Problemes resolts 4

4.1. (Xarxes ad-hoc)

Les xarxes ad-hoc, composades per dispositius sense fils de baixa potència, s'han proposat per situacions com els desastres naturals en què els coordinadors dels treballs de rescat podrien controlar les condicions en zones de difícil accés. La idea és que una gran col·lecció d'aquests dispositius sense fils es podria llançar des d'un avió en una regió per a continuació reconfigurar-se com una xarxa operativa.
Estem parlant de: (a) dispositius relativament barats, el quals (b) es llancen des d'un avió a (c) un territori perillós; i per la combinació de (a), (b) i (c), es fa necessari fer front a la fallida d'un nombre raonable dels dispositius.
Ens agradaria que fos el cas que si un dels dispositius v detecta que està en perill de fallar, transmetés una representació del seu estat a un altre dispositiu a la xarxa. Cada dispositiu té un abast de transmissió limitat, pot comunicar-se amb altres dispositius que es troben com a màxim a d metres d'ell. Com que no volem que es transmeti el seu estat a un dispositiu inoperant, hem d'incloure una mica de redundància: Un dispositiu v ha de tenir un conjunt de k altres dispositius en un radi de d metres de distància. Anomenarem a aquest conjunt una *còpia de seguretat* per al dispositiu v.
Suposeu que després del llançament podem obtenir les coordenades $p_i = (x_i, y_i)$ dels n dispositius operatius que formen la xarxa inicial.
Dissenyeu un algorisme per determinar si és possible triar una còpia de seguretat per a cada dispositiu, amb la propietat addicional que, per algun paràmetre donat b, cap dispositiu apareix en la còpia de seguretat de més de b altres dispositius. L'algorisme ha de proporcionar com a sortida també els conjunts de còpia de seguretat, sempre que es puguin trobar.

Una solución: Resolveremos el problema reduciendolo a un problema de flujo máximo. Para ello, a partir de la entrada, construimos una red $\mathcal{N}$:

- Nodos $V = \{s, t, u_1, \ldots, u_n, v_1, \ldots, v_n\}$
- Aristas y capacidades:
  - Para $i \in [n]$, (s, u_i) con capacidad k y arco (v_i, t) con capacidad b.
  - Para $i, j \in [n]$ con $i \neq j$ y $d(p_i, p_j) \leq d$, (u_i, u_j) con capacidad 1.

Vamos a relacionar una solución de nuestro porblema con flujos en esta red y viceversa.

Supongamos que podemos seleccionar copias de seguridad C_i para todo $i \in [n]$ verificando las restricciones. En este caso, para $j \in C_i$ sabemos que $d(p_i, p_j) \leq d$ y por tanto $(u_i, v_j) \in E$. Definimos la siguiente asignación de flujo: para $i \in [n]$, $f(s, u_i) = k$ y, para $(u_i, v_j) \in E$, $f(u_i, v_j) = 1$ si $j \in C_i$ y $f(u_i, v_j) = 0$ si $j \in C_i$; para $j \in [n]$, $f(u_j, t) = \sum_{i \in [n] i \neq j} f(u_i, v_j)$. Al cumplirse las restricciones, la asignación f es un flujo válido con valor kn.

Supongamos ahora que en la red tenemos un flujo válido f con valor kn. Definimos para $i \in [n]$ el conjunto $C_i = \{j \mid (u_i, v_j) \in E \text{ and } f(u_i, v_j) = 1\}$. Como el valor del flujo es kn, cada vértice v_i recibe k unidades de flujo, por la ley de conservación del flujo, tenemos que $|C_i| = k$. Para la segunda restriccion, observemos que $|\{i \mid j \in C_i\}|$ coincide con el flujo de entrada en v_j, de nuevo por la ley de conservación de flujo esta cantidad tiene que coincidir con $f(v_j, t)$. Finalmente, al ser un flujo válido tenemos $|\{i \mid j \in C_i\}| = f(v_j, t) \leq c(v_j, t) = b$ y se cumple la segunda restricción.

Por lo tanto tenemos que el problema planteado tiene solución si y solo si el valor del flujo máximo en la red construida es kn. Además la última parte de la demostración nos proporcina una manera de obtener una soliución válida cuando este flujo existe a partir d eun flujo con valores enteros. Así tenemos el siguiente algoritmo:

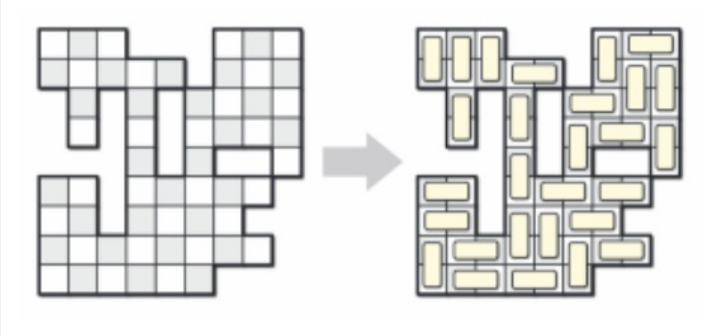
Construir $\mathcal{N}$
$F = MaxFlowFF(\mathcal{N})$
if $|F| < kn$ **then return** No es posible
Obtener $C[i] = \{j \mid (u_i, v_j) \in E \text{ and } f(u_i, v_j) = 1\}$ **return** C

Utilizando Ford-Fulkerson el número máximo de augmentations es kn y podemos resolver el problema en $O(kn(|V| + |E|)$ es decir $O(kn^3)$. Una vez obtenido este flujo f, para $i \in [n]$ podemos obtener el conjunto $C[i]$ con un recorrido de los arcos de G en $O(n^2)$ time.

El algoritmo propuesto tiene coste $O(kn^3)$.

4.2. **(Dòmino)**

Tenim un tauler $n \times n$, amb caselles de dos colors (blanc i negre) alternants per files i columnes. Esborrem un determinat nombre de caselles dels dos colors, de manera que al tauler resultant quedi el mateix nombre de caselles blanques i negres. Descriviu i analitzeu un algorisme per a determinar eficientment si existeix una forma d'omplir el tauler amb fitxes de dòmino (que ocupen un àrea de 2×1 caselles), de manera que totes les caselles que han quedat al tauler quedin cobertes i cap fitxa de dòmino surti del tauler.



A l'exemple de la figura, el tauler percolat de l'esquerra es pot omplir (completament i correctament) amb fitxes de dòmino; per tant, la resposta en aquest cas és positiva. Raoneu com, en cas de respostes afirmatives, es podria calcular la col·locació exacta de les fitxes de dòmino.

Ajut: Penseu que una fitxa de dòmino col·locada sobre el tauler sempre ocuparà una casella blanca i una casella negra que serà adjacent a la blanca. De totes les possibles caselles negres adjacents, l'algorisme haurà decidir quina.

Una solució:

Cobrir dues caselles amb una fitxa de dòmino és equivalent a ocupar una casella blanca i una negra que comparteixen algun costat. Per tant, cobrir el tauler percolat amb fitxes de dòmino és el mateix que aparellar totes les caselles blanques i negres de manera que totes les caselles queden aparellades i cap casella és part de més d'un parell.

Com a graf, podem representar totes les caselles negres com a nodes $b_1, \ldots, b_k$ i totes les caselles blanques com a nodes $w_1, \ldots, w_k$ (on k és el nombre de caselles blanques/negres que resten al tauler, $k \leq \lceil \frac{n^2}{2} \rceil$).

Un arc (b_i, w_j) entre aquest dos grups de nodes correspon a la relació entre una casella negra b_i i les blanques que comparteixen costat (són adjacents) al tauler i, per tant, és possible aparellar-les entre elles sota una fitxa de dòmino. El resultat és un graf bipartit.[1] L'objectiu, donades aquestes restriccions, és trobar un *perfect matching* (o *aparellament*) a un graf bipartit, on cada casella formarà part d'un emparellament.

Per trobar el matching perfecte, ampliem aquest graf per transformar-lo en una xarxa de flux $\mathcal{N}(V, E)$. Com a nodes tindrem:

[1] Observeu que un node d'un determinat color mai tindrà cap arc cap a nodes del mateix color.

- un node font s,
- els nodes $b_1, \ldots, b_k$ representant les caselles negres,
- els nodes $w_1, \ldots, w_k$ representant les caselles blanques,
- un node embornal t.

El nombre total de nodes de la xarxa és, doncs, $2 + 2k \leq 2 + n^2 = O(n^2)$.

Les arestes (dirigides) d'aquesta xarxa seran:

- $\{(s, b_i)\}_{1 \leq i \leq k}$
- $\{(w_j, t)\}_{1 \leq j \leq k}$
- $\{(b_i, w_j) \mid b_i \text{ és adjacent a } w_j\}_{1 \leq i \leq k}$

El nombre total d'arestes de la xarxa és $|E| \leq 2k + 4k \leq 3n^2 = O(n^2)$. Observem que, per a cada node b_i hi haurà, com a molt, quatre arestes (b_i, w_j), corresponents a les quatre adjacències amb fitxes blanques que pot tenir com a màxim.

Considerem que totes les arestes tenen capacitat 1. Fixem aquesta capacitat perquè l'assignació d'una fitxa de dòmino ha de ser única per a cada casella (no es poden sobreposar fitxes) i una unitat de flux representarà precisament aquesta assignació.

Executem l'algorisme de Ford-Fulkerson per a determinar el flux màxim f^* de s a t en aquesta xarxa. Si hi ha un flux de mida k, aleshores hi ha un matching bipartit de mida k. El temps total de l'algorisme és, doncs, $O(|E| \cdot f^*) = O(n^4)$ i, per tant, polinòmic.[2]

Per recuperar la col·locació exacta de les fitxes de dòmino només haurem de mirar quines arestes (b_i, w_j) han quedat amb flux ($f[b_i, w_j] = 1$) després d'aplicar l'algorisme de Ford-Fulkerson (cost $O(n^2)$).

[2] Atenció perquè, com ha de ser, estem expressant el cost de l'algorisme en funció de la mida de l'entrada. Recordeu que el tauler és de mida $n \times n$.

4.3. **(Cohesió social)**

En sociologia, sovint s'estudia un graf $G = (V, E)$ en el qual els nodes representen les persones, i les arestes representen els que són amics entre si. Suposem que l'amistat és simètrica, per la qual cosa podem considerar G com un graf no dirigit.
Volem estudiar aquest graf G, per a trobar grups de persones, molt unides en el sentit de que tots són amics de tots. Una manera de formalitzar aquesta idea seria la següent. Per a un subconjunt $S \subseteq V$ sigui $e(S)$ el nombre d'arestes dintre d'S, és a dir, el nombre d'arestes que tenen tots dos extrems en S. Definim la *cohesió* d'S com $e(S)/|S|$. en aquest tipus de grafs, un paràmetre natural podia ser el conjunt $S \subseteq V$ amb cohesió màxima.

(a) Doneu un algorisme polinòmic que pren com a entrada G i un nombre α racional, i determina si hi ha un conjunt S amb cohesió $> \alpha$.

(b) Doneu un algoritme polinòmic per trobar un conjunt $S \subseteq V$ amb la màxima cohesió.

Una solución

Antes de definir el algoritmo vamos a asociar el problema con un parámetro de una red de flujo. Construimos la siguiente red:

- Por cada $\{u, v\} \in E$ añadimos dos arcos (u, v) y (v, u) con capacidad 1.
- Añadimos vertice s y para cada $u \in V$ un arco (s, u) con capacidad $d(u)$.
- Añadimos vertice t y para cada $u \in V$ un arco (u, t) con capacidad 2α.

Analizemos la capacidad de los $s - t$ cortes (S, T). Supongamos que $S = \{s\} \cup A$ and $T = \{t\} \cup B$.

Notemos que $c(\{s\}, V \cup \{t\}) = 2m$ y $c(V \cup \{s\}, \{t\}) = 2\alpha n$.

Si $\alpha < m/n$, V tiene la cohesión requerida

En caso contrario $2m < 2\alpha n$ y mincut $\leq 2m$.

Para un corte genérico la capacidad del corte es:

$$\begin{aligned} c(S, T) &= \sum_{u \in B} d(u) + \sum_{v \in A} 2\alpha + c(A, B) \\ &= 2\alpha|A| + 2e(B) + 2c(A, B) \\ &= 2\alpha|A| + 2m - 2e(A) \\ &= 2(m + \alpha|A| - e(A)) \end{aligned}$$

$c(S, T) > 2m$ iff $\alpha|A| - e(A) < 0$ iff $\alpha|A| < e(A)$ iff $e(A)/|A| > \alpha$.

Hay un conjunto con cohesion $> \alpha$ iff min-cut $< 2m$

(a) La red se puede construir en tiempo polinómico. Luego calculamos el valor de MaxFlow usando Edmonds Karp en tiempo $O(nm^2)$. Y hacemos la comparación final. Obteniendo un algoritmo con coste polinómico.

(b) Primero calcularemos el valor mas grande de α ($\alpha_{\max}$) para el que existe un conjunto ocn cohesión α. Para ello combinaremos búsqueda binaria con el algoritmo del apartado anterior. Una vez obtenido el valor $\alpha_{\max}$, obtenemos un flujo f con valor máximo usando Edmonds Karp en tiempo $O(nm^2)$ para $\alpha_{\max}$. A partir de f obtenemos el grafo residual y tomamo como A los vertices accesibles desde s en este grafo. De acuerdo con el teorema maxflow-mincut $s \cup A$ define un corte con capacidad máxima y de acuerdo con el resultado anterior A tiene cohesión máxima. Como en el paso anterior el tiempo total es polinómico ya que la cohesión de un conjunto es un valor entre 0 y n.

4.4. (SafeGym).

> ExtremeGym tiene que replanificar los circuitos de entrenamiento extremo incorporando las nuevas restricciones de seguridad sanitaria.
> Para ello ha dividido los espacios en salas con capacidades limitadas. Algunas de estas salas se han conectado entre ellas a través de pasillos unidireccionales, debidamente aislados, en los que también se debe limitar el número de personas que los transitan.
> Los técnicos de la empresa han diseñado un grafo en 5 niveles que define los posibles circuitos de entrenamiento. Un entrenamiento se inicia y finaliza en la zona de acceso al recinto (con capacidad ilimitada) que tiene acceso directo a todas las salas del primer nivel y desde todas las salas del último nivel. El entrenamiento finaliza tras realizar, en el tiempo estipulado, las rutinas asociadas a 5 salas, una en cada nivel. Los pasillos entre salas permiten acceder de una sala de entrenamiento en un nivel a otra (u otras) en el nivel siguiente.
> ExtremeGym quiere un algoritmo eficiente para determinar el nivel de restricción que puede aplicar a la capacidad real de salas y pasillos que permita mantener un flujo mínimo de M personas realizando un entrenamiento en el gimnasio.
> Diseñad un algoritmo eficiente para resolver el siguiente problema:
> Dados el grafo de niveles y la capacidad de salas y pasillos, determinar si para algún valor entero no negativo $k \geq 0$, se puede limitar la capacidad de cada sala y cada pasillo en un factor 2^k manteniendo un flujo de M entrenamientos. En el caso de que este valor exista proporcionar el valor de k más grande que permita el número de personas entrenando deseado.

Una solución: Lo resolveremos mediante un problema de flujo con restricciones. Tenemos restricciones en las salas y en los pasillos. La capacidad de un pasillo la trasladaremos a la capacidad de un arco. Para limitar la capacidad de las salas desdoblaremos los vértices correspondientes las salas en dos, de la forma habitual.

La entrada es $G = (V, E)$ dónde $V = V_1 \cup V_2 \cup \cdots \cup V_5$. A patir de G consideramos la siguiente red de flujo. La red $\mathcal{N}$ tiene

- Nodos: s, t, V, $V' = \{v' \mid v \in V\}$ una copia de V,
- Aristas y capacidades:

$$\begin{aligned} \{(s,a) \mid a \in V_1\} &\quad \text{capacidad } c(a) \\ \{(b,t) \mid b \in V_5\} &\quad \text{capacidad } c(b) \\ \{(a,a') \mid a \in V\} &\quad \text{capacidad } c(a) \\ \{(a',b) \mid a \in V, b \in V, (a,b) \in E &\quad \text{capacidad } c(a,b) \end{aligned}$$

Un camino de s a t tiene la forma $s \to a_1 \to a_1' \to a_2 \to a_2' \to \cdots \to a_5' \to t$, si transporta una unidad de flujo interpretaremos que una persona realiza las rutinas de las salas involucradas. Las capacidad de las aristas aseguran que, en un flujo con valor máximo, en ningún momento se supera el aforo de una sala o la ocupación máxima de un pasillo.

Para resolver el problema primero obtendremos un flujo con valor máximo en la red, esto nos proporcionará el número máximo de personas que podrían entrar simultáneamente en el gimnasio sin violar las restricciones de seguridad.

Algorítmo:

Construir $\mathcal{N}$
$F = MaxFlow(\mathcal{N})$ **return** $(|F|)$

El número de vértices en la red es $N = 2n + 2$, el número de aristas es $M \leq 3n + m$. Como el grafo tiene capacidades y no tenemos cotas sobre ellas utilizaremos EK. Así el coste es $O(M^2N) = O(m^2n)$.

Para calcular el valor de k que nos piden tengo en cuenta que la operación de escalado divide por el mismo factor todas las capacidades de la red, y por lo atanto el mismo factor de reducción aparecerá eb el valor flujo máximo. Por ello, si MF es el valor devuelto por el algoritmo anterior tenemos que buscar el valor mayor de k tal que $MF/2^k \geq M$. Si $MF < M$, no se puede garantizar nunca el mínimo de ocupación. En caso contrario iremos dividiendo por 2 MF y actualizando el valor de k

Input $MF(\geq M)$
$k = 0$
while $MF/2 \geq M$ **do**
    $++k$
    $MF = MF/2$
return k

El número de iteraciones es $O(\log MF)$ ya que siempre dividimos por dos. Así el coste total del algoritmo es $O(m^2n + \log C)$ dónde $C = \sum_{a \in A_1} c(a)$ que es una cota superior a MF.

4.5. (**SafeTrans**)

SafeTrans es dedica al transport de mercaderies perilloses o de gran volum i/o pes per carretera. La companyia està analitzant la possibilitat d'acceptar un encarreg per traslladar els residus emmagatzemats a un dipòsit cap a un altre més segur. Per fer l'anàlisi de viabilitat el seu equip logístic ha definit un graf dirigit $G = (V, E)$ que representa la xarxa de carreteres que permetria connectar els dos dipòsits $s, t \in E$ dintre d'uns límits de quilometratge raonables.
Degut a la natura dels materials a transportar el trasllat s'ha de fer al llarg de com a màxim 5 dies durant la nit i a més s'ha de tenir en compte, en la mesura del possible, el nivell de seguretat que es pot assolir. Per això volen que l'equip informàtic els doni informació sobre la possibilitat de fer el trasllat tenint en compte diferents restriccions. En tots els escenaris, per raons de seguretat un tram de carretera només es pot fer servir com a molt una de les 5 nits.

Escenari 1: Si un tram de carretera es fa servir una nit, aquesta nit només hi pot circular per ell un camió carregat.

Escenari 2: Les condicions de l'escenari 1 es poden relaxar, per un subconjunt de trams de carretera suficientment aïllats. Suposant que $S \subset E$ son els trams suficientment aïllats. En aquest escenari s'ha de garantir: Si un tram de carretera a S es fa servir una nit, aquesta nit poden circular un o dos camions carregats. Si un tram de carretera no és a S i es fa servir una nit, aquesta nit només hi pot circular un camió carregat.

Escenari 2: Una anàlisi alternativa de la densitat dels nuclis de població ha determinat que en la majoria dels casos la població està suficientment aïllada, però que hi han un cert subconjunt de nuclis urbans densament poblats. Suposant que $D \subset V$ és el conjunt de nuclis urbans amb alta densitat de població. En aquest escenari s'ha de garantir que com a molt un camió carregat circuli una nit per $d \in D$. En contrapartida, si un tram de carretera es fa servir una nit, aquesta nit poden circular per ell fins a tres camions carregats.

Proporcioneu algorismes, per a cada escenari, què:

(a) Donats, G, S i D juntament amb s i t, determini un conjunt tan gran com sigui possible de rutes potencials per fer el transport en una nit.

(b) Donats, G, S, D, c, $1 \leq c \leq 5$, i k juntament amb s i t determini si es possible seleccionar rutes per k camions i c dies, i en cas de que sigui possible proporcioni aquestes rutes, assegurant què: cada camió carregat fa com a molt un trajecte per nit; globalment es compleixen les restriccions imposades per l'escenari; i a més el nombre total de trasllats es més gran que $\frac{2}{3}ck$.

Solució:

(a) Resolveremos el problema como un problema de flujo máximo. Para cada escenario tendremos una red de flujo que modela el problema. Sobre esta red calcularemos el flujo máximo usando Ford Fulkerson. Una vez calculado un flujo con valor máximo consideraremos el grafo G' en el que solo aparecen las aristas con flujo positivo. En G' repetiremos el siguiente proceso: obtener un camino de s a t, este camino será una ruta; reducir el valor del flujo en una unidad en todas las aristas del camino; eliminar aquellas en las que el flujo sea 0.

En el escenario 1, tenemos que calcular caminos que no comparten aristas, tal y como hemos visto en clase basta con asignar capacidad 1 a todos los arcos en E y tomat s como fuente y t como sumidero.

En el escenario 2, los arcos en S pueden soportar dos caminos, les asignaremos capacidad 2 y a los que no están en S capaciadad 1. Cualquier conjunto de rutas que cumpla las condiciones se puede convertir en un flujo en el que el valor del flujo en arcos de S es menor o igual que 2 y al revés siempre que el flujo tenga valores enteros. Por lo tanto un flujo entero con valor máximo nos proporciona una solución al problema en este escenario.

En el escenario 3, asignaremos a los arcos capaciadad 3 y tal como hemos visto en clase para conseguir que los caminos no pasen dos veces por el mismo node en D, convertiremos el grafo en un grafo G' donde cada nodo $d \in D$ se remplaza por dos nodos d' y d'', los arcos que entran en d entraran en d', los que salen lo harán de d'' y añadimos un arco (d', d'') con capacidad 1.

En cualquiera de los tres casos el coste de calcular maxflow es $O((n+m)m)$ ya que el valor del flujo máximo es como mucho $3m$. Con el mismo coste se pueden obtener la lista de rutas, realizando un BFS por cada unidad de flujo.

(b) Notemos que los tramos de carretera solo se pueden utilizar en uno de los c días, por ello es necesario que el número total de rutas, calculado en el apartado a) sea $\geq \frac{2}{3}ck$.

Además necesitamos encontrar una partición de E en c conjuntos $E_1, \ldots, E_c$ que nos permita asignar $\leq k$ rutas por día y suficientes rutas a lo largo de los c días. Para una partición $E_1, \ldots, E_c$, resolveremos el problema de flujo máximo como en el apartado a) obteniendo c valores de flujo $f_1, \ldots, f_c$, dónde $f_i = maxflow(V, E_i, s, t, c)$. La partición nos proporciona una solución sí $\sum_{i=1}^{c} \min k, f_i \geq \frac{2}{3}ck$.

En el escenario 1 las rutas son totalmente independientes, ya que no comparten aristas y por lo tanto se pueden asignar a cualquiera de los dias. Una vez resuelto a) asignamos las primeras k rutas al día 1, las siiguientes k al día 2, hasta que acabemos con todas las rutas disponibles o llenemos los c días.

En los escenarios 2 y 3, recorreriamos todas las particiones posibles hasta encontrar una en la que las condiciones se cumplan o concluir que no hay tal solución.

En el escenario 1 el coste es como en el caso a) $O((n+m)m)$, pero podríamos reducirlo a $O((n+m)ck)$ finalizando el algoritmo cuando el valor del flujo llegue al mínimo requerido. En los escenarios 2 y 3 tenemos que recorrer todas las particiones, este número es exponencial si $c > 1$, el algoritmo tiene coste exponencial, aún cuando el coste por partición es polinómico.

4.6. (**Vacunació**).

La Generalitat ha de planificar els trasllats de les vacunes Covid des de l'aeroport del Prat als diferents centres de distribució de la vacuna a Catalunya. Per dur-ho a terme, ha creat un graf de distribució format per les localitats catalanes a on es poden emmagatzemar vacunes amb les condicions de conservació adients (incloent-hi el propi aeroport). Les connexions entre localitats garanteixen que el transport es pot portar a terme des d'una localitat fins a qualsevol altra, seguint una o més connexions, en temps suficientment curt per tal de no trencar la cadena del fred.
Per a cada localitat, fora del Prat, es coneixen la quantitat màxima de caixes de vacunes que es poden emmagatzemar de forma segura.
Per raons de seguretat, el nombre total de caixes de vacunes que poden circular per una localitat no pot superar la capacitat d'emmagatzematge de la localitat.

(a) Dissenyeu un algorisme que, donats el graf de distribució, la quantitat de caixes que es pot emmagatzemar a cada localitat, les localitats que són centres de distribució i, per a cada centre de distribució, el nombre de caixes de vacunes que es volen traslladar des del magatzem de l'aeroport a ell, determini si és possible fer el trasllat respectant les restriccions descrites abans.

Podeu suposar que al magatzem del Prat hi ha més caixes de vacunes de les que es volen distribuir.

(b) En cas que el trasllat no es pugui portar a terme, esteneu l'algorisme precedent per tal d'identificar un conjunt de localitats on un increment de seva capacitat d'emmagatzematge garantiria poder portar a terme el trasllat.

Una solución:

(a) Podemos resolver el problema como un problema de circulación con demandas en una red adecuada. La limitación de transporte está en la cantidad de cajas de vacunas que pueden atravesar una localidad.

Sea $G = (V, E)$ el grafo que nos proporcionan y $a(u)$, $u \in V$, la cantidad de cajas que se pueden almacenar la localidad u. Suponemos que $s \in V$ es el nodo que representa el almacén del Prat. Si $D \subset V - \{s\}$ es el conjunto de centros de distribución, se nos dará $b(w) > 0$, $w \in D$, el número de cajas de vacunas que queremos trasladar desde el Prat al centro de distribución w.

Asumimos que el grafo que nos dan es dirigido, si no lo fuese, tal y como hemos visto en clase se convertiría en dirigido poniendo los arcos (u, v) y (v, u) para cada arista $\{u, v\}$ del grafo no dirigido.

La red $\mathcal{N}$ tiene

- Nodos: V y V', siendo V' una copia de los nodos en V. Si $u \in V$, entonces u' representa su copia en V'.

- Aristas y capacidades:

$$\begin{aligned} \{(u, u') \mid u \in V\} \qquad & \text{capacidad } a(u) \\ \{(u', v) \mid (u, v) \in E\} \qquad & \text{capacidad } \infty \end{aligned}$$

- Demandas:

$$\begin{aligned} u \in D, d(u) &= b(u) \\ d(s) &= -\sum_{u \in D} b(u) \\ u \notin D, u \neq s, d(u) &= 0 \end{aligned}$$

La duplicación de los nodos y la capacidad de la arista que conecta un nodo con su copia garantizan que, en un flujo entero, nunca pasan por un nodo más cajas de las que es posible almacenar en él.

Teniendo en cuenta que el aeropuerto tiene suficientes cajas para satisfacer la demanda, el problema de circulación tiene solución si y solo si podemos realizar el transporte requerido ya que las restricciones solicitadas están garantizadas por la capacidad de las aristas.

Para analizar el coste, asumo que G tiene n vértices y m aristas. Así $\mathcal{N}$ tiene $2n$ vértices y $n + m$ aristas. Como no tenemos cotas en el valor de las demandas y $-d(s)$ puede ser grande, el coste del algoritmo viene dado por el análisis de EK con coste $O(NM^2) = O(n(n + m)^2)$.

(b) Si la circulación no existe, lo que podemos hacer es considerar la red de flujo utilizada para resolver el problema de circulación y el flujo con valor máximo que hemos calculado en dicha red. Utilizando el grafo residual, obtenemos un corte con capacidad mínima. Como ninguna de las aristas del corte puede tener capacidad ∞, ya que no permite el transportar todas las cajas requeridas, cada arista de este corte identifica un vértice.

Miraríamos en cuanto se incrementa el valor del flujo asignando capacidad ∞ a las aristas del corte. Si con esto no es suficiente para que el problema de circulación tenga solución. Repetiremos el proceso, con la nuevo red y el nuevo flujo máximo, tantas veces como sea necesario. El algoritmo acabará en el momento en que la circulación buscada exista.

El conjunto de aristas (u, u') con capacidad ∞, nos proporciona el conjunto de nodos pedidos. Observemos que en cada paso el corte con capacidad mínima no puede contener ninguna de las aristas a las que hemos asignado capacidad ∞, por lo que como mucho repetiremos este proceso $O(n)$ veces, cada una de ellas con el coste de EK $O(n(n+m)^2)$. Así el coste total del algoritmo es $O(n^2(n + m)^2)$.

4.7. **(Flow Bicing)**

La ciutat de Barcelona necessita actualitzar el seu servei de bicicletes compartides de cara a l'estiu. Les bicicletes en funcionament s'estacionen en una xarxa d'n punts d'aparcament distribuida per la ciutat.
Per a cada punt d'aparcament de bicicletes v l'ajuntament té una estimació del nombre d'usuaris α_v que necessitaran una bicicleta a primera hora del matí. A més, la nit anterior s'ha fet una ronda de reconeixement a cada punt v per saber el nombre de bicicletes β_v que hi haurà disponibles l'endemà. Cada dia, per posar en marxa el servei, cal que tots els punt d'aparcament tinguin disponibles tantes bicicletes com s'hagi previst que demanaran els seus usuaris.
L'ajuntament també té una estimació de la quantitat màxima de bicicletes que, en cas que fos necessari, es podrien transportar entre dos punts d'aparcament diferents. Entre tot parell $e = (u, v)$ de punts d'aparcament de la xarxa, es poden transportar fins a $c(e) \geq 0$ bicicletes de u cap a v.
Dissenyeu un algorisme per a poder decidir si, amb l'estimació de necessitat de bicicletes que té l'ajuntament, podem tenir el servei preparat per funcionar a l'endemà. En cas afirmatiu, el vostre algorisme també ha d'explicitar el nombre de bicicletes que s'han de traslladar d'una estació a una altra per deixar-ho tot preparat. En cas negatiu, digueu quins són els punts d'aparcament que no reben totes les bicicletes necessàries a primera hora del matí. Justifiqueu-ne la correctesa i el cost temporal.

Una solució

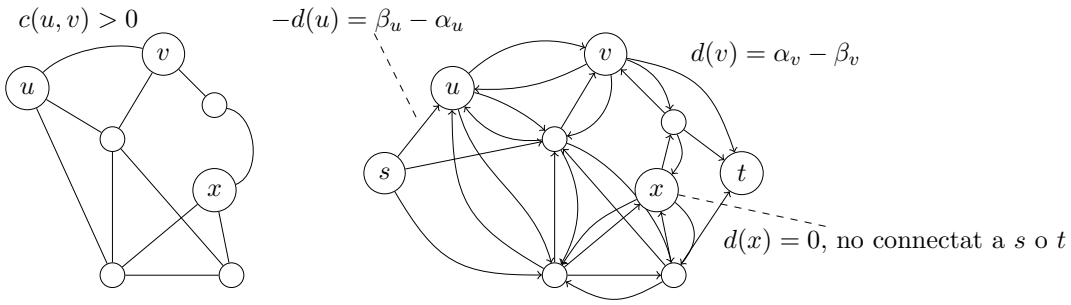
Donat un punt d'aparcament v sigui $d(v) = \alpha_v - \beta_v$. Si $d(v) > 0$ el punt d'aparcament té demanda de bicicletes, que hauràn d'arribar des d'altres punts; si $d(v) < 0$ llavors el punt d'aparcament té un excedent de bicicletes que pot redirigir als altres punts d'aparcament. Sigui V^+ el conjunt de punts d'aparcament tals que $d(v) > 0$ i $D^+ = \sum_{v:d(v)>0} d(v)$; i sigui V^- el conjunt de punts d'aparcament tals que $d(v) < 0$ i $D^- = \sum_{v:d(v)<0} d(v)$. Clarament una condició necessària (pero no suficient) per a poder satisfer totes les demandes és que $|D^-| \geq D^+$.

Definim una xarxa $\mathcal{N}$ on:

- Vèrtexos: $V = \{s, t\} \cup$ punts d'aparcament,
- Arestes:

$$E = \{(s, x) \,|\, d(x) < 0\} \;\cup\; \{(y, t) \,|\, d(y) > 0\} \;\cup\; \{(u, v) \,|\, c(u, v) > 0\}$$
$$\cup\; \{(v, u) \,|\, c(u, v) > 0\}$$

  (el subconjunt {(v,u) — c(u,v)¿0} s'afegiria només si es considera que la informació sobre el transport entre estacions $c(u, v)$ no està assumint un sentit concret).
- Capacitat $c(s, x) = -d(x), \forall x \in V^-$
- Capacitat $c(y, t) = d(y), \forall y \in V^+$
- Capacitat $c(u, v)$ a la resta d'arestes



Si f és un flux màxim sobre $\mathcal{N}$ amb valor $v(f) = D^+$ això vol dir que existeix la manera de transferir bicicletes des dels punts amb excés de tal manera que totes les demandes D^+ són satisfetes. EL valor del flux en $\mathcal{N}$ mai pot ser $> D^+$ ja que D^+ és la capacitat del tall $(V - \{t\}\{t\})$. Si el valor del flux màxim és $< D^+$ llavors algún punt d'aparcament v amb $d(v) > 0$ no pot rebre totes les bicicletes que es necessiten.

Totes les capacitats són enters. Si utilitzem l'algorisme de Ford-Fulkerson el cost serà $O(n^2 D^+)$, doncs $|E| = O(n^2)$.[3] Si D^+ fos potencialment exponencial respecte a n podem utilitzar l'algorisme d'Edmonds-Karp amb cost polinòmic $O(m^2 n)$.

Si el valor del flux màxim f és $v(f) = D^+$ llavors $f(u,v)$ és el nombre de bicicletes que haurem de traslladar des del punt u al punt v, per a tot parell (u,v); si $f(u,v) = 0$ llavors és perquè no hem de traslladar cap bici d'u a v. Si el valor del flux màxim és $v(f) < D^+$ els punts d'aparcament y que no reben totes les bicis necessàries són aquells tals que la corresponent aresta (y,t) no està saturada per f, és a dir, tals que $f(y,t) < d(y)$.

Una solució alternativa: Sigui V' el conjunt de punts d'aparcament. També podem definir la xarxa $\mathcal{N}$ on:

- Vèrtexos: $V = \{s,t\} \cup V'$,
- Arestes: $E = \{(s,v) \,|\, v \in V'\} \cup \{(v,t) \,|\, v \in V'\} \cup \{(u,v) \,|\, u,v \in V', c(u,v) > 0\}$ $\cup \{(v,u) \,|\, u,v \in V', c(u,v) > 0\}$
- Capacitat $c(s,v) = \beta_v, \quad \forall v \in V'$
- Capacitat $c(v,t) = \alpha_v \quad \forall v \in V'$
- Capacitat $c(u,v)$ a la resta d'arestes

I demanar el requeriment que el valor del flux màxim $f^\star = \sum_{v \in V'} \alpha_v$.

[3] El cost pot ser molt millor si el número d'arestes (u,v) amb capacitat $c(u,v) > 0$ fos "$ln(n-1)/2$.

4.8. **(Congrés)**

> Suposeu que esteu organitzant un congrés on els investigadors presentin els articles que han escrit. Els investigadors que vulguin presentar un article envien un document als organitzadors de la conferència. Els organitzadors de la conferència tenen accés a un comitè de revisors que estan disposats a llegir-ne com a molt R articles cadascun. Tots els articles enviats han de ser revisat per, com a mínim, A revisors.
> El congrés té declarats un conjunt de temes. Cada enviament té assignat un tema concret i cada revisor té declarada una especialització per a un conjunt de temes. Els articles sobre un tema determinat només es revisen per part dels revisors experts en aquell tema.
> Els organitzadors del congrés han de decidir (sempre que es pugui) quins avaluadors revisaran cadascun dels articles o, equivalentment, quins articles seran revisats per cada revisor. Es demana:
> Proporcioneu un algorisme eficient per a resoldre aquest problema d'assignació.

Una solución. Lo resolveremos como un problema de circulación en una red de flujo. La red tendrá un vértice por cada artículo $\{v_1, \ldots, v_n\}$ y un vértice por cada revisor $\{r-1, \ldots, r_m\}$. Añadimos las aristas (s, v_i) con cota inferior A y cota superior m, las aristas (r_j, t) con capacidad R y aristas (v_i, r_j) con capacidad 1, siempre que el revisor j sea experto en el tema del artículo v_i.

Una unidad de flujo que se transmita de s a t, pasará a través de un camino $s \to v_i \to r_j \to t$ representando que el trabajo i se asigna al revisor j.

La capacidad 1 en las aristas (v_i, r_j) garantiza que no se asigna un artículo más de una vez a un revisor. La capacidad R en las aristas (r_j, t) evita que se asignen más de R artículos a un revisor. La cota inferior en las otras aristas garantiza que cada artículo tiene al menos A revisiones. La red admite una circulación sii el problema planteado tiene solución.

Sea $t(j)$ el número de temas en el que es experto el revisor j y T la suma de los valores de $t(j)$. La red tiene $N = n + m + 2$ nodos Y $M = n + T + m$ aristas.

El flujo máximo está acotado por mR, el coste será $O(mR(n + m + T))$ si utilizamos la cota superior al flujo y $O((n + m + T)^2(n + m))$ si utilizamos el tamaño de la red. Como $R \leq n$ la mejor cota es $O(mR(n + m + T))$.

4.9. **(VideoFast)**

La compañía VideoFast quiere utilizar la red para transmitir vídeo a sus clientes. Para ello ha contratado una red de comunicación formada por n servidores que representamos con un grafo dirigido $G = (V, E)$ con $|V| = n$ vértices y $|E| = m$ aristas.
Para cada $e \in E$, $b(e) \geq 0$ es el *ancho de banda* de la arista e contratado por VideoFast. El total de bits transmitido por un arco e no puede superar el ancho de banda contratado, $b(e)$. Por otra parte, VideoFast quiere establecer contratos con sus clientes, un subconjunto $X \subset V$. Para cada cliente $x \in X$, un contrato establece el tamaño total (en bits) de la transmisión $t(x)$. VideoFast quiere iniciar la transmisión desde un servidor $s \in V - X$.
Proporcionad un algoritmo con coste polinómico, para determinar si los anchos de banda contratados permiten efectuar o no la transmisión que se prevee establecer en los contratos con los clientes. En caso de que los contratos se puedan cumplir, el algoritmo debe proporcionar un nodo en $V - X$ desde dónde VideoFast pueda iniciar las transmisiones y cumplir con sus compromisos con los clientes.

Una solució
Lo plantearemos como un problema de asignación con restricciones en una red de flujo en la que una unidad de flujo de $s \in V - X$ a $x \in X$ represente la transmisión de un bit de s a t. Para controlar la transmisión añadiremos un nodo t a la red y conectaremos X a t.

La red $\mathcal{N}_v$, para $v \notin X$, tiene

- Nodos: t y V, tomamos $s = v$
- Aristas y capacidades:

$$\begin{array}{rl} e \in E & \text{capacidad } b(e) \\ \{(x,t) \mid x \in X\} & \text{capacidad } t(x) \end{array}$$

Un camino de s a t tiene la forma $s \to \cdots \to x \to t$, si transporta una unidad de flujo interpretaremos que se envía un bit desde s a $x \in X$. La capacidad de las aristas que entran en t garantiza que se puede realizar la transmisión requerida si el flujo máximo en la red es $\sum_{x \in X} t(x)$.
Resolveremos MaxFlow, para cada $v \in V - X$, hasta encontrar un v tal que el valor del flujo máximo en $\mathcal{N}_v$ es $\sum_{x \in X} t(x)$, y devolveremos este nodo, o comprobar que no es posible.

Las capacidades son enteras, pero los valores pueden ser grandes, por ello utilizarremos EK para calcular un flujo con valor máximo. Como tenemos que hacerlo para cada $s \in V - X$, el coste total del algritmo es $O(|V|^2|E|^2)$.

Un altre solució: Se puede resolver el problema planteandolo como como un problema de circuación y utilizar el algoritmo correspondiente.

La red $\mathcal{N}_v$, para $v \notin X$, tiene

- Nodos: V
- Aristas y capacidades:

$$\begin{array}{rl} e \in E & \text{capacidad } b(e) \\ \{(x,t) \mid x \in X\} & \text{capacidad } t(x) \end{array}$$

- Demandas: $d(v) = -\sum_{x \in X} t(x)$ y $d(x) = t(x)$, $x \in X$.

Un altre solució: Se puede resolver el problema planteandolo como un problema de circulación con cotas inferiores y utilizar el algoritmo correspondiente.

La red $\mathcal{N}_v$, para $v \notin X$, tiene

- Nodos: t y V, tomamos $s = v$
- Aristas y capacidades:

$$\begin{array}{rl} e \in E & \text{capacidad } b(e) \\ \{(x,t) \mid x \in X\} & \text{cotas } [t(x), t(x)] \end{array}$$

4.10. **(Escapament)**

Una graella $n \times n$ és un graf no dirigit amb n^2 vèrtexs organitzats en n files i n columnes. Denotem el vèrtex a la i-èsima fila i la j-èsima columna per (i, j). Cada vèrtex (i, j) té exactament quatre veïns $(i-1, j)$, $(i+1, j)$, $(i, j-1)$ i $(i, j+1)$, excepte els vèrtexs a la vora, per als quals $i = 1$, $i = n$, $j = 1$, o $j = n$. Siguin $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ vèrtexs diferents, anomenats *terminals*, a la graella $n \times n$. El problema d'escapament consisteix a determinar si a la graella hi ha m camins disjunts (en vèrtexs) que connectin els terminals amb qualssevol m vèrtexs diferents de la vora de la graella.

a) Doneu i analitzeu un algorisme eficient per a resoldre el problema d'escapament.

b) Ara suposem que l'entrada del problema d'escapament inclou també una llista d'arestes prohibides (és a dir, arestes que no poden aparèixer als camins) i una altra llista d'arestes requerides (és a dir, que han d'aparèixer almenys a un camí). Doneu i analitzeu un algorisme eficient per decidir si aquesta versió del problema d'escapament té solució. En cas que en tingui, el vostre algorisme haurà de proporcionar una solució.

Una solución (Apartado a)

Una forma de resolver el "problema d'escapament" es reducirlo al problema de búsqueda de caminos disjuntos. Para ello, vamos a modelar nuestro problema como un grafo $G = (V, E)$ en la que $V = \{s, t\} \cup P$ siendo $P = \{p_{i,j}\}$ los vértices de la graella. Respecto a las artistas E, para cada vértice $p_{i,j}$ existirán 4 artistas con sus vecinos $(p_{i,j}, p_{i+1,j})$, $(p_{i-1,j}, p_{i,j+1})$, $(p_{i,j}, p_{i,j-1})$, $(p_{i,j}, p_{i,j})$ (excepto los vértices del borde). Además, conectaremos todos vértices del borde con s y los terminales con t (Figura **??**, panel A).

Para determinar si existen m caminos disjuntos (en vértices) que conectan los terminales con cualquiera de los m vértices diferentes del borde de la graella, utilizaremos el Teorema de Menger por el que el número máximo de caminos disjuntos (en aristas) $s \rightsquigarrow t$ en G es igual al número de aristas mínimo que atraviesa un s-t corte, o lo que es lo mismo, igual al valor del máximo flujo f^* en $\mathcal{N}$. Para que el problema tenga solución, al menos se ha de cumplir que $m \leq 4(n-1)$.

No obstante, antes de proceder a buscar caminos disjuntos (en vértices) en nuestro grafo, tenemos que adaptar nuestro grafo no-dirigido para (a) convertir nuestro G en digrafo y (b) imponer la restricción de que los caminos sean disjuntos en vértices. Para ello, aplicaremos dos transformaciones.

(a) Para convertir el grafo no-dirigido G a digrafo $G' = (V', E')$, por cada arista (v, u) vamos a crear dos aristas dirigidas (v, u) y (u, v) de capacidad 1 (Figura **??**, panel B). Un problema podría ser que el flujo f podría usar simultáneamente las dos aristas (v, u) y (u, v). Sin embargo, tal y como se vio en teoría, en tal caso siempre podemos eliminar ambas aristas del flujo f. El flujo resultante es válido y tiene el mismo valor. De tal forma, si repetidamente eliminamos esas "dobles aristas" del flujo f, el flujo resultante f' tendrá el mismo valor.

(b) Para imponer la restricción de buscar caminos vértices disjuntos, vamos a "dividir" cada vértice v de G' en v_i y v_o (Figura **??**, panel C). El vértice v_i (in) estará conectado con las aristas que antes entraban en v. El vértice v_o (out) estará conectado con las aristas que antes salían de v. Adicionalmente, crearemos una arista (v_i, v_o) de capacidad 1. Basándonos en el grafo resultante $G'' = (V'', E'')$, forzamos a que cualquier flujo necesariamente solo pueda pasar una vez por (v_i, v_o) (modelando un vértice de la graella).

Formalmente, construimos una red s-t $\mathcal{N} = (V'', E'', s, t, c : \forall e \in E \to 1)$ en tiempo $O(|V''| + |E''|) = O(n^2)$, lineal en el número de vértices y aristas de la graella. Utilizando el algoritmo de Ford-Fulkerson, podemos computar el flujo máximo f^* en tiempo pseudopolinómico $O(Dn^2)$. Sabemos que el valor del flujo máximo $D = s(f^*)$ no puede exceder la capacidad de los cortes $\langle s, V'' \setminus s\rangle$ y $\langle V'' \setminus t, t\rangle$. Por tanto, $s(f^*) \leq min\{4(n-1), m\} = m$ y podemos acotar el tiempo polinómico de FF con $O(m \cdot n^2)$.

Una solución (Apartado b)

(a) Por un lado, para modelar (u, v) aristas prohibidas, solo tenemos que limitar a cero la capacidad de las aristas (u, v) y (v, u) de nuestro digrafo $G' = (V', E')$ (o eliminarlas directamente).

(b) Por otro lado, para modelar aristas requeridas, podemos modelar el problema como una red con demandas y cotas inferiores. En este caso, las aristas requeridas (u, v) han de tener cota inferior 1. Debido a la transformación a digrafo realizada en el apartado anterior, no podemos aplicar la cota inferior a las aristas (u, v) y (v, u) de forma excluyente. No obstante, siendo R el conjunto de aristas requeridas, podemos plantear buscar una solución en las $2^{|R|}$ redes con demandas posibles. Reducimos el problema a una red con demandas transformando toda aristas requerida $e = (u, v)$, con $c(e) = 1$ y $l(e) = 1$, de la siguiente forma:

   i. Eliminamos la arista dado que $c'(e) = c(e) - l(e) = 0$.
   ii. Actualizamos las demandas en ambos extremos; $d'(u) = d(u) + l(e)$ y $d'(v) = d(v) - l(e)$.

   Por construcción de la red, sabemos que $D = \sum_{v \in T} d(v) = -\sum_{v \in S} d(v)$, donde $S = \{v \in V \mid d(v) < 0\}$ son los vértices origen de las aristas requeridas (vértices fuente) y $T = \{v \in V \mid d(v) > 0\}$ son los vértices destino de las aristas requeridas (vértices sumidero). Es decir, la red $\mathcal{N} = (V, E, c, l, d)$ admite una circulación si y solo si el flujo máximo $s(f^*) = D$. Además, todas las capacidades y demandas en $\mathcal{N}$ son enteras y, por tanto, $\mathcal{N}$ admite una circulación entera para toda arista. Por tanto, el problema puede resolverse en tiempo $O(2^{|R|} \cdot mn^2)$

4.11. **(SuperFast)**

SuperFast és una empresa de transport que està intentant decidir si li interessa participar en una oferta de treball de Amazon a Barcelona. Amazon voldria subcontractar el transport de productes de proximitat a SuperFast. El transport ha de garantir uns compromisos de puntualitat molt estrictes i SuperFast vol una estimació de la quantitat de nous vehicles que hauria d'incorporar a la seva flota i una estimació del cost corresponent al seu ús. Amazon li ha proporcionat els resultats de les seves simulacions de comportament dels clients i, en particular, de les necessitats de transport diàries per uns quants dies. Per un dia es disposa d'una llista de sol·licituds de transport. Cada sol·licitud de transport especifica les coordenades GPS de l'origen i del destí d'un enviament juntament amb l'hora de recollida i d'entrega. SuperFast disposa d'un programa que li proporciona el temps necessari de desplaçament d'un vehicle entre qualsevol parell de posicions de la ciutat coneixent el temps d'inici del trasllat.
En aquest primer estudi SuperFast assumeix el cas pitjor en el què els vehicles no poden portar més d'un enviament. A més, el vehicle ha de ser a la posició d'origen al temps estipulat i no pot deixar el punt de destí fins el temps estipulat de trasllat. Així, un vehicle que transporti un enviament pot encarregar-se d'un altre sempre que el temps de desplaçament entre el destí i el nou origen li permeti arribar l'hora estipulada. També assumeix que l'estimació del temps necessari de desplaçament és acurada.
Disenyeu algorismes amb cost polinòmic que:

(a) Donats un nombre k i les sol·licituds de transport d'un dia, determini si es poden servir amb k vehicles.

(b) Donades les sol·licituds de transports d'un dia, determini el nombre mínim de vehicles que les poden servir ($k_{\min}$)

(c) Donades les sol·licituds de transports de un dia, proporcioni una planificació per a cadascun dels $k_{\min}$ vehicles indicant, per a cada vehicle, la seqüència de sol·licituds de transport que ha de servir i el temps total de desplaçament del vehicle.

Podeu suposar que el càlcul del temps de desplaçament entre dos posicions té un temps constant i que entre l'hora de recollida i la d'entrega hi ha temps suficient per fer-hi el desplaçament amb puntualitat.
Ajut: Podeu pensar un vehicle com una unitat de flux i una sol·licitud de transport com un arc amb fites inferiors i superiors a la seva capacitat.

Una solució

(a) Utilizando la ayuda identificaremos los vehiculos con unidades de flujo y las solicitudes con arcos con capacidad limitada con cota superior e inferior.

De las restricciones del problema sabemos que un vehículo puede servir otro envío siempre que pueda desplazarse con tiempo suficiente desde el destino al nuevo origen, asumiendo que permanece en el destino hasta la hora estipulada. Esta restricción nos indica cual es la red a considerar.

Tendremos, para cada solicitud i, dos nodos o_i y d_i y un arco (o_i, d_i) con capacidad $[1, 1]$, reflejando el hecho de que todas las solicitudes tienen que ser servidas. Añadiremos un arco (d_i, o_j) cuando saliendo del destino de la solicitud i en el tiempo estipulado podamos llegar al origen de la solicitud j en el tiempo estipulado, reflejando la observación previa.

Añadiremos tres vertices adicionales s, x, t y los arcos (s, x) con capacidad k, para forzar que el flujo máximo sea $\leq k$, i, para cada solicitud i, los arcos (x, o_i) i (d_i, t) con capacidad 1.

Notemos que la existencia de un flujo que satisface las restricciones de capacidad nos garantiza la existencia de $\leq k$ caminos que no comparten aristas de s a t que, además, cubren todos los arcos correspondientes a solicitudes. Esto es equivalente a poder servir las peticiones con k o menos vehículos.

Una vez obtenida la red de flujo, podemos determinar si hay un flujo que satisface las restricciones en tiempo polinómico utilizando el algoritmo visto en clase para decidir la existencia de flujos con demandas y cotas inferiores.

(b) Siempre hay una solución con n vehículos, por ello $1 \leq k_{\min} \leq n$ podemos implementar una búsqueda binaria utilizando el algoritmo del apartado (a) que nos permite decidir si las solicitudes se pueden servir con $\leq k$. En total tendermos que hacer $O(\log n)$ ejecuciones del algoritmo del apartado (a).

(c) Una vez encontrado el valor de $k_{\min}$ utilizando el algoritmo del apartado (a) para este valor de k tendremos una flujo de s a t con valor del flujo $k_{\min}$ que satisface la restricción de que todas las solicitudes están servidas. Como hemos visto en clase lo único que tenemos que hacer es aplicar el algoritmo de extracción de caminos disjuntos como vimos en clae. Esto nos da $k_{\min}$ ejecuciones de BFS sobre el grafo formado por los arcos con flujo positivo.

4.12. **(DistFast)**

L'empresa DistFast està interessada a trobar un mecanisme per poder suggerir als seus conductors rutes per fer la distribució de material dintre del centre de Barcelona tan ràpid com sigui possible. En aquests moments DistFast té un model simplificat de Barcelona, que està format per un graf $G = (V, E)$ on els vèrtexs representen cruïlles, i les arestes representen trams de carrer que connecten les cruïlles dels seus extrems. Al mapa tenen marcades un conjunt de cruïlles $B \subseteq V$, que són les més properes als punts on tenen els magatzems d'on ha de sortir el material, i també un altre conjunt $C \subseteq V$, a on s'hi ha de distribuir el material (podeu suposar que $B \cap C = \emptyset$).
Durant el dia van seguint les informacions del trànsit dins de la ciutat i això els permet fer una classificació dels trams de carrers segons tres nivells de saturació de trànsit: alt, mitjà o baix. També determinen amb aquesta informació el subconjunt $D \subseteq V$ de cruïlles que voldrien evitar.
Les rutes són camins que comencen a alguna cruïlla de B i que acaben a alguna de les cruïlles de C. DistFast està interessada a trobar el *màxim nombre possible de rutes* que podrien fer servir els seus conductors, d'acord amb dos criteris diferents:

(a) Els camins no comparteixen vèrtexs (cruïlles) entre ells.

(b) Un tram de carrer molt saturat només es pot fer servir com a molt en una ruta, un de saturació mitjana en, com a molt, 10 i un de baix nivell de saturació no té restriccions. Una cruïlla a D només pot aparèixer en, com a molt, dues rutes.

Doneu algorismes tan eficients com pugueu que, donats G, B, C, D i una avaluació del nivell de saturació del trànsit, determini un conjunt de rutes de grandària màxima d'acord amb cadascun dels dos criteris.

Una solució: Aquest exercici és bàsicament igual al problema 4.16 (RACC). Essencialment, busquem el màxim nombre de rutes a un graf amb certes restriccions. Per a l'**apartat (a)**, resumirem les restriccions del nostre problema per després modelar-les amb una xarxa s-t.

- R1: $G = (V, E)$ és un mapa amb vèrtexs V (cruïlles) i arestes E (carrers).
- R2: $B \subseteq V$ són els vèrtexs d'inici dels camins.
- R3: $C \subseteq V$ són els vèrtexs de final dels camins.
- R4: $D \subseteq V$ són els vèrtexs que no estan continguts en els camins.
- R5: Els camins han de ser de vèrtexs disjunts.

Procedim a modelar i formalitzar el problema i les seves restriccions usant una xarxa s-t $\mathcal{N}$.

- **(R1)**, Com que $G = (V, E)$ és un graf no dirigit, el transformarem en un graf dirigit $G' = (V, E')$ on, per cada aresta $e = (v_1, v_2) \in E$, ara tindrem dos arcs $e_1 = (v_1, v_2)$ i $e_2 = (v_2, v_1)$ $(e_1, e_2 \in E')$, un d'anada i un altre de tornada.
- **(R2)**, connectarem tots els vèrtexs $v \in B$ amb la font s $(\{(s, v) \mid v \in B\})$.
- **(R3)**, connectarem tots els vèrtexs $u \in C$ amb l'embornal (*sink*) t $(\{(u, t) \mid u \in C\})$.
- **(R4)**, eliminarem tot arc que connecti $w \in D$ $(\{(w, x), (x, w) \mid w \in D \wedge x \in V\})$.

- **(R5)**, per limitar la capacitat de les cruïlles (vèrtex disjunts), desdoblarem tots els vèrtexs $v \in V$ en dos (v_{in} i v_{out}). v_{in} rebrà els arcs d'entrada i de v_{out} en sortiran els de sortida. L'arc connectant les dues còpies (v_{in}, v_{out}) tindrà capacitat 1. La resta d'arcs tindrà capacitat ∞.

Un camí de s a t té la forma $s \to v \to \cdots \to u \to t$, si transporta una unitat de flux interpretarem que hi ha una ruta des de $v \in B$ fins a $u \in C$ que compleix amb totes les restriccions. Pel teorema de Menger, el nombre màxim de camins disjunts $s \rightsquigarrow t$ és igual al nombre d'arcs mínim que travessa un tall s-t; és a dir, el valor del flux màxim en $\mathcal{N}$. Per això, per resoldre el problema calcularem un flux amb valor màxim a la xarxa $\mathcal{N}$. Després, en podem extreure les rutes utilitzant el graf format per les arestes amb flux 1. Amb un BFS podem extreure'n un camí, restar una unitat de flux a les arestes d'aquest camí, i repetim el procés fins a extreure'n els f^* camins. Proposo el següent algorisme.

function DistFast(G, B, C, D)

▷ Construir la xarxa s-t

$\mathcal{N} \leftarrow$ ConstruirXarxa(G, B, C, D)

▷ Computar el flux màxim

$F \leftarrow$ MaxFlow($\mathcal{N}$)

▷ Obtenir rutes

$Rutes \leftarrow$ ObtenirRutes($\mathcal{N}, F$)

Amb n (el nombre inicial de vèrtexs) i m (el nombre inicial d'arestes), podem veure que cap transformació aplicada en el modelat incrementa n i m excepte en factors constants. Construir la xarxa s-t costa $O(n + m)$. Calcular el flux màxim usant Ford-Fulkerson té cost pseudopolinòmic $O(f^*(m+n))$, on no pot haver-hi més camins disjunts que cruïlles ($f^* \leq n$), per tant $O(n(m + n))$. Noteu que, si decidim utilitzar Edmonds-Karp, tindrem un cost $O(mn(m + n))$ que no sembla millorar la fita de FF. Finalment, extreure'n n camins com a màxim tindrà cost $O(n(m + n))$. En definitiva, el nostre algoritme té cost $O(n(m + n))$ en el pitjor cas.

A l' **apartat (b)**, s'apliquen les restriccions **{R1, R2, R3}** i afegim les següents.

- R6: Arestes amb saturació alta, $a \in S_{alta} \subseteq E$, només poden pertànyer a 1 ruta.
- R7: Arestes amb saturació mitjana, $m \in S_{mitjana} \subseteq E$, només poden pertànyer a 10 rutes.
- R8: Arestes amb saturació baixa, $b \in S_{baixa} \subseteq E$, poden pertànyer a ∞ rutes.
- R9: $D \subseteq V$ vèrtexs poden pertànyer a 2 rutes.

En aquest cas **{R6,R7,R8}**, només hem d'ajustar la capacitat de les arestes en S_{alta}, $S_{mitjana}$ i S_{baixa} als valors 1, 10 i ∞, respectivament. A més **(R9)**, només desdoblarem els vèrtexs en D i l'arc connectant les dues còpies tindrà capacitat 2. La resta de la resolució és idèntica a l'apartat (a).

4.13. **(Un joc bi-personal)**

Tenim dos jugadors P_0 i P_1 i dues llistes, X i Y. La llista X conté els noms d'n actrius amb les pel·lícules on han participat, i la llista Y conte els noms d'n actors i les pel·lícules on han participat. Cada jugador disposa de còpies de les llistes X i Y amb la informació completa de qui ha aparegut en una pel·lícula amb qui.
El joc consisteix en el següent, el jugador P_0 diu el nom d'una actriu $x_1 \in X$, aleshores P_1 ha de donar el nom d'un actor $y_1 \in Y$ que ha aparegut a una pel·lícula amb x_1, a continuació P_0 ha de donar el nom d'una actriu $x_2 \in X$, $x_2 \neq x_1$, que ha aparegut a una pel·lícula amb y_1, i així successivament fins que un dels dos jugadors no pot donar cap nom diferent. Noteu que la restricció important és que no es poden repetir noms al llarg de la partida. El jugador que per primer cop no pot donar cap nom és el jugador que perd.
Podeu observar que a una partida P_0 i P_1 generen col·lectivament una seqüència

$$x_1, y_1, x_2, y_2, \ldots, x_i, y_i, \ldots,$$

de manera que cada actor/actriu en la seqüència ha coprotagonitzat al costat de l'actriu/actor immediatament anterior, i no es repeteixen noms.
Una estratègia per al jugador P_0 és un algorisme que pren com a entrada una seqüència $x_1, y_1, x_2, y_2, \ldots, x_i, y_i$ i genera un x_{i+1} legal, si això és possible. De la mateixa manera podem definir una estratègia per a P_1.

- Doneu una formalització de les propietats que es compleixin quan una partida és guanyadora per a P_0 i el mateix quan ho és per a P_1.
- Dissenyeu un algorisme de temps polinòmic per a decidir si un dels dos jugadors pot forçar una estratègia on ell guanyi. Ajuda: penseu com farien servir els jugadors un *matching* entre actrius i actors en dissenyar les seves respectives estratègies.

Una solució:

Para analizar las estratégias del juego y formalizar las propiedades que nos piden considero el grafo bipartido $G = (X, Y, E)$ en el que tenemos un vértice por cada actriz (X) y un vértice por cada actor (Y) y una arista entre actriz y actor si han trabajado juntos en alguna película.

Una partida ganadora para P_0 es una secuencia $x_1, y_1, x_2, y_2, \ldots, x_i, y_i$ representa una partida ganadora para P_1 si y solo si $x_1, y_1, x_2, y_2, \ldots, x_i, y_i$ es un camino simple en G tal que y_i no tiene ningún vecino fuera de $x_1, \ldots x_i$. Notemos también que la secuencia tiene longitud impar.

Una partida ganadora para P_1 es una secuencia $x_1, y_1, x_2, y_2, \ldots, x_i$ representa una partida ganadora para P_0 si y solo si $x_1, y_1, x_2, y_2, \ldots, x_i$ es un camino simple en G tal que x_i no tiene ningún vecino fuera de $y_1, \ldots y_{i-1}$. Notemos también que la secuencia tiene longitud impar.

Siguiendo la ayuda, podemos ver que en una partida ganadora para P_0, $x_1, y_1, x_2, y_2, \ldots, x_i, y_i$, aparecen dos matchings entrelazados: $(x_1, y_1), \ldots (x_i, y_i)$ determinado por las respuestas de P_1 y $(y_1, x_2), \ldots (y_{n-1}, x_n)$ determinado por las respuestas de P_0. Notemos que los dos matchings

tiene el mismo número de aristas.

En una partida ganadora para P_1, $x_1, y_1, x_2, y_2, \ldots, x_i$, aparecen dos matchings entrelazados: $(x_1, y_1), \ldots (x_{i-1}, y_{i-1})$ determinado por las respuestas de P_1 y $(y_1, x_2), \ldots (y_{i-1}, x_i)$ determinado por las respuestas de P_0. Notemos que en este caso el segundo matching tiene una arista menos que el primero.

Parece que una forma de forzar que P_1 gane (independientemente de lo que haga P_0) es cuando podemos garantizar que no hay un matching para P_0 de longitud mayor que el usado por P_1. Esto pasa cuando el grafo tiene un matching perfecto.

Si G tiene un matching perfecto M y P_1 responde siempre de acuerdo con la pareja establecida por M, P_1 gana, ya que sea la que sea la actriz que proponga P_0, siempre tiene actor emparejado con ella en M que $P-1$ todavía no a propuesto.

Si G no tiene un matching perfecto, considero un matching M de cardinalidad máxima. Como en M no es perfecto al menos hay una actriz que no está emparejada en M. Si P_0 empieza con una actriz no emparejada en M y responde siempre de acuerdo con M ganará independientemente de lo que haga P_1. Para que P_1 gane tendria que encontrar una manera de solapar con el matching M formando un camino que empiece en X en un actriz fuera de M y acabe en Y en un actor fuera de M, pero entonces podríamos ampliar M en al menos una arista y no sería un matching de tamaño máximo.

A la vista de este análisis si G tiene un matching perfecto P_1 tiene una estrategia ganadora y en caso contrario P_0 tiene una estratégia ganadora.

Para obtener un matching the tamaño máximo en un grafo bipartido hemos visto en clase un algoritmo basado en flujo con coste $O(n(n+m)$. Dependiendo de si es perfecto o no podemos decidir cual de los dos jugadores tiene una estratégia ganadora y la estratégia que le llevará a ganar.