Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Tópicos Avanzados en Optimización Combinatoria y Teoría de Grafos

PRÁCTICA 3

Alumno: Rodríguez Jeremías

5 de mayo de 2017

1. Ejercicio 2

1.1. Apartado A

Sea G conexo, y sea $T \subseteq E(G)$ un bosque maximal respecto a la cantidad de aristas. Es decir:

$$e \in E(G) \Rightarrow T \cup \{e\} \text{ no es}$$

Supongamos que T no es un árbol. Entonces tiene al menos dos componentes conexas. Como el grafo G es conexo, habrá alguna arista e que conecta dos vértices de componentes distintas.

El arista e cláramente no pertenece a T (pues une dos componentes distintas). Como ambas componentes son árboles, $T \cup \{e\}$ seguirá siendo un bosque (no podrá haber otro par de vértices conectados formando un ciclo).

Luego, T no era bosque maximal, pues hallamos un bosque con más aristas. Absurdo.

$\therefore T$ es árbol

1.2. Apartado B

Sea (G, c) una instancia del problema del arbol recubridor de menor peso. Asumo que c es un vector de costos positivos asociado a cada arista y que G es conexo.

Sea $m = max_{e\ \in E(G)} c_e$ es el máximo costo de una arista en c; y defino c' tal que $c'_e = 1 + m - c_e$.

Hechas estas definiciones, propongo la siguiente reducción entre los dos problemas:

$$\begin{aligned} f(G, c) &= (G, c') \\ h &= id \end{aligned}$$

Veamos que, efectivamente, es una reducción. Sea g un algoritmo que resuelve el problema del bosque de peso máximo. Debo probar que $h \circ g \circ f$ resuelve el problema del árbol recubridor de costo mínimo para (G,c).

Es decir, debo demostrar que si $T \subseteq E(G)$ es un bosque de peso máximo en G con costos c', T es un árbol de peso mínimo en G con costos c.

Como G es conexo, T será un árbol de peso máximo en G con costos c' (Apartado anterior). Supongamos que T no es un árbol de peso mínimo en G con costos c. Entonces existe otro árbol T' tal que:

$$\sum_{e \in T'} c_e < \sum_{e \in T} c_e$$

Nótese que al ser árboles recubridores del mismo grafo, ambos tendrán la misma cantidad de aristas n. Por la desigualdad anterior se cumple:

$$n.(m+1) - \sum_{e \in T'} c_e > n.(m+1) - \sum_{e \in T} c_e$$

$$\sum_{e \in T'} (m + 1 - c_e) > \sum_{e \in T} (m + 1 - c_e)$$

$$\sum_{e \in T'} c'_e > \sum_{e \in T} c'_e$$

Entonces T no es bosque de peso máximo de G con costos c'. Absurdo. Luego, T es un árbol de peso mínimo en G con costos c.

$\therefore\ h \circ g \circ f$ resuelve el problema del árbol recubridor de costo mínimo para (G,c).

1.3. Apartado C

Sea $G = (V, E)$ grafo; definimos $S = E$, $I = \{F \subseteq S \mid F \text{ es un bosque de } G\}$. Veamos que es matroide:

1. $\varnothing \in F$ pues es un bosque (no posee ciclos).

2. Sea $F_1 \in I$ y $F_2 \subseteq F_1$. Como F_1 es un bosque, no posee ciclos. Luego, cualquier subconjunto de aristas de F_1 no podrá poseer ciclos. Entonces, $F_2 \in I$.

3. Sean $F_1, F_2 \in I$ tales que $|F_1| < |F_2|$. Si hay un arista $e \in F_2 - F_1$ que conecta dos componentes diferentes de F_1, entonces $F_1 \cup \{e\} \in I$.

   Supongamos que no existe ningún arista $e \in F_2 - F_1$ conectando dos componentes distintas de F_1. Es decir, para toda arista $e \in F_2 - F_1$, ambos extremos están en la misma componente de F_1.

   Sea k_1 el número de componentes conexas de F_1. Entonces, el número de componentes conexas k_2 de F_2, verificará $k_2 \geqslant k_1$.

   Entonces $|F_1| = |V| - k_1 \geqslant |V| - k_2 = |F_2|$. Absurdo.

$\therefore$ (S, I) es matroide.

1.4. Apartado D

Debo probar que el algoritmo de Kruskal resuelve el Problema del Árbol Generador Mínimo (es decir, el resultado es un árbol recubridor de peso mínimo). No lo voy a resolver utilizando la sugerencia porque no logré llegar al resultado usándola.

Sea G el grafo input, c el vector de costos y O el grafo output.
En primer lugar, observemos que el conjunto de aristas de O es un bosque, pues en cada iteración se agrega un arista siempre y cuando no forme ciclos con las anteriores. Entonces el resultado final no tendrá ciclos.

Veamos que si el G es conexo, O será un árbol. Supongamos que no es árbol, entonces tomamos dos componentes conexas de O. En el grafo G habrá al menos un arista e conectando vos vértices de estas componentes de O, pues G es conexo. Consideramos, de entre estas aristas e, la de menor peso. Kruskal debería haberla agregado en la iteración que la consideró, pues no generaba ciclos. Absurdo. Luego O es un árbol.

Falta ver que es de peso mínimo. Supongamos que no es de peso mínimo. Entonces hay otro árbol recubridor T tal que la sumatoria de sus aristas es de menor costo. Recordemos que tanto T como O tienen la misma cantidad de aristas pues son árboles recubridores. Sean:

$o_1,\ o_2$ ... o_p las aristas de O ordenadas de menor a mayor costo
$t_1,\ t_2$ ... t_p las aristas de T ordenadas de menor a mayor costo

Sea $k = min_i\ tq\ c_{t_i} < c_{o_i}$. Claramente ese k existe pues T tiene menor peso total que O. Pero entonces, el algoritmo de kruskal hubiera elegido en el paso k al arista t_k en vez del arista o_k. Absurdo.
Luego el árbol hallado es de peso mínimo.

2. Ejercicio 10

Sea $G = (V, E)$, c vector de costos y $a \in V$ tal que para todo $v \in V$ hay un av camino dirigido. Voy a probar el contrarecíproco:

Hay un ciclo de costo total negativo $\Leftrightarrow$ $\exists\ v \in V$, no hay un camino dirigido av de costo mínimo.

$\Rightarrow$) Supongamos que hay un ciclo de costo total negativo $-|k|$ y sea c un vértice perteneciente a este ciclo. Supongamos que hay un ac-camino dirigido de costo mínimo n. Entonces, podemos armar un ac camino dirigido utilizando ese camino ac y luego dando una vuelta al ciclo c-c. Este nuevo camino dirigido ac tendrá costo $n - |k| < n$. Absurdo pues el costo mínimo era n. Por lo tanto, no hay ac-camino de costo mínimo para c.

$\Leftarrow$) Supongamos que para un vértice v, no hay un camino dirigido av de costo mínimo. Esto es, para todo $k \in \mathbb{R}$, hay un camino av de costo menor a k.

Sabemos que la cantidad de caminos sin ciclos de a hacia v es finita. Sea c el mínimo costo de estos caminos sin ciclos desde a hacia v. Por hipótesis habrá un camino de costo menor a c.

La única forma de que exista es que contenga algún ciclo. Si todos los ciclos que contiene son no negativos, entonces el costo del camino no sería menor a c. Luego tiene algún ciclo de costo negativo.

3. Ejercicio 11

Asumo que no hay ciclos negativos (y que hay siempre un camino aw).

3.1. Apartado A

Sea un aw-camino dirigido con n vértices:

$$v_1\ v_2\ ...\ v_n,\ v_1 = a,\ v_n = w.$$

Veamos, por inducción, lo siguiente:

P(k): $k \leqslant n \ \Rightarrow\ L(v_k) \leqslant c(v_1\ ...\ v_k)$ (El coste del camino hasta v_k)

<u>Caso base</u>: P(1). $L(v_1) = L(a) = 0$ por hipótesis, y 0 es igual al costo de un camino vacío.
<u>Paso recursivo</u>: Supongo que vale $P(k-1)$:

$$L(v_{k-1}) \leqslant c(v_1\ ...\ v_{k-1}),$$

¿$L(v_k) \leqslant c(v_1\ ...v_{k-1}\ v_k)$?

Tenemos un arista $e = v_{k-1}v_k$ de costo c_e. Por ser L potencial factible, tenemos que:

$$L(v_k) \leqslant L(v_{k-1}) + c_e$$

Uniendo estas dos desigualdades:

$$L(v_k) \leqslant c(v_1\ ...\ v_{k-1}) + c_e = c(v_1\ ...\ v_{k-1})$$

Luego vale P(k) para todo k. En particular vale para k=n, y resulta:

$$L(w) = L(v_n) \leqslant c(v_1\ ...\ v_n) = c(P_w) \text{ con } P_w \text{ un aw-camino arbitrario.}$$

3.2. Apartado B

Sea P_w un camino de costo $L(w)$. Sea P'_w un aw-camino. Sabemos por el apartado anterior que $L(w) \leqslant c(P'_w)$. Es decir, $c(P_w) \leqslant c(P'_w)$. Luego P_w es camino de costo mínimo.

4. Ejercicio 13

4.1. Apartado A

El algoritmo siempre termina porque:

1. Cada iteración siempre termina (recorrer todas las aristas, que son finitas)
2. La cantidad máxima de iteraciones está limitada a 20.

Luego, en una cantidad finita de pasos terminará.

4.2. Apartado B

Si al terminar el algoritmo *hubo_actualización* es falso, entonces al seguir iterando no podríamos modificar más las entradas de L.

Veamos que, el L resultante en este caso es potencial factible:

- En primer lugar, L(a)=0 se cumple en todas las iteraciones.
- En segundo lugar, ¿Para toda $vw \in A$, $L(v) + c_{vw} \geqslant L(w)$? Como *hubo_actualización* es falso, para todo $vw \in A$, $L(v) + c(vw) \geqslant L(w)$ pues nunca se ejecutó en la iteración anterior la modificación de *hubo_actualización* en el paso 2.

Luego, como L es potencial factible y ademas siguiendo el camino av de las etiquetas hallamos un camino con costo exáctamente L(v), por (11.B) resulta ser camino de menor costo.