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. 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 aulas necesarias para planificar todas las conferencias. Cada aula Sj contiene todas las conferencias que se darán en esta aula.
- *Función Solución*: El conjunto de Candidatos C está vacío.
- *Función de Factibilidad*: Es posible añadir la conferencia al aula actual sin que se solape con ninguna otra ya añadida.
- *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 usando el número de aulas mínimo.

3. Descripción del algoritmo

Proponemos la siguiente solución:

```
Conjunto de Conferencias C; // Input
Conjunto de aulas S; //Inicialmente empty
C.sort('tiempo de inicio ascendente')

aulaActual; //Empty

While C not empty{
        conf = primer elemento de C compatible con a

        // Si no podemos planificar ninguna conferencia en el
        // aula actual, abrimos otra nueva
        if (conf no existe){
                S.add(aulaActual)
                aulaActual = vacia;
        }
        else{
                aulaActual.add(conf);
                C.borrar(conf)
        }
}
```

4. Demostración del algoritmo

Antes de demostrar que nuestro algoritmo lleva a cabo una planificación correcta, tenemos que definir primero que es:

Una planificación correcta para este problema no utiliza más aulas que el máximo de conferencias que se solapan en el tiempo.

Introducimos ahora la forma de planificar incorrectamente:

Sea $[x, y]$ una conferencia (x el instante de inicio e y el instante de fin) y $[x + c, k]$ otra conferencia ($c > 0$). Ambas son planificables en la supuesta iteración actual de nuestro algoritmo i, que está planificando conferencias en un aula i, pero incompatibles entre sí ($y > x + c$). Suponemos que elegimos planificar $[x+c, k]$ en vez de $[x, y]$, esta última quedará en la lista de candidatos (por lo que será planificada en una iteración posterior a i), al ser incompatible con la recién seleccionada.

Esta decisión que hemos tomado en la iteración i puede condicionar en una iteración j ($j > i$) de esta manera: En la iteración j se ha planificado una tercera conferencia $[q, w]$, esta conferencia no es compatible con $[x, y], w > x$ (que seguirá por tanto en la lista de candidatos y el algoritmo tendrá que iterar y por tanto utilizar por lo menos otro aula), sin embargo esta conferencia si es compatible con $[x + c, k]$ ($x + c > w$). Por las decisiones que hemos tomado, tenemos que planificar estas 3 conferencias en 3 aulas distintas, sin embargo, las podríamos haber planificado en solo 2 con el criterio de tomar la que empiece antes ya que $[x, y]$ se hubiese planificado en la iteración i y las otras dos en la j.

Con esto último no afirmamos que no se pueda hacer una planificación correcta con un criterio distinto al de elegir siempre la que empiece antes. Sin embargo, con ciertos conjuntos de conferencias, se corre el riesgo de utilizar aulas extra sin necesidad, con nuestra función de selección esto no sucede.