# ASIGNACIÓN DE QUIRÓFANOS EN UN HOSPITAL Y GENERACIÓN DE COLUMNAS

## ENTREGA 3

**Grupo MCA - N:**

Andrés Janeiro Villar

Diego Pérez Castro

Mario Rodríguez Pastrana

Lucía Rosas Otero

Fecha: 26/11/2024

## 1. Modelo 1

El modelo completo se ha formulado y resuelto en un código Python, el cual se adjunta junto con este informe bajo el nombre “Entrega 3. Ejercicio 1”.

Primeramente, se importaron los dos archivos Excel que contenían parte de los datos del problema:

- Excel con los costes de cada operación según el tipo de quirófano (importado como DataFrame). Se utilizaron los índices de la tabla (primera columna) para crear una lista con los quirófanos existentes.
- Excel con los datos de las operaciones programadas (importado como DataFrame). Se utilizaron los índices de la tabla (primera columna) para crear una lista con las operaciones existentes.

No obstante, el enunciado pide realizar este primer ejercicio para una especialidad quirúrgica concreta, en este caso Cardiología Pediátrica, por lo cuál nos quedaremos únicamente con los datos relativos a esta especialidad tal y como se establece en la siguiente línea de código:

```
programacion = pd.read_excel("241204_datos_operaciones_programadas.xlsx", index_col=0)
programacion = programacion[programacion["Especialidad quirúrgica"] == "Cardiología Pediátrica"]
```

Además, se iteró sobre cada operación comparándola con el resto de las operaciones, para identificar incompatibilidades (dos operaciones son incompatibles de realizarse en el mismo quirófano si sus horarios se solapan). Estas incompatibilidades se registraron en un diccionario *op_incompatibilidades*, donde el primer elemento indica la operación en cuestión y el segundo elemento es una lista de las operaciones incompatibles con la primera. Para generar este diccionario, se empleó el siguiente código:

```
op_incompatibles = {}
for i in operaciones:
    I_i = pd.to_datetime(programacion.loc[i,"Hora inicio"])
    F_i = pd.to_datetime(programacion.loc[i,"Hora fin"])
    L = []
    for j in operaciones:
        if i==j:
            continue
        else:
            I_j = pd.to_datetime(programacion.loc[j, "Hora inicio"])
            F_j = pd.to_datetime(programacion.loc[j, "Hora fin"])
            if (I_i >= I_j and I_i < F_j) or (F_i > I_j and F_i <= F_j) or (I_i <= I_j and F_i >= F_j):
                L.append(j)
    op_incompatibles[i] = L
```

Con los dos archivos Excel importados y su contenido procesado, se dio comienzo a la formulación del problema. Para ello se definió:

- Conjuntos: *I* (operaciones), *J* (quirófanos) y *L* (operaciones incompatibles). La elaboración de estos tres elementos ya se justificó al explicar la importación de los archivos Excel.
- Parámetros: $C_{ji}$ (costes). Se usó el mismo DataFrame importado con el primer archivo Excel.

Con todo listo, se comenzó con el problema de optimización; que en este caso busca minimizar el coste total de la asignación quirúrgica del servicio de Cardiología Pediátrica.

La única variable a emplear, $x_{ij}$, tiene valor 1 si la operación *i* se asigna al quirófano *j*, y valor 0 en caso contrario.

La primera restricción obliga a que cada operación esté asignada por lo menos a un quirófano. La segunda restricción permite que no haya operaciones incompatibles en el mismo quirófano al mismo tiempo.

Tras definir las variables, función objetivo, restricciones y ejecutar el problema, se procedió a imprimir el valor de la función objetivo y del estado del problema.

- El valor de la función objetivo, que minimiza el coste total, es de 1510 €.

```
El valor de la función objetivo (coste total) es de:  1510.0 €
```

- El estado del problema es óptimo.

```
El estado del problema es:  Optimal
```

Por último, para mostrar el valor de cada variable, se decidió indicar, además del quirófano asignado, el equipo de cirugía asociado a cada operación.

```
Asignación operaciones - quirófanos:
- La operación 20241204 OP-18 (Equipo Encarnación Díaz) se asigna al Quirófano 65
- La operación 20241204 OP-88 (Equipo Teresa Ramírez) se asigna al Quirófano 71
- La operación 20241204 OP-67 (Equipo Inmaculada Serrano) se asigna al Quirófano 58
- La operación 20241204 OP-2 (Equipo Pablo Domínguez) se asigna al Quirófano 40
- La operación 20241204 OP-133 (Equipo Ángel Castillo) se asigna al Quirófano 23
- La operación 20241204 OP-68 (Equipo Alfonso Rodríguez) se asigna al Quirófano 34
- La operación 20241204 OP-107 (Equipo Sergio Navarro) se asigna al Quirófano 50
- La operación 20241204 OP-44 (Equipo Ricardo Suárez) se asigna al Quirófano 21
- La operación 20241204 OP-159 (Equipo Dolores González) se asigna al Quirófano 61
- La operación 20241204 OP-12 (Equipo David Guerrero) se asigna al Quirófano 42
- La operación 20241204 OP-5 (Equipo Pedro Muñoz) se asigna al Quirófano 24
- La operación 20241204 OP-57 (Equipo Alicia González) se asigna al Quirófano 11
- La operación 20241204 OP-138 (Equipo Pablo Marín) se asigna al Quirófano 4
```

A mayores, para personalizar la impresión de la solución y ayudar a los cirujanos a la visualización de la ocupación de los quirófanos durante el día, se imprimió el número de operaciones que van a ocurrir en cada quirófano, tal y como se muestra en la siguiente imagen:

```
Número de operaciones por quirófano:
- El Quirófano 4 ubicará 1 operación
- El Quirófano 11 ubicará 1 operación
- El Quirófano 21 ubicará 1 operación
- El Quirófano 23 ubicará 1 operación
- El Quirófano 24 ubicará 1 operación
- El Quirófano 34 ubicará 1 operación
- El Quirófano 40 ubicará 1 operación
- El Quirófano 42 ubicará 1 operación
- El Quirófano 50 ubicará 1 operación
- El Quirófano 58 ubicará 1 operación
- El Quirófano 61 ubicará 1 operación
- El Quirófano 65 ubicará 1 operación
- El Quirófano 71 ubicará 1 operación
```

## 2. Modelo 2 (homogéneo)

En esta variante del modelo anterior, se tratará de llegar a una programación de quirófanos factible, donde los costes de cada operación son independientes del quirófano donde se realice. Se tendrán en cuenta únicamente las operaciones de 4 especialidades distintas.

- Se calcula el coste medio de realizar cada operación.

```
costes = pd.read_excel("241204_costes.xlsx", index_col=0)

coste_medio = costes.mean()
print("\nCostes de cada operación")
print(coste_medio)
```

- Se filtran las operaciones para tener en cuenta únicamente las de las especialidades requeridas.

```
especialidades = ["Cardiología Pediátrica", "Cirugía Cardíaca Pediátrica",
                  "Cirugía Cardiovascular", "Cirugía General y del Aparato Digestivo"]
programacion = pd.read_excel(
    "241204_datos_operaciones_programadas.xlsx", index_col=0)
programacion_filtrada = programacion[
    (programacion["Especialidad quirúrgica"] == "Cardiología Pediátrica") |
    (programacion["Especialidad quirúrgica"] == "Cirugía Cardíaca Pediátrica") |
    (programacion["Especialidad quirúrgica"] == "Cirugía Cardiovascular") |
    (programacion["Especialidad quirúrgica"] =="Cirugía General y del Aparato Digestivo")
]
```

Dado que generar todas las posibles combinaciones de operaciones es computacionalmente inviable, se sugiere crear un subconjunto de planificaciones relevantes.

Para ello, se seguirá una lógica que consiste en ordenar las operaciones por su hora de inicio y coger la primera de ellas (operación base) para generar planificaciones partiendo de ésta, de manera que:

1. Se añade la “operación base” a la planificación, eliminándola de la lista de operaciones pendientes.

```
operacion_actual = operaciones_pendientes[0]
planificacion.append(operacion_actual)
operaciones_pendientes.remove(operacion_actual)
```

2. Se comprueba si la siguiente operación de la lista podría realizarse después o no (si es compatible o no). En caso de que si, se añade a la planificación y se borra de la lista de las operaciones que todavía no han sido asignadas (pendientes). En caso de que no, se pasa a la siguiente. De esta forma se asegura que el tiempo entre operaciones es el mínimo posible.

```
while orden < len(operaciones_pendientes):
    operacion_siguiente = operaciones_pendientes[orden]
    compatibilidad = True
    I_op_s = pd.to_datetime(programacion_ordenada.loc[operacion_siguiente, "Hora inicio"])
    F_op_s = pd.to_datetime(programacion_ordenada.loc[operacion_siguiente, "Hora fin"])
    for j in planificacion:
        I_j = pd.to_datetime(programacion_ordenada.loc[j, "Hora inicio"])
        F_j = pd.to_datetime(programacion_ordenada.loc[j, "Hora fin"])
        if (I_op_s >= I_j and I_op_s < F_j) or (F_op_s > I_j and F_op_s <= F_j) or (I_op_s <= I_j and F_op_s >= F_j):
            compatibilidad = False
            break
    if compatibilidad == True:
        planificacion.append(operacion_siguiente)
        operaciones_pendientes.remove(operacion_siguiente)
    else:
        orden = orden + 1
```

3. Se procede de esta manera, creando distintas planificaciones hasta repetir el bucle la cantidad de veces necesarias que hagan que la lista de operaciones pendientes quede vacía, de manera que se van añadiendo las diferentes planificaciones creadas a la lista "planificaciones".

```
while operaciones_pendientes:
    planificacion = []
    operacion_actual = operaciones_pendientes[0]
    planificacion.append(operacion_actual)
    operaciones_pendientes.remove(operacion_actual)
    orden = 0
    while orden < len(operaciones_pendientes):
        operacion_siguiente = operaciones_pendientes[orden]
        compatibilidad = True
        I_op_s = pd.to_datetime(programacion_ordenada.loc[operacion_siguiente, "Hora inicio"])
        F_op_s = pd.to_datetime(programacion_ordenada.loc[operacion_siguiente, "Hora fin"])
        for j in planificacion:
            I_j = pd.to_datetime(programacion_ordenada.loc[j, "Hora inicio"])
            F_j = pd.to_datetime(programacion_ordenada.loc[j, "Hora fin"])
            if (I_op_s >= I_j and I_op_s < F_j) or (F_op_s > I_j and F_op_s <= F_j) or (I_op_s <= I_j and F_op_s >= F_j):
                compatibilidad = False
                break
        if compatibilidad == True:
            planificacion.append(operacion_siguiente)
            operaciones_pendientes.remove(operacion_siguiente)
        else:
            orden = orden + 1
    planificaciones.append(planificacion)
```

De esta forma, se ha creado una posible combinación de planificaciones que asegura el máximo aprovechamiento del tiempo partiendo de la operación base

Para aumentar el número de posibles planificaciones, este proceso se repite tomando como "operación base" cada una de las operaciones. Para estos nuevos casos, será necesario añadir un nuevo paso 0, anterior al paso 1, en el que se reorganiza la lista de operaciones ordenadas, de manera que tras las últimas operaciones del día se añadan las primeras de la mañana hasta llegar a la operación base, que será la primera de la lista.

```
planificaciones = []
for i in operaciones:
    operaciones_pendientes = operaciones[operaciones.index(i):] + operaciones[:operaciones.index(i)].copy()
```

Una vez tenemos el conjunto de posibles planificaciones, se procede a definir y resolver el modelo de programación lineal, cuyo objetivo será disminuir el coste total.

- Se definen los conjuntos y parámetros

```
# Conjuntos
I = operaciones

J = quirofanos

# Parámetros
coste_planificacion = {}
for k in K:
    lista_costes = []
    op_planificacion = K[k]
    for i in op_planificacion:
        coste_i = coste_medio.loc[i]
        lista_costes.append(coste_i)
    coste_planificacion[k] = sum(lista_costes)
```

Se crea un diccionario para almacenar los costes de cada planificación (K)

- Se define el modelo

```
# Inicializar modelo
modelo_2 = lp.LpProblem("Entrega_3_Ejercicio_2", lp.LpMinimize)
```

- Se definen las variables

```
# Variables
b = lp.LpVariable.dicts("Operación_en_planificación", [(i, k) for i in I for k in K], lowBound=0, cat=lp.LpBinary)
y = lp.LpVariable.dicts("Uso_quirófano", [k for k in K], lowBound=0, cat=lp.LpBinary)
```

$b_{ij}$: es una variable binaria que vale 1 si la operación i pertenece a la planificación *k* y 0 si no pertenece.

$y_k$: es una variable binaria que vale 1 si se lleva a cabo una planificación determinada y 0 si no. Cada planificación corresponde a un quirófano.

- Se crea la función objetivo

```
# Función objetivo
modelo_2 += lp.lpSum(coste_planificacion[k] * y[k] for k in K)
```

El objetivo será minimizar la suma de los costes de las operaciones de cada planificación

- Se definen las restricciones

```
# Restricciones
for i in I:
    modelo_2 += lp.lpSum(y[k] for k in K if i in planificaciones[k]) >= 1
```

Cada operación solo podrá estar asignada a una planificación dentro de la programación diaria.

Por último, se resuelve el modelo, dando como resultado:

```
El valor de la función objetivo (coste total) es de: 57923.63 €

El estado del problema es: Optimal
```

Correspondiente a la siguiente distribución:

```
En total se requerirán 23 quirófanos

La asignación de operaciones a cada quirófano es:
- Quirófano 1: ['20241204 OP-126']
- Quirófano 2: ['20241204 OP-30']
- Quirófano 3: ['20241204 OP-34']
- Quirófano 4: ['20241204 OP-167']
- Quirófano 5: ['20241204 OP-68', '20241204 OP-36']
- Quirófano 6: ['20241204 OP-44', '20241204 OP-73']
- Quirófano 7: ['20241204 OP-163', '20241204 OP-159', '20241204 OP-121', '20241204 OP-143', '20241204 OP-57',
'20241204 OP-105', '20241204 OP-9']
- Quirófano 8: ['20241204 OP-1', '20241204 OP-22']
- Quirófano 9: ['20241204 OP-88']
- Quirófano 10: ['20241204 OP-165']
- Quirófano 11: ['20241204 OP-107', '20241204 OP-102', '20241204 OP-99']
- Quirófano 12: ['20241204 OP-78', '20241204 OP-125']
- Quirófano 13: ['20241204 OP-110']
- Quirófano 14: ['20241204 OP-133']
- Quirófano 15: ['20241204 OP-156']
- Quirófano 16: ['20241204 OP-135']
- Quirófano 17: ['20241204 OP-104']
- Quirófano 18: ['20241204 OP-2', '20241204 OP-83']
- Quirófano 19: ['20241204 OP-35']
- Quirófano 20: ['20241204 OP-138', '20241204 OP-70', '20241204 OP-164']
- Quirófano 21: ['20241204 OP-148', '20241204 OP-18', '20241204 OP-67', '20241204 OP-59', '20241204 OP-139',
'20241204 OP-12', '20241204 OP-5']
- Quirófano 22: ['20241204 OP-23', '20241204 OP-117']
- Quirófano 23: ['20241204 OP-21', '20241204 OP-55']
```

Llegados a este punto, nos damos cuenta de que el coste será constante, ya que el número de operaciones no varía y su coste asociado no depende de dónde se realice la operación. Surgen dos posibles propuestas para poder minimizar el número de quirófanos en uso, de manera que la programación quede más optimizada:

- Cambiar la función objetivo para que minimice el número de quirófanos (minimizar el número de elementos de la lista de programación.
- Añadir un coste asociado fijo al uso de los quirófanos para que así al minimizar los costes se tenga en cuenta que el uso de más quirófanos repercute en el coste y es desfavorable.

Optamos por volver a realizar el problema con la siguiente función objetivo:

```
# Función objetivo
#modelo_2 += lp.lpSum(coste_planificacion[k] * y[k] for k in K)
modelo_2 += lp.lpSum(y[k] for k in K)
```

En esta simulación se vuelve a obtener como resultado 23 quirófanos, lo que indica que debido al algoritmo que se ha empleado para definir las planificaciones, éstas eran las óptimas.

## 3. Modelo 3 (generación columnas)

En este apartado se plantea un enfoque basado en la generación de columnas para abordar el problema. A continuación, se explica la elaboración del enfoque adoptado y sus principales ventajas.

Una vez se leen los archivos Excel necesarios y se proporcionan los quirófanos disponibles a partir del archivo de costes, así como el orden de las operaciones planificadas, se lleva a cabo la generación de todas las combinaciones factibles de operaciones que no se solapan temporalmente. Estas planificaciones formarán el conjunto inicial de columnas para el modelo maestro que se definirá más adelante.

```
planificaciones = []
for i in operaciones:
    operaciones_pendientes = operaciones[operaciones.index(i):] + operaciones[:operaciones.index(i)].copy()
    print(f"\nOperación {i}: {len(planificaciones)} planificaciones")
    while operaciones_pendientes:
        planificacion = []
        operacion_actual = operaciones_pendientes[0]
        planificacion.append(operacion_actual)
        operaciones_pendientes.remove(operacion_actual)
        orden = 0
        while orden < len(operaciones_pendientes):
            operacion_siguiente = operaciones_pendientes[orden]
            compatibilidad = True
            I_op_s = pd.to_datetime(programacion_ordenada.loc[operacion_siguiente, "Hora inicio"])
            F_op_s = pd.to_datetime(programacion_ordenada.loc[operacion_siguiente, "Hora fin"])
            for j in planificacion:
                I_j = pd.to_datetime(programacion_ordenada.loc[j, "Hora inicio"])
                F_j = pd.to_datetime(programacion_ordenada.loc[j, "Hora fin"])
                if (I_op_s >= I_j and I_op_s < F_j) or (F_op_s > I_j and F_op_s <= F_j) or (I_op_s <= I_j and F_op_s >= F_j):
                    compatibilidad = False
                    break
            if compatibilidad == True:
                planificacion.append(operacion_siguiente)
                operaciones_pendientes.remove(operacion_siguiente)
            else:
                orden = orden + 1
        planificaciones.append(planificacion)
```

Una vez llegados a este punto, se define el modelo maestro con Pulp, donde la función objetivo debe minimizar el número de quirófanos utilizados, y las restricciones aseguran que cada operación se asigna al menos a un quirófano.

```python
def modelo_maestro(K):
    #Conjuntos
    I = operaciones

    #Inicializar modelo
    modelo_3 = lp.LpProblem("Entrega_3_Ejercicio_3", lp.LpMinimize)

    #Variables
    y = lp.LpVariable.dicts("Uso_quirófano", [k for k in K], lowBound=0, cat=lp.LpBinary)

    #Función objetivo
    modelo_3 += lp.lpSum(y[k] for k in K)

    #Restricciones
    for i in I:
        modelo_3 += lp.lpSum(y[k] for k in K if i in planificaciones[k]) >= 1

    #Resolver modelo
    modelo_3.solve()

    return modelo_3, y
```

Con el modelo maestro definido, se procede a generar nuevas variables para el mismo, evaluando la información dual del modelo maestro para identificar soluciones candidatas con costes reducidos. Por lo tanto, con esta función (generar columnas) obtendremos una lista de nuevas planificaciones que cumplirán las restricciones de compatibilidad.

Para lograrlo se recorren todas las restricciones del modelo maestro y se crean combinaciones de operaciones para evaluar su compatibilidad como se puede ver en la imagen.

```python
def generar_columnas(maestro, operaciones, planificaciones, programacion_ordenada):
    duales = {}
    for i in maestro.constraints.values():
        if hasattr(i, "pi"):
            duales[i.name] = i.pi
    nuevas_planificaciones = []
    for i in range(len(operaciones)):
        for j in range(i + 1, len(operaciones)):
            planificacion_candidata = [operaciones[i], operaciones[j]]
            compatible = True
            for m in range(len(planificacion_candidata)):
                for n in range(m + 1, len(planificacion_candidata)):
                    op1_inicio = programacion_ordenada.loc[planificacion_candidata[m], "Hora inicio"]
                    op1_fin = programacion_ordenada.loc[planificacion_candidata[m], "Hora fin"]
                    op2_inicio = programacion_ordenada.loc[planificacion_candidata[n], "Hora inicio"]
                    op2_fin = programacion_ordenada.loc[planificacion_candidata[n], "Hora fin"]
                    if (op1_inicio >= op2_inicio and op1_inicio < op2_fin) or (op1_fin > op2_inicio and op1_fin <= op2_fin) or (op1_inicio <= op2_inicio and op1_fin >= op2_fin):
                        compatible = False
                        break
                if not compatible:
                    break
            if compatible:
                coste_reducido = 1 - sum(duales.get(op, 0) for op in planificacion_candidata)
                if coste_reducido < 0:
                    nuevas_planificaciones.append(planificacion_candidata)
    return nuevas_planificaciones
```

Posteriormente, se procede a la generación e iteración de columnas hasta que no se encuentren mejoras en el modelo maestro y con ello, se resuelva el modelo.

```python
#Generación inicial de columnas
nuevas_planificaciones = generar_columnas(maestro, operaciones, planificaciones, programacion_ordenada)
for planificacion_nueva in nuevas_planificaciones:
    nuevo_id = len(K)
    K[nuevo_id] = planificacion_nueva

#Iteración de columnas hasta resolver modelo
while nuevas_planificaciones:
    maestro, y = modelo_maestro(K)
    nuevas_planificaciones = generar_columnas(maestro, operaciones, planificaciones, programacion_ordenada)
```

Por último, se procederá a imprimir el estado del problema, junto con las variables y la función objetivo definida con anterioridad.

```python
#Imprimir función objetivo, variables y estado del problema
asignaciones_quirofanos = {}
contador = 1
for i,j in y.items():
    if j.varValue == 1:
        operaciones_asignadas = K[i]
        asignaciones_quirofanos[contador] = operaciones_asignadas
        contador = contador + 1

print(f"\nEn total se requerirán {round(lp.value(maestro.objective),0)} quirófanos")

print(f"\nEl estado del problema es: {lp.LpStatus[maestro.status]}")

print("\nLa asignación de operaciones a cada quirófano es:")
for i,j in asignaciones_quirofanos.items():
    print(f"Quirófano {i}: {j}")
```

Una vez diseñado y ejecutado el código se obtuvo que el número mínimo de quirófanos necesarios es de 93.
El estado del problema fue Óptimo, asegurando la validez de la solución obtenida.

```
En total se requerirán 93.0 quirófanos

El estado del problema es: Optimal
```

Cada quirófano se asignó a un conjunto específico de operaciones, como se detalla a continuación:

```
Quirófano 1: ['20241204 OP-98']
Quirófano 2: ['20241204 OP-163']
Quirófano 3: ['20241204 OP-125']
Quirófano 4: ['20241204 OP-3', '20241204 OP-60']
Quirófano 5: ['20241204 OP-16']
Quirófano 6: ['20241204 OP-100']
Quirófano 7: ['20241204 OP-107', '20241204 OP-102', '20241204 OP-72']
Quirófano 8: ['20241204 OP-167']
Quirófano 9: ['20241204 OP-127']
Quirófano 10: ['20241204 OP-162']
Quirófano 11: ['20241204 OP-45']
Quirófano 12: ['20241204 OP-71']
Quirófano 13: ['20241204 OP-53']
Quirófano 14: ['20241204 OP-4', '20241204 OP-159', '20241204 OP-42', '20241204 OP-24',
'20241204 OP-11', '20241204 OP-151', '20241204 OP-39', '20241204 OP-80', '20241204 OP-131']
Quirófano 15: ['20241204 OP-74']
Quirófano 16: ['20241204 OP-8']
Quirófano 17: ['20241204 OP-23']
Quirófano 18: ['20241204 OP-17']
Quirófano 19: ['20241204 OP-161', '20241204 OP-85']
Quirófano 20: ['20241204 OP-174']
Quirófano 21: ['20241204 OP-171', '20241204 OP-78']
Quirófano 22: ['20241204 OP-140']
Quirófano 23: ['20241204 OP-31']
Quirófano 24: ['20241204 OP-124', '20241204 OP-108', '20241204 OP-132', '20241204 OP-117',
'20241204 OP-27', '20241204 OP-79', '20241204 OP-20', '20241204 OP-99', '20241204 OP-67',
'20241204 OP-145']
Quirófano 25: ['20241204 OP-104']
Quirófano 26: ['20241204 OP-119']
Quirófano 27: ['20241204 OP-1']
Quirófano 28: ['20241204 OP-10']
Quirófano 29: ['20241204 OP-49', '20241204 OP-86', '20241204 OP-47']
Quirófano 30: ['20241204 OP-33']
```

```
Quirófano 31: ['20241204 OP-63']
Quirófano 32: ['20241204 OP-122']
Quirófano 33: ['20241204 OP-7']
Quirófano 34: ['20241204 OP-65', '20241204 OP-120', '20241204 OP-105', '20241204 OP-25',
'20241204 OP-68']
Quirófano 35: ['20241204 OP-43']
Quirófano 36: ['20241204 OP-93']
Quirófano 37: ['20241204 OP-50']
Quirófano 38: ['20241204 OP-150', '20241204 OP-9', '20241204 OP-165']
Quirófano 39: ['20241204 OP-64']
Quirófano 40: ['20241204 OP-175']
Quirófano 41: ['20241204 OP-97', '20241204 OP-141']
Quirófano 42: ['20241204 OP-34']
Quirófano 43: ['20241204 OP-59', '20241204 OP-146', '20241204 OP-13', '20241204 OP-22',
'20241204 OP-172', '20241204 OP-101']
Quirófano 44: ['20241204 OP-76', '20241204 OP-128', '20241204 OP-139', '20241204 OP-168',
'20241204 OP-134', '20241204 OP-90']
Quirófano 45: ['20241204 OP-144']
Quirófano 46: ['20241204 OP-88']
Quirófano 47: ['20241204 OP-2']
Quirófano 48: ['20241204 OP-69']
Quirófano 49: ['20241204 OP-14', '20241204 OP-18']
Quirófano 50: ['20241204 OP-55']
Quirófano 51: ['20241204 OP-158', '20241204 OP-118', '20241204 OP-92', '20241204 OP-82',
'20241204 OP-83', '20241204 OP-121']
Quirófano 52: ['20241204 OP-109']
Quirófano 53: ['20241204 OP-77']
Quirófano 54: ['20241204 OP-94', '20241204 OP-40', '20241204 OP-148']
Quirófano 55: ['20241204 OP-66']
Quirófano 56: ['20241204 OP-153']
Quirófano 57: ['20241204 OP-30']
Quirófano 58: ['20241204 OP-170']
Quirófano 59: ['20241204 OP-110']
```

```
Quirófano 60: ['20241204 OP-160', '20241204 OP-46', '20241204 OP-58', '20241204 OP-70',
'20241204 OP-164', '20241204 OP-166', '20241204 OP-42', '20241204 OP-24', '20241204 OP-143',
'20241204 OP-57', '20241204 OP-41', '20241204 OP-130']
Quirófano 61: ['20241204 OP-173']
Quirófano 62: ['20241204 OP-87', '20241204 OP-166', '20241204 OP-42', '20241204 OP-91',
'20241204 OP-113', '20241204 OP-149', '20241204 OP-54', '20241204 OP-32', '20241204 OP-157',
'20241204 OP-21']
Quirófano 63: ['20241204 OP-113', '20241204 OP-27', '20241204 OP-79', '20241204 OP-61',
'20241204 OP-12', '20241204 OP-5', '20241204 OP-91']
Quirófano 64: ['20241204 OP-137']
Quirófano 65: ['20241204 OP-35']
Quirófano 66: ['20241204 OP-56']
Quirófano 67: ['20241204 OP-136']
Quirófano 68: ['20241204 OP-133']
Quirófano 69: ['20241204 OP-112']
Quirófano 70: ['20241204 OP-142']
Quirófano 71: ['20241204 OP-126']
Quirófano 72: ['20241204 OP-29']
Quirófano 73: ['20241204 OP-38']
Quirófano 74: ['20241204 OP-44', '20241204 OP-103', '20241204 OP-96', '20241204 OP-138',
'20241204 OP-154', '20241204 OP-32', '20241204 OP-157', '20241204 OP-73']
Quirófano 75: ['20241204 OP-156']
Quirófano 76: ['20241204 OP-26']
Quirófano 77: ['20241204 OP-48']
Quirófano 78: ['20241204 OP-115']
Quirófano 79: ['20241204 OP-28']
Quirófano 80: ['20241204 OP-15']
Quirófano 81: ['20241204 OP-72', '20241204 OP-116', '20241204 OP-36', '20241204 OP-106',
'20241204 OP-75', '20241204 OP-37']
Quirófano 82: ['20241204 OP-54', '20241204 OP-32', '20241204 OP-19', '20241204 OP-111',
'20241204 OP-169', '20241204 OP-22', '20241204 OP-149']
Quirófano 83: ['20241204 OP-81']
Quirófano 84: ['20241204 OP-135']
```

```
Quirófano 85: ['20241204 OP-129']
Quirófano 86: ['20241204 OP-95', '20241204 OP-114']
Quirófano 87: ['20241204 OP-62']
Quirófano 88: ['20241204 OP-51', '20241204 OP-147']
Quirófano 89: ['20241204 OP-84']
Quirófano 90: ['20241204 OP-117', '20241204 OP-172', '20241204 OP-18', '20241204 OP-157',
'20241204 OP-152', '20241204 OP-155', '20241204 OP-89', '20241204 OP-132']
Quirófano 91: ['20241204 OP-6']
Quirófano 92: ['20241204 OP-123']
Quirófano 93: ['20241204 OP-52']
```