Práctica 3. Algoritmos voraces - Salas de conferencias

Noelia Escalera Mejías Alejandro Menor Molinero
Javier Núñez Suárez Adra Sánchez Ruiz
Jesús Torres Sánchez

7 de abril de 2019

1. Descripción del problema

El problema a resolver es el siguiente:

Un centro educativo va a realizar un ciclo de n conferencias durante un día. Cada conferencia tiene establecido su horario, la conferencia i empieza a la hora s i y termina a la hora f i . Se desea que esta actividad interfiera lo mínimo posible con las actividades normales del centro.
Por tanto, nuestro algoritmo debe ser capaz de planificar todas las conferencias en su horario establecido usando el menor número de aulas posibles, partiendo de que dos conferencias no se pueden planificar en el mismo aula si sus horarios se solapan.

2. Planteamiento del algoritmo voraz

En primer lugar debemos crear un algoritmo Greedy que pueda solucionar el problema. Tras un análisis e interpretación del enunciado, obtenemos esta primera aproximación:

- Partimos de una cola/vector/lista de conferencias, y un vector de soluciones vacío.
- Ordenamos la lista de conferencias de acuerdo a la hora-in.
- Mientras queden elementos en la lista de conferencias candidatas:
  - Comparamos la hora-in de la primera conferencia de las candidatas con la hora-fin de la última conferencia de la lista de soluciones asociada a la componente del vector actual.
  - Si la actividad no se solapa, se incluye en la lista de soluciones y se extrae de la lista de candidatos.

- Cuando hayamos recorrido los candidatos, si aún quedan, incrementamos la componente del vector (nueva aula) y repetimos el proceso para las conferencias restantes.

3. Identificación de elementos

- *Conjunto de Candidatos (C)*: el conjunto de candidatos está formado por todo el conjunto de conferencias a planificar; $Ci(i = 0..n-1)$
- *Conjunto de Seleccionados (S)*: inicialmente vacío, contendrá las todas las conferencias Ci agrupadas en las distintas salas Sj
- *Función Solución*: vector en el que cada componente se corresponde con un aula distinta, cada una de ellas con una lista de conferencias asociadas.
- *Función de Factibilidad*: no se pueden seleccionar actividades que se solapen en el aula actual.
- *Función de Selección*: en este caso, hemos decidido utilizar como criterio de selección la conferencia de **menor hora de inicio**. Posteriormente, estudiaremos su optimalidad.
- *Función Objetivo*: planificar todas las conferencias en el menor número de aulas.

4. Demostración (falta pulirla)

En esta demostración debemos tener presente que la solución óptima tiene que abrir tantas aulas como conferencias se solapen a la vez (preguntar si esto hay que demostrarlo también por favor).

- Sea c_1 una conferencia
- Sea c_2 la conferencia inmediatamente posterior y compatible con c_1
- Con nuesro algoritmo $aula(c_1) = aula(c_2)$. Llamemos A a la solución que logra esto.
- Supongamos que en la solución óptima $aula(c_1) \neq aula(c_2)$
- Entonces se podría dar $n_{aulas} = n_{conferencias\ solapadas\ a\ la\ vez} + 1$. Esto es absurdo ya que contradecimos la condición para que sea solución óptima.
- Luego la solución A es la óptima.

5. Demostración (versión 2)

Según nuestro criterio de selección, el algoritmo va a ir comprobando y seleccionando aquellas actividades que tienen un menor tiempo de inicio. Para comprobar su optimalidad, podemos llevar a cabo una reducción al absurdo:

Supongamos que nuestra solución no es óptima; esto implica que, si nuestro algoritmo voraz ha determinado que la solución es n aulas, hay una solución minimal que puede planificar las conferencias en k aulas, de modo que $k < n$.

Como sabemos, en la primera iteración, el máximo número de conferencias solapadas es k (máximo número de aulas necesarias). Por la naturaleza de nuestro algoritmo, al seleccionar primero la conferencia que menos tarda, tenemos que en cada iteración el número de solapamientos que tiene cada conferencia se reduce en una unidad (hasta ser seleccionada). De este modo, al llegar a la iteración k, como inicialmente el máximo número de conferencias solapadas era k, el número de solapamientos restantes equivale a 0.

Al no quedar solapamientos, nuestro algoritmo puede planificar todas las actividades en un mismo aula. De este modo, el número de aulas obtenidas con nuestro algoritmo es k, que es el número óptimo de aulas, por lo que llegamos a un absurdo.