Ejercicios Tema 3

Algoritmia para problemas difíciles

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

25 de noviembre de 2022

Ejercicios sobre algoritmos aproximados

Ejercicio 1 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 2 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 3 Supón que actúas como consultor para la autoridad portuaria de un pequeño país del Pacífico. En la actualidad su facturación es de varios miles de millones de euros por año, y sus beneficios están limitados casi enteramente por la velocidad a la que pueden descargar los barcos que llegan a puerto.
Aquí está el problema básico que afrontan. Llega un barco, con n contenedores de peso $w_1, w_2, \ldots, w_n$. En el puerto hay esperando un conjunto de camiones, cada uno de los cuales puede transportar K unidades de peso. (Puedes suponer que K y cada w_i son enteros.) Puedes cargar varios contenedores en cada camión, sujeto a la restricción de peso K; el objetivo es minimizar el número de camiones necesarios para transportar todos los contenedores. Este problema es NP-difícil (no tienes que demostrarlo).
Un algoritmo voraz que puedes usar para este problema es el siguiente: Empieza con un camión vacío y empieza a apilar contenedores $1, 2, 3, \ldots$ en él hasta que llegues a un contenedor que haga sobrepasar el límite de peso. Ahora declara este camión "cargado" y envíalo; entonces continúa el proceso con un nuevo camión. Este algoritmo, considerando los camiones de uno en uno, puede no alcanzar la forma más eficiente de cargar todos los contenedores en la colección de camiones disponible.

1. Da un ejemplo de un conjunto de pesos y un valor de K para los que este algoritmo no usa el mínimo número posible de camiones.
2. Demuestra que sin embargo el algoritmo es una 2-aproximación del óptimo.

Ejercicio 4 Sea VC el problema de, dado un grafo, encontrar el conjunto más pequeño de vértices que cubra todas las aristas del grafo (para cada arista, al menos uno de los dos extremos pertenece al conjunto). Diseñar un algoritmo que aproxime la solución del problema utilizando la estrategia voraz de seleccionar primero el vértice de mayor grado. Demostrar que el algoritmo diseñado tiene un ratio de aproximación acotado por $\log n$. Encontrar un caso en que se alcance dicha cota de $\log n$.

Ejercicio 5 Encontrar una implementación del algoritmo APROXSC visto en clase (página 46) con coste en tiempo $O(\sum_{i=1}^{m} |S_i|)$.

Ejercicio 6 Dado el algoritmo de aproximación para *Set Cover* visto en clase, encontrar una entrada para la que se alcance el ratio de aproximación $\log n$.

Ejercicio 7 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 una cota constante para el ratio de aproximación (ayuda, argumentar no puede haber dos cajas llenas a menos de la mitad).

Ejercicio 8 MOCHILA INVERSA Tenemos n objetos, el objeto $i$ésimo tiene un peso p_i y un beneficio b_i. Esta vez el problema trata de llenar la mochila con objetos con beneficio total al menos un valor B dado (todos los pesos, los beneficios y B son reales positivos). El objetivo es **minimizar** el peso de un conjunto de objetos que tengan beneficio al menos B.

1. Diseñar un algoritmo eficiente que aproxime la solución óptima a este problema.
2. Justificar qué cota de tiempo cumple el algoritmo.
3. Justificar qué ratio de aproximación cumple dicho algoritmo.

Ambas cotas deben ser razonables (idealmente tiempo polinómico y aproximación constante o incluso mejor) pero no es necesario que sean ajustadas. **PISTAS:** (i) Se trata de usar programación dinámica asumiendo primero que los pesos son enteros, cada posición de la matriz depende únicamente de hasta 2 posiciones de la fila anterior. (ii) Se trata de cambiar de escala algunos datos, hay que pensar qué datos cambiar de escala y si son más convenientes los redondeos por exceso o por defecto.

Ejercicio 9 BINPACKING. Supongamos que se nos da un conjunto de n objetos, donde el tamaño s_i del objeto número i satisface $0 < s_i < 1$. Queremos meter todos los objetos en el número mínimo de cajas tamaño uno. Cada caja puede contener cualquier subconjunto de los objetos cuyo tamaño total no sobrepase 1. La heurística del *primer ajuste* toma cada vez un objeto y lo coloca en la primera caja en la que aun cabe. Sea $S = \sum_{i=1}^{n} s_i$.

1. Argumentar que el número óptimo de cajas necesarias es al menos $\lceil S \rceil$.
2. Argumentar que la heurística del primer ajuste deja a lo sumo una caja llena a menos de la mitad.
3. Demostrar que el número de cajas utilizadas por la heurística de primer ajuste nunca es más que $\lceil 2S \rceil$.
4. Demuestra que la heurística del primer ajuste es una 2-aproximación.
5. Dar una implementación eficiente de la heurística primer ajuste, y analizar su complejidad en tiempo.

Ejercicio 10 El problema de *Equilibrado de Carga Hetereogénea* consiste en asignar procesos a procesadores intentando minimizar el "makespan" o 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, el procesador j tiene velocidad $v_j \geq 1$. El proceso i tarda tiempo t_i/v_j en ejecutarse en el procesador j.
Sea

$$T = \frac{\sum_{i=1}^{n} t_i}{\sum_{j=1}^{m} v_j}.$$

Es sencillo razonar que el makespan óptimo es mayor o igual que T. Supongamos que cada proceso no es demasiado grande, en concreto podemos asumir que para cada i, $t_i \leq T/2$.

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

Ejercicio 11 Los responsables de PLAZA quieren aumentar la velocidad de descarga de los aviones que llegan al aeropuerto de Zaragoza, pero también quieren limitar gastos.

El problema concreto es el siguiente. Llega un avión, con n bultos de peso $p_1, p_2, \ldots, p_n$ (los p_i son reales positivos). Los bultos no se pueden trocear. En el aeropuerto hay esperando un conjunto de furgonetas, cada una de los cuales puede transportar un peso máximo $M > 0$. Se pueden cargar varios bultos en cada furgoneta, sujeto a la restricción de peso M (asumimos que para todo i, $p_i \leq M$); el objetivo es minimizar el número de furgonetas necesarias para transportar todos los bultos.

Explora al menos dos algoritmos voraces para ese problema, uno que empieza con una furgoneta vacía y empiece a apilar bultos en ella hasta que llegues a un bulto que haga sobrepasar el límite de peso, en ese momento envías la furgoneta y continúas el proceso con una nueva furgoneta vacía. El segundo algoritmo también es voraz pero mantiene todas las furgonetas que está cargando en la pista rellenándolas todo lo posible hasta terminar de descargar el avión.

1. Detalla ambos algoritmos y justifica que son eficientes.
2. Para cada uno de los dos algoritmos, justifica que el ratio de aproximación es menor o igual que una constante.
3. Da ejemplos en los que un algoritmo funcione mejor que el otro.

Ejercicio 12 Sea d un entero positivo. Encontrar un algoritmo con ratio de aproximación menor o igual que $1 + d$ que resuelva el siguiente problema:
Dado un grafo (no dirigido, sin autoaristas) $G = (V, A)$ que cumpla que para cualquier vértice u el grado o número de vecinos de u es menor o igual que d, encontrar el conjunto independiente U de G tal que el número de elementos de U sea el mayor posible.

Ejercicio 13 Recordemos que en el problema de la mochila tenemos n objetos, cada uno con un peso p_i y un valor b_i. También tenemos un límite de peso C, 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 C – es decir, $\sum_{i \in S} p_i \leq C$. Si el óptimo es un conjunto O con $\sum_{i \in O} p_i \leq C$ y cuyo valor total es $\sum_{i \in O} b_i = B$, en clase hemos visto un algoritmo que, dado ϵ, encuentra de forma eficiente un conjunto A con peso total $\sum_{i \in A} p_i \leq C$ y valor total al menos $\sum_{i \in A} b_i \geq B/(1 + \epsilon)$.
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 C y B; y te dicen que hay un subconjunto O cuyo peso total es $\sum_{i \in O} p_i \leq C$ y cuyo valor total es $\sum_{i \in O} b_i = B$. Para un determinado $\epsilon > 0$ fijo, diseña un algoritmo en tiempo polinómico que encuentre un subconjunto de objetos A tal que $\sum_{i \in A} p_i \leq (1 + \epsilon)C$ y $\sum_{i \in A} b_i \geq B$. En otras palabras, quieres que A alcance un valor de al menos la cota dada B, pero se le permite exceder el límite de peso C por un factor de $1 + \epsilon$.
Ejemplo. Supongamos que te dan cuatro objetos, con pesos y valores como sigue:

$$(p_1, b_1) = (5, 3), \quad (p_2, b_2) = (4, 6)$$
$$(p_3, b_3) = (1, 4), \quad (p_4, b_4) = (6, 11)$$

También te dan $C = 10$ y $B = 13$ (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 $\epsilon = 0,1$. Esto

quiere decir que debes encontrar (con tu algoritmo aproximado) un subconjunto de peso como mucho $(1 + 0,1) * 10 = 11$ 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 B ya que se permite sobrecargar un poco la mochila.)

Ejercicio 14 Diseñar un algoritmo para TSP métrico basado en una variante del visto en clase pero que utilice la estrategia del perfect matching para encontrar un recorrido euleriano. Demostrar que el ratio de aproximación es como mucho 1,5.

En caso de entregar alguno de estos ejercicios, la fecha límite es el martes 6 de diciembre.

Antes de realizar cualquiera de estos ejercicios el alumno debe seleccionarlo en moodle.

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

A partir del lunes 28 de noviembre se puede elegir un segundo ejercicio de entre los todavía disponibles enviando un correo a la profesora.