Capítol 3: Recorreguts, camins i arbres generadors òptims

Matemàtica discreta
Grau d'Enginyeria Informàtica
Escola d'Enginyeria
Universitat Autònoma de Barcelona

Contingut

Motivació

Per saber si existeix algun camí possible entre dues ciutats (vèrtexs) en una xarxa d'aerolínies (graf) i quina és la ruta més econòmica, moltes vegades cal examinar l'estructura del graf i trobar el camí de cost mínim.
Per tal d'examinar el graf sovint cal fer-ne un recorregut de manera sistemàtica.

Exploració de grafs

Per recórrer un graf considerarem dues estratègies, les més comunes.

1. La **Breadth First Search (BFS)**: coneguda com a recerca per amplada prioritària. Dóna preferència a l'exploració en paral·lel de totes les alternatives possibles des del vèrtex on ens trobem.
2. La **Depth First Search (DFS)**: coneguda com a recerca en profunditat prioritària. En aquest cas es dóna preferència a l'obertura d'una única via d'exploració a partir del vèrtex actual.

Tots dos tenen una complexitat $O(m) \leq O(n^2)$.

Exploració de grafs

BFS

1. Etiqueteu v_1 amb un 0.
2. $j = 0$.
3. A partir del vèrtex etiquetat j, cerqueu tots els vèrtexs no etiquetats adjacents. Si no hi ha cap vèrtex adjacent ja hem acabat.
4. Etiqueteu tots els vèrtexs trobats en el pas anterior amb $j + 1$.
5. $j = j + 1$, torneu a 3.

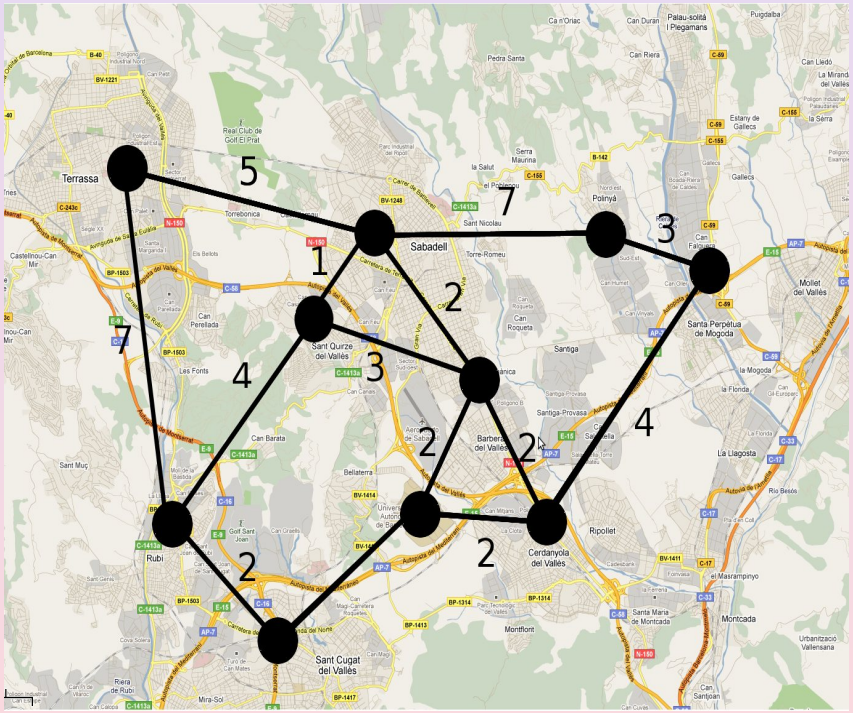
Exploració de grafs

DFS

1. Inicialment totes les connexions aresta-vèrtex no estan etiquetades.
   $v = v_1$
2. Si a v totes les connexions són etiquetades, aneu a 4.
3. Escolliu una connexió no etiquetada, etiqueteu-la amb un 0 i useu l'aresta per arribar a l'altre extrem v_i.
   Si v_i té alguna connexió etiquetada, etiqueteu la connexió d'entrada v_i amb un 0 i torneu enrere cap a v.
   Si no, etiqueteu la connexió d'entrada a v_i amb un 1; $v = v_i$.
   Torneu a 2.
4. Si totes les connexions tenen una etiqueta diferent d'1, hem acabat.
5. Si no, useu la connexió etiquetada amb un 1 per anar cap a l'altre extrem v_i
   $v = v_i$ i torneu a 2.

Camins de cost mínim

L'objectiu dels **camins de cost mínim** és determinar el camí òptim que uneix dos vèrtexs determinats.
Un exemple típic d'un graf és una xarxa de carreteres que connecten un conjunt de ciutats.



Camins de cost mínim

El problema algorísmic més natural en aquest graf és trobar el camí més curt o de cost mínim que uneix dues ciutats s (start) i t (target) en un graf ponderat.

Un graf ponderat es denota per (G, w) on $G = (V, A)$ és un graf i w és una funció $w : A \to \mathbb{R}$ que assigna pesos a les arestes del graf. Els pesos, en aquest exemple, ens poden indicar els costos que volem avaluar, ja sigui km, temps que cal emprar, preu dels peatges, etc.

Camins de cost mínim

A partir d'un **graf ponderat** (G, w) i un camí $C : v_0, v_1, ..., v_k$ definim el **pes del camí** C com

$$w(C) = \sum_{i=1}^{k} w(v_{i-1}, v_i),$$

i la **distància entre dos vèrtexs** $u, v \in V$ com

$$d_G(u, v) = min\{w(C) \mid C \text{ és un } u - v \text{ camí}\}.$$

Camins de cost mínim

Per trobar el camí de cost mínim en (G, w) amb $w \geq 0$ dirigit entre dos vèrtexs s i t a partir del vèrtex inicial v_0 utilitzarem l'algorisme de **Dijkstra**.

En canvi, si volem trobar el cost mínim entre totes les parelles de vèrtexs cal utilitzar l'algorisme de **Floyd**.

Dijkstra

1. $\lambda(s) = 0; \lambda(v_j) = \infty \quad \forall v_j \neq s$..
2. $T = V$.
3. Sigui $v_k \in T$ tal que $\lambda(v_k)$ és mínima.
4. Si $v_k = t$, ja hem acabat.
5. $\forall a_i = (v_k, v_q)$, si $v_q \in T$ i $\lambda(v_q) > \lambda(v_k) + C(a_i)$ aleshores $\lambda(v_q) = \lambda(v_k) + C(a_i)$.
6. $T = T - \{v_k\}$. Torneu a 3.

La complexitat és $O(n^2)$.

Floyd

1. $m = 1$.
2. Per a tota parella v_i, v_j, tal que $1 \leq i, j \leq n$, calculeu:

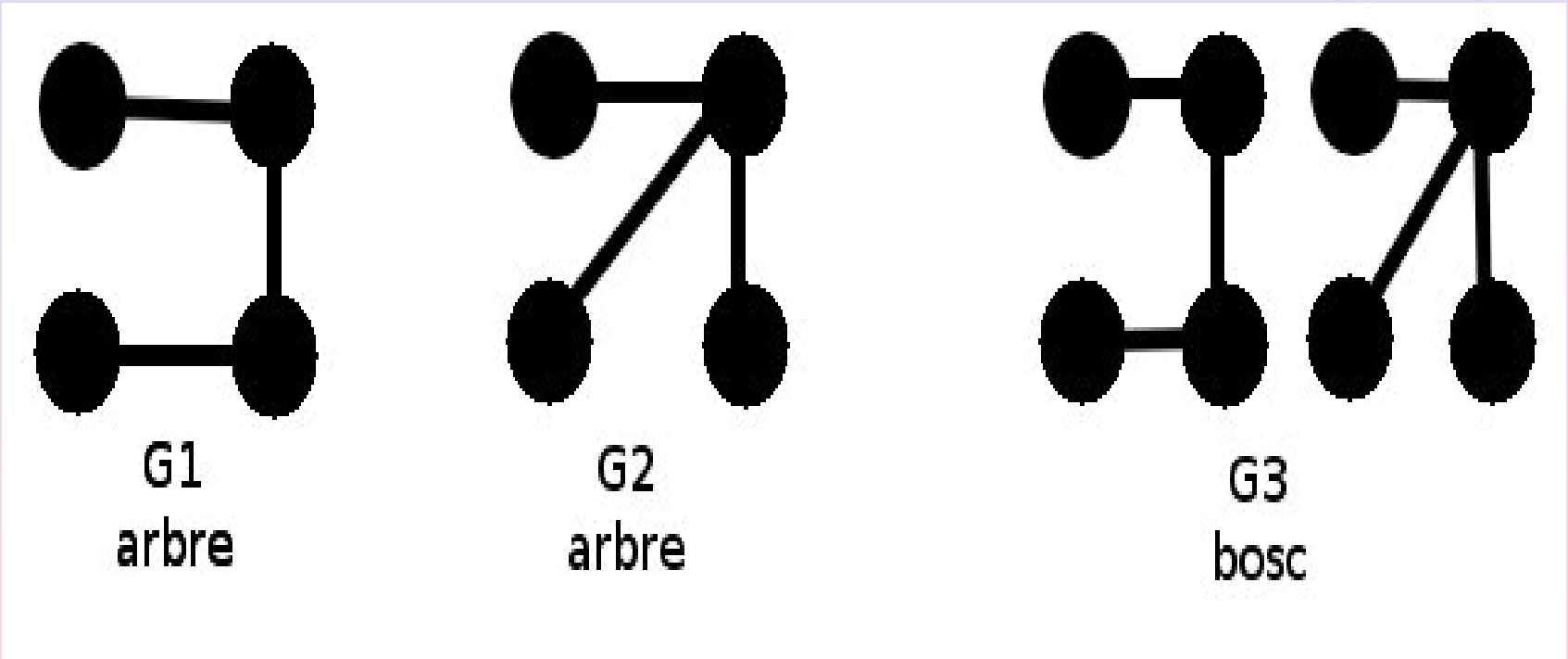
$$\lambda_{ij}^{m} = min\{\lambda_{ij}^{m-1}, \lambda_{im}^{m-1} + \lambda_{mj}^{m-1}\}$$

3. Si $m = n$, hem acabat. Si no, $m = m + 1$ i torneu a 2.

On n és l'ordre del graf que estem avaluant.
La complexitat és $O(n^3)$. Menor que si s'aplica Dijkstra n^2 vegades. Però aplicant Dijkstra n vegades és suficient si cada vegada explorem tots els vèrtexs. Aleshores la complexitat és la mateixa.

Caracterització dels arbres

Un arbre simètric és un graf simètric connex sense cicles. Si eliminem la condició de connectivitat obtenim un bosc.



Observeu que cada component d'un bosc és un arbre.

Caracterització dels arbres

Sigui G un graf d'ordre n i mida m.

Teorema 1.

G és un arbre $\Leftrightarrow$ entre cada parella de vèrtexs diferents de G hi ha camí únic.

Teorema 2.

G és un arbre $\Leftrightarrow$ no conté circuits i $n = m + 1$.

Teorema 3.

G és un arbre $\Leftrightarrow$ és connex i $n = m + 1$.

Arbres generadors òptims

A partir d'un graf $G(V, A)$ tot subgraf de G que sigui un arbre i contingui tots els vèrtexs V s'anomena **arbre generador** de G.

Sigui $G(V, A)$ simètric amb un cost no negatiu $c(a_k)$ associat a cada aresta $a_k = (v_i, v_j) \in A$.
Per determinar a G un arbre generador simètric de cost total mínim, que també podem anomenar òptim, podem utilitzar l'algorisme de **Kruskal** o bé el de **Prim**. Aquests dos algorismes es basen en la següent propietat:

Teorema 4.

A partir d'un graf $G(V, A)$ simètric i connex, sigui $U \subset V$ i $a_m = (v_i, v_j)$ una aresta de cost mínim entre aquelles que tenen $v_i \in U$ i $v_j \in V - U$. Aleshores, existeix un arbre generador de cost mínim a G que conté a_m.

Arbres generadors òptims

Kruskal

1. Comenceu amb un graf $G_0(V_0 = V, A_0 = \emptyset)$.
   $k = 0$
2. Ordeneu les arestes de G de forma creixent en funció del cost.
3. Començant pel cap de la llista ordenada, preneu la primera aresta a^* no usada i que NO forma circuit a G_k.
   $k = k + 1$
   $G_k = G_{k-1} \cup \{a^*\}$
4. Si $k < n - 1$ torneu a 3.
   Si no, heu acabat.

La complexitat és $O(m \log m) \leq O(n^2 \log n)$. L'algorisme de Prim té complexitat $O(n^2)$. Depenent de si el graf té moltes arestes o poques interessarà més usar un algorisme o l'altre.