Optimización en redes. Flujos en redes (Network Flows NF)

Andrés Ramos
Universidad Pontificia Comillas
http://www.iit.comillas.edu/aramos/
Andres.Ramos@comillas.edu

Optimización en redes. Flujos en redes (*Network Flows* NF)

1. Terminología

2. Camino mínimo

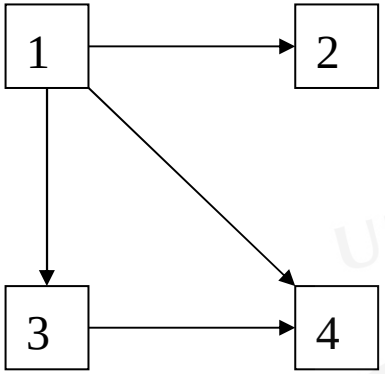
3. Árbol generador mínimo

4. Flujo máximo

5. Flujo de coste mínimo

6. Algoritmos de recorrido de grafos

Terminología

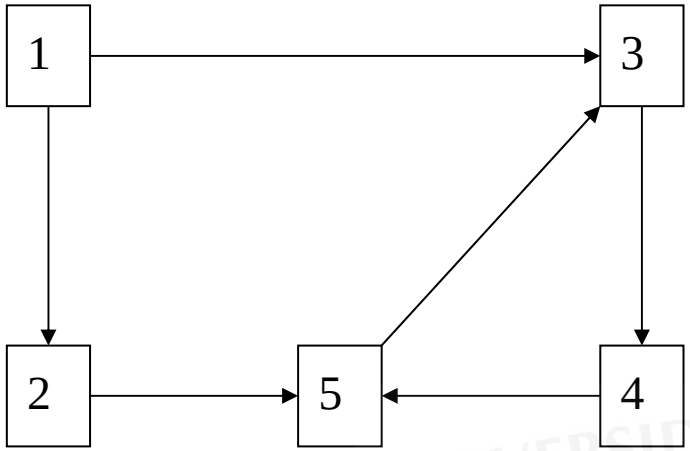


Red
Grafo

Nodos	Vértices	Nudos
Arcos	Aristas	Ramas

$$N = \{1, 2, 3, 4\}$$
$$A = \{(1, 2), (1, 3), (1, 4), (3, 4)\}$$

Arco dirigido	aquél que permite el flujo en un solo sentido
Red dirigida	conjunto de nodos unidos por arcos dirigidos
Camino	secuencia de arcos que conectan nodos
***Camino dirigido* (*camino*)**	secuencia de arcos dirigidos en el sentido del nodo final
***Camino NO dirigido* (*cadena*)**	secuencia de arcos cuyo sentido puede ser hacia o desde el nodo final
Ciclo	camino que empieza y acaba en el mismo nodo



Camino dirigido (camino) (1, 2) (2, 5) (5, 3) (3, 4)
Camino NO dirigido (cadena) (1, 2) (2, 5) (5, 4)

Ciclo dirigido (circuito) (5, 3) (3, 4) (4, 5)
Ciclo NO dirigido (ciclo) (1, 2) (2, 5) (5, 3) (3, 1)

Nodos conexos	dos nodos son conexos si la red contiene al menos una cadena entre ellos
Red conexa	red donde cada pareja de nodos es conexa
Árbol	red conexa sin ciclos
Árbol generador	red conexa sin ciclos que tiene exactamente *n*-1 arcos para recorrer todos los *n* nodos
Capacidad de un arco	máximo flujo por un arco
Nudo de generación	aquél cuyo flujo saliente excede al entrante
Nudo de demanda	aquél cuyo flujo entrante excede al saliente
***Nudo de transbordo* (*conexión*)**	aquél que conserva el flujo

Representación de un grafo (i)

Matriz de **incidencia**: filas nodos, columnas aristas,

elementos $b_{ij} = \begin{cases} 1 & \text{si el nodo } i \text{ pertenece a la arista } j \\ 0 & \text{otro caso} \end{cases}$

Grafo dirigido: -1 nodo inicial, +1 nodo final.

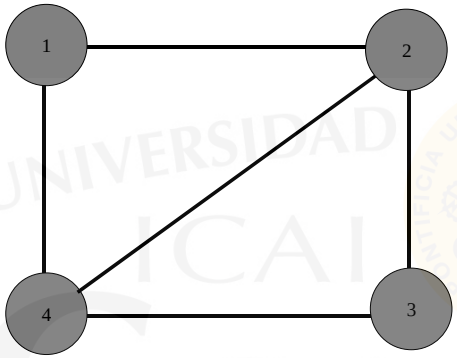
Dos elementos por columna. Matriz unimodular.

Matriz de **adyacencia**: filas nodos, columnas nodos,

elementos $a_{ij} = \begin{cases} 1 & \text{si existe un arco (arista) del nodo } i \text{ al } j \\ 0 & \text{otro caso} \end{cases}$

Matriz simétrica en caso de redes no dirigidas.

Representación de un grafo (ii)



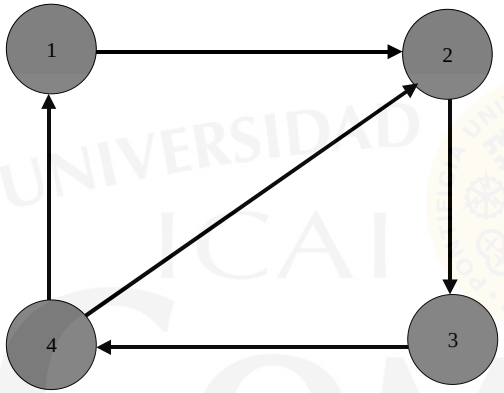
$$B = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Matriz de incidencia

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Matriz de adyacencia

Representación de un grafo (iii)



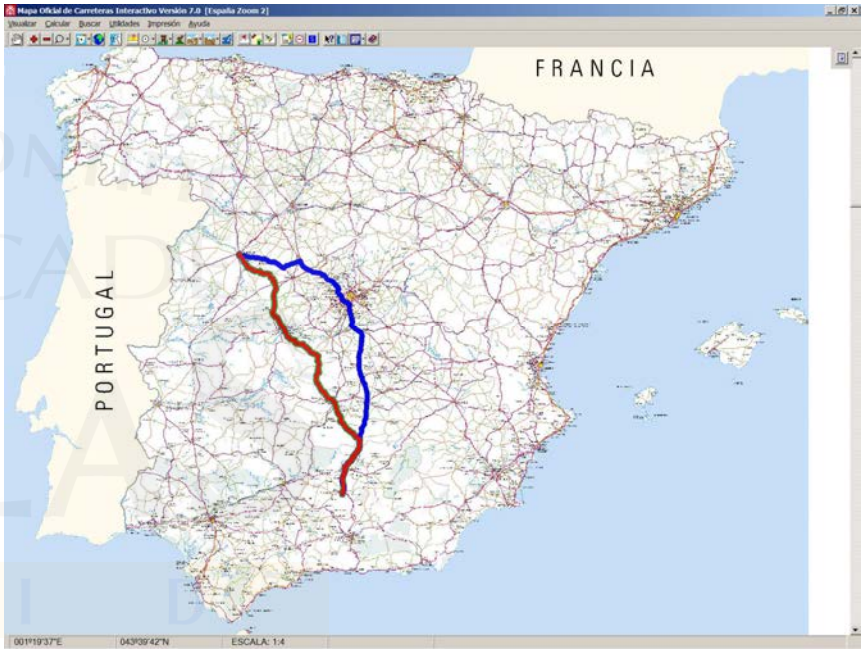
$$B = \begin{pmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \end{pmatrix}$$

Matriz de incidencia

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Matriz de adyacencia

Problema del camino mínimo



Hipótesis:

- Red conexa no dirigida
- Cada arco no dirigido lleva asociada una distancia no negativa

Otras posibilidades:

- Redes dirigidas
- Restricciones en los caminos

Objetivo:

- Encontrar la distancia (coste, tiempo, etc.) mínima entre el origen y el resto de nodos

iMetro: Subway best route calculator (http://www.iit.comillas.edu/imetro/)

Procedimiento del problema del camino mínimo

Algoritmo de Dijkstra

PASO 0: INICIO, SE ETIQUETAN LOS NODOS CON LA LONGITUD DEL ARCO QUE LOS UNE CON EL ORIGEN

$u_1 = 0$, $u_j = d_{1j}$ $j = 2,...,n$. $P = \{1\}$, $T = \{2,..,n\}$, $pred(j) = 1$, $j = 2,...,n$

PASO 1: DESIGNAR LA ETIQUETA PERMANENTE DE LA ITERACIÓN: VÉRTICE CON MENOR VALOR DE LA ETIQUETA TRANSITORIA

Buscar $k \in T$ tal que $u_k = \min_{j \in T} \{u_j\}$. Hacer $P = P \cup \{k\}$ y $T = T - \{k\}$.

Si $T = \varnothing$, parar.

PASO 2: REVISAR LAS ETIQUETAS TRANSITORIAS

$\forall j \in T$ calcular $u_j = \min\{u_j, u_k + d_{kj}\}$, si se modifica poner $pred(j) = k$. Volver al PASO 1.

Procedimiento del problema del camino mínimo

Algoritmo de Bellman-Ford

- Utilizar sólo para el ***caso de arcos con distancia negativa***

u_j^m longitud del camino mínimo del vértice 1 al vértice j usando a lo sumo m arcos

PASO 1: INICIAR CON LONGITUDES DE LOS ARCOS DEL NODO 1 A LOS DEMÁS NODOS

$u_1^1 = 0$, $u_j^1 = d_{1j} \quad j = 2,...,n$. Poner $m = 1$

PASO 2: CALCULAR LAS DISTANCIAS MÍNIMAS USANDO A LO SUMO $m+1$ ARCOS

Calcular $u_j^{m+1} = \min\left\{u_j^m, \min_{k \neq j}\left\{u_k^m + d_{kj}\right\}\right\}$.

Si $u_j^{m+1} = u_j^m \quad \forall j$, parar[1]. Si no, poner $m = m+1$ y volver al paso 2.

[1] Obsérvese que para $m = n$ seguro se cumplirá $u_j^n = u_j^{n-1}$ ya que se trata de construir un árbol.

Problema del árbol generador mínimo

Aplicaciones:

- Redes de telecomunicación, de distribución o de transporte

Hipótesis:

- Red conexa no dirigida
- Distancia no negativa en cada arco

Objetivo:

- Encontrar la cadena de longitud mínima que recorre todos los nodos sin ciclos (es decir, el árbol generador de mínimo peso).

Procedimiento del problema del árbol generador mínimo

Algoritmo de Prim

1. *Selección arbitraria de cualquier nodo y conexión al nodo diferente más cercano*
   Hacer $C_0 = \varnothing$, $\overline{C}_0 = V$. Elegir un vértice cualquiera $i \in V$ y hacer $C_1 = \{i\}$, $\overline{C}_1 = V - \{i\}$.
   Sea $k = 2$ y $T = \varnothing$.

2. *Selección del nodo no conectado más cercano a cualquier nodo conectado y establecimiento de un arco nuevo.*
   Seleccionar $j^* \in \overline{C}_{k-1}$, un nodo no conectado tal que se une a algún vértice ya conectado (de C_{k-1}) con la arista de menor peso e_{k-1}. Hacer $C_k = C_{k-1} \cup \{j^*\}$, $\overline{C}_k = \overline{C}_{k-1} - \{j^*\}$ y $T = T \cup \{e_{k-1}\}$.
   Si $k = n$, parar. Si no, poner $k = k + 1$ y repetir el paso 2.

Los empates se pueden solucionar de manera arbitraria sin pérdida del óptimo. Indican la posible existencia de múltiples soluciones óptimas.
La distancia óptima no depende de la elección del nodo inicial.

Problema de flujo máximo

Hipótesis:

- Red conexa dirigida sin bucles
- Nodos *origen* y *destino*, el resto son de transbordo
- Cada arco tiene una capacidad asociada

Objetivo:

- Maximizar el flujo total desde el origen al destino

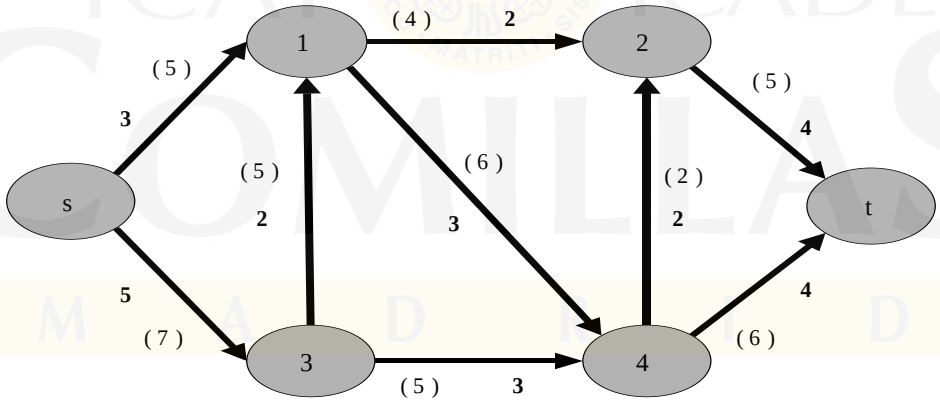
Métodos de resolución:

- Programación lineal
- Algoritmo del camino de aumento (Ford-Fulkerson)

Flujo compatible

Aquél que verifica:

- El flujo no supera la capacidad de ningún arco ($\varphi_{ij} \leq c_{ij} \quad \forall (i,j)$)
- Verifica la *ley de conservación de flujo*: $\forall i \neq s,t \quad \sum_{j/(j,i)\in U} \varphi_{ji} - \sum_{j/(i,j)\in U} \varphi_{ij} = 0$



Problema de flujo máximo: determinar el flujo compatible que transporta la mayor cantidad de flujo de la fuente (origen) al sumidero (destino).

Método del camino de aumento (Ford-Fulkerson)

Basado en tres conceptos principales: *red residual*, *camino de aumento* y *corte*

Capacidad residual de un arco:

Se indica como un número junto al nodo para su flujo saliente.

Asignar una cantidad en un arco es incrementar en un sentido y decrementar en el otro.

Red residual:

Indica el flujo sobrante de cada arco para asignarle flujo adicional sin violar su capacidad.

Camino de aumento:

Camino dirigido del origen al destino en la red residual con todos sus arcos con capacidad residual *estrictamente* positiva.

Capacidad residual del camino de aumento:

Mínimo de las capacidades residuales de los arcos.

Corte:

Partición de la red en dos subredes (componentes) cualesquiera, conteniendo una el origen y otra el destino.

Capacidad del corte:

Suma de las capacidades de los arcos que unen ambas subredes (cruzan el corte) en el sentido de origen a destino.

Flujo por un corte:

Suma de los flujos que atraviesan el corte cada uno con su sentido.

Teorema de máximo flujo mínimo corte:

Para una red conexa dirigida con un origen y un destino el valor del flujo máximo es igual al valor mínimo de las capacidades de corte de cualquier corte de la red.

En el óptimo no existen caminos de aumento en la red residual y se obtiene un corte con capacidad de corte cero en la red residual.

Algoritmo de Ford-Fulkerson

1. Obtención de la red residual a partir de la red inicial

2. Identificar un camino de aumento
   Si no existe tal camino de aumento los flujos son óptimos
3. Calcular la capacidad residual del camino de aumento c^*
4. Disminuir en c^* la capacidad residual de cada arco de dicho camino en el sentido de origen a destino. Incrementar en c^* las capacidades residuales en el sentido opuesto

Procedimiento para encontrar un camino de aumento en redes de gran tamaño.

1. Etiquetar todos los nodos alcanzables desde el origen con un solo arco con capacidad residual estrictamente positiva.
2. Repetir recursivamente el proceso para los nodos anteriores hasta etiquetar todos.

Algoritmo de Ford-Fulkerson

PASO 1: COMENZAR CON UN FLUJO COMPATIBLE Y OBTENER LAS CAPACIDADES RESIDUALES DE LOS ARCOS

Poner flujo 0 $\varphi_{ij} = 0 \; \forall (i, j) \in U$ y capacidades residuales $c^*_{ij} = c_{ij} \; \forall (i, j) \in U$.

Etiquetar $s \rightarrow (-, \infty)$

PASO 2: OBTENER UN CAMINO DE AUMENTO, INCLUSO REDIRECCIONANDO FLUJO ANTERIORMENTE ASIGNADO

Sea $i \in V$ un nodo ya etiquetado y sea $j \in V$ un nodo sin etiquetar tal que existe el arco (i, j) o el arco (j, i). Hacer según el caso:

a) Si $(i, j) \in V$ y la capacidad residual no es 0 $(c^*_{ij} > 0)$ puede aumentar flujo, calcular el mínimo de la capacidad residual de los arcos anteriores y este arco y etiquetar el nodo, es decir, calcular $\delta_j = \min\{\delta_i, c^*_{ij}\}$ y etiquetar el nodo j con $j \rightarrow (i+, \delta_j)$.

b) Si $(j, i) \in V$ y se está enviando flujo por ese arco $(\varphi_{ji} > 0)$ se puede redireccionar parte de ese flujo (se puede disminuir en ese arco), entonces calcular $\delta_j = \min\{\delta_i, \varphi_{ji}\}$ y etiquetar el nodo j con $j \rightarrow (i-, \delta_j)$.

Repetir hasta que el sumidero esté etiquetado e ir al paso 3 o hasta que no sea posible etiquetar más nodos (que se habrá acabado) e ir al paso 4.

PASO 3: AUMENTAR EL FLUJO EN EL CAMINO OBTENIDO Y RECALCULAR LAS CAPACIDADES RESIDUALES

Recorrer el camino de nodos etiquetados aumentando el flujo en los arcos etiquetados positivamente y disminuyéndolo en los etiquetados negativamente en la cantidad con que se etiquetó el sumidero δ_t (capacidad residual del camino). Recalcular las capacidades residuales (disminuir en los arcos positivos y aumentar en los negativos). Borrar las etiquetas, excepto la de la fuente, y volver al paso 2.

PASO 4: PARAR, NO ES POSIBLE AUMENTAR EL FLUJO

El flujo obtenido es máximo y el corte de capacidad mínima viene dado por los nodos etiquetados en una componente y los no etiquetados en otra (es decir, el corte son los arcos que vayan de una componente a otra).

Problema de flujo de coste mínimo

Hipótesis:

- Red conexa dirigida con *n* nodos
- Al menos un *origen* y un *destino*

Variables $x_{ij} \geq 0$ flujo a través del arco $i \rightarrow j \quad \forall i, j$

Parámetros

c_{ij} coste unitario del flujo $i \rightarrow j$

u_{ij} capacidad del arco $i \rightarrow j$

b_i flujo neto generado en el nodo i

$b_i > 0$ nodo generador.

$b_i < 0$ nodo consumidor.

$b_i = 0$ nodo de transbordo.

$$\min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

- Balance o conservación del flujo en cada nudo i

$$\sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = b_i \qquad \forall i = 1, \ldots, n$$

$$0 \le x_{ij} \le u_{ij} \qquad \forall \text{ arco } i \to j$$

Se supone que la oferta es igual a la demanda del producto $\sum_{i=1}^{n} b_i = 0$

Propiedad de soluciones enteras:

Si b_i y u_{ij} son enteros todas las variables básicas en cada solución básica son enteras.

Casos particulares del problema de flujo de coste mínimo

Problema de transporte

Problema de asignación de tareas

- Número de nodos de generación (personas) = número de nodos de demanda (tareas)
- Demanda en cada nodo consumidor = 1
- Generación en cada nodo generador = 1

Problema de transbordo

- Problema de flujo de coste mínimo sin restricciones de capacidad en los arcos.

Problema de camino mínimo

- Origen: nodo generador con generación = 1
- Destino: nodo consumidor con demanda = 1
- Red no dirigida: red dirigida con arcos en ambos sentidos
- Distancia: coste por unidad de flujo independiente del sentido
- Arcos no acotados (acotados con límite 1)

Problema de flujo máximo

- Anular los costes del flujo c_{ij} de los arcos existentes
- Seleccionar una cantidad F (cota superior del máximo flujo posible) y asignarlo como generación en el nudo origen y demanda en el nodo destino
- Añadir un arco entre O y D con coste c_{ij} elevado y capacidad ilimitada
- La minimización del coste hace que se envíe la mayor cantidad posible de flujo por los arcos de la red existente

Problema de flujo compatible con coste mínimo

En este problema se incorpora un nuevo elemento en la red, el *coste*. Se supone asociado a cada arco un coste unitario de envío de flujo a través del arco, d_{ij}. El problema consiste en enviar una cantidad de flujo conocida, θ, a través de la red desde la fuente (origen) al sumidero (destino), con coste total mínimo.

$$\min \quad \sum_{(i,j)\in U} d_{ij}\varphi_{ij}$$

$$\sum_{i/(s,i)\in U} \varphi_{ij} = \theta$$

$$\sum_{i/(i,t)\in U} \varphi_{ij} = -\theta$$

$$\sum_{i/(j,i)\in U} \varphi_{ji} - \sum_{i/(i,j)\in U} \varphi_{ij} = 0 \quad \forall j \neq s,t$$

$$0 \leq \varphi_{ij} \leq c_{ij} \quad \forall (i,j) \in U$$

Algoritmos de recorridos en grafos

Recorren los vértices y aristas de un grado con ciertas condiciones

1. Ciclo y camino euleriano
   - Problema del cartero chino

2. Ciclo hamiltoniano
   - Problema del viajante

Ciclo y camino euleriano

Ciclo euleriano: ciclo que pasa por todos los nodos y atraviesa *exactamente una vez cada arista*.

Camino euleriano: camino entre dos nodos que pasa por todos los vértices y atraviesa *exactamente una vez cada arista*.

Teorema de Euler:
Sea $G = (V, E)$ un grafo o multigrafo. G posee un *ciclo euleriano* si y sólo si G es conexo y todos sus vértices son de grado par (número de aristas que inciden en el vértice).

Un grafo G tiene un *camino euleriano* si y sólo si es conexo y tiene a lo sumo dos vértices de grado impar (que serían inicio y final del camino).

Determinar un ciclo euleriano

Idea: Determinar todos los ciclos posibles de un grafo y unirlos entre sí.

Algoritmo:
Sea $A=(a_{ij})$ la matriz de adyacencia del grafo $G=(V,E)$.

PASO 1: CONDICIÓN DE EXISTENCIA

Si existe un vértice de grado impar ($\exists i \in V$ tal que $\sum_j a_{ij}$ no es par), parar ya que no existe ciclo euleriano. En otro caso, poner $p=0$ e ir al paso 2.

PASO 2: CONSTRUCCIÓN DE UN CICLO

Poner $p=p+1$. Elegir k una fila no nula de la matriz A y buscar un ciclo de la siguiente forma:

a) Definir $n=k$ y $C=\{v_k\}$
b) Buscar m tal que $a_{nm}>0$ (es decir, que todavía sea posible ir del vértice v_n al vértice v_m.
c) Incluir el vértice v_m en C ($C=C\cup\{v_m\}$) y hacer $a_{nm}=a_{mn}=0$ (eliminar la existencia de esa arista pues no puede volver a ser utilizada).

d) Poner $n = m$. Si $n \neq k$ (no se ha cerrado el ciclo) volver al paso 2b). Si no, almacenar el ciclo C haciendo $C_p = C$ e ir al paso 3.

PASO 3: CONDICIÓN DE PARADA

Si existe alguna arista sin utilizar ($\exists a_{ij} \neq 0$) volver al paso 2. Si no, ir al paso 4.

PASO 4: FORMACIÓN DEL CICLO EULERIANO A PARTIR DE LOS CICLOS ALMACENADOS

Buscar dos ciclos almacenados C_i y C_j con al menos un vértice común, v_k. Eliminar el vértice de uno de ellos, C_i, y sustituirlo por el otro ciclo, C_j. Eliminar este último, C_j, del almacén de ciclos. Repetir el proceso hasta que quede un único ciclo, que será un ciclo euleriano.

Problema del cartero chino

Dado un grafo no dirigido $G=(V,E)$ conexo con distancias en las aristas, determinar un recorrido que pase *al menos una vez por cada arista* y cuya longitud sea mínima.

Corresponde al problema de un cartero que tiene que pasar por todas las calles de una red para repartir el correo recorriendo la menor distancia posible.

Si el grafo tiene un ciclo euleriano es evidente que ese ciclo será la solución al problema. Si no lo tiene (algún vértice es de grado impar), alguna o algunas aristas han de ser recorridas más de una vez.

Ciclo y camino hamiltoniano

Sea $G = (V, E)$ un grafo no dirigido o un multigrafo. G tiene un *ciclo hamiltoniano* si existe un ciclo elemental (no repite vértices) que contiene todos los vértices del grafo. Un *camino hamiltoniano* es un camino elemental que contiene todos los vértices de G.

Dado un ciclo hamiltoniano, la eliminación de cualquier arista da un *camino hamiltoniano.* Al revés, sin embargo, no siempre es factible.

Problema del viajante

Consiste en encontrar el circuito hamiltoniano de longitud mínima en un grafo valorado, exigiendo que visite cada vértice *exactamente una vez* o que se visite *al menos una vez*.

Grafo no dirigido: problema del viajante *simétrico.*
Grafo dirigido: problema del viajante *asimétrico.*

Algoritmo bioptimal para el TSP simétrico sobre el grafo G'

Paso previo: construir el grafo G', es decir, un grafo completo con distancias entre los nodos las distancias mínimas entre esos dos nodos en el grafo original.

PASO 1: INICIO

Seleccionar un nodo origen s. Elegir otro nodo t, que sea el de distancia mínima desde el origen ($d(s,t) \leq d(s,j) \quad \forall j \neq t$). Poner $l = t$ y guardar los nodos s y l como "visitados".

PASO 2: CONSTRUCCIÓN DEL RECORRIDO INICIAL MEDIANTE ESTRATEGIA GREEDY

Seleccionar t entre los nodos no visitados de modo que sea el de menor distancia al nodo l de los visitados. Añadir t al final del recorrido y poner $l = t$. Si hay nodos que no están en el recorrido repetir el paso 2, en otro caso, añadir s al recorrido e ir al paso 3.

PASO 3: GUARDAR EL RECORRIDO Y CALCULAR SU LONGITUD

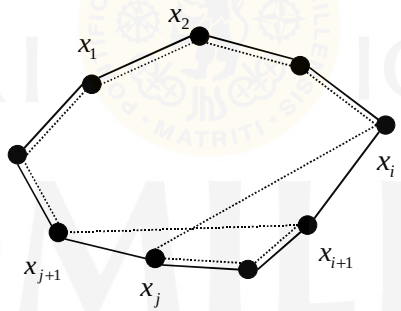
El recorrido será $\{x_1, x_2, ..., x_n, x_1\}$ y su longitud L. Poner $i = 1$.

PASO 4: INICIAR EL CAMBIO DE LIGADURAS DEL RECORRIDO

Poner $j = i + 2$.

PASO 5: EVALUAR EL CAMBIO DE LIGADURAS

Considerar el recorrido $\{x_1,...,x_i,x_j,x_{j-1},...,x_{i+1},x_{j+1},...,x_1\}$ creado por el intercambio de ligaduras (ver en la figura adjunta el recorrido inicial en trazo liso y el nuevo recorrido en trazo punteado) y obtener su longitud L'. Si $L' < L$ (es mejor este recorrido), considerar el nuevo recorrido como base para hacer los cambios y volver al paso 3. En otro caso ir al paso 6.



PASO 6: CONDICIÓN DE PARADA O VUELTA A INTERCAMBIAR LIGADURAS

Poner $j = j+1$. Si $j \le n$, ir al paso 5. Si no, poner $i = i+1$. Si $i \le n-2$, ir al paso 4. Si no, parar.