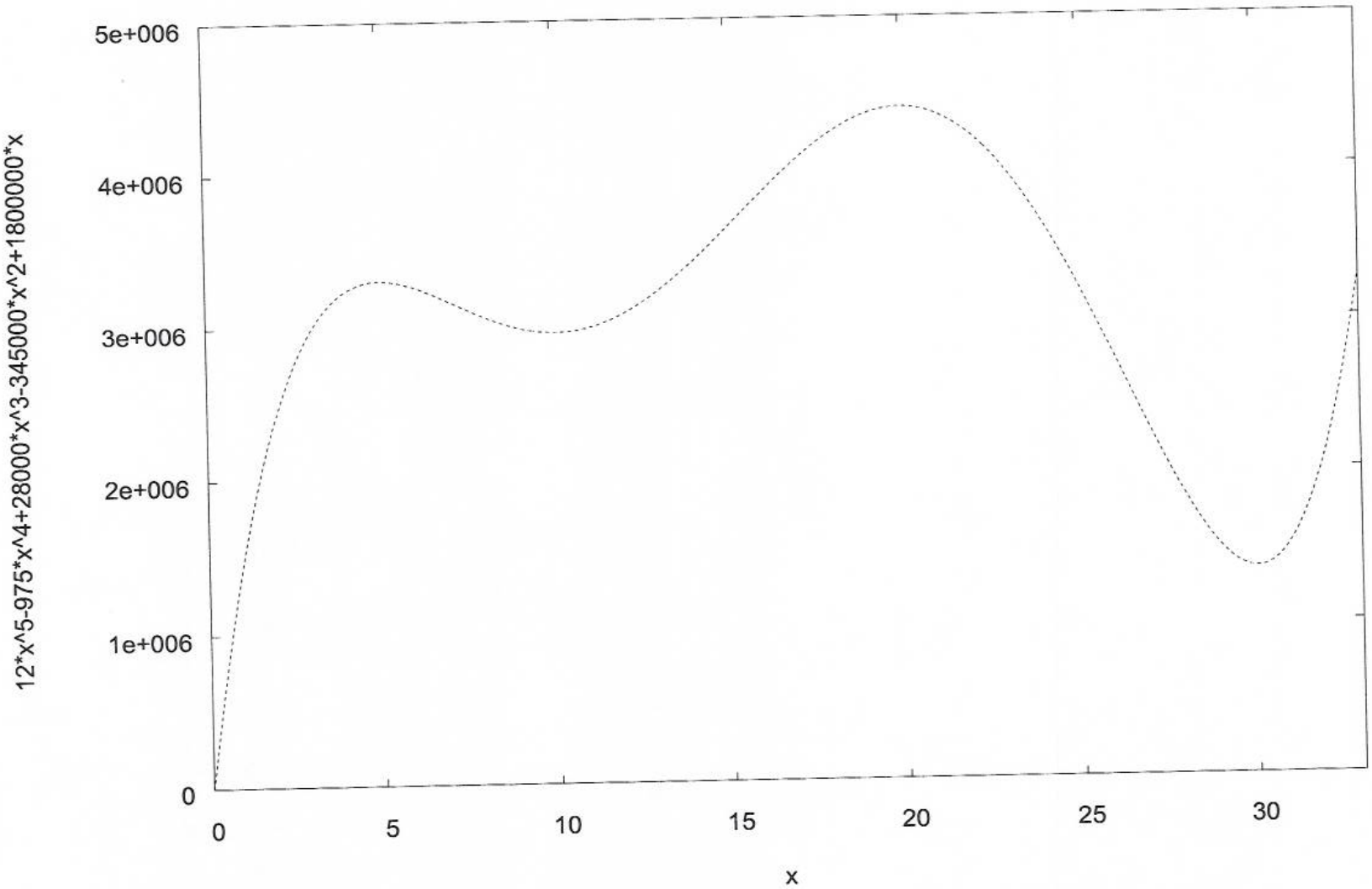
Un **método heurístico** es un procedimiento que trata de descubrir una solución factible muy buena, pero no necesariamente una solución óptima, para un problema específico. No puede darse una garantía acerca de la calidad de la solución que se obtiene, pero un método heurístico bien construído puede proporcionar una solución que al menos está cerca de ser óptima (o concluir que no existen tales soluciones). El procedimiento también debe ser suficientemente eficiente como para manejar problemas muy grandes. Con frecuencia, el procedimiento es un *algoritmo iterativo* novedoso, donde cada iteración implica la realización de una búsqueda de una nueva solución que puede ser mejor que la solución encontrada con anterioridad. Cuando el algoritmo termina después de un tiempo razonable, la solución que proporciona es la mejor que se puede encontrar en cualquier iteración.

Con frecuencia, los métodos heurísticos se basan en ideas relativamente simples de sentido común acerca de la forma en que se debe buscar una buena solución. Estas ideas deben ajustarse al problema específico de interés.
En consecuencia, los métodos heurísticos tienden a ser *ad hoc* por naturaleza, es decir, cada método se elabora para abordar un tipo específico del problema en vez de una variedad de aplicaciones.
Durante mucho tiempo, si no existía un algoritmo disponible para encontrar la solución óptima a un problema, se empezaba desde cero. El panorama ha cambiado con el desarrollo de metaheurísticas. Una **metaheurística** es un método de solución general que proporciona una estructura general y también unos criterios para desarrollar un método heurístico específico que se ajuste a un tipo particular de problema.

La metaheurística se ha convertido en una de las técnicas más importantes dentro de las herramientas que utilizan los profesionales de la Investigación Operativa. Algunos de los metaheurísticos utilizados con más frecuencia son las denominadas técnicas de búsqueda tabú y los algoritmos genéticos.

Para ilustrar la naturaleza de la metaheurística vamos a considerar el siguiente problema moderadamente difícil:
Maximizar
$f(x) = 12x^5 - 975x^4 + 28000x^3 - 345000x^2 + 1800000x$,
sujeto a
$0 \leq x \leq 31$.
En la siguiente figura se representa la función objetivo $f(x)$ sobre los valores factibles de la única variable, x. Esta gráfica muestra que el problema tiene tres óptimos locales, uno en $x = 5$, otro en $x = 20$ y el tercero en $x = 31$, encontrándose el óptimo global en $x = 20$.

Problema moderadamente difícil de maximización



La función objetivo $f(x)$ es tan complicada que sería difícil determinar dónde se encuentra el óptimo global sin el beneficio de observar la gráfica de la figura. Se podría usar el cálculo, pero esto requeriría resolver una ecuación polinomial de cuarto grado (tras igualar a cero la primera derivada) para determinar dónde se encuentran los puntos críticos. Incluso es difícil determinar que $f(x)$ tiene óptimos locales múltiples en vez de un solo óptimo global.

Para problemas de este tipo, un método heurístico común es ejecutar un **procedimiento de mejora local**, que comienza con una solución de prueba inicial y después, en cada iteración, busca en la vecindad de la solución de prueba para tratar de encontrar una mejor solución que la actual. Este proceso continúa hasta que no se puede encontrar una solución mejorada en la vecindad de la solución de prueba actual.

Este tipo de procedimiento puede verse como un procedimiento de “escalada" que se mantiene en ascenso en la gráfica de la función objetivo (si el objetivo es una maximización) hasta que, en esencia alcanza una cumbre. Un procedimiento de mejora local bien construído, por lo general convergerá hacia un óptimo local (una cumbre), pero se detendrá incluso si ese óptimo local no es un óptimo global (la cumbre más alta).

Desventaja de un procedimiento de mejora local: Cuando se aplica un procedimiento de mejora local bien construído a un problema de optimización con múltiples óptimos locales, el procedimiento convergerá hacia un óptimo local y se detendrá. El óptimo local que encuentra depende del punto en el que el procedimiento inicia su búsqueda.

Por lo tanto, el procedimiento encontrará el óptimo global sólo si inicia la búsqueda en la vecindad de dicho óptimo global. Para tratar de evitar esta desventaja, se puede reiniciar el procedimiento de mejora local cierto número de veces a partir de soluciones de prueba aleatorias. Con frecuencia, cuando se reinicia desde una parte nueva de la región factible se llega a un nuevo óptimo local. Si se repite esta rutina cierta cantidad de veces se incrementa la posibilidad de que el mejor óptimo local que se obtuvo en realidad sea el óptimo global.

Por lo general, estos métodos heurísticos implican la generación de una secuencia de soluciones de prueba factibles, donde cada nueva solución de prueba se obtiene al hacer cierto tipo de ajustes menores en la solución de prueba actual. Se han sugerido algunos métodos para ajustar la solución de prueba actual. Por la facilidad de su implantación, un método popular usa el siguiente tipo de ajuste.

Un **subviaje inverso** ajusta la secuencia de ciudades visitadas en la solución de prueba actual mediante la selección de una subsecuencia de las ciudades y simplemente invertir el orden en el cual se visita esa subsecuencia de ciudades. La subsecuencia invertida puede consistir cuando menos en dos ciudades.

Para ilustrar un subviaje inverso, supongamos que la solución de prueba inicial del ejemplo de la figura anterior es visitar las ciudades en orden numérico:

1-2-3-4-5-6-7-1 Distancia=69

Por ejemplo, si selecciona la secuencia 3-4 y se invierte, se obtiene la siguiente nueva solución de prueba:

1-2-4-3-5-6-7-1 Distancia=65

Observamos que la nueva solución de prueba elimina exactamente dos ligaduras del viaje previo y los reemplaza por exactamente dos nuevas ligaduras para formar el nuevo viaje. Ésta es una característica de cualquier subviaje inverso, incluso en los que la subsecuencia de ciudades invertida está formada por más de dos ciudades. Por tanto, un subviaje inverso particular es posible sólo si las dos nuevas ligaduras correspondientes en realidad existen. Este ejemplo sugiere el siguiente método heurístico para buscar una buena solución factible para cualquier problema del agente viajero.

Algoritmo del subviaje inverso:

- **Paso inicial.** Empezamos con cualquier viaje factible como solución de prueba inicial.
- **Iteración.** Para la solución de prueba actual, consideramos todas las formas posibles de realizar un subviaje inverso (excepto el inverso del viaje completo). Seleccionamos el que proporciona la mayor disminución en la distancia viajada para que sea la nueva solución de prueba (los empates se rompen de manera arbitraria).
- **Regla de detención.** El proceso se detiene cuando ninguna inversión de un subviaje mejora la solución de prueba actual, la cual se acepta como la solución final.

Vamos a aplicar este algoritmo al ejemplo. Se inicia con 1-2-3-4-5-6-7-1 como la solución de prueba inicial. Existen cuatro posibles subviajes que mejoran esta solución:

	1-2-3-4-5-6-7-1	Distancia=69
inverso 2-3	1-3-2-4-5-6-7-1	Distancia=68
inverso 3-4	1-2-4-3-5-6-7-1	Distancia=65
inverso 4-5	1-2-3-5-4-6-7-1	Distancia=65
inverso 5-6	1-2-3-4-6-5-7-1	Distancia=66

Las dos soluciones con distancia igual a 65 empatan ya que proporcionan la mayor disminución en la distancia viajada, entonces supongamos que se elige de manera arbitraria la primera de ellas: 1-2-4-3-5-6-7-1, como la siguiente solución de prueba y con lo que se completa la primera iteración.

Segunda iteración: en este caso, existe sólo un subviaje que proporciona una mejora, como se muestra a continuación:

	1-2-4-3-5-6-7-1	Distancia=65
inverso 3-5-6	1-2-4-6-5-3-7-1	Distancia=64

A continuación se tratará de encontrar el inverso que mejorará esta nueva solución de prueba. Sin embargo, no existe ninguna, por lo que se detiene el algoritmo del subviaje inverso con esta solución de prueba como la solución final.
La cuestión es si 1-2-4-6-5-3-7-1 es la solución óptima.
Desafortunadamente no, pues la solución óptima resulta ser: 1-2-4-6-7-5-3-1 Distancia=63 (o 1-3-5-7-6-4-2-1 al invertir la dirección de este viaje completo)
Sin embargo, no se puede llegar a esta solución al realizar la inversión de un subviaje que mejore 1-2-4-6-5-3-7-1.

El algoritmo del subviaje inverso es un ejemplo de un *procedimiento de mejora local*. Encuentra una mejoría sobre la solución de prueba actual en cada iteración. Cuando ya no puede encontrar una mejor solución se detiene porque la solución de prueba actual es un óptimo local. Algunas metaheurísticas permiten encontrar un óptimo global mediante el proceso de “escapar" de un óptimo local.

Búsqueda tabú

Esta metaheurística utiliza algunas ideas de sentido común que permiten que el proceso de búsqueda se aparte de los óptimos locales.
Conceptos básicos. Se incluye como una subrutina algún **procedimiento de búsqueda local** que parezca apropiado para el problema objeto de estudio, donde un **procedimiento de búsqueda local** opera como uno de mejora local con la salvedad de que no requiere que cada nueva solución de prueba sea mejor que la solución de prueba anterior.
El proceso comienza con este procedimiento como uno de **mejora** local de la manera usual, es decir, aceptando sólo una solución mejorada en cada iteración, para encontrar un óptimo local. Una estrategia de la búsqueda tabú es que continúa la búsqueda permitiendo *movimientos sin mejora* hacia las mejores soluciones en la vecindad del óptimo local.

Una vez que se alcanza un punto en el que se pueden encontrar mejores soluciones en la vecindad de la solución de prueba, se aplica de nuevo el procedimiento de mejora local para encontrar un nuevo óptimo local.
Debido a la analogía con la escalada de un monte, algunas veces este proceso se conoce como **enfoque del ascenso más empinado/descenso más suave** ya que cada iteración selecciona el movimiento disponible que sube más la pendiente, o, cuando no hay disponible algún movimiento hacia arriba, selecciona el movimiento que baja menos en la pendiente. Si todo sale bien, el proceso seguirá un patrón como el que se muestra en la siguiente figura, donde se deja atrás un óptimo local con la intención de escalar hacia el óptimo global.

El peligro de este enfoque es regresar al mismo óptimo local y para evitarlo se prohíbe en forma temporal los movimientos que pudieran hacer regresar al proceso a una solución obtenida recientemente. Una **lista tabú** registra esos movimientos prohibidos, los cuales se conocen como *movimientos tabú*. La única excepción a la prohibición de un movimiento de este tipo se presenta cuando un movimiento tabú es mejor que la mejor solución factible que se haya encontrado hasta ese momento. Este uso de *memoria* en un cierto sentido es una característica distintiva de estas metaheurísticas y que las relaciona con el campo de la inteligencia artificial. Incorpora además algunos conceptos avanzados como la *intensificación*, que implica la exploración de una parte de la región factible con mucha intensidad cuando parece que va a contener buenas soluciones y la *diversificación* , que implica forzar la búsqueda en áreas de la región factible no exploradas con anterioridad.

Esquema de un algoritmo de búsqueda tabú básico.
Paso inicial. Comenzar con una solución de prueba inicial factible.
Iteración. Utilizar un procedimiento de búsqueda local apropiado para definir los movimientos factibles en la vecindad local de la solución de prueba actual. No considerar la realización de ningún movimiento incluido en la lista tabú actual a menos que ese movimiento genere una mejor solución que la mejor solución de prueba que se haya encontrado hasta ahora. Determinar cuál de los movimientos restantes proporciona la mejor solución y adoptar esta solución como la próxima solución de prueba, sin importar si es mejor o peor que la solución de prueba actual.
Actualizar la lista tabú para evitar el regreso a la última solución de prueba actual. Si la lista tabú ya está llena, eliminar el elemento más antiguo de la lista para proporcionar más flexibilidad a los movimientos futuros.

Regla de detención. Utilizar algún criterio de detención, como puede ser fijar un número de iteraciones, una cantidad fija de tiempo del CPU, un número fijo de iteraciones consecutivas que no produzcan ninguna mejoría al mejor valor de la función objetivo, o si no existen movimientos factibles en la vecindad local de la solución de prueba actual. Como solución final, se acepta la mejor solución de prueba que se haya encontrado. Este esquema deja algunas cuestiones sin ser respondidas, como cuál es el procedimiento de búsqueda local que debe ser utilizado, cómo debe definirse la *estructura de vecindad* que indica las soluciones vecinas inmediatas de cualquier solución de prueba, la forma en la que los movimientos tabú se deben representar en la lista, el tiempo que debe mantenerse un movimiento tabú en la lista y la regla de detención que debe utilizarse.

Ejemplo del problema del agente viajero. Existen ciertos paralelismos entre un problema del árbol de expansión mínima y uno del agente viajero:

- El problema consiste en elegir las ligaduras que deben incluirse en la solución.
- El objetivo es minimizar el costo total o la distancia asociada con el número de ligaduras que se incluyen en la solución.
- Existe un procedimiento de búsqueda local intuitivo que implica la adición y eliminación de ligaduras en la solución de prueba actual para obtener la nueva solución de prueba. Mientras que en el problema del árbol de expansión, en cada iteración se realiza la adición y la eliminación de una sola ligadura, en el problema del agente viajero se agregan y se borran un par de ligaduras en cada iteración.

Como consecuencia de este paralelismo, el algoritmo de búsqueda tabú básico para problemas del agente viajero puede ser muy similar al descrito para el problema del árbol de expansión con la posibilidad de responder de forma similar a las preguntas clave, como indicamos a continuación:

1. Algoritmo de búsqueda local: En cada iteración, seleccionaremos el mejor vecino inmediato de la solución de prueba actual que no esté descartado por la lista tabú.

2. Estructura de vecindad: Un vecino inmediato de la solución de prueba actual es aquél al que se llega por medio de un *subviaje inverso*.

3. Forma de los movimientos tabú: Enumerar las ligaduras de forma que un subviaje inverso sea tabú si las dos ligaduras que se eliminan por esta inversión se encuentran en la lista, con lo que se evitará regresar con rapidez a la solución de prueba anterior.

4. Adición de un movimiento tabú: En cada iteración, tras elegir las dos ligaduras que deben agregarse a la solución de prueba actual, incorporarlas a la lista tabú.
5. Longitud máxima de la lista tabú: Cuatro (dos de cada una de las dos iteraciones más recientes). Siempre que se agregue un par de ligaduras a una lista llena, eliminamos las dos ligaduras que han permanecido por más tiempo en la lista.
6. Regla de detención. Detenemos el proceso tras tres iteraciones consecutivas sin mejorar el mejor valor de la función objetivo o en cualquier iteración donde la solución de prueba actual no tenga vecinos inmediatos que no estén descartados por la lista tabú.

Vamos a aplicar este algoritmo de búsqueda tabú al ejemplo ya considerado y empezaremos con la misma solución de prueba inicial, 1-2-3-4-5-6-7-1. Aplicando el algoritmo del subviaje inverso con esta solución de prueba inicial, habíamos llegado en dos iteraciones a un óptimo local en 1-2-4-6-5-3-7-1, en cuyo punto se detenía el algoritmo. Por tanto, para el algoritmo de búsqueda tabú, empezaremos como sigue:
Solución de prueba inicial: 1-2-3-4-5-6-7-1 Distancia=69.
Lista tabú: En blanco.
Iteración 1: Se elige invertir 3-4.
Ligaduras borradas: 2-3 y 4-5.
Ligaduras agregadas: 2-4 y 3-5.
Lista tabú: Ligaduras 2-4 y 3-5.
Nueva solución de prueba: 1-2-4-3-5-6-7-1 Distancia=65.

Iteración 2: Se elige invertir 3-5-6.
Ligaduras borradas: 4-3 y 6-7.
Ligaduras agregadas: 4-6 y 3-7.
Lista tabú: Ligaduras 2-4, 3-5, 4-6 y 3-7.
Nueva solución de prueba: 1-2-4-6-5-3-7-1 Distancia=64.
Sin embargo, en lugar de terminar, el algoritmo de búsqueda tabú escapa de este óptimo local porque se traslada hacia el mejor vecino inmediato de la solución de prueba actual aunque su distancia sea mayor. Debido a la disponibilidad limitada de ligaduras entre pares de ciudades, sólo tenemos un vecino inmediato, que es el que se describe a continuación:
Si invertimos 3-7: 1-2-4-6-5-7-3-1 Distancia=66.
Podemos verlo en la figura correspondiente.
El correspondiente vecino inmediato se selecciona como la próxima solución de prueba, como se resume a continuación:

Iteración 3: Se elige invertir 3-7.
Ligaduras borradas: 5-3 y 7-1.
Ligaduras agregadas: 5-7 y 3-1.
Lista tabú: Ligaduras 4-6, 3-7, 5-7 y 3-1.
Nueva solución de prueba: 1-2-4-6-5-7-3-1 Distancia=66.
Notemos que una de las ligaduras borradas es 5-3, que estaba en la lista tabú al final de la iteración 2. Esto es correcto, ya que un subviaje inverso es tabú sólo si *las dos* ligaduras borradas están en la lista tabú.
La nueva solución tiene los dos vecinos inmediatos que se describen a continuación:
Invertimos 2-4-6-5-7: 1-7-5-6-4-2-3-1 Distancia=65.
Invertimos 5-7: 1-2-4-6-5-7-3-1 Distancia=63.
Como el último de estos vecinos tiene la distancia más corta, se convierte en la próxima solución de prueba, como se resume a continuación:

Iteración 4: Se elige invertir 5-7 (ver la correspondiente figura).
Ligaduras borradas: 6-5 y 7-3.
Ligaduras agregadas: 6-7 y 5-3.
Lista tabú: Ligaduras 5-7, 3-1, 6-7 y 5-3.
Nueva solución de prueba: 1-2-4-6-7-5-3-1 Distancia=63.
Esta nueva solución resulta ser la solución óptima. Si no se conoce este hecho, el algoritmo de búsqueda tabú intentará ejecutar más iteraciones. Sin embargo, no existen vecinos disponibles sin violar las condiciones que impone la lista tabú con lo que la regla de detención que hemos definido da por finalizado el algoritmo en este punto. En general, no hay garantía de que la solución final del algoritmo de búsqueda tabú básico sea una solución óptima, aunque en este caso ocurre.

La programación dinámica

- Es una técnica matemática útil para la toma de una serie de decisiones interrelacionadas.
- Proporciona un procedimiento sistemático para determinar la combinación óptima de decisiones.
- En contraste con la programación lineal, no cuenta con una formulación matemática estándar “del problema de programación dinámica", sino que se trata de un enfoque de tipo general para la solución de problemas. Además, las ecuaciones específicas que se usan se deben desarrollar para que representen cada situación individual.

Un ejemplo de programación dinámica en un problema de la ruta más corta

Se quiere organizar un viaje en cuatro etapas desde el estado de California (A) al estado de New York (J). Existen diferentes opciones en cuanto a los estados que se deben elegir como puntos intermedios (Arizona, Utah o Montana, Oklahoma, Missouri o Iowa, Carolina del Norte o Ohio). La siguiente figura muestra las posibles rutas. Las distancias del estado *i* al estado *j*, son:

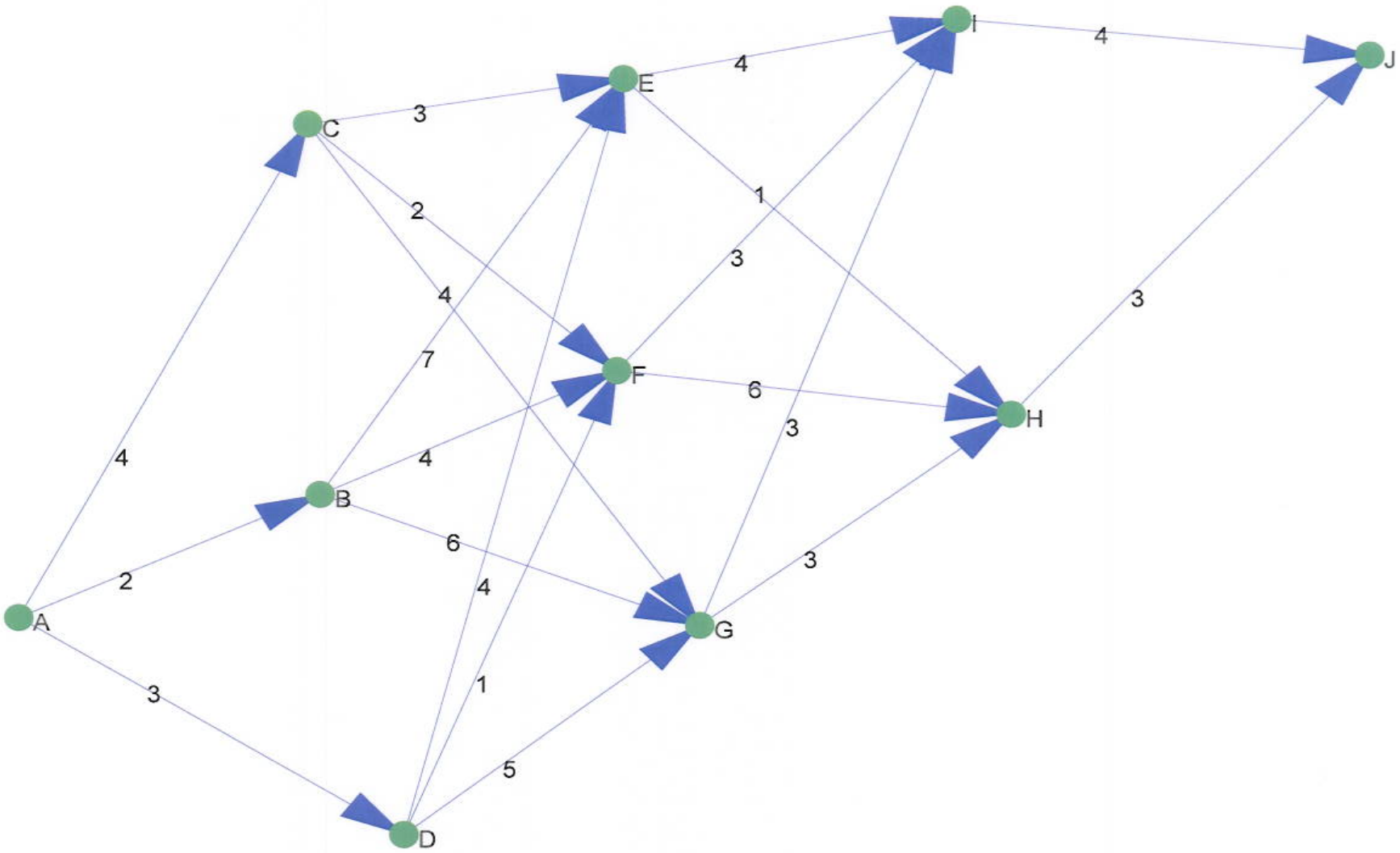
	B	C	D
A	2	4	3

	E	F	G
B	7	4	6
C	3	2	4
D	4	1	5

	H	I
E	1	4
F	6	3
G	3	3

	J
H	3
I	4

Sistema de caminos y distancias de California a New York



Un procedimiento es elegir la ruta más corta en cada etapa sucesiva, pero esto no conduce a una decisión óptima global: la ruta $A \rightarrow B \rightarrow F \rightarrow I \rightarrow J$ tiene una distancia total de 13.

El procedimiento de enumeración de todas las posibilidades implica considerar las 18 posibles rutas y para cada una de ellas la distancia correspondiente.

La programación dinámica proporciona una solución con menor esfuerzo que la enumeración exhaustiva y el ahorro computacional es enorme cuando se trata de versiones grandes del problema. La idea es encontrar la solución óptima para una parte del problema original e ir agrandando el problema de manera gradual y encontrando su solución a partir de la que le precede, hasta resolver el problema completo.

Concretamente, se comienza con el problema sencillo de suponer que se ha llegado casi al final del viaje y sólo queda una etapa más. La solución óptima obvia de este problema reducido es ir del estado actual (aquél en el que se encuentre, sea cual sea) al destino final (estado J).

En cada una de las iteraciones siguientes, el problema aumenta de una en una el número de etapas que le quedan por recorrer para completar el viaje.

En cada problema aumentado se puede encontrar la solución óptima del lugar al que debe dirigirse desde cada estado posible, donde se toma en cuenta los resultados obtenidos en la iteración anterior.

Formulación

Sean X_n $(n = 1, 2, 3, 4)$ las variables de decisión que representan el destino inmediato de la etapa n (es decir, el n-ésimo viaje que se realizará). En este caso, la ruta seleccionada es $A \rightarrow X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4$, donde $X_4 = J$.

Sea $f_n(s, X_n)$ el coste total de la mejor *política* global para enfrentar las etapas *restantes* cuando se encuentra en el estado s, listo para iniciar la etapa n y elige X_n como destino inmediato. Dados entonces un estado s y una etapa n, sea X_n^* el valor para el estado destino X_n, no necesariamente único, que minimiza $f_n(s, X_n)$, y sea $f_n^*(s)$ ese valor mínimo correspondiente de $f_n(s, X_n)$. Por tanto,

$$f_n^*(s) = \min_{X_n} f_n(s, X_n) = f_n(s, X_n^*),$$

donde $f_n(s, X_n) =$ coste inmediato (etapa n) + coste futuro mínimo (correspondiente a las etapas n+1 y siguientes)=$c_{sX_n} + f_{n+1}^*(X_n)$.

Procedimiento de solución

Los valores de c_{sX_n} están dados por las tablas presentadas al enunciar el ejemplo al establecer $i = s$ (el estado actual) y $j = X_n$ (un posible destino inmediato). Como el destino final (estado J) se alcanza al terminar la etapa 4, claramente $f_5^*(J) = 0$.

El objetivo es determinar $f_1^*(A)$ y la ruta correspondiente. La técnica de la programación dinámica la encuentra al determinar de manera sucesiva $f_4^*(s), f_3^*(s), f_2^*(s)$, para cada uno de los estados posibles s y usar después $f_2^*(s)$ para encontrar $f_1^*(s)$.

Cuando sólo queda una etapa por recorrer (n=4), la ruta de ahí en adelante está perfectamente determinada por el estado actual s (ya sea H o I), así como su destino final, $X_4 = J$, de manera que la ruta de esa última jornada, si se inicia en s, será necesariamente $s \rightarrow J$. Por lo tanto, $f_4^*(s) = f_4(s, J) = c_{s,J}$.

La solución al problema para $n = 4$ es:

s	$f_4^*(s)$	x_4^*
H	3	J
I	4	J

Cuando hay dos etapas por recorrer (n=3) el procedimiento de solución requiere unos cuantos cálculos. Por ejemplo, supongamos que nos encontramos en el estado F. Entonces, como se describe en el diagrama, debe ir al estado H o al estado I con unos costes inmediatos respectivos de $c_{F,H} = 6$ o $c_{FI} = 3$. Si elige H, el coste adicional mínimo al llegar ahí se presenta en la tabla anterior como $f_4^*(H) = 3$ y en consecuencia, el coste total de esta decisión es 6+3=9. Si en su lugar elige el estado I, el coste total es 3+4=7, que es menor. Por lo tanto la solución óptima es $X_3^* = I$, puesto que proporciona el coste mínimo $f_3^*(F) = 7$.

Razonando análogamente para los otros dos estados, $s = E$ y $s = G$, los resultados del problema para $n = 3$, teniendo en cuenta que $f_3(s, X_3) = c_{sX_3} + f_4^*(X_3)$, $s = E, F$ o G y $X_3 = H$ o I:

s/X_3	H	I	$f_3^*(s)$	X_3^*
E	4	8	4	H
F	9	7	7	I
G	6	7	6	H

Razonando análogamente, los resultados del problema para $n = 2$, teniendo en cuenta que $f_2(s, X_2) = c_{sX_2} + f_3^*(X_2)$, $s = B, C$ o D y $X_2 = E, F$ o G, son:

s/X_2	E	F	G	$f_2^*(s)$	X_2^*
B	11	11	12	11	E o F
C	7	9	10	7	E
D	8	8	11	8	E o F

En el primer y tercer renglones de esta tabla se observa que E y F empatan como el valor que minimiza X_2, de manera que el destino inmediato desde cualquiera de los estados B o D debe ser $X_2^* = E$ o F.

Si se pasa al problema de la primera etapa ($n = 1$), con las cuatro etapas por recorrer, los cálculos son similares a los realizados en la segunda etapa (n=2), excepto que ahora sólo hay un estado de inicio posible, $s = A$, con posibles destinos $X_1 = B, C$ o D. Por ejemplo, para $X_1 = B$, $f_1(A, B) = c_{A,B} + f_2^*(B) = 2 + 11 = 13$. Razonando igual con los otros dos destinos se obtienen los resultados que se muestran en la siguiente tabla para $n = 1$, con $f_1(s, X_1) = c_{sX_1} + f_2^*(X_1)$:

s/X_3	B	C	D	$f_1^*(s)$	X_1^*
A	13	11	11	11	C o D

Ahora es posible identificar una solución óptima a partir de las cuatro tablas. Los resultados del problema con $n = 1$ indican que el viaje que parte de A debe elegir como destino inmediato C o D. Supongamos que se elige C. Con $n = 2$, el resultado de $s = C$ es $X_2^* = E$. Esto conduce al problema de $n = 3$, que resulta en $X_3^* = H$ con $s = E$, y el problema con $n = 4$ indica que $X_4^* = J$ con $s = H$. Por lo tanto una ruta óptima es

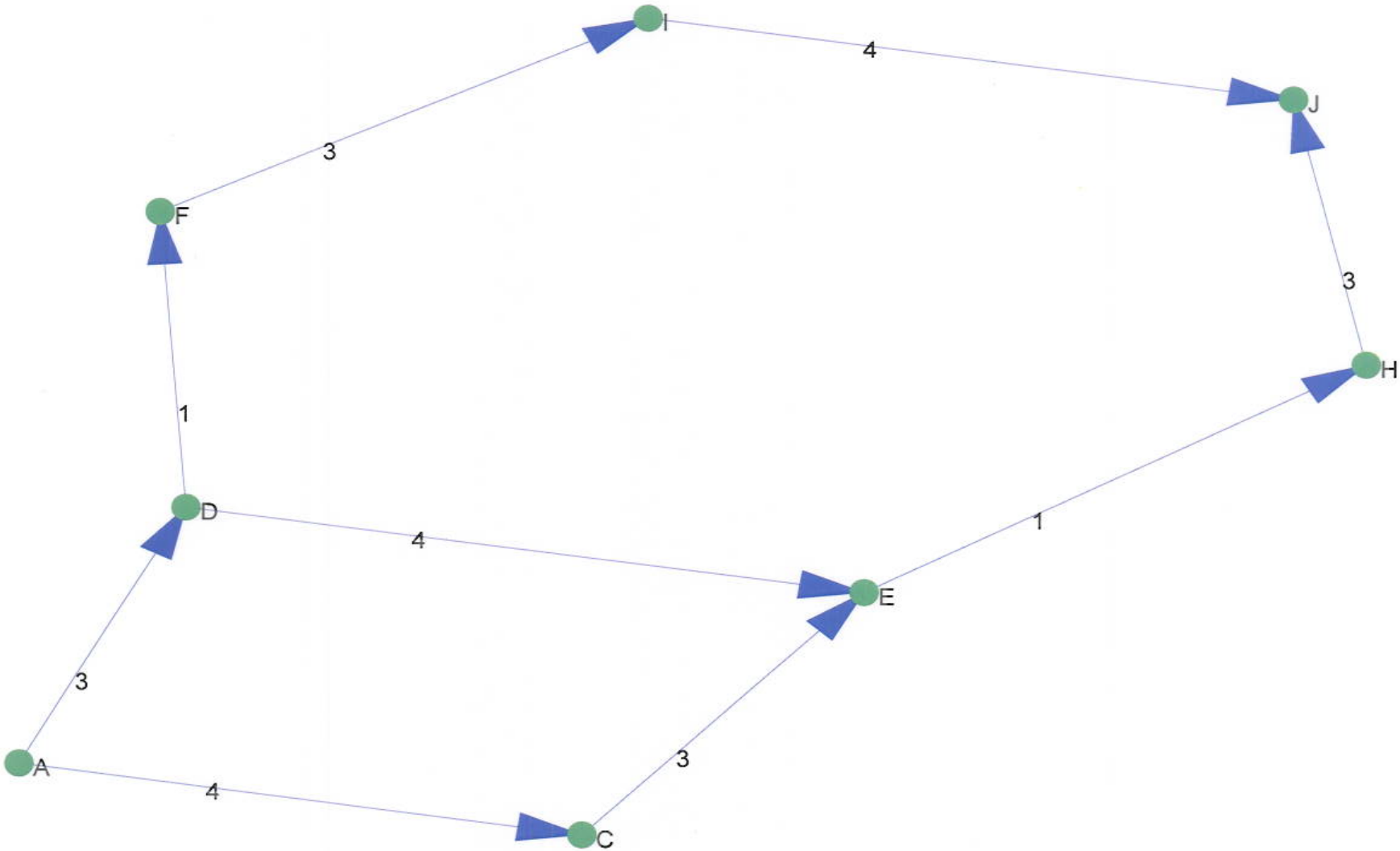
$$A \to C \to E \to H \to J.$$

Si se elige $X_1^* = D$, se obtienen otras dos rutas óptimas:

$$A \to D \to E \to H \to J \text{ y}$$

$$A \to D \to F \to I \to J.$$

Descripción gráfica de la solución de programación dinámica



Características de los problemas de programación dinámica

Una manera de reconocer una situación que se pueda formular como un problema de programación dinámica es poder identificar una estructura análoga a la del problema anterior. Las características básicas que distinguen a los problemas de programación dinámica son:

1. El problema se puede dividir en **etapas**, cada una de las cuales requiere de una **política de decisión**. Se requiere tomar una *serie de decisiones interrelacionadas*, cada una de las cuales corresponde a una etapa del problema.

2. Cada etapa tiene cierto número de **estados** asociados con su inicio. Son las distintas *condiciones posibles* en las que se puede encontrar el sistema en cada etapa del problema. El número de estados puede ser finito o infinito.

3. El efecto de la política de decisión en cada etapa es *transformar el estado actual en un estado asociado con el inicio de la siguiente etapa*, en ocasiones según una distribución de probabilidad. El procedimiento sugiere que los problemas de programación dinámica se pueden interpretar en términos de *redes*. Cada *nodo* corresponde a un *estado*. La red consiste en columnas de nodos, donde cada *columna* corresponde a una *etapa*, de forma que el flujo que sale de un nodo sólo puede ir a un nodo de la siguiente columna a la derecha. El valor asignado a cada arco que conecta dos nodos se interpreta como la *contribución inmediata* a la función objetivo que se obtiene al tomar esa política de decisión. En muchos casos, el objetivo corresponde a encontrar la trayectoria *más corta* o bien la *más larga* a través de la red.

4. El procedimiento de solución está pensado para encontrar una **política óptima** para manejar el problema completo, es decir, una receta para elaborar la política de decisión óptima para cada etapa en cada uno de los estados posibles. En cualquier problema, la programación dinámica proporciona una política sobre qué hacer en todas las circunstancias posibles. Esto puede ser valioso en muchas situaciones que incluyen un análisis de sensibilidad.

5. Dado el estado actual, una *política óptima para las etapas restantes es independiente* de la política adoptada en *etapas anteriores*. Por lo tanto, la decisión inmediata óptima depende sólo del estado actual y no de cómo se llegó ahí. Éste es el **principio de optimalidad** de la programación dinámica.

6. El procedimiento de solución inicia cuando se determina la *política óptima* para la última etapa.

7. Se dispone de una **relación recursiva** que identifica la política óptima para la etapa n, dada la política óptima para la etapa $n+1$. La forma precisa de la relación recursiva difiere de un problema a otro de programación dinámica, aunque con la siguiente notación:

- N: número de etapas.
- n: etiqueta de la etapa actual ($n = 1, 2, \cdots, N$).
- s_n: estado actual de la etapa n.
- X_n: variable de decisión de la etapa n.
- X_n^*: valor óptimo de X_n (dado s_n).

- $f_n(s_n, X_n)$: contribución a la función objetivo de las etapas $n, n+1, \cdots, N$, si el sistema se encuentra en el estado s_n en la etapa n, la decisión inmediata es X_n y en adelante se toman decisiones óptimas $f_n^*(s_n) = f_n(s_n, X_n^*)$.

La relación recursiva siempre tendrá la forma

$$f_n^*(s_n) = \max_{X_n}\{f_n(s_n, X_n)\} \text{ o } f_n^*(s_n) = \min_{X_n}\{f_n(s_n, X_n)\},$$

- 21. Resolver manualmente el siguiente problema del transporte, partiendo de una solución inicial obtenida por el método de la esquina noroeste:

2	2	5	6	10
3	1	8	5	20
8	9	4	9	15
17	18	5	5	

- 22. Determinando una solución inicial por el método del coste mínimo, resolver manualmente el problema del transporte con vector de existencias (10,15,25) y con vector de demandas (5,10,20,15) supuesto que la matriz de costes es $\begin{pmatrix} 8 & 3 & 5 & 2 \\ 4 & 1 & 6 & 7 \\ 1 & 9 & 4 & 3 \end{pmatrix}$.

- 23. Resolver el siguiente problema del transporte manualmente, partiendo de una solución inicial obtenida por el método del coste mínimo:

7	5	30	1	2	40
2	4	10	8	5	42
9	6	5	7	8	50
10	20	30	50	22	

- 24. (Hillier y Liberman, 2005) La METRO WATER DISTRICT es una dependencia que administra la distribución de agua en cierta región geográfica grande. La región es bastante árida, por lo que el distrito debe comprar y traer agua del exterior. Las fuentes de esta agua importada son los ríos Colombo, Sacron y Calorie. El distrito revende el agua a los usuarios de la región. Sus clientes principales son los departamentos de aguas de las ciudades de Berdoo, Los Devils, San Go y Hollyglass.

Es posible hacer llegar agua a cualquiera de estas ciudades desde cualquiera de los tres ríos, con la excepción de que no hay forma de abastecer a Hollyglass con agua del río Calorie. Para los demás casos, los costes de abastecimiento por cada combinación de río y ciudad (en décenas de dolares por pie de acre[3]) aparecen recogidos en la siguiente tabla y también las cantidades disponibles en los tres ríos y las solicitudes de cada ciudad (en millones de pies de acre) para el próximo verano. Resolver el problema de la administración del distrito de asignación del agua disponible de manera que se minimice el coste total, haciendo uso de una herramienta informática. Además, dibujar el grafo que representa los canales de abastecimiento de las ciudades desde los ríos.

[3] 1 pie de acre=1233'48 m^3

Costes	Berdoo	L. Devils	S. Go	Holly.	Fic.	Dispon.
Colombo	16	13	22	17	0	70
Sacron	14	13	19	15	0	60
Calorie	19	20	23	*M*	0	90
Solic.	50	60	30	40	40	**220**

Notemos que la imposibilidad de abastecimiento de Hollyglass por el río Calorie, se refleja en la tabla con una **penalización** en la tabla de costes de *M* (un coste tan grande como se quiera). Por otro lado, el problema no es inicialmente equilibrado al haber un exceso en las disponibilidades de 40 unidades. Para “equilibrar" el problema se incluye un **destino ficticio**, para el que los abastecimientos son a coste cero, con una solicitud precisamente de 40 unidades.

- 25. El jefe de un bufete de abogados está interesado en la utilización más efectiva de sus recursos de personal buscando la forma de hacer las mejores asignaciones de abogado-cliente. Llegan 4 nuevos clientes y encuentra que 4 abogados pueden hacerse cargo de los casos. Cada uno de ellos sólo se puede hacerse cargo de un caso. Para decidir la mejor asignación se tiene en cuenta una tasa de efectividad (de 1 a 9) construida sobre actuaciones anteriores de dichos abogados, ya que no todos son igual de especialistas en todo tipo de procesos:

Tasa de efectividad de cada abogado según caso cliente

Abogado	Divorcio	Fusión	Desfalco empresarial	Herencias
1)	6	2	8	5
2)	9	3	5	8
3)	4	8	3	4
4)	6	7	6	4

Determinar la asignación más efectiva.

- 26. (Asignación generalizada.) Un sistema de procesamiento compartido tiene 3 ordenadores y tiene que procesar 6 tareas. Todas las tareas se pueden realizar en cualquier ordenador y no se pueden fraccionar. Los tiempos de procesamiento de cada tarea i en cada ordenador j, t_{ij}, y los tiempos disponibles de cada ordenador son:

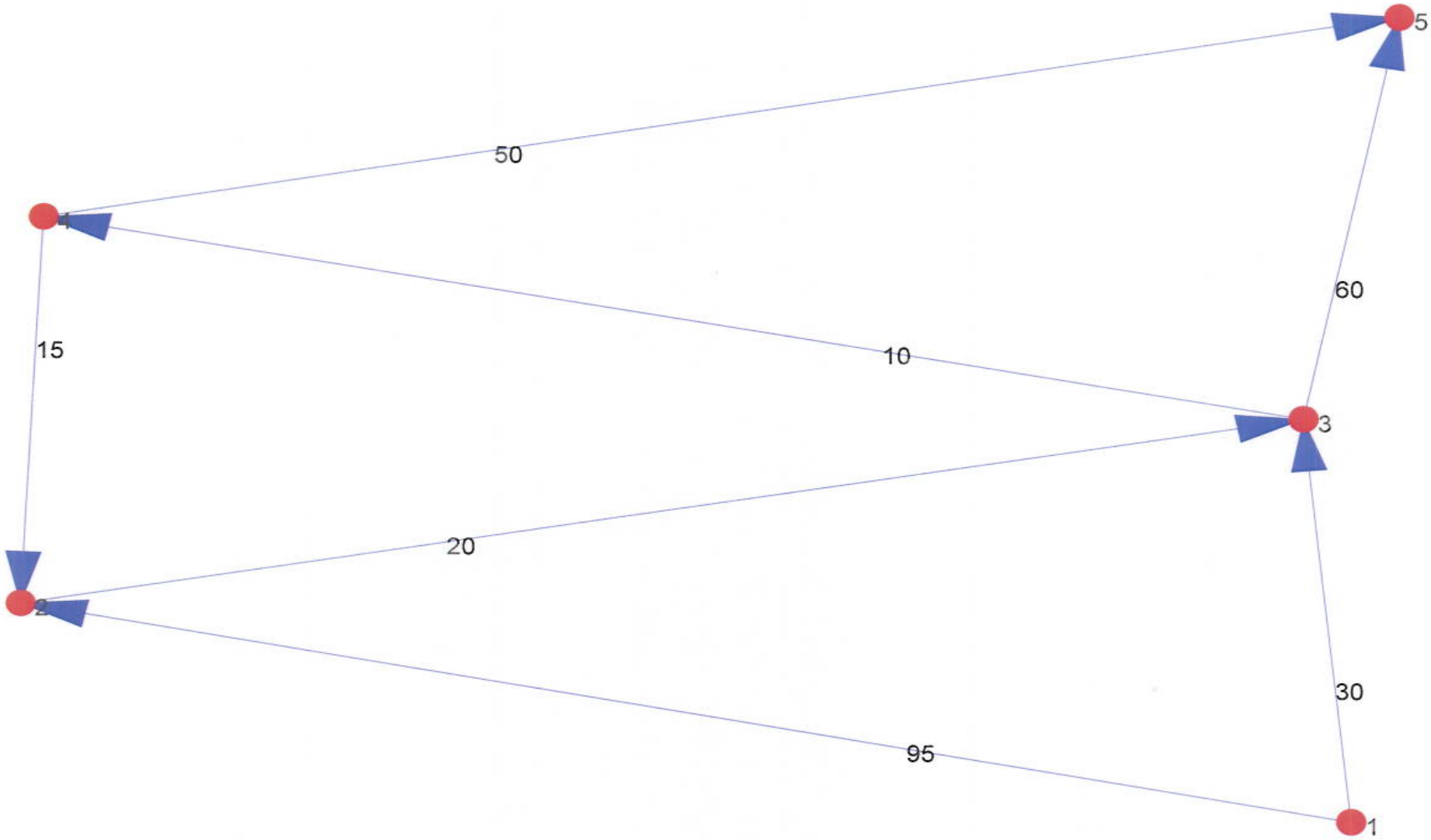
Tarea/Ordenador	O1	O2	O3
T1	18	16	12
T2	14	21	19
T3	23	27	33
T4	16	24	23
T5	17	24	24
T6	25	28	30
T. disp.	47	41	46

Decidir a qué ordenador debemos mandar cada tarea si queremos minimizar el tiempo total de procesamiento. Utilizar, por ejemplo, Solver de Excel para la resolución.

- 27. En la siguiente red, cinco ciudades están conectadas por medio de posibles rutas. La distancia en millas[4] entre cada dos ciudades está sobre el arco que las conecta. Encontrar la distancia más corta entre 1 y 5.
- 28. La administración de Seervada Park necesita determinar los caminos en los cuales tender las líneas telefónicas para conectar todas las estaciones con una longitud mínima de cable.

[4] 1 milla=1609'34 metros

Red de caminos entre cinco ciudades



- 29. Una empresa de telefonía por cable se encuentra en el proceso de proporcionar el servicio por Internet a nueve urbanizaciones y se quiere determinar la red más económica que permita llevar la conexión a las urbanizaciones. En la tabla siguiente aparecen los costes de conexión entre las posibles uniones, a partir de la cual se puede dibujar el grafo.

	1	2	3	4	5	6	7	8	9
1		3		3	4				
2	3		3	2					
3		3		4			4	6	
4	3	2	4		5	3	5		
5	4			5		4			
6				3	4		4	8	3
7			4	5		4		2	
8			6			8	2		5
9						3		5	

- 30. **Ajuste de una función lineal a una muestra de observaciones** Se ha medido el contenido de oxígeno, Y, en miligramos / litro, del lago Worther, en Austria, a una profundidad de X metros, obteniéndose los siguientes 7 pares de observaciones:

$$X \quad 15 \quad 20 \quad 30 \quad 40 \quad 50 \quad 60 \quad 70$$

$$Y \quad 6'5 \quad 5'6 \quad 5'4 \quad 6'0 \quad 4'6 \quad 1'4 \quad 0'1$$

Ajusta una recta a los datos anteriores, utilizando un modelo de programación lineal. Además, predecir, para una profundidad comprendida entre 75 y 70 metros, el contenido en oxígeno.

- 31. En una investigación relativa a las naciones europeas se quiere estudiar el nivel de penetración de Internet en los hogares (Y) (% de hogares con acceso a Internet) en función de los salarios brutos (X_1) (media anual en paridades de poder adquisitivo) y la población (X_2) (entre 25 y 64 años) que ha completado al menos la segunda etapa de la Educación Secundaria. La tabla siguiente recoge los datos para una muestra de 10 paises.

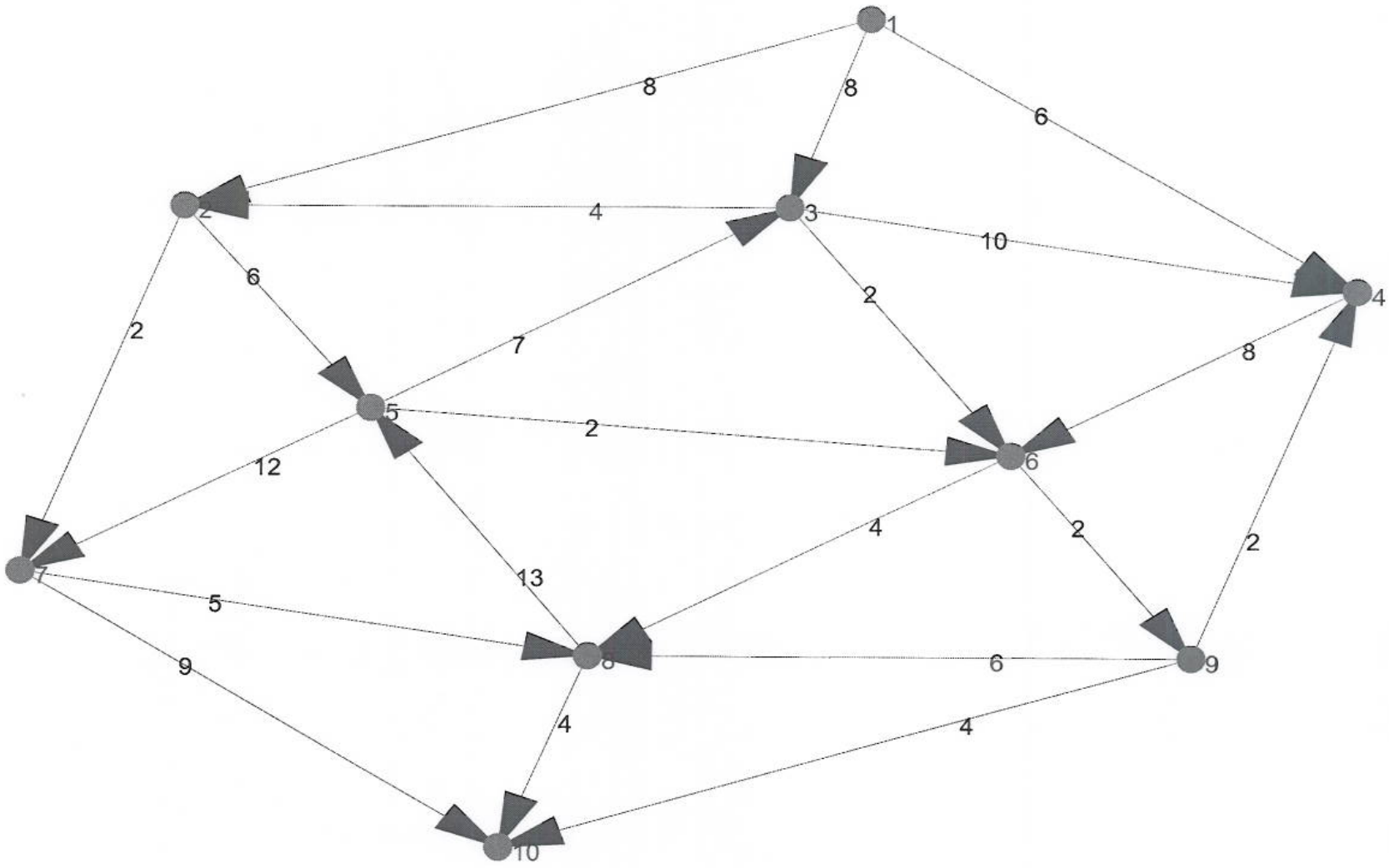
  Se quiere construir la función lineal que ajuste estas observaciones utilizando un modelo de programación lineal. Además, si se sabe que en Hungría el salario medio fue de 13518 euros y el tanto por cien de población con Educación Secundaria o Superior es 79'2, dar una estimación de su porcentaje de hogares con Internet.

	Salarios en Euros (2006)	% Población con Educación Secundaria o Superior (2007)	% Hogares con Internet (2008)
Luxemburgo	46085	65’7	80
Reino Unido	38069	73’4	71
Austria	35566	80’1	69
Dinamarca	35145	75’5	82
Suecia	29263	84’6	84
Italia	28946	52’3	42
Eslovaquia	11598	89’1	58
Estonia	12195	89’1	58
Letonia	9820	85	53
Rumanía	7485	75	30

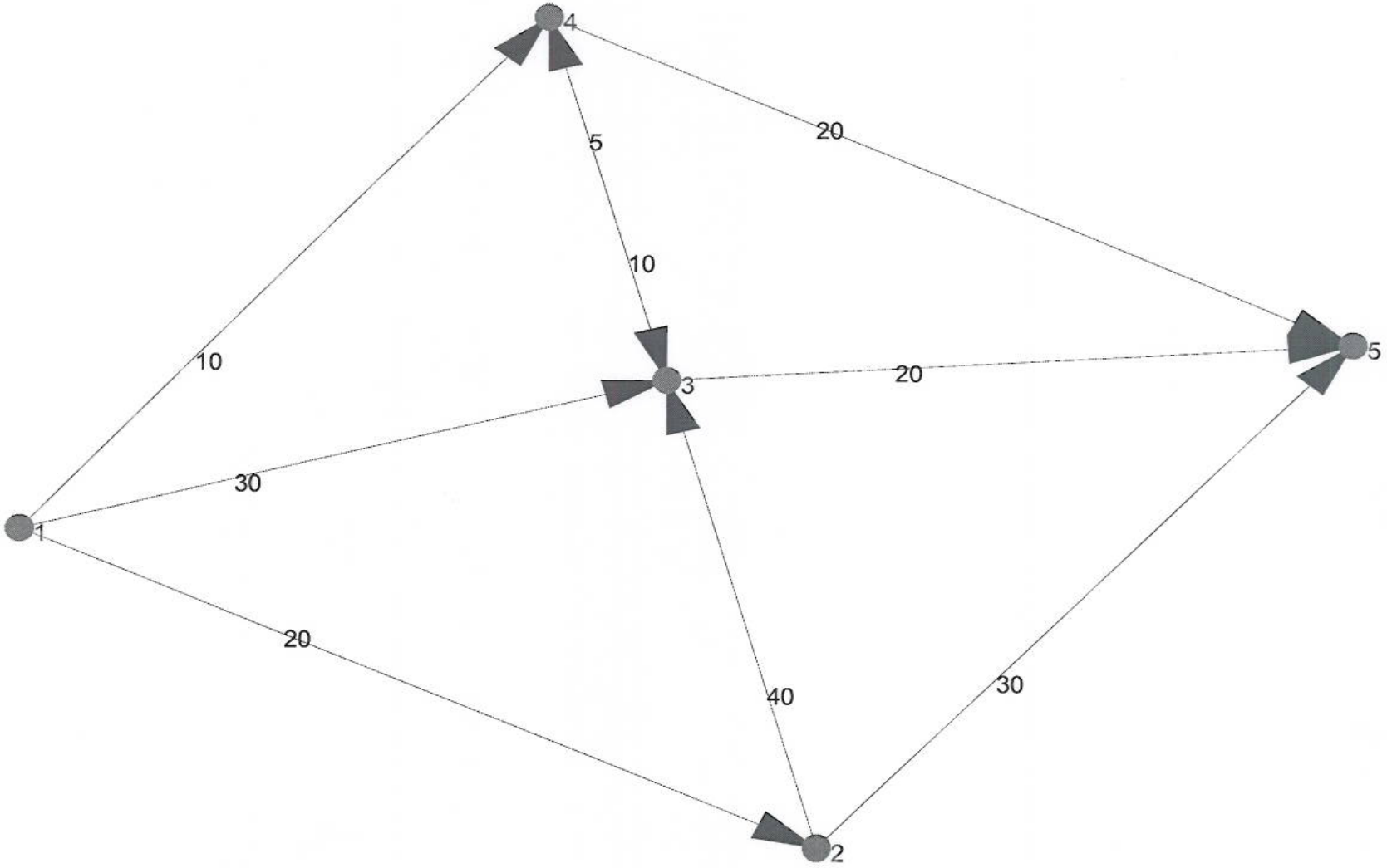
Fuente: España en cifras 2009. INE.

- 32. En la siguiente red de calles, determinar el máximo flujo que puede atravesarla y cuál es la calle, que de ser cortada en una hora punta en la que la red está a tope de su capacidad, provocaría un colapso mayor.
- 33. En la siguiente red con 5 nodos, determinar el máximo flujo que puede atravesarla.

Red de calles



Red con 5 nodos



- 34. La entrenadora de un equipo de natación debe asignar competidoras para la prueba de 200 metros combinado por equipos para mandarlas a una competición. Como muchas de sus mejores nadadoras son rápidas en más de un estilo, no le es fácil decidir a qué estilo asignar a cada una. Las cinco mejores nadadoras y sus mejores tiempos (en segundos) en cada estilo son:

Tipo de nado/Nadadora	1	2	3	4	5
Espalda	31	33	30	31	32
Crol	32	31	30	31	31
Braza	31	32	30	32	33
Mariposa	30	31	31	30	32
Libre	30	30	31	30	31

La entrenadora quiere minimizar la suma de los mejores tiempos correspondientes. Formúlese este problema como un *problema de asignación* y resolverlo.

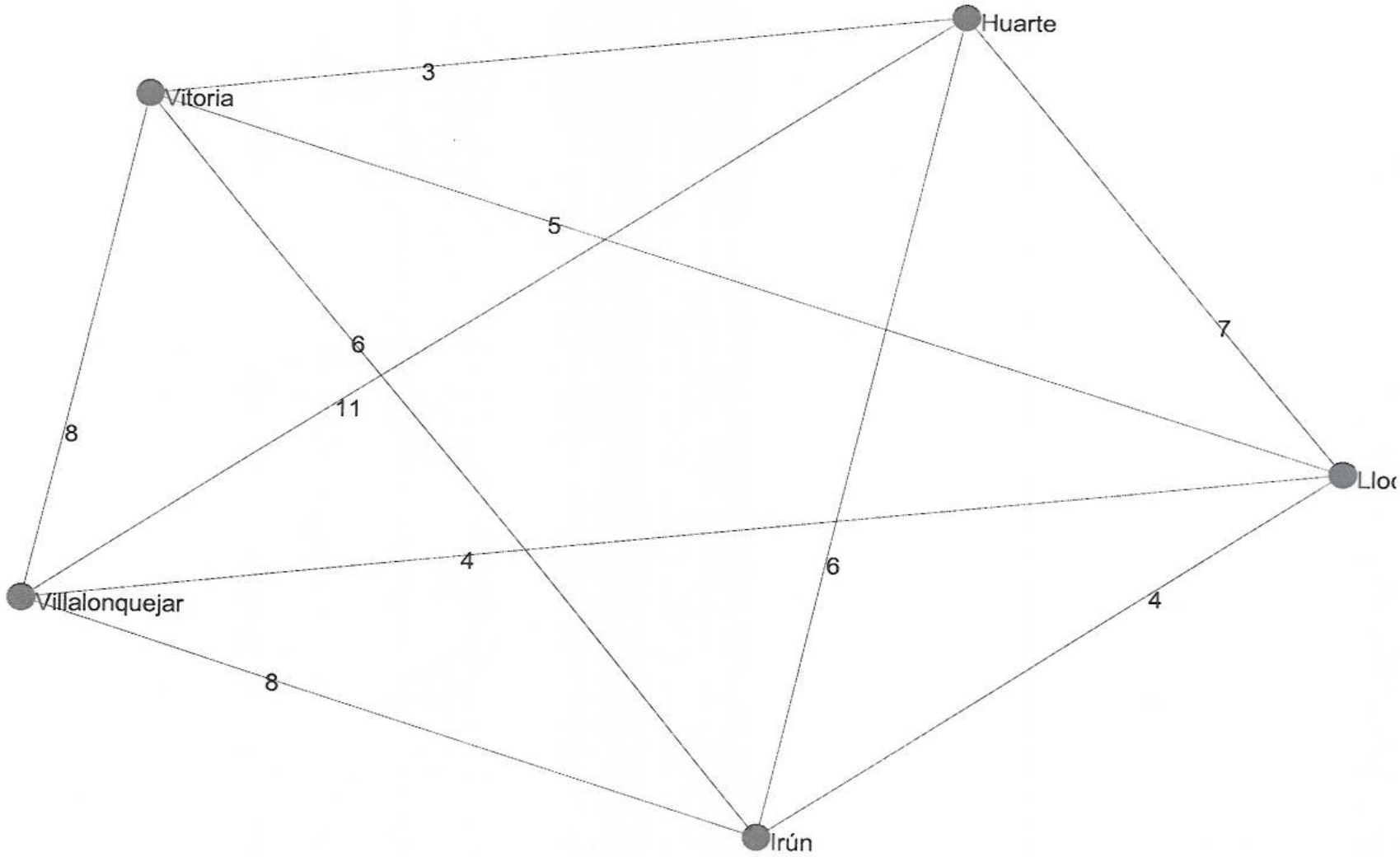
- 35. La empresa Bentzeler sita en el Polígono de Villalonquejar (Burgos) fabrica componentes de automóviles y recoge diferentes piezas, que usa como materias primas, en diferentes factorías de la geografía nacional. La recogida de estas piezas debe seguir una periodicidad determinada. Un problema con el que se encuentra es fijar en qué fechas se recogen las diferentes piezas y además confeccionar las rutas diarias correspondientes, de forma que el coste total de transporte sea mínimo.

  Partimos de un primer problema en el que se sabe que el camión que transporta las materias primas habrá de acudir desde Villalonquejar a las localidades de Vitoria, Llodio, Irún y Huarte para regresar al final de nuevo a Villalonquejar.

La siguiente figura recoge las distancias entre estas localidades. Se desea determinar la ruta que tiene que seguir el camión con el objetivo de minimizar la distancia total recorrida.

- Enumerar todos los viajes posibles, excluyendo aquéllos que sólo invierten el orden de los viajes de la lista presentada con anterioridad. Calcular la distancia de cada uno de estos viajes y con esto identificar el viaje óptimo.
- Aplicar el algoritmo del subviaje inverso a este problema; para ello comience con 1-2-4-3-5-1 como la solución de prueba inicial.

Red de viajes de la empresa Bentzeler



Ramificación y acotación

- 36. Resolver el siguiente problema usando la técnica de ramificación y acotación.

  Max $5X_1 + 27X_2$

  sujeto a

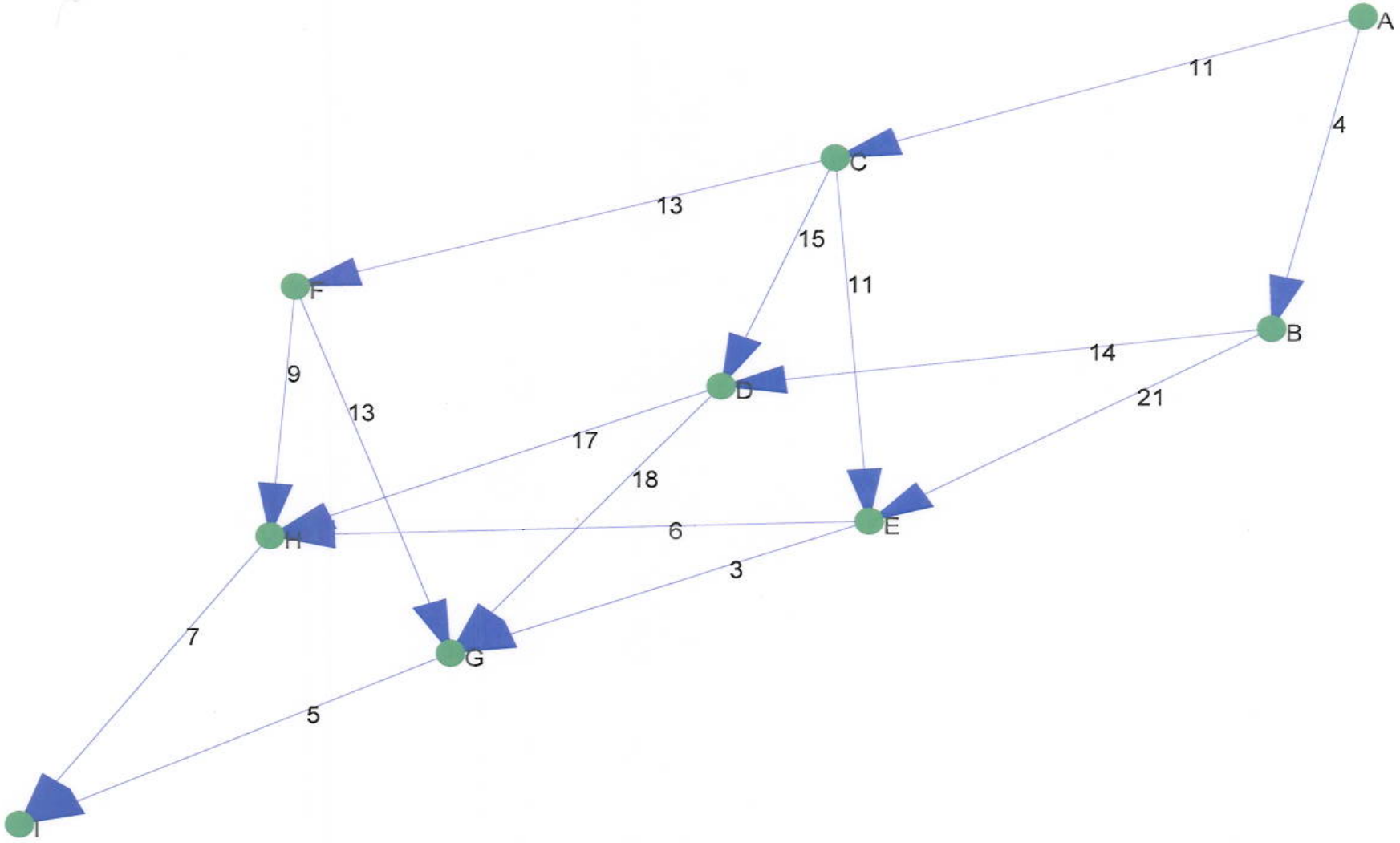
  $2X_1 + 11X_2 \leq 59$

  $X_1 - X_2 \leq 7$

  con $X_1, X_2 \geq 0$ y enteras.

Programación dinámica

- 37


A
11
4
C
13
15
11
F
B
14
9
D
21
13
17
18
H
6
E
3
7
G
5
I

Soluciones e indicaciones de algunos ejercicios

Ejercicios 30/31: Ajuste de una función lineal a una muestra de observaciones

Supongamos que tenemos un conjunto de observaciones (x^i, y^i), $i = 1, 2, \cdots, m$, donde cada x^i es un vector con n elementos y cada y^i es un número real. Queremos encontrar un vector $a \in \mathbb{R}^n$ y una constante b tal que

$$x^{it} a + b \approx y^i \ \forall i = 1, 2, \cdots, m$$

es decir $x_1^i a_1 + \cdots + x_n^i a_n + b \approx y^i \ \forall i = 1, 2, \cdots, m$.

La función $Y = a_1 X_1 + \cdots a_n X_n + b$ va a "explicar" el comportamiento de una cierta variable (dependiente) Y a partir de los valores de las variables (independientes) $X_1, \cdots, X_n$ y la construimos con la ayuda de m observaciones de dichas variables a las que la función va a ajustar. La función permite hacer predicciones de Y a partir de los valores de las otras n variables.

Por ejemplo, m puede ser el número de personas en una población sometida a estudio y los componentes de cada x^i pueden ser los ingresos de la persona i, el número de años de educación, el valor de su vivienda, el número de hijos, etc. Cada y^i puede representar la cantidad de dinero que paga en su declaración de la renta.

Para encontrar el “mejor" par (a, b), necesitamos medir el error en el ajuste entre $x^{it}a + b$ e y^i para todos los i. Una posible técnica es la suma de los valores absolutos de los errores de ajuste, esto es,

$$\sum_{i=1}^{m} \mid x^{it}a + b - y^i \mid .$$

Podemos formular un problema de programación lineal para encontrar el par (a, b) que minimiza esta medida. Primero, definimos la matriz x (observada) y los vectores y (observado) y r (variables) como:

$$x = \begin{pmatrix} x^{1^t} \\ x^{2^t} \\ \vdots \\ x^{m^t} \end{pmatrix}, \quad y = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{pmatrix}, \quad r = \begin{pmatrix} r^1 \\ r^2 \\ \vdots \\ r^m \end{pmatrix}.$$

A continuación, escribimos el programa lineal como sigue:

$$\min_{a,b,r} \; z = e^t r$$

s. a $-r \leq xa + be - y \leq r$.

En esta formulación e es un vector de m unos, por lo que el objetivo es la suma de los elementos de r. Las restricciones aseguran que cada r^i no es menor que el valor absoluto $\mid x^{i^t} a + b - y^i \mid$ mientras que el hecho de que estemos minimizando la suma de los r^i asegura que cada uno de ellos se escoge no mayor de lo necesario. Así, el proceso de minimización escoge cada r^i igual a $\mid x^{i^t} a + b - y^i \mid$.

Cuando $n = 1$ (esto es, cada x^i tiene un único elemento), el problema tiene una interpretación geométrica sencilla.

Si dibujamos x^i en el eje horizontal e y^i en el eje vertical, esta formulación encuentra la recta en dimensión dos, de parámetros a y b, tal que la suma de las distancias verticales desde los puntos a la recta se hace mínima.