IN7K3-1 Flujos en Redes - Verano 2011	18 de octubre 2011

Auxiliar 1-Pauta

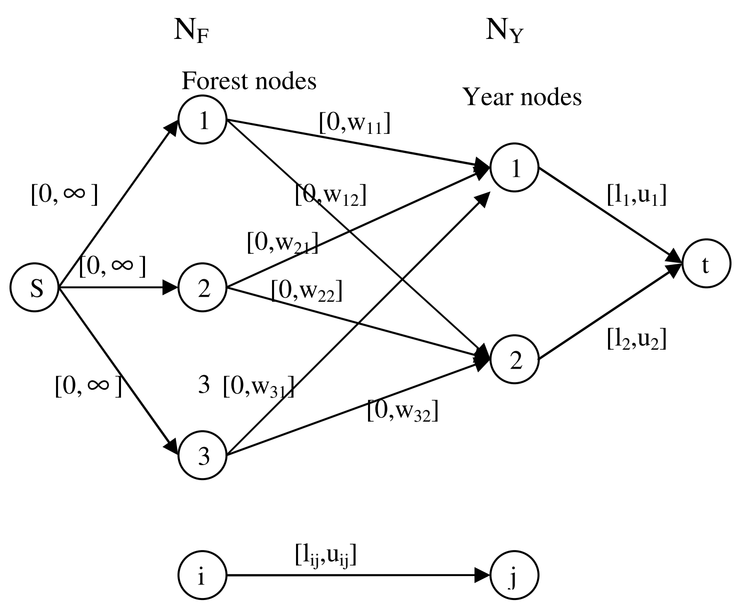
Profesor: Fernando Ordóñez — *Auxiliar: Renaud Chicoisne*

Ejercicio 1.4

Construir el grafo bipartido $G = (N_p \cup N_j, A)$ tal que cada nodo de N_p corresponda a una persona, cada nodo de N_j corresponda a un trabajo, y $A = N_p \times N_j$. El peso de cada arco (i,j) es d_{ij}. Un *maximum weight matching* en G maximiza la utilidad total de las asignaciones.

Ejercicio 1.10

Construir el grafo $G = (\{s\} \cup N_F \cup N_Y \cup \{t\}, A)$, adonde s es el nodo $fuente$, N_F contiene los nodos correspondiendo a cada bosque, N_Y contiene los nodos correspondiendo a cada año, y para cada par bosque-año $[i,j]$, existe un arco (i,j) de capacidad w_{ij}. Para cada nodo $i \in N_F$, existe un arco (s,i) de capacidad infinita. Para cada nodo $j \in N_Y$, existe un arco (j,t) con cotas inferior y superior l_j y u_j respectivamente. Luego, se resuelve a problema de flujo máximo de s a t. El flujo x_{ij} pasando por cada arco (i,j) (ie: para cada par bosque-año $[i,j]$) representa la cantidad de madera a cosechar en el bosque i durante el año j. Se puede ver esta formulación en la siguiente figura.



Ejercicio 2.10

Necesario. Supongamos que el arco (i,j) pertenece a algun ciclo de G. Sea $i - j - i_1 - i_2 -$

$\ldots - i_k - i$ este ciclo. Entonces, incluso si se borra el arco (i, j), $j - i_1 - i_2 - \ldots - i_k - i$ es un camino en el grafo resultante, i.e., incluso si se borra el arco (i, j) del grafo, el nodo i queda conectado al nodo j por algun camino. En consequencia, si el grafo G era originalmente conexo, lo sigue siendo despues de borrar el arco (i, j) de G.

Suficiente. Supongamos que el grafo queda conexo despues de borrar el arco (i, j). Porque el grafo queda conectado despues de haberle quitado el arco (i, j), el nodo i está conectado al nodo j por algún camino. Sea $j - i_1 - i_2 - \ldots - i_k - i$ este camino en el grafo resultante. Entonces $i - j - i_1 - i_2 - \ldots - i_k - i$ forma un ciclo en el grafo original y (i, j) pertenece a este ciclo.

Ejercicio 2.28

Necesario Si G es fuertemente conexo, cualquier subconjunto $S \subset N$ de nodos vecinos$(S) \neq \emptyset$. Sino fuera el caso, el corte $[S, \bar{S}]$ no tuviera ningun arco $forward$ y ningun nodo en $\bar{S}$ sería alcanzable desde cualquier nodo de S.

Suficiente. Supongamos que tenemos vecinos$(S) \neq \emptyset$ para cualquier subconjunto $S \subset N$. Sean k y l dos nodos arbitrarios de G. Aplicamos el siguiente método. Empezamos con ningún nodo marcado. Marcamos el nodo k y tratamos de marcar lo más nodos posible con la siguiente iteración: Si (i, j) es un arco del grafo, el nodo i está marcado y el nodo j no. Entonces se marca j y pred$(j) = i$. Sea S el conjunto de nodos marcados durante la ejecución del algoritmo. Porque neighbor$(S) \neq \emptyset$ para cualquier conjunto $S \subset N$, mientras tengamos $S \neq N$, se puede marcar más nodos. Entonces, este método logrará marcar el nodo I y los índices de los predecesores entragarán un camino dirigido del nodo k al nodo I. En consequencia, demostramos que cada par de nodos (k, I) está conectado por un camino dirigido. En otras palabras, G es fuertemente conexo.

Ejercicio 2.50

Sea $G = (N, A)$ el grafo en lo cual el problema de flujo de costo mínimo está definido. Construimos el grafo $G = (N \cup \{s, t\}, A1)$, adonde $A1 = A \cup \{(s, i) : i \in N, b(i) > 0\} \cup \{(i, t) : i \in N, b(i) < 0\} \cup \{(t, s)\}$. Pongamos las cotas inferiores y superiores de cada arco(s, i) o (i, t) a $|b(i)|$. El arco (t, s) tiene capacidad ∞ y no tiene cota inferior. Esos arcos adicionales tienen costo cero. Se puede observar que existe una correspondencia uno a uno entre las soluciones factibles del problema original y las del problema transformado.

Ejercicio 3.32

(a) Dado un árbol de *depth first search* (dfs) teniendo su raíz en 1, un nodo j es un descendiente del nodo i si el camino de 1 a j pasa por i. Dado que el ordenamiento del dfs está construido según el órden en lo cual los nodos son visitados order$(i) >$ order(j). Se nota que la misma demostración vale tambien para un árbol de *breadth first search* (bfs).

(b) En un grafo de dos nodos, queda evidente que el arbol dfs tiene todos los descendientes de un nodo ordenados consecutivamente. Supongamos que es cierto para un grafo de n nodos, y ocupamos un razonamiento por inducción para mostrar que es también cierto para un grafo de $n+1$ nodos. En el dsf de un grafo de $n+1$ nodos, todos los nodos son descendientes del nodo 1 y son numerados consecutivamente. Sea i cualquier nodo del grafo. Cuando el algoritmo dfs visita el nodo i, y asigna order(i), le pone al final del arreglo LIST. Dado que las siguientes iteraciones sacarán y insertarán nodos al final del arreglo LIST, todos los otros nodos del arreglo LIST no influencian el algoritmo hasta que i sea sacado definitivamente. Esto fue aplicar el dfs-arbol en el subgrafo de los descendientes de i. Por inducción, sabemos que dfs aplicado al subgrafo numerará todos los descendientes secuencialmente, dándoles números de 1 a l. Entonces la numeración correspondiente para cualquier nodo del grafo de $n+1$ nodos será secuencial de order(i) to order(i) $+ l - 1$.

Ejercicio 3.50

Se sabe que cualquier ciclo dirigido del flujo contiene s, t, exactamente un nodo del conjunto $\{1, \ldots, k\}$, y exactamente un nodo del conjunto $\{k+1, k+2, \ldots, 2k\}$. Entonces, encontrar una decomposición se reduce al problema de encontrar un *matching* entre los nodos $\{1, \ldots, k\}$ y los nodos $\{k+1, k+2, \ldots, 2k\}$. Existen $k!$ *matchings* posibles. Para contar los *matchings* posibles, consideramos que el nodo 1 puede ser apareado con un nodo de $k+1$ a $2k$, lo que ofrece k posibilidades distintas. El nodo 2 puede ser apareado con cualquier nodo de $k+1$ a $2k$, excepto el apareado con 1, que son $k-1$ posibilidades. De la misma manera, el nodo 3 puede ser apareado con $k-2$ nodos, 4 con $k-3$, etc... hasta el ultimo nodo k que puede asociarse con solamente uno. En consequencia, el numero total de *matchings* es $k(k-1)(k-2)\cdots 1 = k!$