Esenciais USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

OS PROBLEMAS DE OPTIMIZACIÓN EN REDES

algúns problemas clásicos de optimización en redes e a súa resolución

Alejandro Saavedra Nieves e Balbina Casas Méndez (Departamento de Estatística, Análise Matemática e Optimización)

Problemas de optimización en redes

Neste esencial imos facer un pequeno percorrido polas versións máis sinxelas dalgúns dos problemas clásicos de optimización en redes. Debido a que gran cantidade dos asuntos que nos rodean se organizan en redes, esta clase de problemas ten infinidade de aplicacións, entre outras, as seguintes:

- deseño de rutas de vehículos ou de autoestradas, que minimicen a distancia entre dous vértices dados,
- o deseño de redes de comunicacións, coma o trazado de redes de fibra óptica de maneira que se cubran os vértices da maneira máis económica posible,
- determinar a cantidade máxima de enerxía que se pode enviar a través dunha rede eléctrica.

Matematicamente, para a modelización deste tipo de situacións é preciso introducir o concepto de grafo. Un grafo queda determinado por un conxunto de vértices ou nodos, que denotaremos por *N*, que están conectados mediante os arcos ou arestas (no caso de grafos non dirixidos) ou as conexións (no caso de seren dirixidos) no conxunto *M*. Neste contexto, denomínase fluxo a calquera ben que circule polas conexións ou arcos do grafo.

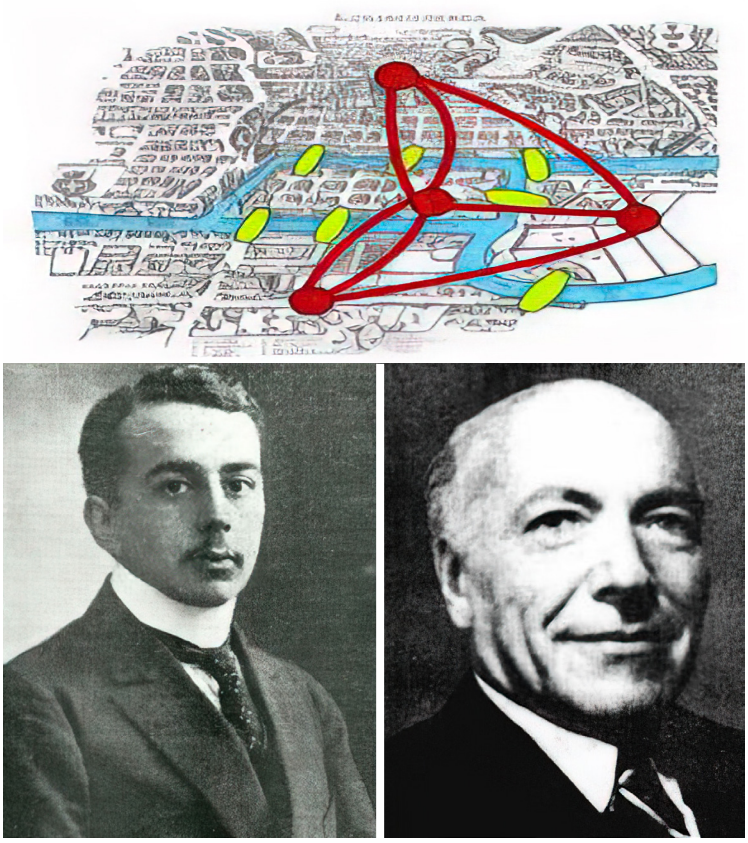
A historia da teoría de Grafos: desde o problema das pontes de Königsberg ata a intelixencia artificial

Historicamente, a orixe remóntase a 1736, ano no que Euler publicou un artigo no que resolvía o problema das pontes de Königsberg (a actual Kaliningrado).

Esta cidade tiña dúas illas e sete pontes sobre o río Pregel e o problema era atopar un camiño que empezase nun punto calquera e retornase ao mesmo punto pasando unha única vez por cada unha das sete pontes. Euler foi quen de probar que non había camiño ningún con esas propiedades.

De feito, probou que, dado un grafo calquera, o anterior problema ten solución se e só se o número de arcos que inciden en cada nodo é par.

Desde entón, os grafos e as súas aplicacións suscitaron o interese de moitos matemáticos. Sirvan como exemplo König ou Egerváry, da escola de matemáticos húngaros, que se aplicaron ao estudo dos grafos bipartitos, ou Kuhn, que propuxo un algoritmo que resolve os problemas de asignación que veremos máis adiante. Actualmente, na era da intelixencia artificial, o deseño de heurísticas e metaheurísticas é un campo de grande interese, onde o problema do axente viaxeiro, que presentamos na última sección do esencial, é un continuo referente e fonte de ideas.



Pontes sobre o río Pregel, König e Egerváry

Para a resolución desta clase de problemas, cabe destacar que se poden modelar coma un problema de programación lineal (enteira ou non). Este feito permite a resolución destes problemas mediante algoritmos xerais de programación lineal e enteira, como o símplex ou ramificación e acotación. Adicionalmente, a estrutura de rede subxacente permite definir algoritmos específicos e altamente competitivos desde un punto de vista computacional.

No que segue abordaremos diferentes situacións clásicas no contexto dos problemas de optimización en redes. Ademais de modelizalos matematicamente, ilustrarémolos mediante exemplos reais e presentaremos recursos para a súa resolución mediante o emprego do software R.

1. O problema de fluxo en redes a custo mínimo

Nesta sección estudarase o **problema de fluxo en redes con custo** mínimo. Este consiste en determinar o fluxo que pasará por cada arco de maneira que o custo asociado sexa mínimo e que se cumpran as restricións de conservación de fluxo e as impostas polas capacidades.

Cabe dicir que este é un problema moi xeral, de tal forma que os problemas que imos ver nas catro secciones seguintes son casos especiais.

Sexa $G=(N,M)$ o grafo subxacente á rede. Denotamos por M_{Oi} ao conxunto de conexións que se orixinan no nodo i e por M_{Ti} ao conxunto de ligazóns que se rematan no nodo i. Desta forma, o problema de fluxo en redes a custo mínimo admite a seguinte formulación.

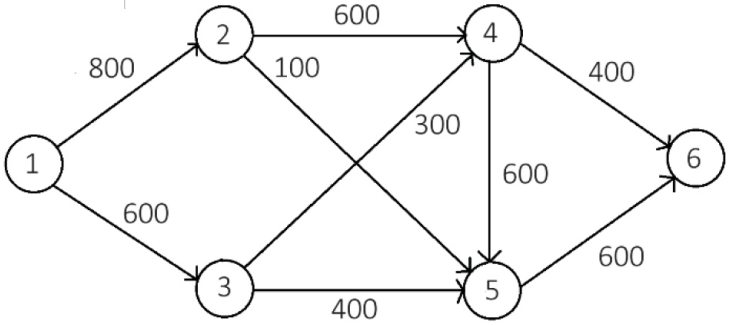
$$\begin{array}{lll} \text{minimizar} & \sum_{k\in M} c_k f_k & \\ \text{suxeito a} & \sum_{k\in M_{O_i}} f_k - \sum_{k\in M_{T_i}} f_k = 0,\ i\in N & \\ & f_k \le u_k, k\in M & \\ & f_k \ge l_k, k\in M & \end{array} \quad \text{ou} \quad \begin{array}{ll} \text{minimizar} & c\cdot f \\ \text{suxeito a} & B_{n\times m}\cdot f = 0 \\ & I_{m\times m}\cdot f \le u \\ & \mathrm{I}_{m\times m}\cdot f \ge l, \end{array}$$

onde B_{nxm} é a coñecida matriz de incidencia, que permite representar matricialmente o grafo G, e que queda definida por b_{ik}=1, se i é o nodo inicial de f_k, b_{ik}=-1, se i é o nodo final de f_k e 0, noutro caso. Ademais, $c=(c_k)_{k\in M}$ denota o vector de costes asociado a cada arco.

A función obxectivo é o custo que supón o fluxo dos f_k. Por outro lado, a primeira lista de restricións son as restricións de conservación de fluxo, unha por nodo. Estas indican que todo os fluxos que entran polas conexións incidentes nun nodo, saen polas conexións que se orixinan nel. Por cada conexión k, temos dúas restricións de capacidade, unha para as cotas superiores, μ_k, e outra para as cotas inferiores, l_k.

Exemplo

O grafo asociado ás principais rúas dunha cidade móstrase a continuación, indicando para cada conexión o número máximo de vehículos permitidos. Cada hora, uns 900 vehículos cruzan a cidade dende o nodo 1 ao 6.



Se o tempo en atravesar cada conexión se mostra na seguinte táboa, búscase minimizar o tempo total requirido polos 900 vehículos para cruzar a cidade.

Arco	*Tempo*
(1,2)	10
(1,3)	50
(2,5)	70
(2,4)	30
(5,6)	30
(4,5)	30
(4,6)	60
(3,5)	60
(3,4)	10

En primeiro lugar, introducimos en R os elementos que definen o problema. Cabe destacar que, por comodidade, introduciuse a conexión artificial (6-1) con custo 0 e capacidade mínima e máxima de 900 vehículos.

```
> nnodes<-6; nedges<-10
> costes<-c(10, 50, 70, 30, 30, 30, 60, 60, 10, 0)
```

>>>

<<<

```
> upper_bound = c(800, 600, 100, 600, 600, 600, 400, 400, 300, 900)
> lower_bound = c(rep(0,9),900)
> from = c(1, 1, 2, 2, 5, 4, 4, 3, 3, 6)
> to = c(2, 3, 5, 4, 6, 5, 6, 5, 4, 1)
> B<-matrix(0,ncol=nedges,nrow=nnodes)
> for (j in 1:nedges){B[from[j],j]<-1;B[to[j],j]<--1}
```

Para a resolución do problema de fluxo máximo a custo mínimo asociado, usamos a libraría `lpSolve` do software R, tratándoo como un problema de programación lineal básico.

```
> library(lpSolve)
> pfcm<- make.lp(nnodes, nedges)
> set.objfn(pfcm,costes)
> for (j in 1:nnodes){add.constraint(pfcm, B[j,],"=", 0)}
> set.bounds(pfcm, lower = lower_bound)
> set.bounds(pfcm, upper = upper_bound)
> solve(pfcm)
[1] 0
> get.objective(pfcm)
[1] 95000
> get.variables(pfcm)
[1] 700 200 100 600 500 400 400 0 200 900
```

Á vista da solución obtida, os 900 vehículos tardarán uns 95 000 minutos ao seguir a distribución de fluxos de vehículos que se indica: 700 vehículos, entre os nodos 1 e 2; 200, entre o 1 e o 3; 100, entre o 2 e o 5; 600, entre o 2 e o 4; 500, entre o 5 e o 6; 400 entre o 4 e 5 e o 4 e 6; 0, entre o 3 e o 5; e 200 entre o 3 e o 4.

2. O problema de fluxo máximo

Nesta sección estudarase o **problema de fluxo máximo**. Este consiste en maximizar o fluxo que pasará entre dous nodos prefixados da rede cando as conexións teñen limitada a súa capacidade.

Denótase por F ese fluxo máximo a determinar e supoñeremos, sen perda de xeneralidade, que queremos enviar fluxo do nodo 1 (nodo fonte) ao nodo n (nodo sumidoiro).

maximizar F

suxeito a

$$\sum_{k\in M_{O_1}} f_k - \sum_{k\in M_{T_1}} f_k = F$$

$$\sum_{k\in M_{O_n}} f_k - \sum_{k\in M_{T_n}} f_k = -F$$

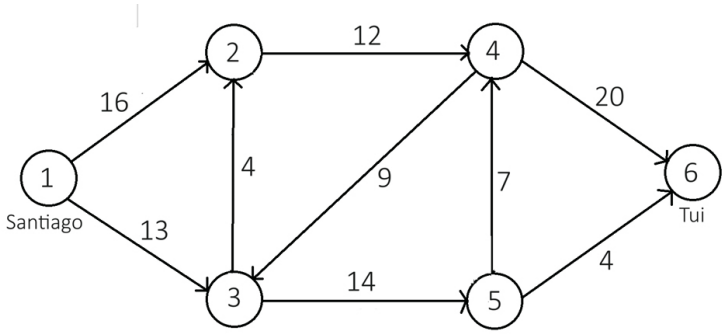
$$\sum_{k\in M_{O_i}} f_k - \sum_{k\in M_{T_i}} f_k = 0, i \neq 1, n$$

$$f_k \leq u_k, k \in M$$

$$f_k \geq 0, k \in M$$

Exemplo

Supoñamos que queremos organizar o envío de camións entre Santiago de Compostela (s) e Tui (t) e a capacidade de cada conexión no seguinte grafo asociado, representa o número de camións que poden ir pola estrada que representan cada hora.



Para a resolución do problema de fluxo máximo asociado, usamos a libraría `igraph` do software R. En *E*, gárdase a información relativa ás conexións (nodo de orixe, de destino e de capacidade).

```
>E<-rbind(c(1,2,16),c(1,3,13),c(3,2,4),c(2,4,12),c(4,3,9),c(3,5,14),c(5,4,7),
c(4,6,20),c(5,6,4))
> colnames(E) <- c("from", "to", "capacity")
> g <- graph_from_data_frame(as.data.frame(E))
> max_flow(g, source=V(g)[1], target=V(g)[6])$value
[1] 23
> max_flow(g, source=V(g)[1], target=V(g)[6])$flow
[1] 12 11 0 12 0 11 7 19 4
```

Unha vez resolto, sabemos que o fluxo máximo que soporta a rede é de 23 camións á vez, con 12 camións na conexión 1-2; 11, na conexión 1-3; 12, na conexión 2-4; 11, na conexión 3-5; 7, na 5-4; 19, na 4-6; e 4 na 5-6.

3. O problema do transporte

Nesta sección estudarase o **problema do transporte**. Para a súa definición formal introducimos a seguinte notación. Dado un grafo dirixido $G = (N, M)$, dise que G é un grafo bipartito se o conxunto de nodos se pode dividir en dous subconxuntos N_1 e N_2 tales que todos os arcos se orixinan en N_1 e rematan en N_2. Nun problema do transporte, cada nodo i de N_1 chámase nodo de subministración e terá asociada unha capacidade $s_i > 0$ e cada nodo j de N_2 chámase nodo de demanda e terá asociada unha demanda $d_j > 0$. Cada arco $k = (i, j)$ representa unha canle de distribución cun custo asociado c_k.

Sexa $G = (N, M)$ un grafo bipartito con nodos de subministración e demanda dados por N_1 e N_2, respectivamente. Dado un vector de custos c e vectores de demandas s e d, o **problema do transporte** consiste en usar os puntos de subministración para satisfacer todas as demandas a custo mínimo.

O problema de programación lineal que o modela admite a seguinte formulación.

$$\text{minimizar} \quad \sum_{k\in M} c_k f_k$$

$$\text{suxeito a} \quad \sum_{k\in M_{O_i}} f_k \le s_i, i \in N_1$$

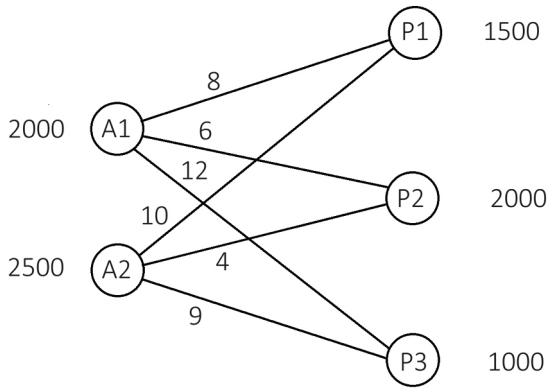
$$\sum_{k\in M_{T_j}} f_k = d_j, j \in N_2$$

$$f_k \ge 0, k \in M$$

Para que o problema teña solución, as demandas non poderán exceder as subministracións, é dicir, debe cumprirse que $\sum_{i\in N_1} s_i \ge \sum_{j\in N_2} d_j$. Tipicamente, no problema do transporte temos que os nodos de N_1 representan almacéns e os nodos en N_2 tendas ou clientes. A primeira restrición indica que do elemento i de N_1 poden saír, como moito, s_i unidades. A segunda restrición dinos que a cada elemento j de N_2 deben chegar d_j unidades.

Exemplo

Supoñamos que unha empresa produtora de barras de pan ten dous almacéns, A1 e A2, desde os que debe enviar pan a tres panadarías, P1, P2 e P3. As ofertas e as demandas (en unidades de barras de pan) e os custos de envío danse no seguinte grafo. Se identificamos cada arco k co par (i,j), onde $i \in \{1,2\}$ e $j \in \{1,2,3\}$ o problema do transporte neste caso ten un grafo como segue:



A continuación, resolvemos o problema de asignación que resulta usando a libraría `lpSolve` no software R.

```
> library(lpSolve)
> costes<-matrix(c(8,6,10,10,4,9),nrow=2,byrow=T)
> row.signs<-rep("=",2)
> row.rhs<-c(2000,2500)
> col.signs<-rep("=",3)
> col.rhs<-c(1500,2000,1000)
lp.transport(costes, "min", row.signs, row.rhs, col.signs, col.rhs)$objval
[1] 2950
> lp.transport(costes, "min", row.signs, row.rhs, col.signs, col.rhs)$solution
 [,1] [,2] [,3]
[1,] 1500 0 500
[2,] 0 2000 500
```

A solución óptima indícanos que do almacén A1 saen 1 500 unidades á panadaría P1 e 500 á P3, e do almacén A2, 2 000 unidades a P2 e 500 a P3.

4. O problema de asignación

Nesta sección estudarase o problema de asignación, que pode verse coma un caso particular do problema do transporte. Novamente, consideramos un grafo bipartito no que N_1 e N_2 teñen o mesmo número de elementos e as demandas son unitarias ($|N_1| = |N_2|$, $s = d = (1, \ldots, 1)$). Cada arco $k \in M$ ten un custo asociado c_k.

Sexa $G = (N, M)$ un grafo bipartito, particionado a través dos conxuntos N_1 e N_2. Dado un vector de custos c, o **problema de asignación** consiste en emparellar, a custo mínimo cada elemento de N_1 cun único elemento de N_2.

Para a súa modelización, consideramos o seguinte problema de programación lineal e enteira con variables binarias.

$$\begin{aligned} &\text{minimizar} && \sum_{k \in M} c_k f_k \\ &\text{suxeito a} && \sum_{k \in M_{O_i}} f_k = 1, i \in N_1 \\ & && \sum_{k \in M_{T_j}} f_k = 1, j \in N_2 \\ & && f_k \geq 0, k \in M \end{aligned}$$

O primeiro grupo de restricións indica que cada elemento de N_1 ten que ser asignado exactamente a un elemento de N_2. O segundo grupo de restricións dinos que cada elemento de N_2 ten que ser asignado a un único elemento de N_1. Por ser un caso particular do problema del transporte, a restrición de que os fluxos han de ser binarios poderá substituírse por unha condición de non negatividade.

Exemplo

Supoñamos que catro contratistas concursan para conseguir a construción de catro edificios, de maneira que cada edificio debe ser asignado a un un único contratista. O tempo que cada contratista precisa para construír cada edificio vén dado na seguinte táboa. O problema de asignación busca aquela asignación que minimiza a suma total do tempo empregado na construción dos catro edificios.

	E1	E2	E3
C1	58	48	54
C2	66	70	78
C3	106	104	95

A continuación, resolvemos o problema de asignación que resulta usando a libraría `lpSolve` no software R.

```
> library(lpSolve)
> Tempos<-matrix(c(58,48, 54, 66, 70, 78, 106, 104, 95), nrow=3, byrow=TRUE)
> lp.assign(Tempos)
Success: the objective function 219
> lp.assign(Tempos)$solution
 [,1] [,2] [,3]
[1,] 0 1 0
[2,] 1 0 0
[3,] 0 0 1
```

É dicir, ao contratista C1 foille asignado o edificio E2 ($x_{12}=1$), ao C2, o E1 ($x_{21}=1$) e ao C3 o E3 ($x_{33}=1$), cun custo asociado de 219.

5. O problema do camiño máis curto

O **problema do camiño máis curto** é un grafo dirixido $G = (N, M)$ con custos $c \in R^n$ sobre o que atopar o camiño de menor custo do nodo 1 ao nodo n de G. O custo dun camiño é a suma dos custos dos arcos que o forman.

A función obxectivo é o custo que supón o fluxo dos f_k. Por outro lado, a primeira lista de restricións son as restricións de conservación de fluxo, unha por nodo.

$$\text{minimizar} \quad \sum_{k\in M} c_k f_k$$

$$\text{suxeito a} \quad \sum_{k\in M_{O_i}} f_k - \sum_{k\in M_{T_i}} f_k = 0 \qquad i \neq 1, n$$

$$\sum_{k\in M_{O_1}} f_k - \sum_{k\in M_{T_1}} f_k = 1$$

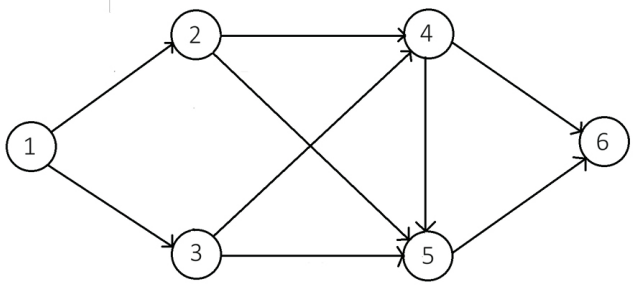
$$\sum_{k\in M_{O_n}} f_k - \sum_{k\in M_{T_n}} f_k = -1$$

$$f_k \leq 1, k \in M$$

$$f_k \geq 0, k \in M$$

Exemplo

Retomando o exemplo da Sección I, queremos saber cal é o camiño que menos tempo require para cruzar a cidade, é dicir, cal é o camiño entre o nodo 1 e o nodo 6 máis curto.



Arco	*Tempo*
(1,2)	10
(1,3)	50
(2,5)	70
(2,4)	30
(5,6)	30
(4,5)	30
(4,6)	60
(3,5)	60
(3,4)	10

>>>

<<<

Para determinar o camiño máis curto facemos uso do paquete `igraph` de R.

```
> library(igraph)
> nodes <- data.frame(id = c(1:6))
> edges <- data.frame(from = c(1, 1, 2, 2, 5, 4, 4, 3, 3),
      to = c(2, 3, 5, 4, 6, 5, 6, 5, 4),
      lower_bound = 0,
      upper_bound = 1,
      time = c(10, 50, 70, 30, 30, 30, 60, 60, 10))
> g <- graph_from_data_frame(d = edges,directed = TRUE,vertices = nodes)
> shortest_paths(g,from=V(g)[1],to=V(g)[6])
[[1]]
+ 4/6 vertices, named, from 1768974:
[1] 1 2 4 6
```

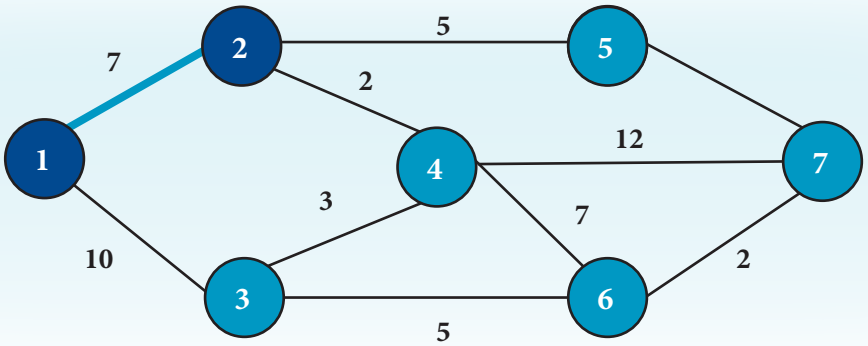
Do resultado que se proporciona, o camiño entre os nodos 1 e 6 que menos tempo require é o que pasa polos nodos 2 e 4.

6. O problema da árbore de expansión mínima

Para a definición do **problema da árbore de expansión mínima** (en inglés, *minimum spanning tree*, abreviadamente MST), pártese dun grafo non dirixido no que os distintos arcos teñen asignado un custo. Este problema consiste en seleccionar un conxunto de arcos, que constitúa unha árbore, que involucre a todos os nodos do grafo e cuxo custo sexa mínimo. O problema de programación lineal que o modela admite a seguinte formulación.

$$\begin{aligned} &\text{minimizar} && \sum_{k\in M} c_k f_k \\ &\text{suxeito a} && \sum_{k\in M} f_k = n-1 \\ & && \sum_{k\in M(X)} f_k \le |X|-1, X \subsetneq N \\ & && f_k \in \{0,1\} \end{aligned}$$

Para cada arco temos que elixir se o introducimos na árbore ou non, o que dá lugar a un problema de programación enteira con variables binarias. A función de custo é a habitual e está encamiñada a minimizar os custos de conexión. O número de arcos é exactamente n-1 e ningún subconxunto de N estará conectado mediante tantas arestas como nodos ten o conxunto e desta forma asegurámonos que non haberá ciclos. Grazas a estas restricións a estrutura final será unha árbore. Aínda que este é un problema altamente combinatorio a súa estrutura permite resolvelo mediante algoritmos moi rápidos.



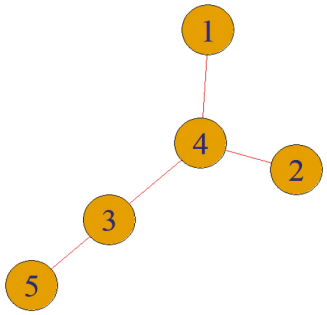
Exemplo

Coa idea de mellorar a velocidade das conexións a internet desde os distintos edificios do campus, a Universidade de Santiago de Compostela decidiu redobrar o cableado de fibra óptica. Para iso, quere conectar un novo cableado de última xeración que, para ser efectivo, deberá conectar entre si a todos os 5 edificios do campus. Doutra banda, dada a delicada situación económica que atravesa a institución, non quere gastar máis do necesario na nova instalación. O custo de instalar o novo cableado entre calquera par de edificios é proporcional á distancia entre eles.

Usando o software R, cargamos a libraría `igraph`, introducimos as conexións que compoñen o grafo e as distancias de cada conexión. Co comando `mst()` obtemos a solución do problema que tamén podemos visualizar.

Árbore de expansión mínima

```
>library(igraph)
>g <- make_graph(edges = c(1,2, 1,3, 1,4, 1,5, 2,3, 2,4,
2,5, 3,4, 3,5, 4,5), n=5, directed = FALSE)
>distancias<-c(42, 98, 37, 106, 61, 30, 64, 61, 10, 71)
>mst(g, weights=distancias, algorithm=NULL)
[1] 1--4 2--4 3--4 3—5
>g <- make_graph(c(1,4, 2,4, 3,4, 3,5), n = 5, directed=
FALSE) %>%
>set_edge_attr("color", value = "red")
>plot(g, vertex.size=40, vertex.label.cex=3)
```



Observando os datos introducidos en R, vemos que é posible conectar entre si directamente calquera par de edificios, a diferentes custos. A solución óptima reside en conectar ao edificio 4, os edificios 1, 2 e 3 e conectar tamén a este último o edificio 5.

Como mostra dun algoritmo específico para resolver problemas de optimización en redes presentamos, pola súa sinxeleza, o algoritmo de Prim que resolve o problema da árbore de expansión mínima. Este algoritmo aplicado ao exemplo anterior proporciona unha solución idéntica á brindada por R.

Algoritmo de Prim

Paso 1.	Eliximos un nodo arbitrario i e definimos o conxunto $X=\{i\}$.
Paso 2.	Buscamos un nodo de $N\backslash X$ de entre os máis próximos a X, sexa este nodo j. Engádese o nodo j a X. Engádese á árbore o arco que une j a X a mínimo custo. Repítese o PASO 2 mentres que X sexa distinto de N.

7. O problema de emparellamento de máxima cardinalidade

No **problema de emparellamento de máxima cardinalidade** (*maximal matching* en inglés) búscase un emparellamento dous a dous dos nodos dun grafo bipartito non dirixido, por medio de arestas que non teñan nodos en común e conseguindo emparellar o maior número de nodos. Este problema pódese transformar nun de fluxo máximo ou resolverse directamente por medio dun modelo de programación enteira binaria. O modelo móstrase a continuación:

$$\text{maximizar} \quad \sum_{k\in M} f_k$$

$$\text{suxeito a} \quad \sum_{k\in M,\ i\in k} f_k \le 1, i \in N$$

$$f_k \in \{0,1\}$$

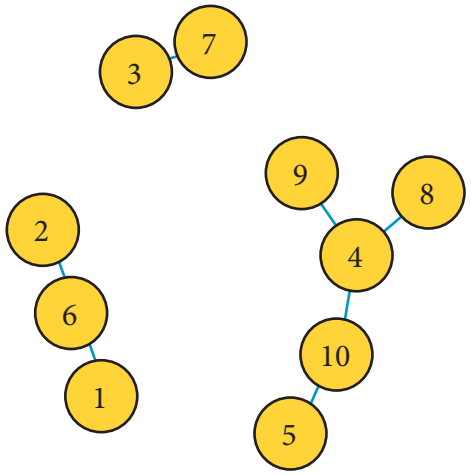
Para cada arco temos que elixir se o introducimos no emparellamento ou non, o que dá lugar a un problema de programación enteira con variables binarias. A función obxectivo está encamiñada a maximizar o número de nodos emparellados e as restricións impiden que cada nodo forme parte de máis dun arco do emparellamento.

Exemplo

Exemplo. Unha compañía que manufactura altofalantes estéreo debe asociar 10 altofalantes individuais en parellas antes de poder vendelos como un *pack*. O funcionamento dos altofalantes depende da súa resposta de frecuencia. A compañía detectou sete posibles parellas por mor de garantir unha pequena diferenza entre as respostas dos dous altofalantes do *pack* a certas frecuencias. Deséxase seleccionar as parellas de altofalantes de modo que se maximice o número deles que se poñen á venda.

Coa libraría `igraph` de R debuxamos o grafo cos 10 nodos (altofalantes) e os sete arcos (posibles parellas de calidade). Como cada altofalantes só pode entrar nun *pack* coa libraría `lpSolveAPI` obtemos o emparellamento de máxima cardinalidade.

Grafo de posibles parellas de altofalantes




```
>library(igraph)
> g <- make_graph(c(1,6, 2,6, 3,7, 4,8, 4,9,
4,10, 5,10), n = 10, directed=
+ FALSE) %>%
+ set_edge_attr("color", value = "red")
> plot(g, vertex.size=40, vertex.label.cex=3)
> library(lpSolveAPI)
> #arco 1: (1:6)
> #arco 2: (2:6)
> #arco 3: (3:7)
> #arco 4: (4:8)
> #arco 5: (4:9)
> #arco 6: (4:10)
> #arco 7: (5:10)
> altofalantes<- make.lp(3, 7)
> set.objfn(altofalantes,c(1,1,1,1,1,1,1))
> set.row(altofalantes,1,c(1,1,0,0,0,0,0))
> set.row(altofalantes,2,c(0,0,0,0,0,1,1))
> set.row(altofalantes,3,c(0,0,0,1,1,1,0))
> set.rhs(altofalantes,c(1,1,1))
> set.constr.type(altofalantes,rep("<=",3))
> set.type(altofalantes,c(1,2,3,4,5,6,7),
"binary")
> lp.control(altofalantes,sense="max")
> solve(altofalantes)
[1] 0
> get.variables(altofalantes)
[1] 0 1 1 0 1 0 1
```

Como vemos na solución proporcionada por R, poñeríanse á venda 4 *packs* cos altofalantes 2 e 6, 3 e 7, 4 e 9 e 5 e 10 e os altofalantes 1 e 8 non serían seleccionados.

O problema de emparellamento pódese ver como unha xeneralización do problema de asignación onde o grafo subxacente non é bipartito. Ademais está relacionado con outros problemas. Un deles e con moitas aplicacións é o de cobertura de vértices que consiste en determinar un conxunto de nodos que inclúa, polo menos, un extremo de cada aresta do grafo. Unha *cobertura mínima de vértices* é un tal que ningún outro teña menos vértices. Para o caso dun grafo bipartito, debemos a König o seguinte resultado.

Hai moitas demostracións deste teorema e unha fai uso do problema do fluxo máximo. No exemplo anterior é fácil ver que unha cobertura de vértices mínima é {3, 4, 5, 6}. Existen outras variantes do problema de emparellamento e todas elas contan con interesantes aplicacións prácticas ademais de que algunhas delas constitúen unha etapa dentro de algoritmos que resolven outros problemas máis complexos, como o que presentamos a continuación.

Teorema

En calquera grafo bipartito, o número de arestas nun emparellamento máximo é igual ao número de vértices nunha cobertura de vértices mínima.

8. O problema do axente viaxeiro

Nesta sección abórdase o **problema do axente viaxeiro**. Para iso, introducimos algúns conceptos necesarios. Un *circuíto hamiltoniano* nun grafo G é unha secuencia de vértices na que só o primeiro e o último coinciden e que contén a todos os vértices de G. Un grafo que contén a un circuíto hamiltoniano chámase grafo hamiltoniano.

O problema do axente viaxeiro (en inglés, *traveling salesman problema* ou *TSP*) consiste en determinar un circuíto hamiltoniano en G que minimice o tempo que se tarda en percorrelo. Na práctica, o axente viaxeiro ten que visitar todos os nodos (as cidades) e quere tardar o menor tempo posible sen repetir cidades.

O problema de programación lineal que o modela admite a formulación da dereita.

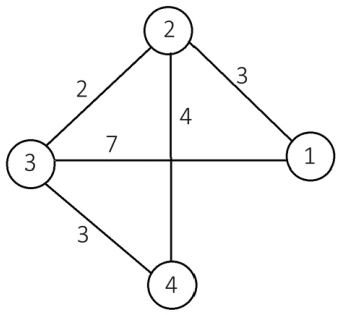
Búscase minimizar o fluxo na rede, de tal maneira que se satisfaga a conservación de fluxo. Isto é, se o axente visita unha cidade, este debe saír e non volver visitala. A terceira das restricións permite evitar a formación de subciclos.

$$\begin{aligned} &\text{minimizar} && \sum_{k\in M} c_k f_k \\ &\text{suxeito a} && \sum_{k\in M_{O_i}} f_k = 1, i \in N \\ & && \sum_{k\in M_{T_j}} f_k = 1, j \in N \\ & && \sum_{k\in M(X)} f_k \leq |X| - 1, X \subsetneq N \\ & && f_k \in \{0,1\} \end{aligned}$$

A diferenza dos problemas das seccións anteriores, o problema do axente viaxeiro é ben coñecido pola dificultade da súa resolución a pesar da sinxeleza da súa formulación.

Exemplo

Sexa G o grafo que se mostra a continuación e que describe un problema do axente viaxeiro, en termos das posibles arestas e os seus custos asociados. O noso axente conduce o camión dunha cooperativa agrícola galega e debe visitar as cidades nas que vende os produtos da cooperativa. Así debe visitar 4 cidades a custo mínimo, de forma que visite todas as cidades sen repetir ningunha e volvendo ao punto do que partiu.



A continuación, resolvemos o problema do axente viaxeiro que resulta usando a libraría `TSP` no software R. En primeiro lugar, construímos a matriz de distancias entre os nodos, facendo que esta resulte simétrica (é dicir que a distancia entre i e j sexa a mesma que entre j e i). Dado que a conexión (1-4) non existe, asignámoslle un custo infinito.

```
> costes<-c(3,2,4,3,7)
> from = c(1, 2, 2, 3, 1)
> to = c(2, 3, 4, 4, 3)
> M<-matrix(Inf,ncol=4,nrow=4)
> for (j in 1:5){
+ M[from[j],to[j]]<-costes[j]
+ M[to[j],from[j]]<-costes[j]
+ }
```

Para obter a solución, facemos uso da función `TSP()`.

```
> library(TSP)
> tps=TSP(M)
> axente=solve_TSP(tps,"two_opt")
> names(axente)
[1] "2" "1" "3" "4"
> axente
object of class 'TOUR'
result of method 'two_opt' for 4 cities
tour length: 17
```

>>>

<<<

Pódese comprobar que o axente, se comeza na segunda cidade, debe volver á primeira, para despois visitar a terceira e a cuarta respectivamente.

A libraría ten implementados diversos algoritmos de resolución. A continuación resolvemos o exemplo usando un algoritmo de inserción no canto do 2-opt. O custo total do circuíto é o mesmo coas dúas solucións: 17.

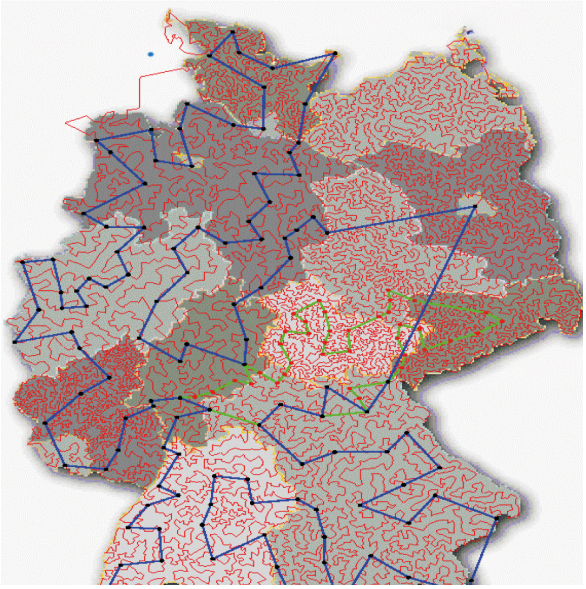
```
> axente=solve_TSP(tps,"cheapest_insertion")
> names(axente)
[1] "3" "4" "2" "1"
> tour_lenght(axente)
> axente
object of class 'TOUR'
result of method 'cheapest_insertion' for 4 cities
tour length: 17
```

Cabe dicir que na actualidade non se conseguiron resolver de maneira exacta problemas do axente viaxeiro con máis de 86 000 cidades. Os algoritmos aproximados e, sobre todo, os heurísticos son os máis utilizados na práctica para resolver o problema do axente viaxeiro e as súas variantes, incluíndo os denominados xenericamente problemas de rutas de vehículos de grande importancia na organización industrial. Finalizamos cunha interesante recensión histórica do problema do axente viaxeiro.

A primeira referencia coñecida ao problema do axente viaxeiro atópase no manual *Der Handlungsreisende - wie er sein soll und was er zu thun hat, um Auftraege zu erhalten und eines gluecklichen Erfolgs in seinen Geschaeften gewiss zu sein - Von einem alten Commis-Voyageur,* publicado en 1832. Este manual non contén un tratamento matemático do TSP, pero si unha descrición precisa do problema. Nun artigo de Schrijver, preséntase un exemplo dun TSP de 45 cidades do manual alten Commis-Voyageur.

As cidades alemás aparecen de novo na solución récord do TSP de Groetschel para un caso de 120 cidades en 1977 e en 2001 Applegate, Bixby, Chvátal, and Cook atopan unha viaxe óptima para 15 112 cidades de Alemaña.

Na figura presentamos as tres viaxes alemás. O percorrido do alten Commis-Voyageur aparece en verde, o de Groetschel en azul e o das 15 112 cidades en vermello.



Conclusións

Os grafos constitúen unha poderosa ferramenta que unida ás técnicas de investigación operativa serve de apoio á toma de decisións en moitos problemas que aparecen na vida real e suscita multitude de problemas matemáticos de grande interese. Hai outros problemas que non se tratan aquí como o chamado da mochila, coloreado de grafos... Contamos ademais con ferramentas informáticas de libre acceso que poden facilitar a representación e resolución destes problemas. Para profundar e ampliar os contidos aquí presentados pódese facer uso de bibliografía como a que citamos máis abaixo.

Referencias:

[1] Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. (1993). Network Flows: Theory, Algorithms and Applications. Prentice Hall.
[2] Diestel R. (2017 , 5ª edición). Graph Theory. Springer.
[3] Hillier, F. & Lieberman, G. (2010). Introducción a la Investigación de Operaciones. McGraw-Hill.
[4] Korte, B. & Vygen, J. (2001, 2ª edición) Combinatorial Optimization. Theory and Algorithms. Springer.
[5] Papadimitriou, C. H. & Steiglitz, K. (1998). Combinatorial Optimization: Algorithms and Complexity. Dover.

https://dx.doi.org/10.15304/_____

ISBN 978-_____