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

Auxiliar 11 - "Algoritmos Online, Aproximados y Probabilísticos y Aleatorizados"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: Dustin Cobas

P1. Tareas y Procesadores

Tenemos m procesadores idénticos. Como entrada recibimos una lista $t_1, t_2, \ldots, t_n$ de tareas con tiempos de procesamiento $p_1, p_2, \ldots, p_n > 0$, respectivamente. Las tareas son recibidas secuencialmente, y sólo cuando una tarea llega conocemos su tiempo de procesamiento. Cada tarea debe ser asignada a un procesador inmediatamente, y la decisión no puede ser cambiada. La *carga* de un procesador es la suma de los tiempos de procesamiento de todas las tareas que le son asignadas. El costo de un algoritmo que resuelve el problema de asignación es la máxima carga entre sus procesadores.

Considere el siguiente algoritmo para el problema anterior. Cada tarea es asignada al procesador con menor carga (en caso de empate, se elige cualquiera).

a) Demuestre que este algoritmo es $\left(2 - \frac{1}{m}\right)$ −competitivo.

b) Demuestre que esta cota es óptima para el algoritmo.

P2. Partition

El problema de *Partition* consiste en dada una lista de números $s_1, s_2, \ldots, s_n$, particionar los índices en dos conjuntos A y B de modo que $A \cup B = \{1, 2, \ldots, n\}$ y

$$\max\left(\sum_{i \in A} s_i, \sum_{i \in B} s_i\right)$$

es mínima. *Partition* es un problema NP-completo. Diseñe una $(1+\epsilon)$-aproximación para el problema.

P3. Corte Mínimo de un Grafo

Un corte o *cut* de un grafo no dirigido conexo $G = (V, E)$, con $|V| = n$ y $|E| = m$, es un subconjunto E' de E tal que el grafo sin estas aristas $(V, E \setminus E')$, tiene dos o más componentes conexas. El corte mínimo del grafo o *min-cut* corresponde al corte del grafo que tiene menor cantidad de aristas.

Para resolver este problema utilizaremos el siguiente algoritmo:

$(V', E') = (V, E)$
for $i \leftarrow 1 \dots n-2$ **do**
  Elegimos una arista $e \in E'$ uniformemente al azar y la contraemos del grafo (V', E') actualizando V' y E'.
end
return E'

a) Muestre que el algoritmo encuentra un *cut* del grafo.

b) Muestre que la probabilidad que el algoritmo encuentre el *min-cut* es al menos $\frac{2}{n(n-1)}$.

c) Muestre como reducir la probabilidad de error (no encontrar el *min-cut*) a $\frac{1}{n^2}$.

Soluciones

P1. Tareas y Procesadores

a) Enumeremos los procesadores en orden creciente de carga luego de correr ALG. El costo de ALG queda expresado como la carga del último de estos procesadores, el m. Consideremos ahora la asignación de la última tarea del procesador m. Por como funciona el algoritmo, cuando se le fue asignada esta tarea, el procesador m tenía la menor carga entre los procesadores. Por lo tanto, el período de ocio de cada uno de los $m-1$ procesadores anteriores no puede ser mayor que la duración de esta última tarea (sino este procesador no hubiese sido escogido como el de carga menor). De este modo, estos períodos de ocio no pueden exceder la duración de la tarea más larga, $max\ p_i$. Con esto se tiene que:

$$m \times ALG \leq \sum_{i=1}^{n} p_i + (m-1)\max p_i$$

$$ALG \leq \frac{1}{m}\sum_{i=1}^{n} p_i + \left(1 - \frac{1}{m}\right)\max p_i$$

Finalmente, el óptimo no puede ser menor a la carga promedio (pues este es el óptimo del problema relajado), ni a la máxima duración entre las tareas (pues algún procesador debe procesarle).

$$ALG \leq OPT + \left(1 - \frac{1}{m}\right) OPT = \left(2 - \frac{1}{m}\right) OPT$$

b) Que la cota sea óptima quiere decir que el ALG no consigue un radio competitivo menor a $\left(2 - \frac{1}{m}\right)$. Esto lo podemos hacer mostrando que hay un input para el cual ALG alcanza exactamente $\left(2 - \frac{1}{m}\right) OPT$.

Consideremos $m(m-1)$ tareas de tiempo 1 y luego una tarea de tiempo m. En este caso ALG distribuirá equitativamente las primeras tareas dando carga $(m-1)$ a cada procesador y luego la última tarea se la dará a alguno de los procesadores quien quedará con carga $2m-1$ obteniendo este costo. Sin embargo, el OPT distribuye de manera equitativa estas tareas dándole carga m a cada uno de los procesadores.

De esta forma $ALG = 2m - 1 = \left(2 - \frac{1}{m}\right) m = \left(2 - \frac{1}{m}\right) OPT$.

P2. Partition

Para ahorrar notación llamemos $w(A) = \sum_{i \in A} s_i$, $w(B) = \sum_{i \in B} s_i$ a los pesos de los conjuntos, $2W = w(A) + w(B) = \sum_{i=1}^{n} s_i$, al peso total (notemos que $|OPT| \geq W$, pues en el mejor caso ambos conjuntos quedan perfectamente balanceados) y $\left\lceil \frac{1}{\epsilon} \right\rceil - 1 = m < n$ ($\epsilon \approx \frac{1}{m+1}$). Y hacemos lo siguiente:

- Ordenar el input en orden decreciente.
- **Primera fase:** Encontrar partición óptima A', B' de los primeros m elementos por fuerza bruta ($\mathcal{O}(2^m) = \mathcal{O}(2^{1/\epsilon})$).
- **Segunda fase:** Ir agregando el resto de los elementos al conjunto que tenga menor peso en ese momento.

Para mostrar que esto es una $1+\epsilon$ aproximación supongamos que al final el algoritmo termina con $w(A) \geq w(B)$ (por lo que $|ALG| = w(A)$), así calculamos $\frac{|ALG|}{|OPT|} \leq \frac{w(A)}{W}$. Para seguir con el cálculo consideremos el momento en el que se agrega a A el último número s_k:

- Si este fue agregado en la primera fase, entonces el resultado del algoritmo es una solución óptima (pues A termina con los mismos elementos de una solución óptima del problema más pequeño).
- Por otro lado, si s_k fue agregado en la segunda fase, por el funcionamiento del algoritmo se debe cumplir que $w(A) - s_k \leq w(B) \Rightarrow w(A) \leq W + \frac{s_k}{2}$, y por lo tanto $\frac{|ALG|}{|OPT|} \leq 1 + \frac{s_k}{2W}$. Por otro lado, como todos los elementos anteriores a s_k son mayores a él, se cumple que $2W \geq (m+1)s_k$, y entonces $\frac{|ALG|}{|OPT|} \leq 1 + \frac{s_k}{(m+1)s_k} = 1 + \epsilon$.

P3. Corte Mínimo de un Grafo

a) Supongamos que los nodos en V' (final) son v' y u' y sean $v, u \in E$, nodos que pasaron a ser parte de v', u' en el algoritmo respectivamente. E' (final) es un corte del grafo pues en $(V, E \setminus E')$ no existe un camino de $v \to u$ o dicho de otro modo, todo camino de $v \to u$ pasa por E' (lo cual es cierto pues v' y u' corresponde a una "partición" de V, y E' corresponde a las aristas entre ambas partes, por lo que todo camino que vaya de una parte a otra debe tomar una de estas aristas).

b) Supongamos que un *min-cut* es C, de k aristas. Observemos lo siguiente:

1. Para que el algoritmo responda C se debe cumplir que en cada iteración la arista escogida no esté en C.
2. El grado de los nodos del grafo debe ser al menos k. Y por lo tanto el grafo tiene al menos $nk/2$ aristas.

Definamos entonces E_i como el evento que en la i-ésima iteración del algoritmo escojamos una arista que no está en C. La probabilidad que nos interesa calcular entonces es $\mathbb{P}(E_1 \cap E_2 \cap \ldots E_{n-2}) = \mathbb{P}(E_1)\mathbb{P}(E_2|E_1)\ldots\mathbb{P}(E_{n-2}|E_1 \cap E_2 \cap \ldots E_{n-3})$.

Veamos que en general $\mathbb{P}(E_i|E_1 \cap E_2 \cap \ldots E_{i-1})$ corresponde a la probabilidad que no se escoja una arista de C en el paso i dado que en ninguno de los pasos anteriores esto ha sucedido. Dado que no se ha escogido ninguna arista de C para contraer tenemos un grafo de $n - i + 1$ nodos cuyo *min-cut* es de tamaño $\geq k$ (si hubiese un corte más chico C no sería mínimo) y por la observación 2. el grafo debe tener al menos $(n - i + 1)k/2$ aristas

así la probabilidad de escoger una arista de C es $\leq 2/(n-i+1)$ y por lo tanto la de no hacerlo $\geq 1-2/(n-i+1) = \frac{n-i-1}{n-i+1}$. Luego, la probabilidad que el algoritmo encuentre C es mayor o igual a :

$$\prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \frac{2}{n(n-1)}$$

c) Si repetimos el algoritmo $n(n-1)\ln n$ veces de manera independiente y retornamos el menor de los cortes encontrados como *min-cut*. En este caso la probabilidad que ninguno de las ejecuciones encuentre un *min-cut* es menor o igual a:

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\ln n} \leq e^{-2\ln n} = \frac{1}{n^2}$$