

Examen final de Algoritmia

17 de junio de 2010

Duración: 2:30 horas

Entregad las respuestas a los enunciados en tres bloques separados:

Bloque A: enunciado 1, bloque B: enunciado 2 y 3, bloque C: enunciados 4 y 5.

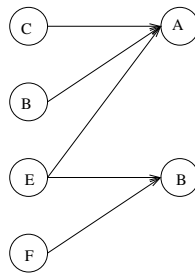
Las notas de este examen y las del curso se publicarán antes del día 2 de julio de 2010.

La revisión será el día 2 de julio de las 10 a las 11 horas en el despacho 235 del edificio Omega.

Podéis optar por entregar el ejercicio 4 o no. En el caso que no se entregue se sustituirá la puntuación del ejercicio 4 de este examen por la parte proporcional de la nota obtenida en la primera entrega obligatoria de problemas.

Para cada algoritmo debéis justificar por una parte su corrección y por otra proporcionar un análisis de su coste.

Exercici 1 (2 punts) Considereu la xarxa d'enllaços de la figura següent:



- (a) Doneu els valors de Hub i Auth, quan executeu HITS (amb $k = 2$) a la xarxa de la figura. Mostreu els valors abans i després de la etapa final de normalització
- (b) Supposeu que voleu crear una nova pàgina web de X, i afegir-la a la xarxa, per tal que pugui aconseguir una puntuació (normalitzada) de Auth tan gran com sigui possible. Una cosa que pots provar es crear una segona pàgina Y, de manera que existeixi una aresta de Y cap a X. En fer això, es natural que us pregunteu si ajuda o perjudica a l'Auth de X el fet de deixar que Y també tingui enllaços cap a altres nodes. En concret, suposem que afegim X i Y a la xarxa. Per fer això necessitem especificar quins enllaços tindran. Tenim dues opcions, a la primera opció, només hi ha una aresta de Y cap a X, mentre que a la segona opció, hi han enllaços de Y cap a d'altres autoritats més de fortes que X (amb m es valor Auth).

Opció 1: Afegir nous nodes X i Y a la xarxa, crear una aresta de Y a X

Opció 2: Afegir nous nodes X i Y i crear enllaços de Y cap a A, B, i X.

Per a cada una d'aquestes dues opcions, podeu dir com evoluciona Auth d'X? (Es a dir, doneu els valors normalitzats d'Auth per A, B, i X (amb $r = 2$). Per a quina de les dues opcions X obtindrà un valor més alt d'Auth? (tenint en compte la normalització)?

Enunciado 2 (2 punts) El *problema de la evacuación* tiene la siguiente definición. Nos dan un grafo dirigido $G = (V, E)$ que representa una red de carreteras. Una cierta colección de nodos $X \subseteq V$ se designan como *nodos habitados*. Otra colección de nodos S son designados como *nodos seguros*. Suponemos que S y X son disjuntos. En el caso de una emergencia, queremos rutas de evacuación desde los nodos habitados a los seguros. Un conjunto de *rutas de evacuación* se define como un conjunto de caminos en G tales que: (i) cada nodo en X aparece como el inicio de un camino, (ii) el último nodo en un camino aparece en S , (iii) los caminos no comparten aristas.

Cada ruta de evacuación proporciona una ruta a través de la cual los habitantes de un nodo poblado pueden escapar a un nodo seguro sin compartir ninguna parte del camino con habitantes de otros nodos.

Un conjunto de *rutas de evacuación mixtas* se define como un conjunto de caminos en el aparece como el inicio de un camino, (ii) el último nodo en un camino aparece en S .

Este tipo de rutas proporciona rutas de escape en las que parte del camino es compartido entre habitantes de distintas ciudades.

Supongamos que además cada arco $e \in E$ tiene asignada una capacidad $c(e)$.

- Dados $G = (V, E, c)$ y $S, X \subseteq V$, muestra como decidir en tiempo polinómico si un conjunto de rutas de evacuación existe o no. En caso de que si que existan,

el algoritmo debe proporcionar, para cada nodo poblado, la capacidad del arco de capacidad mínima de la ruta que se inicia en él.

- Dados $G = (V, E, c)$, $S, X \subseteq V$, y una petición de tráfico $r(x)$, para $x \in X$, muestra como decidir en tiempo polinómico si existe un conjunto de rutas mixtas de evacuación que permitan enviar el tráfico solicitado $r(x)$ desde X a un nodo seguro, sin superar en ningún momento la capacidad de un arco en la red. En caso de que sea posible el algoritmo tiene que devolver además, para cada nodo $x \in X$, un plan de evacuación indicando: (i) una colección de caminos con origen en x y final en un nodo seguro, (ii) el tráfico asociado a cada uno de estos caminos. El total de tráfico asignado entre todos los caminos del plan de escape tiene que cubrir la totalidad de la petición de tráfico $r(x)$.

Enunciado 3 (2.5 puntos) Consideramos el siguiente escenario: tenemos un conjunto N de n ciudades con distancias entre ellas. Queremos seleccionar un subconjunto C de k ciudades en las que queremos ubicar un centro comercial. Asumiendo que las personas que viven en una ciudad compraran en el centro comercial más próximo se quiere buscar una ubicación C de manera que todas las ciudades tengan un centro comercial a distancia menor que r en la que intentamos minimizar r sin perder cobertura. Para ello diremos que C es un r -recubrimiento si todas las ciudades están a distancia como mucho r de una ciudad en C . Sea $r(C)$ el mínimo r para el que C es un r -cover. Nuestro objetivo es encontrar C con k vértices para el que $r(C)$ es mínima.

1. Demuestra que si $k \geq n$, la solución formada por todas las ciudades es óptima. A partir de ahora asumiremos que $k < n$.
2. Diseña un algoritmo que dado C calcule $r(C)$. Analiza su coste.
3. Considera el siguiente algoritmo:

Para cada ciudad $c \in N$ definimos $D(c)$ como el máximo de la distancia de c al resto de ciudades y $d(c)$ como el mínimo de estas distancias.

Una ciudad c es central si $D(c)$ es mínimo.

$|C| = \emptyset$

- **mientras** $|C| \neq k$ y $N \neq \emptyset$

 Obtener una ciudad central c .

$C \leftarrow C \cup \{c\}$

 Eliminar de N c y todas las ciudades a distancia menor que $\frac{D(C)-d(c)}{2}$ de c .

Es un algoritmo de aproximación con factor constante?

Enunciado 4 (2 puntos) El *problema de asignación generalizado* (en inglés, generalized assignment problem, con siglas GAP) se puede describir como se detalla a continuación:

Sean dados n artículos y m contenedores. Cada contenedor j tiene capacidad $c_j \geq 0$. Adicionalmente, para cada contenedor j , el artículo i tiene asociados un beneficio $p_{ij} \geq 0$ y un peso $w_{ij} \geq 0$. Una *solución* es un conjunto $U \subseteq \{1, \dots, n\}$ y una asignación de los artículos de U a los contenedores. Una *solución factible* es una solución en la cual, para cada contenedor j , la suma de los pesos de los artículos asignados es como máximo c_j . El *valor de la función objetivo* de una solución factible es la suma de los beneficios para cada asignación artículo-contenedor. El objetivo es encontrar una solución factible que maximice esta función objetivo.

GAP es NP -difícil, y es incluso APX -difícil de aproximar. Matemáticamente, el problema de asignación generalizado se puede formular de la siguiente manera:

$$\text{maximizar } \sum_{i=1}^n \sum_{j=1}^m p_{ij} x_{ij} \quad (1)$$

sujeito a:

$$\begin{aligned} \sum_{i=1}^n w_{ij} x_{ij} &\leq c_j & j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} &\leq 1 & i = 1, \dots, n \\ x_{ij} &\in \{0, 1\} & i = 1, \dots, n, \quad j = 1, \dots, m \end{aligned}$$

Dada su intratabilidad, nos gustaría resolver el problema GAP con una metaheurística, un algoritmo voraz iterado (en inglés *iterated greedy*, con siglas IG). Recuerda, que las componentes básicas de un IG son (1) una heurística greedy para la construcción de soluciones, (2) un método para la destrucción parcial de la solución, y (3) un criterio de aceptación.

Cuestiones:

1. **(1 punto)** Desarrolla un algoritmo voraz para el problema GAP y proporciona el pseudocódigo del algoritmo y una explicación en palabras.

NOTA: Ten en cuenta que, para obtener la puntuación máxima, el algoritmo tiene que *tener sentido*.

2. **(1 punto)** Desarrolla un método para la destrucción parcial de una solución. Explica además un posible método de aceptación.

NOTA: Ten en cuenta que el método de destrucción tiene que ser compatible con el algoritmo voraz que has desarrollado. Eso implica que, después de la destrucción parcial de una solución s y su reconstrucción mediante el algoritmo voraz desarrollado, la solución s' resultante debería ser, en general, diferente a s .

Enunciado 5 (1.5 puntos) Los algoritmos evolutivos se basan en la evolución natural. Dichos algoritmos trabajan a base de operadores como el cruce de individuos (en inglés: *crossover*) y la mutación.

1. **(1 punto)** Desarrolla un operador de cruce para el problema GAP especificado en el enunciado 1. El operador debería tomar dos soluciones como entrada y proporcionar una sola solución como salida.

NOTA: Ten en cuenta que, para obtener la puntuación máxima, el operador tiene que tener sentido (por ejemplo, un operador inteligente que preserve las partes que los padres tienen en común).

2. **(0.5 puntos)** Algunos operadores de cruce producen potencialmente soluciones no factibles. Menciona brevemente las tres medidas más comunes para tratar con soluciones no factibles.