CHE652

Report

Elective Surgery Scheduling

Problem Statement:

The hospital's objective is to efficiently schedule a set of elective surgeries within available operating room (OR) blocks over a fixed planning horizon. The primary goal is to minimize the total cost associated with performing surgeries, postponing surgeries, and managing overtime.

Assumptions:

1. The hospital has a known set of elective surgeries (I) that need to be scheduled.
2. There are a fixed number of available OR blocks (B), each with a predetermined duration (L).
3. Each elective surgery (i) has a deterministic duration (d_i), cost of performing and cost of postponing.
4. Emergency surgeries are not considered in this scheduling process.
5. The hospital incurs a fixed cost (c_i) for performing each surgery, and a higher cost (c_i') for postponing surgeries to the next planning horizon.
6. Overtime costs (c_o) are incurred if the cumulative duration of surgeries in a block exceeds the block duration.

Introduction

Hospitals are complex and expensive systems to manage. One department of particular interest that poses major managerial challenges is the operating room (OR) department. The OR department generates about 40–70% of revenues and incurs 20–40% of operating costs in a hospital [1, 4]. It also demands significant hospital resources and directly influences patient flow and efficiency of care delivery. Thus, hospital managers are constantly seeking better OR and surgery scheduling approaches to improve OR utilization, surgical care, and quality, as well as to minimize operational costs. Stochasticity is an intrinsic characteristic of OR and surgery scheduling problems since surgical activities are subject to multiple sources of uncertainty.

METHODOLOGY

PARAMETERS:

I: the number of elective surgeries (20 in this case)

B: the number of available OR blocks (10 in this case)

L: the duration of each OR block (300 minutes)

d: the duration of each surgery (a list of 20 values)

c: the cost of each surgery (a list of 20 values)

c_p: the penalty cost for not scheduling a surgery (a list of 20 values)

c_o: the cost of overtime per minute(5 in this case)

The values of all the parameters can be referred to from the code attached in the appendix.

DECISION VARIABLES :

y[i, b, p]: a binary variable indicating whether surgery i is assigned to block b at time slot p

t[b, p]: the start time of the surgery scheduled in block b at time slot p

o[b]: the overtime for block b

Objective Function:

The objective function combines three components:

1. **Minimize surgery assignment costs (obj1):**
   - It sums up the costs of assigning each surgery to a specific block and time slot.
   - $\sum_i \sum_b \sum_p c[i] \cdot y[i, b, p]$
2. **Minimize penalty costs for unassigned surgeries (obj2):**
   - It sums up the penalty costs for surgeries that are not assigned to any block or time slot.
   - $\sum_i c_p[i] \cdot (1 - \sum_b \sum_p y[i, b, p])$
3. **Minimize overtime costs (obj3):**
   - It sums up the overtime costs for each block.
   - $c_o \cdot \sum_b o[b]$

The total objective function is the sum of these three components.

CONSTRAINTS:

***note first constraint is very imp as it give rise to two cases being discussed and is the only difference between those cases.**

1. **Each surgery is assigned exactly once:**

$$\sum_{b=1}^{B} \sum_{p=1}^{B} y_{i,b,p} = \begin{cases} 1 & \text{if all surgeries must be scheduled without postponements} \\ \leq 1 & \text{if postponements are allowed} \end{cases}$$

Explanation: For each surgery i, the sum of $y_{i,b,p}$ over all blocks b and time slots p should be equal to 1 if all surgeries must be scheduled without postponements. If postponements are allowed, the sum should be less than or equal to 1.

2. **At most one surgery per block per time slot:**

$$\sum_{i=1}^{I} y_{i,b,p} \leq 1 \quad \text{for } b = 1, 2, \ldots, B \text{ and } p = 1, 2, \ldots, B$$

Explanation: For each block b and time slot p, the sum of $y_{i,b,p}$ over all surgeries i should be at most 1, indicating that at most one surgery can be assigned to block b at time slot p.

3. **Time scheduling constraint:**

$$t_{b,p} = \sum_{i=1}^{I} (t_{b,p-1} + d_i \cdot y_{i,b,p-1}) \quad \text{for } b = 1, 2, \ldots, B \text{ and } p = 2, 3, \ldots, B$$

   Explanation: For each block b and time slot p greater than 1, the start time $t_{b,p}$ is equal to the sum of the start time of the previous time slot $t_{b,p-1}$ and the duration of the surgery i assigned in the previous time slot multiplied by the binary variable $y_{i,b,p-1}$.

4. **Surgery start time must be non-negative:**

$$t_{b,p} \geq 0 \quad \text{for } b = 1, 2, \ldots, B \text{ and } p = 1, 2, \ldots, B$$

   Explanation: For each block b and time slot p, the start time $t_{b,p}$ should be non-negative, indicating that the surgery is scheduled to start at that time.

.. **Calculation of overtime for each block:**

$$o[b] = \left(\sum_{i=1}^{I} \sum_{p=1}^{B} (t_{b,p} + d_i \cdot y_{i,b,p}) \right) - L \quad \text{for } b = 1, 2, \ldots, B$$

Explanation: For each block b, the overtime $o[b]$ is calculated as the total time spent on surgeries in that block ($\sum_{i=1}^{I} \sum_{p=1}^{B} (t_{b,p} + d_i \cdot y_{i,b,p})$) minus the duration L of the block.

.. **Non-negative overtime for each block:**

$$o[b] \geq 0 \quad \text{for } b = 1, 2, \ldots, B$$

Explanation: The overtime $o[b]$ for each block b should be non-negative, ensuring that there is no negative overtime.

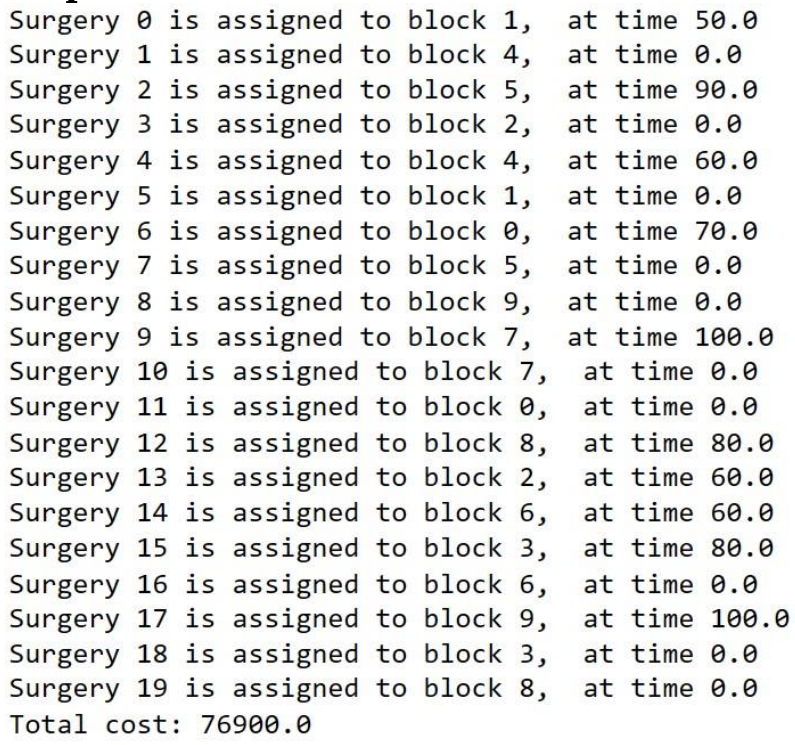
Surgery Start Time within Block Duration

$t_{b,p} \leq L \quad \text{for } b = 1, 2, \ldots, B \text{ and } p = 1, 2, \ldots, B$

For each block b and time slot p, the starting time $t_{b,p}$ of a surgery cannot exceed the duration L of the block. This ensures that surgeries start within the specified block durations.

RESULTS :

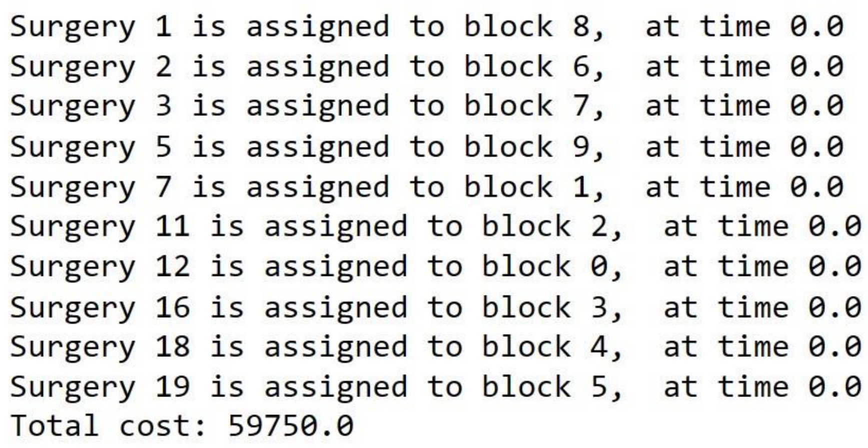
1) **Postponements NOT allowed:**



```
Surgery 0 is assigned to block 1,  at time 50.0
Surgery 1 is assigned to block 4,  at time 0.0
Surgery 2 is assigned to block 5,  at time 90.0
Surgery 3 is assigned to block 2,  at time 0.0
Surgery 4 is assigned to block 4,  at time 60.0
Surgery 5 is assigned to block 1,  at time 0.0
Surgery 6 is assigned to block 0,  at time 70.0
Surgery 7 is assigned to block 5,  at time 0.0
Surgery 8 is assigned to block 9,  at time 0.0
Surgery 9 is assigned to block 7,  at time 100.0
Surgery 10 is assigned to block 7,  at time 0.0
Surgery 11 is assigned to block 0,  at time 0.0
Surgery 12 is assigned to block 8,  at time 80.0
Surgery 13 is assigned to block 2,  at time 60.0
Surgery 14 is assigned to block 6,  at time 60.0
Surgery 15 is assigned to block 3,  at time 80.0
Surgery 16 is assigned to block 6,  at time 0.0
Surgery 17 is assigned to block 9,  at time 100.0
Surgery 18 is assigned to block 3,  at time 0.0
Surgery 19 is assigned to block 8,  at time 0.0
Total cost: 76900.0
```

Here idea is that no surgery is postponed and all the surgeries are considered for the scheduling. Taking I=20 surgeries, B=10 OR blocks, and 300 minutes as the length of each block with 5 as the overtime cost and same rest parameters. Here each surgery is scheduled and given a slot and adding the cost of each surgery and the overtime for each block we get our cost.

2) **Postponements allowed:**



```
Surgery 1 is assigned to block 8,  at time 0.0
Surgery 2 is assigned to block 6,  at time 0.0
Surgery 3 is assigned to block 7,  at time 0.0
Surgery 5 is assigned to block 9,  at time 0.0
Surgery 7 is assigned to block 1,  at time 0.0
Surgery 11 is assigned to block 2,  at time 0.0
Surgery 12 is assigned to block 0,  at time 0.0
Surgery 16 is assigned to block 3,  at time 0.0
Surgery 18 is assigned to block 4,  at time 0.0
Surgery 19 is assigned to block 5,  at time 0.0
Total cost: 59750.0
```

Taking I=20 surgeries, B=10 OR blocks, and 150 minutes as the length of each block with 5 as the overtime cost and same rest parameters . As the duration of the surgeries are defined to be from 60-170 minutes , 2-3 surgeries can't fit depending upon the block into its length and is postponed because of the extra overtime cost incurred would be much more than its postponement cost. Thus, reducing the overtime cost per minute may reduce chances of postponing.

CONCLUSION:

Elective surgery scheduling optimization plays a critical role in managing surgery, postponement, and overtime costs. By minimizing these expenses through efficient resource allocation, while ensuring patient satisfaction and safety, healthcare facilities can achieve significant cost savings and enhance overall operational performance and patient outcomes.

CONTRIBUTIONS: PRIYANSH SINGH (210775)

References:

1. Mixed-integer linear programming based solution methods for a stochastic surgery schedulingproblem - 14th AIMMS-MOPTA Optimization Modeling Competition - Team Bern Unicorns
2. Model Building in Mathematical Programming by H. Paul Williams

Appendix:

priyansh-singh-che652-project

April 12, 2024

[56]:
```
import gurobipy as gp
from gurobipy import GRB

# Define problem parameters
I = 20  # Number of elective surgeries
B = 10  # Number of available OR blocks
L =300   # Duration of each OR block (in minutes)

# Surgery durations and costs
d = [140, 60, 150, 60, 120, 50, 170, 90, 100, 140, 100, 70, 150, 140, 160, 100,␣
 ↪60, 160, 80, 80]# duration of surgeries
c =␣
 ↪[300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300]#␣
 ↪Cost of surgeries
c_p =␣
 ↪[500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500]#␣
 ↪Cost of postponing
c_o = 5  # Cost of overtime per minute

# Create the Gurobi model
model = gp.Model("Elective Surgery Scheduling")

# Define decision variables
y = model.addVars(I, B, range(B), vtype=GRB.BINARY, name="y")
t = model.addVars(B, range(B), vtype=GRB.CONTINUOUS, name="t")
o = model.addVars(B, vtype=GRB.CONTINUOUS, name="o")

# Objective function
obj1 = gp.quicksum(c[i] * y[i, b, p] for i in range(I) for b in range(B) for p␣
 ↪in range(B))
obj2 = gp.quicksum(c_p[i] * (1 - gp.quicksum(y[i, b, p] for b in range(B) for p␣
 ↪in range(B))) for i in range(I))
obj3 = c_o * gp.quicksum(o[b] for b in range(B))
obj = obj1 + obj2 + obj3
model.setObjective(obj, GRB.MINIMIZE)

# Constraints
```

```
model.addConstrs(gp.quicksum(y[i, b, p] for b in range(B) for p in range(B)) ==␣
↪1 for i in range(I))
model.addConstrs(gp.quicksum(y[i, b, p] for i in range(I)) <= 1 for b in␣
↪range(B) for p in range(B))
model.addConstrs(t[b, p] == gp.quicksum(t[b, p-1] + d[i] * y[i, b, p-1] for i␣
↪in range(I)) for b in range(B) for p in range(1, B))
model.addConstrs(t[b, p] >= 0 for b in range(B))
model.addConstrs(o[b] == gp.quicksum(t[b, p] + d[i] * y[i, b, p] for i in␣
↪range(I) for p in range(B)) - L for b in range(B))
model.addConstrs(o[b] >= 0 for b in range(B))
model.addConstrs(t[b,p]<= L for b in range(B) )


# Solve the problem
model.optimize()

# Print the solution
for i in range(I):
    for b in range(B):
        for p in range(B):
            if y[i, b, p].X > 0.5:
                print(f"Surgery {i} is assigned to block {b},  at time {t[b, p].
↪X}")

# Print the final objective function value (total cost)
print(f"Total cost: {model.ObjVal}")
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (win64 - Windows 11+.0
(22631.2))

CPU model: 12th Gen Intel(R) Core(TM) i5-1240P, instruction set [SSE2|AVX|AVX2]
Thread count: 12 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 250 rows, 2110 columns and 8120 nonzeros
Model fingerprint: 0x346a476e
Variable types: 110 continuous, 2000 integer (2000 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [5e+00, 2e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+02]
Presolve removed 190 rows and 1680 columns
Presolve time: 0.01s
Presolved: 60 rows, 430 columns, 1250 nonzeros
Variable types: 30 continuous, 400 integer (400 binary)
Found heuristic solution: objective 96900.000000
Found heuristic solution: objective 88900.000000
```

```
Root relaxation: objective 7.690000e+04, 284 iterations, 0.01 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0    76900.000000 76900.0000  0.00%     -    0s

Explored 1 nodes (284 simplex iterations) in 0.07 seconds (0.01 work units)
Thread count was 16 (of 16 available processors)

Solution count 3: 76900 88900 96900

Optimal solution found (tolerance 1.00e-04)
Best objective 7.690000000000e+04, best bound 7.690000000000e+04, gap 0.0000%
Surgery 0 is assigned to block 1,  at time 50.0
Surgery 1 is assigned to block 4,  at time 0.0
Surgery 2 is assigned to block 5,  at time 90.0
Surgery 3 is assigned to block 2,  at time 0.0
Surgery 4 is assigned to block 4,  at time 60.0
Surgery 5 is assigned to block 1,  at time 0.0
Surgery 6 is assigned to block 0,  at time 70.0
Surgery 7 is assigned to block 5,  at time 0.0
Surgery 8 is assigned to block 9,  at time 0.0
Surgery 9 is assigned to block 7,  at time 100.0
Surgery 10 is assigned to block 7,  at time 0.0
Surgery 11 is assigned to block 0,  at time 0.0
Surgery 12 is assigned to block 8,  at time 80.0
Surgery 13 is assigned to block 2,  at time 60.0
Surgery 14 is assigned to block 6,  at time 60.0
Surgery 15 is assigned to block 3,  at time 80.0
Surgery 16 is assigned to block 6,  at time 0.0
Surgery 17 is assigned to block 9,  at time 100.0
Surgery 18 is assigned to block 3,  at time 0.0
Surgery 19 is assigned to block 8,  at time 0.0
Total cost: 76900.0
```

[47]:

```
```

[ ]:

```
```