Entrega 3: Asignación de quirófanos en un hospital

Enes ZIDOURI
Guillaume TRAN-RUESCHE

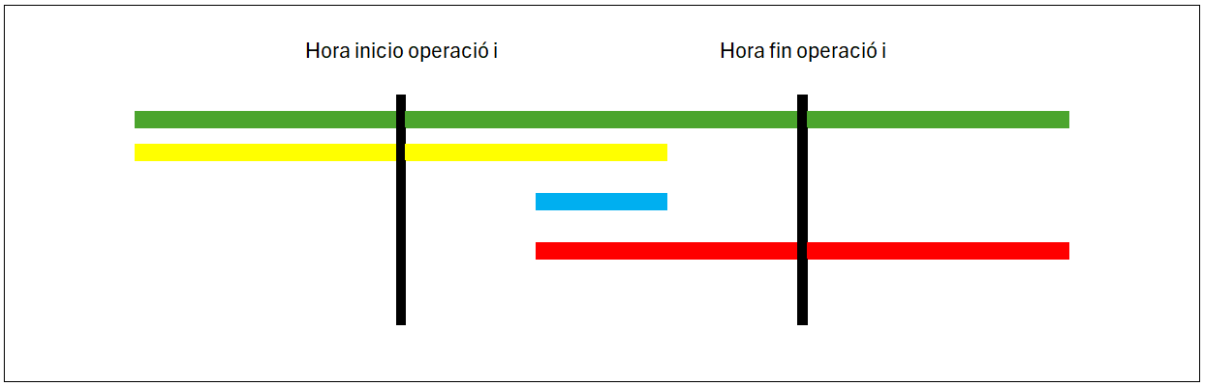
1. Modelo 1

Para nosotros, la primera dificultad de esta pregunta, era de importar los datos de manera eficaz con el módulo *pandas* de Python. Para facilitar la comprensión general, y los diferentes tests que realizar para evaluar nuestros algoritmos, decidimos simplificar los nombres de los quirófanos y de las operaciones. Entonces, en nuestro trabajo, el quirófano "Quirófano 34" se llamará "Q-34". Igualmente, la operación "20241204 OP-68" será "OP-68". Los datos utilizados en la pregunta una están representados en la tabla siguiente. Además, decidimos que los indexes del dataframe de las operaciones son las operaciones, y los del dataframe de los costes serán los quirófanos. Entonces, `df_costes.loc['Q-3', 'OP-68']` dará el coste de hacer la operación 68 en el quirófano 3.

Código operación	Equipo de Cirugía	Especialidad quirúrgica	Hora inicio	Hora fin
20241204 OP-68	Equipo Alfonso Rodríguez	Cardiología Pediátrica	4-12-24 2:45	4-12-24 3:30
20241204 OP-57	Equipo Alicia González	Cardiología Pediátrica	4-12-24 7:25	4-12-24 8:00
20241204 OP-133	Equipo Ángel Castillo	Cardiología Pediátrica	4-12-24 2:25	4-12-24 3:45
20241204 OP-12	Equipo David Guerrero	Cardiología Pediátrica	4-12-24 4:15	4-12-24 4:50
20241204 OP-159	Equipo Dolores González	Cardiología Pediátrica	4-12-24 3:25	4-12-24 4:25
20241204 OP-18	Equipo Encarnación Díaz	Cardiología Pediátrica	4-12-24 11:30	4-12-24 12:25
20241204 OP-67	Equipo Inmaculada Serrano	Cardiología Pediátrica	4-12-24 12:50	4-12-24 1:40
20241204 OP-2	Equipo Pablo Domínguez	Cardiología Pediátrica	4-12-24 1:40	4-12-24 3:00
20241204 OP-138	Equipo Pablo Marín	Cardiología Pediátrica	4-12-24 7:25	4-12-24 8:10
20241204 OP-5	Equipo Pedro Muñoz	Cardiología Pediátrica	4-12-24 4:55	4-12-24 5:40
20241204 OP-44	Equipo Ricardo Suárez	Cardiología Pediátrica	4-12-24 3:00	4-12-24 3:50
20241204 OP-107	Equipo Sergio Navarro	Cardiología Pediátrica	4-12-24 2:50	4-12-24 4:00
20241204 OP-88	Equipo Teresa Ramírez	Cardiología Pediátrica	4-12-24 12:12	4-12-24 3:30

Datos utilisados para el primer modelo

Como explicado en el enunciado, la mayor dificultad es establecer los conjuntos L_i que representan las operaciones incompatibles con la operación i. Para crearlo, hemos identificado 3 situaciones posibles de incompatibilidad, representadas en la imagen siguiente:



Posibilidades de incompatibilidad

Entonces, definimos 3 criterios de incompatibilidad. Sea O_i la hora de inicio de la operación O, O_f la hora de fin, P_i la hora de inicio de la operación P, y P_f la de fin. Tenemos incompatibilidad si:

- $P_i \leq O_i$ y $P_f > O_i$ (caso verde y amarillo)
- $P_i < O_f$ y $P_f \geq O_f$ (caso verde y rojo)
- $P_i \geq O_i$ y $P_f \leq O_i$ (caso azul)

Por fin, para evitar hacer dos bucles completos, y calcular dos veces la incompatibilidad entre dos operaciones, definimos el primer bucle que recoge todas las operaciones i, y el segundo j tal que $j > i$, y cada vez que tenemos una incompatibilidad, añadimos i en el conjunto de incompatibilidad de j, y j en el de i.

Una vez el modelo creado con *pulp* (sin dificultades), Conseguimos la solución siguiente con un coste total de 1510.

```
Optimal
1510.0
OP-138 Q-4
OP-57 Q-11
OP-44 Q-21
OP-133 Q-23
OP-5 Q-24
OP-68 Q-34
OP-2 Q-40
OP-12 Q-42
OP-107 Q-50
OP-67 Q-58
OP-159 Q-61
OP-18 Q-65
OP-88 Q-71
```

Solución del primer modelo

Con esta solución podemos ver que tenemos un quirófano diferente para cada operación. No está sorprendente porque no hay ninguno ventaja en hacer diferentes operaciones en el mismo quirófano (no hay costes “fijos” como en la entrega previa). Para verificar, intentamos fijar el costo (Q-24, OP-68) a 0, y, como esperado, el modelo asignó la operación OP-68 en el quirófano Q-24, con la operación OP-5, porque no hay incompatibilidad entre las dos.

2. Modelo 2 (homogéneo)

El objetivo del modelo 2 era minimizar el coste del uso del quirófano seleccionando los horarios adecuados.

El principal reto era generar los horarios.
La estrategia utilizada para generar los horarios fue la siguiente: en primer lugar, se crea un conjunto vacío. A continuación, repasamos cada operación y comprobamos si es incompatible con las operaciones de los distintos horarios. Si hay un programa con el que la

operación es compatible, se añade la operación. Si no, se crea un nuevo horario que incluya la operación.

```
PS C:\M2\MCA\entrega-3-gc-grupo-mca-g> & C:/Users/enesz/anaconda3/python.exe c:/M2/MCA/entrega-3-gc-grupo-mca-g/entrega2-2
.py
oui [['OP-68', 'OP-57', 'OP-12', 'OP-18', 'OP-67', 'OP-5', 'OP-143', 'OP-21'], ['OP-133', 'OP-138', 'OP-167', 'OP-99', 'OP
-9', 'OP-121'], ['OP-159', 'OP-2', 'OP-22', 'OP-59', 'OP-105'], ['OP-44', 'OP-70', 'OP-102'], ['OP-107', 'OP-148', 'OP-139
'], ['OP-88', 'OP-117', 'OP-36'], ['OP-35', 'OP-165'], ['OP-125', 'OP-83'], ['OP-110', 'OP-104', 'OP-126'], ['OP-55'], ['O
P-23'], ['OP-73', 'OP-78'], ['OP-34'], ['OP-1'], ['OP-135'], ['OP-164'], ['OP-30'], ['OP-163'], ['OP-156']]
Welcome to the CBC MILP Solver
```

Todos los planes creados para el modelo

Realizamos varias pruebas para determinar si la generación de horarios influía en el resultado final. Para ello, generamos varios conjuntos diferentes de calendarios realizando las operaciones de forma aleatoria.

En una segunda prueba, también intentamos determinar si el tamaño de las programaciones influía en el coste final. Por ejemplo, limitamos el número de operaciones a 2 y 8.

Cuando vimos que el coste final aumentaba, decidimos no limitar el número de operaciones por programa y asignar una operación a un único programa.

El modelo nos da una solución óptima con un coste total de 57.923.

```
Optimal
57923.62626262626
['OP-68', 'OP-57', 'OP-12', 'OP-18', 'OP-67', 'OP-5', 'OP-143', 'OP-21']
['OP-133', 'OP-138', 'OP-167', 'OP-99', 'OP-9', 'OP-121']
['OP-159', 'OP-2', 'OP-22', 'OP-59', 'OP-105']
['OP-44', 'OP-70', 'OP-102']
['OP-107', 'OP-148', 'OP-139']
['OP-88', 'OP-117', 'OP-36']
['OP-35', 'OP-165']
['OP-125', 'OP-83']
['OP-110', 'OP-104', 'OP-126']
['OP-55']
['OP-23']
['OP-34']
['OP-1']
['OP-135']
['OP-164']
['OP-30']
['OP-163']
['OP-156']
```

Solución del segundo modelo

Hay un contraste interesante. Algunos planes son capaces de albergar hasta 8 operaciones. Otros, en cambio, no son óptimos. De hecho, en varias ocasiones vemos que un bloque se utiliza para una sola operación. Esto nos lleva al objetivo del modelo 3.

3. Modelo 3 (Generación de columnas)

Para crear este modelo, empezamos en creando el código del problema de “Cutting Stock” visto en clase. Eso permitió construir en primero el modelo unidimensional, luego el modelo de generación con los precios sombras, y finalmente, el enlace entre los dos modelos, el intercambio de datos (precios sombras, nuevo patrón, etc.) y la condición de parada.
Una vez realizado, hemos hecho una analogía entre el modelo de “Cutting Stock” y el nuestro para hacer las modificaciones de adaptación. En nuestro caso, no hay una demanda d_f por finales, pero una condición de asignación de una operación a al menos 1 quirófano para las restricciones del problema unidimensional. También, fue difícil, pero entendimos que las restricciones del modelo de generación, eran todas las incompatibilidades entre las operaciones.
Una vez el modelo realizado y después de múltiples fallos, conseguimos el valor de **93 quirófanos**. Este valor parece un poquito decepcionante porque hace en promedio un poquito menos de 2 operaciones por quirófano, lo que no parece óptimo. Sin embargo, puede ser el valor óptimo si las operaciones tienen muchas incompatibilidades. También, con nuestra función de generación ingenua de las primeras planificaciones, conseguimos un total de 95 quirófanos. Entonces, con nuestro algoritmo de generación de columnas, su utilización puede ser criticable.

Aquí están las dificultades que nos costaban mucho tiempo:

- *Problemas en las restricciones del modelo de generación*. En la primera versión de estas restricciones, hemos recuperado los L_i y hemos añadido la entera lista como restricción. Por ejemplo, para la operación OP-68 y solo los datos del primer servicio, tuvimos las incompatibilidades ['OP-133', 'OP-159', 'OP-2', 'OP-44', 'OP-107', 'OP-88'], entonces, definimos las restricciones como $y_{68} + y_{133} + y_{159} + y_2 + y_{44} + y_{107} + y_{88} \leq 1$ al lugar de los variables dos a dos ($y_{68} + y_{133} \leq 1,\ y_{68} + y_{159} \leq 1, y_{68} + y_2 \leq 1, \ldots$)
- *Problema en la creación de la nueva planificación.* Cuando el código creaba la nueva planificación, los números de apariciones de las operaciones en el vector creado no correspondieron con el orden de las operaciones en la lista de planificaciones. Era un problema difícil de ver, y lo detectamos con el problema siguiente.
- *Variable igual a None.* Ejecutando el modelo en una pequeña instancia, a veces tuvimos un problema de variable y_j igual a *None*. Después de alguna investigación, entendimos que era porque, cuando la variable no tiene impacto en el modelo (como por ejemplo la operación OP-5 que no tiene incompatibilidad en los datos del primer servicio, entonces no importa que y_5 vale 0 o 1), es posible que el modelo no considera este variable. Entonces, cuando añadimos un valor None en una nueva planificación, no funcionaba. Para resolver este problema, si la lista de las incompatibilidades está vacía, añadimos una restricción para fijar el valor de esta variable a 1, para estar seguro de que la variable aparezca en la solución. Esta solución crea un otro problema que no resolvamos por el momento, porque solo impacta el tiempo de cálculo. En efecto, por el momento añadimos en el problema

de generación cada restricción dos veces (por ejemplo, $y_{68} + y_{133} \leq 1$ y $y_{133} + y_{68} \leq 1$) lo que no es necesario. Eso puede simplificarse de manera sencilla (pero no tuvimos el tiempo de hacerlo)., aunque necesitemos también las listas vacías para el problema previo.