MA-47A Optimización Combinatorial **26 de Abril, 2003**

Pauta Control No. 1

Prof. Cátedra: M. Kiwi *Prof. Auxiliar: G. Espinoza*

PROBLEMA 1:

(i).- Sean H, e y f como en el enunciado. Observar que $(H\cup\{e\})\setminus\{f\}$ también es un árbol generador de G. Por minimalidad de H, tenemos que $c(H) \leq c((H \cup \{e\}) \setminus \{f\}) = c(H) + c(e) - c(f)$. Luego, $c(e) \geq c(f)$.

(ii.1).- Supongamos que G es Euleriano. Luego, existe un ciclo Euleriano C en $G = (V, E)$. Como C es Euleriano, el número de arcos incidentes en $v \in V$ es igual a dos veces el número de veces en que C pasa por v_0, es decir, una cantidad par. Como v es arbitrario, se tiene que no existen nodos en G en los que incidan un número impar de arcos.

Supongamos ahora que en todo nodo de G inciden un número par de arcos. Sea C un camino en G cerrado y que recorre el máximo número de arcos de G una sola vez. Si C pasa por todos los arcos de G sería un ciclo Euleriano y estaríamos listos. Supongamos entonces que hay algún arco de $e_0 = v_0v_1 \in E$ que no esta en C.

Podemos suponer que C pasa por v, pues de lo contrario, tomamos un nodo u cualquiera por el que pase C, y por conexidad sabemos que hay un camino $\mathcal{P}$ de u a v. El último arco $e'_0 = v'_0v'_1$ de $\mathcal{P}$ que no pertenece a C cumple con la propiedad deseada, i.e., que C pasa por v_0.

Ahora, notar que por hipótesis, en $G \setminus (C \cup \{e_0\})$ v_1 tiene un número impar de arcos incidentes. Luego, hay al menos un arco $e_1 = v_1v_2 \in E \setminus (C \cup \{e_0\})$. Si $v_2 \neq v_0$, se puede repetir el argumento cuantas veces sea necesario hasta concluir que existe un camino $e_0, e_1, \ldots, e_k \in E \setminus C$ de v_0 a v_0, i.e., cerrado.

Sin pérdida de generalidad podemos denotar los arcos de C por $e'_0, \ldots, e'_{k'}$ de manera que la cola de e'_0 y la cabeza de $e'_{k'}$ correspondan precisamente a v_0. Sigue que $e_0, \ldots, e_k, e'_0, \ldots, e'_{k'}$ es un camino en G cerrado y que recorre $k' > 0$ más arcos que C. Esto contradice la maximalidad de C.

(ii.2).- El argumento de la parte anterior es constructivo y da lugar a un algoritmo. Una implementación eficiente corresponde a la de la Figura 1.

La implementación considera que el grafo se representa mediante un lista de adyacencia, que se denota $Lista(E)$. Inicialmente esta lista contiene todos los arcos del grafo. En los pasos 10–20 se eliminan arcos de la $Lista(E)$. En particular, $Extraer(e = vw, Lista(E))$ denota la operación consistente en extraer de la $Lista(E)$ un arco e cualesquiera incidente en v (el otro extremo de e distinto a v se denota por w). $Lista(C)$ corresponde a una lista doblemente linkeada que al comienzo de el paso 10 represena un circuito C en G (al terminar el algoritmo C corresponde a un circuito Euleriano en el caso que el grafo sea Euleriano). Cada elemento de la $Lista(C)$ representa un nodo del grafo. Nuevos nodos se insertan en la lista vía el procedimiento $Insertar(w, v, Lista(C))$, que consiste en agregar a $Lista(C)$ un elemento representado el nodo w a continuación del elemento representado por v (en el caso particular en que $v = NULL$, se crea la $Lista(C)$ con un elemento representando a w). El procedimiento $Next(v, Lista(C), Lista(E))$ retorna, ENTRADA: Grafo $G = (V, E)$.

CÓDIGO:

1. **Si** G no es conexo **entonces**
2.   **return** "Grafo no Euleriano"
3. **Para todo** $v \in V$
4.   **Si** $|\delta(v)|$ es par **entonces**
5.   **return** "Grafo no Euleriano"
6. $/\star$ G es Euleriano $\star/$
7. $Lista(C) \leftarrow \emptyset$
8. $v^* \leftarrow NULL;\ v \leftarrow v^*;$
9. **Mientras** $(v^* \leftarrow Next(v^*, Lista(C), Lista(E))) \neq NULL$
10.   **Si** $\delta(v^*) = \emptyset$ **entonces** ir a 9.
11. $Extraer(e = v^*w, Lista(E))$
12. $Insertar(v^*, v, Lista(C))$
13. $v \leftarrow v^*$
14.   **Mientras** $w \neq v^*$
15. $Insertar(w, v, Lista(C))$
16. $v \leftarrow w$
17. $Extraer(e = vw, Lista(E))$
18. **return** $Lista(C)$

Figura 1: Algoritmo de determinación de un ciclo Euleriano.

- un nodo v^* tal que $\delta(v^*) \neq \emptyset$ si $v = NULL$ (este caso sólo ocurre cuando la $Lista(C)$ esta vacia),
- el siguiente nodo representado por un elemento a continuación de v^* en $Lista(C)$ que tenga arcos incidentes entre los representados en $Lista(E)$,
- $NULL$ si $Lista(E) = \emptyset$.

El paso 1–2 se puede implementar vía el algoritmo de búsqueda en profundidad o búsqueda horizontal, luego, toma $O(|V| + |E|)$. Los pasos 4–5 toman, dado que el grafo esta representado por una lista de adyacencia, exactamente $O(|\delta(v)|)$. Luego, los pasos 3–5 toman en total $O(|V| + |E|)$. Como para un grafo conexo $|V| = O(|E|)$, hemos concluido que toma $O(|E|)$ decidir si un grafo es Euleriano.

Si el algoritmo llega al paso 7, se tiene que la entrada corresponde a un grafo Euleriano. Los pasos 7–8, 10, 13 y 16 toman $O(1)$. Además, es fácil implementar el procedimiento $Extraer(vw, Lista(E))$ en tiempo $O(1)$ de forma que retorne el primer arco en la lista de adyacencia de v en $Lista(E)$. Cómo insertar en una lista doblemente linkeada toma $O(1)$, concluimos que todos los pasos 10–13 y 15–17 toman $O(1)$. Cómo en cada ejecución de 10–13 y 15–17 la cantidad de arcos en $Lista(E)$ decrece en al menos 1, estos se ejecutan a lo más $O(|E|)$ veces.

Lo único que queda por verificar es el tiempo total que toma ejecutar el paso 9 es $O(|E|)$. En efecto el trabajo total que implica ejecutar el paso 9 es a lo más el número de arcos que tiene un circuito Euleriano, i.e., $O(|E|)$.

PROBLEMA 2:

(i).- Para calcular π^s basta cualquier algoritmo de los vistos en clase para determinar, en digrafos sin dicircuitos de costo negativo, el costo mínimo de los caminos en G a todos los nodos del digrafo partiendo de s. En particular, el algoritmo de Bellman–Ford puede ocuparse (no así el algoritmo de Dijkstra puesto que la función de costos no es necesariamente no–negativa).

La determinación de π^t se puede abordar en forma análoga, pero invirtiendo la orientación de los arcos de G y tomando t como el punto de partida.

La complejidad que se obtiene al ocupar Bellman–Ford es $O(|V||E|)$.

(ii).- Sea $\mathcal{P} = e_0, \ldots, e_l$ un (s,t)–dicamino que ocupa $e_i = e = vw$ de costo mínimo. Por sub–optimalidad de los caminos de costo mínimo[1], se tiene que $\mathcal{P}_s = e_0, \ldots, e_{i-1}$ y $\mathcal{P}_t = e_{i+1}, \ldots, e_l$ son (s,v) y (w,t)–dicaminos respectivamente en G de costo mínimo. Luego,

$$c(\mathcal{P}) = c(\mathcal{P}_s) + c(e) + c(\mathcal{P}_t) = \pi^s(v) + c(e) + \pi^t(w)\,.$$

(iii).- Considere el algoritmo de la Figura 2. La correctitud del algoritmo se desprende de (ii).

ENTRADA:		Digrafo $G = (V,E)$ y $c : E \to \mathbb{R}$.
CÓDIGO:	1.	Calcular π^s, π^t y un (s,t)–dicamino $\mathcal{P}$ de costo mínimo vía Bellman–Ford
	2.	$M \leftarrow +\infty$
	3.	**Para todo** $e = vw \in E \setminus \mathcal{P}$
	4.	$M \leftarrow \min\{M, \pi^s(v) + c(e) + \pi^t(w)\}$
	5.	**Return** M y $\pi^s(t)$

Figura 2: Algoritmo de determinación del costo de un primer y segundo (s,t)–dicamino de costo mínimo.

Como ya se vió en (i), el paso 1 toma $O(|V||E|)$. El paso 2 claramente toma $O(1)$ y el paso 3 toma $O(|E|)O(1) = O(|E|)$. Luego, en total el algoritmo toma $O(|V||E|)$ (la constante escondida en la $O(\cdot)$ es, asintóticamente, el doble que la constante involucrada en el tiempo de ejecución de Bellman–Ford).

PROBLEMA 3:

(i).- Supongamos que x es un flujo factible óptimo. Sea $\mathcal{P} = e_0, \ldots, e_k$ un camino aumentador en $G(x)$. Sean $\mathcal{P}^+$ y $\mathcal{P}^-$ los arcos de G que $\mathcal{P}$ recorre hacia adelante y hacia atrás respectivamente. Sean además,

$$\begin{aligned} \epsilon^+ &= \min\{u(e) - x(e) \,:\, e \in \mathcal{P}^+\}\,, \\ \epsilon^- &= \min\{x(e) \,:\, e \in \mathcal{P}^-\}\,. \end{aligned}$$

Observar que $\epsilon^+ > 0$ puesto que por definición de $G(x)$ y $\mathcal{P}^+$ se tiene que $u(e) > x(e)$. Análogamente, se verifica que $\epsilon^- > 0$.

[1] I.e., un dicamino contenido en un dicamino de costo mínimo es también un camino de costo mínimo.

Sea entonces $\epsilon = \min\{\epsilon^+, \epsilon^-\}$ y $x' \in \mathbb{R}^E$ tal que

$$x'(e) \quad = \quad \begin{cases} x(e) + \epsilon\,, & e \in \mathcal{P}^+\,, \\ 0\,, & e \notin \mathcal{P}\,, \\ x(e) - \epsilon\,, & e \in \mathcal{P}^-\,. \end{cases}$$

Es fácil comprobar que x' es un flujo factible en G y que tiene un valor

$$f_s(x') = \sum_{e \in E : t(e) = s} x'(e) - \sum_{e \in E : h(e) = s} x'(e) = f_s(x) + \epsilon > f_s(x)\,.$$

Esto contradice la optimalidad de x.

(ii).- Sea $G = (L, R, E)$ un grafo bipartido.

Sea M un cuplaje en G de tamaño máximo. Cualquier recubrimiento de G debe contener al menos uno de los extremos de cada arco en M. Luego,

$$\max\{|M| \,:\, M \text{ cuplaje de } G\} = |M| \leq \min\{|C| \,:\, C \text{ recubrimiento de } G\}\,.$$

Obtendremos el teorema como consecuencia del Teorema del Flujo–máximo/Corte–mínimo. Para ello, al grafo bipartido $G = (L, R, E)$ le asociamos la red G' de la Figura 3. Formalmente, G' es un

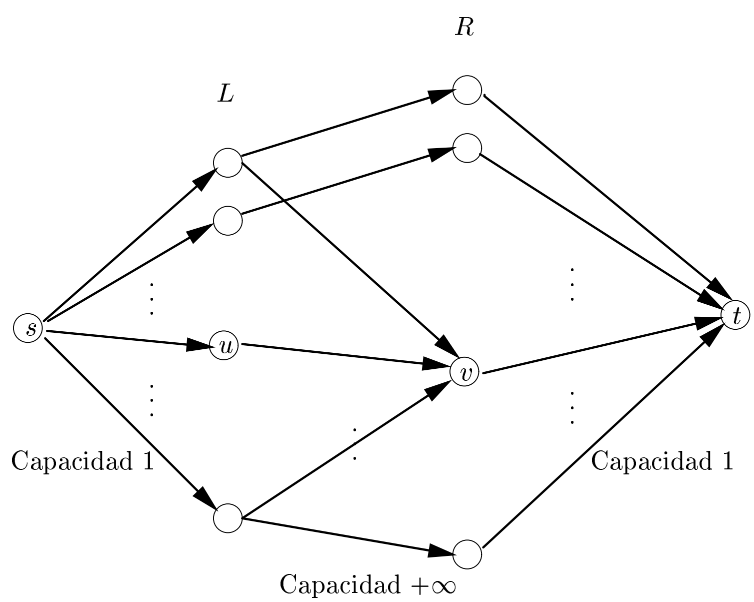

Figura 3: Red G' asociada al grafo bipartido G.

digrafo tal que $V(G') = V(G) \cup \{s, t\}$ y $e = vw \in E(G')$ si

- $v = s$ y $w \in L$, o
- $vw \in E$, o

- $v \in R$ y $w = t$.

La capacidad de los arcos saliendo de s es 1, la de los arcos entrando en t también, todo los arcos restantes tienen capacidad $+\infty$.

Por integralidad de las capacidades, existe un flujo factible máximo x integral. Notar que la integralidad de x y la capacidad unitaria de los arcos que salen de s implica que a cada nodo de L puede llegar o 0 o 1 unidad de flujo. Lo mismo sucede con cada nodo en R, nuevamente por la integralidad de x y ahora porque por un arco entrante a t sólo puede desaguarse a lo más una unidad de flujo. De aquí que para cualquier $e = uv \in E(G')$, $u, v \notin \{s, t\}$, que acarrea flujo es tal que de u no sale ni a v llega otro arco con flujo. De lo anterior se deduce que cada arco de G da lugar a un arco en G' por donde pasa o 0 o 1 unidad de flujo. Sea M el conjunto de arcos de G que dan lugar a un arco $e \in G'$ tal que $x(e) = 1$. Claramente, M es un cuplaje de G tal que

$$f_s(x) \quad = \quad |M|\,.$$

Por otro lado, es fácil asociar a cualquier cuplaje M en G un flujo x en G' tal que $f_s(x) = |M|$. Luego,

$$f_s(x) \quad = \quad \max\{|M| \,:\, M \text{ es un cuplaje en } G\}\,.$$

Del Teorema del Flujo–Máximo/Corte–Mínimo, concluimos que

$$\max\{|M| \,:\, M \text{ es un cuplaje en } G\} \quad \geq \quad \min\{u(\delta(S)) \,:\, \delta(S) \text{ es un } (s,t)\text{–corte en } G'\}\,.$$

Sea entonces S tal que $\delta(S)$ es un (s,t)–corte en G' tal que $u(\delta(S))$ es mínimo. Notar que $u(\delta(S)) < +\infty$. Sean $L_s = L \cap S$, $L_t = L \cap \overline{S}$, $R_s = R \cap S$, y $R_t = R \cap \overline{S}$ (ver Figura 4). Como $u(\delta(S))$ es finito, no pueden existir arcos en G' que vayan de L_s a R_t y debe tenerse que $u(\delta(S)) = |\delta(S)| = |L_t| + |R_s|$. Luego, $C = L_t \cup R_s$ es un recubrimiento de G, tal que

$$\min\{|C| \,:\, C \text{ es un recubrimiento de } G\} \leq |C| = u(\delta(S)) \leq \max\{|M| \,:\, M \text{ es un cuplaje en } G\}\,.$$

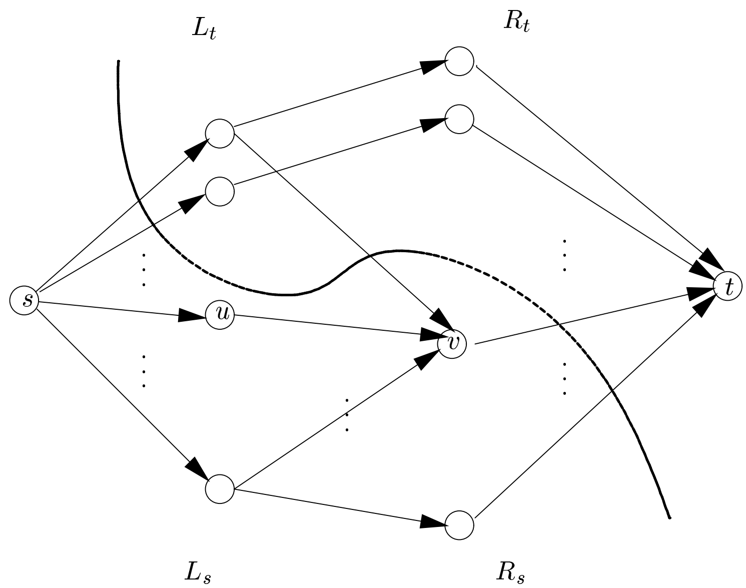

Figura 4: $\delta(S)$ corte y definición de L_s, L_t, R_s y R_t.