Ejercicios Tema 3

Algoritmia para problemas difíciles

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura – Universidad de Zaragoza

27 de noviembre de 2017

Ejercicios sobre algoritmos aproximados

Ejercicio 1 Considerar la siguiente heurística para VC (Cobertura de vértices óptima). Construir un árbol recorrido en profundidad del grafo y borrar todas las hojas de este árbol. Demostrar que el tamaño de este cubrimiento es como mucho dos veces mayor que el óptimo.

Ejercicio 2 Se debe realizar la mudanza de una biblioteca, para la cual se ha contratado una empresa de mudanzas que cobra por el número total de cajas a transportar. Se dispone de k_1 cajas de un metro de alto, k_2 cajas de 50 centímetros de alto y k_3 cajas de 25 centímetros de alto. Tenemos n libros de distintos grosores g_i que se colocarán horizontalmente en las cajas.

Queremos minimizar el coste de la mudanza. Encontrar una aproximación voraz y justificar el ratio de aproximación (ayuda, argumentar no puede haber dos cajas llenas a menos de la mitad).

Ejercicio 3 Supongamos que te dan un conjunto de enteros positivos $A = \{a_1, \ldots, a_n\}$ y un entero positivo B. Un subconjunto $S \subseteq A$ se llama *factible* si la suma de los números de S no sobrepasa B:

$$\sum_{a_i \in S} a_i \leq B.$$

La suma de los números de S se llamará la *suma total* de S.
Te gustaría seleccionar un subconjunto factible S de A cuya suma total sea lo mayor posible.
Ejemplo. Si $A = \{8, 2, 4\}$ y $B = 11$, entonces la solución óptima es $S = \{8, 2\}$.

1. Este es un algoritmo para este problema.

   ```
   S = ∅
   T = 0
   for i = 1, 2, ..., n
         do if T + a_i ≤ B
               Añadir a_i a S
               T = T + a_i
   Resultado S
   ```

   Dar una entrada para la que la suma total del conjunto S devuelto por este algoritmo es menos de la mitad de la suma total de cualquier otro posible subconjunto factible de A.

2. Dar un algoritmo de aproximación para este problema basado en el visto en clase para Mochila.

3. Dar un algoritmo de aproximación en tiempo polinómico para este problema con la siguiente garantía: Devuelve un conjunto factible $S \subseteq A$ cuya suma total es de al menos de la mitad del tamaño de la máxima suma total de cualquier conjunto factible $S' \subseteq A$. Tu algoritmo debe tener un tiempo de ejecución como mucho $O(n \log n)$.

Ejercicio 4 En el problema MAXSAT nos dan un circuito booleano en CNF C con una única salida, y queremos encontrar una asignación que satisface el mayor número posible de cláusulas de C.

1. Demostrar que si este problema se puede resolver en tiempo polinómico, también se puede resolver en tiempo polinómico SAT.
2. Este es un algoritmo muy inocente,

   1 **for** cada variable
   2 **do** asigna su valor a cierto o falso lanzando una moneda

   Supón que tu entrada tiene m cláusulas, de las cuales la número j tiene k_j literales. Demostrar que el número *medio* de cláusulas satisfechas por este algoritmo es

   $$\sum_{j=1}^{m}\left(1-\frac{1}{2^{k_j}}\right) \geq \frac{m}{2}.$$

   En otras palabras, es una 2-aproximación en media. Y si las cláusulas contienen todas k literales, entonces el ratio de aproximación mejora a $1 + 1/(2^k - 1)$.
3. ¿Puedes hacer este algoritmo determinista? (*Pista:* En lugar de lanzar una moneda para cada variable, selecciona el valor que satisface más cláusulas de las todavía no satisfechas. ¿Qué fracción de las cláusulas se satisface al final?)

Notación: Un *literal* es una variable booleana afirmada o negada (p. ej. x, $\neg y$, etc). Una *cláusula* es una disyunción de literales (p. ej. $(x \vee \neg y)$). Un *circuito en CNF* es una conjunción de cláusulas (p. ej. $(x \vee \neg y \vee z) \wedge (\neg z \vee y) \wedge (\neg x)$). Una asignación de las variables satisface una cláusula cuando la hace cierta, una asignación de las variables satisface un circuito cuando hace ciertas todas sus cláusulas.

Ejercicio 5 Recordemos que en el problema de la mochila tenemos n objetos, cada uno con un peso w_i y un valor v_i. También tenemos un límite de peso W, y el problema es seleccionar un conjunto de elementos S del mayor valor posible sujeto a la condición de que el peso total no exceda W – es decir, $\sum_{i\in S} w_i \leq W$. Si se nos dice que existe un subconjunto O cuyo peso total es $\sum_{i\in O} w_i \leq W$ y cuyo valor total es $\sum_{i\in O} v_i = V$ para algún V, entonces el algoritmo aproximado visto en clase puede encontrar un conjunto A con peso total $\sum_{i\in A} w_i \leq W$ y valor total al menos $\sum_{i\in A} v_i \geq V/(1+\epsilon)$. Así, el algoritmo aproxima el mejor valor, manteniendo los pesos estrictamente por debajo de W. (Por supuesto, devolver el conjunto O siempre es una solución válida, pero ya que el problema es NP-difícil, no esperamos poder encontrar siempre O; el ratio de aproximación de $1+\epsilon$ significa que los otros conjuntos A, con un poco menos de valor, pueden ser respuestas válidas también.)
Ahora, como es bien sabido, siempre se puede meter un poco más simplemente "sentándose encima de la maleta", es decir, sobrepasando un poco el límite de peso permitido. Esto sugiere otra manera diferente de formalizar la aproximación para el problema de la mochila, que es la siguiente:
Supongamos , como antes, que nos dan n objetos con pesos y valores, además de los parámetros W y V; y te dicen que hay un subconjunto O cuyo peso total es $\sum_{i\in O} w_i \leq W$ y cuyo valor total es $\sum_{i\in O} v_i = V$. Para un determinado $\epsilon > 0$ fijo, diseña un algoritmo en tiempo polinómico que encuentre un subconjunto de objetos A tal que $\sum_{i\in A} w_i \leq (1+\epsilon)W$ y $\sum_{i\in A} v_i \geq V$. En otras palabras, quieres que A alcance un valor de al menos la cota dada V, pero se le permite exceder el límite de peso W por un factor de $1+\epsilon$.
Ejemplo. Supongamos que te dan cuatro objetos, con pesos y valores como sigue:

$$\begin{array}{ll}(w_1, v_1) = (5, 3), & (w_2, v_2) = (4, 6)\\ (w_3, v_3) = (1, 4), & (w_4, v_4) = (6, 11)\end{array}$$

También te dan $W = 10$ y $V = 13$ (ya que, de hecho, el subconjunto formado por los tres primeros objetos tiene peso total como mucho de 10 y tiene un valor de 13). Por último, te dan $\epsilon = 1$. Esto quiere decir que

debes encontrar (con tu algoritmo aproximado) un subconjunto de peso como mucho $(1 + ,1) * 10 = 11$ y valor de al menos 13. Una solución válida sería el subconjunto formado por el primer y cuarto objetos, con valor $14 \geq 13$. (Notar que este es un caso en que puedes alcanzar un valor estrictamente mayor que V ya que se permite sobrecargar un poco la mochila.)

Ejercicio 6 Encontrar una implementación del algoritmo APROXSC visto en clase (página 45) con coste en tiempo $O(\sum_{i=1}^{m} |S_i|)$.

Ejercicio 7 El problema de Equilibrado de Carga consiste en asignar procesos a procesadores intentando minimizar el tiempo máximo utilizado por cada procesador. Es decir, tenemos n procesos y m procesadores, el proceso i tarda tiempo t_i y no puede ser dividido entre varios procesadores.

Vamos a considerar una variante más general, a la que llamaremos *Carga Hetereogénea* en la que no todos los procesadores tienen la misma velocidad. El procesador j tiene velocidad v_j, con $1 \leq v_j \leq 3$. El proceso i tarda tiempo t_i/v_j en ejecutarse en el procesador j.

Dar un algoritmo voraz en tiempo polinómico que sea una aproximación para dicha variante *Carga Heterogénea*. Justificar el ratio de aproximación de dicho algoritmo.

Ejercicio 8 En el problema básico de *Equilibrado de Carga* estamos interesados en asignar trabajos a máquinas para minimizar la carga máxima de una máquina (tiempo máximo o "makespan"). En muchas aplicaciones, es natural considerar casos en los que se tiene acceso a máquinas con diferente poder de cálculo, por lo que un trabajo dado puede realizarse más rápido en una máquina que en otra. La pregunta es entonces ¿cómo se deben asignar los trabajos a las máquinas en estos sistemas heterogéneos?

Aquí está un modelo básico para este problema, *Equilibrado de Carga Heterogéneo*. Supongamos que disponemos de m máquinas *lentas* y k máquinas *rápidas*. Las máquinas rápidas pueden realizar el doble de trabajo por unidad de tiempo que las máquinas lentas. Dado un conjunto de n trabajos: el trabajo i tarda tiempo t_i en una máquina lenta y tiempo $t_i/2$ en una máquina rápida. Se trata de asignar cada trabajo a una máquina, el objetivo es minimizar el "makespan" o tiempo máximo sobre todas las máquinas del tiempo de trabajo.

Dar un algoritmo en tiempo polinómico que sea una 3-aproximación para *Equilibrado de Carga Heterogéneo*.

En caso de entregar alguno de estos ejercicios, la fecha límite es el domingo 10de diciembre.

Antes de realizar cualquiera de estos ejercicios el alumno debe enviar un correo a `elvira@unizar.es` indicando qué ejercicio desea realizar.

Cualquier fuente utilizada en la resolución de estos ejercicios debe ser indicada claramente en la solución.