Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 9 - "Algoritmos Aproximados"

Profesor: Pablo Barceló
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

7 de Junio del 2017

P1. Knapsack

El "problema de la mochila" es un problema de optimización que busca llevar la mayor cantidad de valor en una mochila de espacio limitado. Más precisamente, se nos da una lista de n items con sus respectivos valor: v_i y espacio que ocupa: s_i y queremos llevarnos la mayor cantidad de "valor" en una mochila que tiene capacidad S.

a) De un algoritmo de programación dinámica y muestre su complejidad.

b) Considere el algoritmo que ordena de forma decreciente las densidades de los items (v_i/s_i) y luego mete a la mochila en este orden hasta que no le quede espacio.
Muestre que no existe una constante k para la cual lo anterior sea una k-aproximación.

c) Mejore el algoritmo anterior a una $2-$aproximación.

P2. "Smarter" Vertex Cover

En clases se vio una 2-aproximación para el problema de cubrimiento por vértices, el siguiente algoritmo busca aprovechar el hecho que vértices de grado alto cubren muchas aristas.

Input: $G = (V, E)$
$E' \leftarrow E$;
$C \leftarrow \emptyset$;
while $E' \neq \emptyset$ **do**
  Tomar $u \in (V \setminus C)$ de mayor grado en $G' = (V \setminus C, E')$;
  $C \leftarrow C \cup \{u\}$;
  $E' \leftarrow E' \setminus \{e \in E' : u \text{ es incidente a } e\}$;
end
return C

Muestre que no existe una constante ρ para la cual este sea una ρ aproximación del problema.

P3. Set Covering (no online)

El problema de cobertura de conjuntos consiste en dado un conjunto U de tamaño n y una serie de subconjuntos de U, $S_1, S_2, \ldots, S_m$, cuya unión es U. Encontrar conjunto de mínimo tamaño $C \subseteq [m]$, tal que $\bigcup_{i \in C} S_i$. Muestre que el algoritmo que va tomando el conjunto con más elementos y sacando sus elementos del problema es una $(\ln n)$-aproximación.

Soluciones

P1. a) Definimos $K[i,x]$ como la solución del problema de la mochila con los elementos $i, i+1, \ldots, n$ con una mochila de capacidad x. Veamos que :

$$K[i,x] = max(\underbrace{K[i+1,x]}_{\text{no cargar item } i}, \underbrace{K[i+1,x-s_i]+v_i}_{\text{cargar item } i})$$

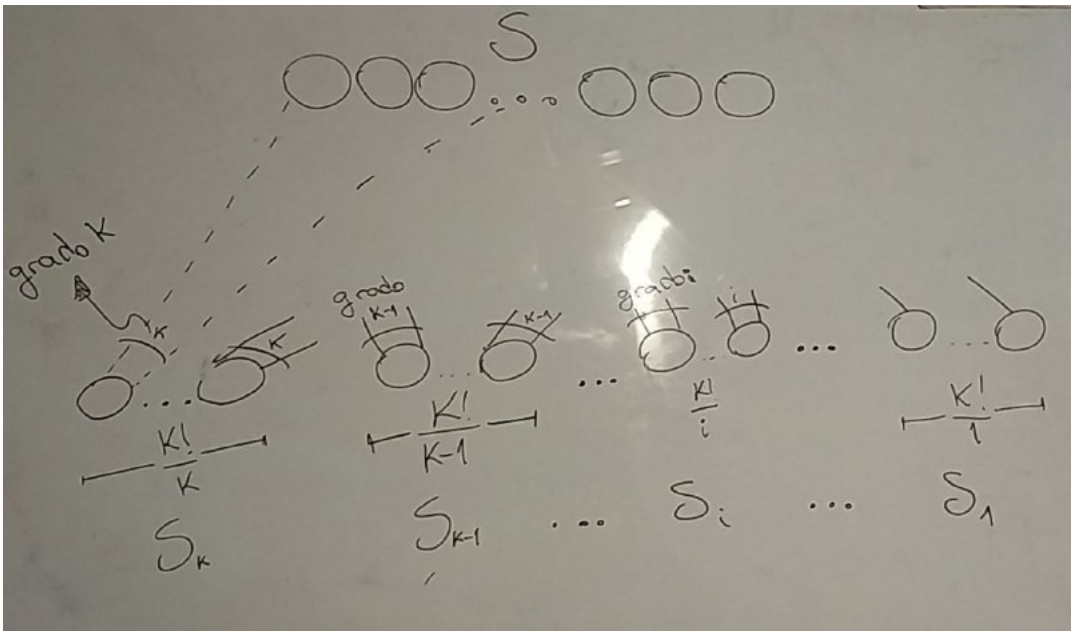
lo que nos indica que el problema tiene subestructura óptima y puede ser resuelto con un algoritmo de programación dinámica. El costo de este algoritmo es el de llenar la tabla de programación dinámica, que es el tamaño de la tabla (pues casilla cuesta calcular un máximo, $\mathcal{O}(1)$) la cual es $\mathcal{O}(nS)$. Notar que esto no contradice el hecho que Knapsack sea NP-Completo pues $\mathcal{O}(nS)$ es exponencial en el tamaño de la entrada ($\mathcal{O}(n \log S)$).

b) Como esto es un problema de maximización ser una k-aproximación significa que k veces el valor obtenido por el algoritmo es siempre mayor o igual al valor óptimo ($k \cdot ALG \geq OPT$). Por lo tanto, para mostrar que el algoritmo **no es** una k aproximación , debemos mostrar que para cualquier k existe un input en el cual $k \cdot ALG < OPT$.
Sea entonces un k y el siguiente input $S = 2k$, $(v_1, s_1) = (2,1), (v_2, s_2) = (3k, 2k)$. Como el primer elemento tiene mayor densidad que el segundo el algoritmo lo elige y después no puede poner al segundo, por lo que $ALG = 2$, mientras que el óptimo corresponde a llevar el segundo elemento con $OPT = 3k$, en donde se cumple que $k \cdot ALG = 2k < 3k = OPT$.

c) Si al aplicar el algoritmo anterior nos quedamos en la mochila con los elementos $1, \ldots, i$. Entonces nos quedaremos con esos elementos siempre y cuando $\sum_{k=1}^{i} v_k > v_{i+1}$, en otro caso nos quedamos con el elemento $i+1$.
Mostremos ahora que este algoritmo es una 2-aproximación, es decir, $2ALG \geq OPT$. Para esto veamos primero que el OPT no puede ser mayor que la suma de los valores de los primeros (en el orden de decreciente de densidad dado) $i+1$ elementos, puesto que la solución del problema relajado (donde se nos permite llevar fracciones de elementos) corresponde a llevar a los elementos más densos y a una fracción del $i+1$ que quepa en la mochila. Esto es:

$$\begin{aligned} OPT &\leq \sum_{k=1}^{i+1} v_k \\ &= \sum_{k=1}^{i} v_k + v_{i+1} \\ &\leq max\left(\sum_{k=1}^{i} v_k, v_{i+1}\right) + max\left(\sum_{k=1}^{i} v_k, v_{i+1}\right) \\ &= 2ALG \end{aligned}$$

P2. Tomemos un k cualquiera y construyamos el siguiente grafo bipartito. La partición izquierda corresponderá a un conjunto de $k!$ vértices S. La partición de la derecha serán k conjuntos disjuntos de vértices, $S_1, \ldots, S_k$; donde S_i será $\frac{k!}{i}$ vértices de grado i que con sus aristas cubren todo S.



En este grafo, el óptimo corresponde a cubrir las aristas con los $k!$ vértices de S, por lo que $OPT = k!$. Sin embargo, el algoritmo tomaría[1] primero los vértices de S_k (pues son de grado k), luego los de S_{k-1}, los de S_{k-2}, ..., los de S_1 que en total son $ALG = \frac{k!}{k} + \frac{k!}{k-1} + \ldots + \frac{k!}{1} = k! \left(\sum_{i=1}^{k} \frac{1}{i} \right) = k! H_k$.[2]

Finalmente, si suponemos que el algoritmo es una ρ-aproximación, $ALG \leq \rho OPT$, pero con el grafo anterior tenemos que: $k! H_k \leq \rho k! \Rightarrow H_k \leq \rho$. Sin embargo, H_k es creciente y no acotado[3], por lo que la última desigualdad es una contradicción.

[1] Como no se especifica que hacer frente a empate nosotros como adversario podemos tomar la decisión por el algoritmo.
[2] Número armónico.
[3] La serie diverge.

P3. Sea t la solución óptima, es decir, $OPT = t$ y nombremos U_k al conjunto de elementos luego de la iteración k del algoritmo. Notemos que:

- $U_0 = U$.
- En el paso k, U_k puede ser cubierto con t conjuntos (pues hay t que cubren todo U).
- Por palomar, al menos uno de ellos cubre al menos $\frac{|X_k|}{t}$ elementos.
- Como el algoritmo escoge el conjunto que tiene más elementos, en este paso escogerá uno que tiene al menos $\frac{|X_k|}{t}$.
- Luego, se cumple que: $|X_{k+1}| \leq \left(1 - \frac{1}{t}\right) |X_k|$.
- Y en general: $|X_i| \leq \left(1 - \frac{1}{t}\right)^i n$.
- Si el número de iteraciones es $t \ln n$, tenemos :

$$\begin{aligned} |X_{t \ln n}| &\leq \left(1 - \frac{1}{t}\right)^{t \ln n} n \\ &= \left(\left(1 - \frac{1}{t}\right)^t\right)^{\ln n} n \\ &< \left(\frac{1}{e}\right)^{\ln n} n \\ &= 1 \end{aligned}$$

esto significa que $|X_{t \ln n}| = 0$, es decir, el algoritmo luego de $t \ln n$ iteraciones ya terminó, por lo que escoge $\leq t \ln n$ conjuntos. Demostrando de esta manera que es una $(\ln n)$-aproximación ($ALG \leq t \ln n = OPT \ln n$).