Hospital Scheduling Using Bin Packing Approach

Objective:

The objective of this approach is to efficiently schedule surgeries in hospital operating rooms while considering surgery durations, room availability, and surgery priorities.

Method:

1. **Model Parameters:**
    - Number of surgeries, types, rooms, and days.
    - Operating hours per day.
    - Surgery durations, priorities, and total surgeries of each type.
2. **Model Creation:**
    - Initialize Gurobi model.
    - Define decision variables `x` representing whether a surgery of a particular type is scheduled in a specific room on a given day.
3. **Objective Function:**
    - Maximize the total weighted priority of scheduled surgeries.
4. **Constraints:**
    - Limit the total duration of surgeries in each room on each day to the operating hours.
    - Ensure each surgery is assigned to at most one room on one day.
    - Limit the total number of surgeries of each type scheduled.
5. **Solving the Model:**
    - Optimize the model to find the optimal surgery schedule using gurobi GRB
6. **Output**
    - Outputs a html visualisation via plotly to show the 5 day 3 room schedule, along with missed surgery frequencies.
    - Outputs 3 data frames showing OR utilizaition per day, schedule of surgeries, missed surgery frequencies.

In [ ]:
```
from gurobipy import Model, GRB

# Model paramaters
num_surgeries = 25
num_types = 3  # Assuming types are numbered 1 through 3
num_rooms = 3
num_days = 5
operating_hours = 8

# Surgery Distribution
```

```
surgery_durations = {1: 5, 2: 2, 3: 6}
surgery_priorities = {1: 25, 2: 20, 3: 30}
total_surgeries_of_each_type = {1: 10, 2: 8, 3: 7}

# Model
m = Model("surgery_scheduling")

# Variable
x = m.addVars(num_surgeries, num_types+1, num_rooms, num_days, vtype=GRB.BINARY, name=

factor = 1 # priority score adjustment factor (to be adjusted to alter impact of surge

# Objective: maximize surgeries scheduled * weighted priority
m.setObjective(sum(x[i,j,k,l] * factor * surgery_priorities[j] for i in range(num_surg
                   for j in range(1, num_types+1) for k in range(num_rooms)
                   for l in range(num_days)), GRB.MAXIMIZE)

# Constraints:
#surgery durations cant exceed 8 hours per day per operating room
for k in range(num_rooms):
    for l in range(num_days):
        m.addConstr(sum(x[i,j,k,l] * surgery_durations[j] for i in range(num_surgeries
                        for j in range(1, num_types+1)) <= operating_hours,
                    f"duration_room{k+1}_day{l+1}")

# Surgeries can only be assigned to a maximum of 1 room in 1 day
for i in range(num_surgeries):
    m.addConstr(sum(x[i,j,k,l] for j in range(1, num_types+1) for k in range(num_rooms
                    for l in range(num_days)) <= 1,
                f"assign_surgery{i+1}")

#Ensures that we are selecting from the portfolio of surgeries
for j in range(1, num_types+1):
    m.addConstr(sum(x[i,j,k,l] for i in range(num_surgeries) for k in range(num_rooms)
                    for l in range(num_days)) <= total_surgeries_of_each_type[j],
                f"total_type{j}")

# Solve
m.optimize()

# Post-optimization processing for outputs
scheduled_surgeries = []
missed_surgeries = {j: total_surgeries_of_each_type[j] for j in range(1, num_types+1)}

# Identify scheduled surgeries and update missed surgeries count
for i in range(num_surgeries):
    for j in range(1, num_types+1):
        for k in range(num_rooms):
            for l in range(num_days):
                if x[i,j,k,l].X > 0.5:  # If surgery i of type j is scheduled in room
                    scheduled_surgeries.append((i+1, j, k+1, l+1))
                    missed_surgeries[j] -= 1

# Output scheduled surgeries
print("Scheduled Surgeries:")
for surgery in scheduled_surgeries:
    print(f"Surgery {surgery[0]} of type {surgery[1]} is scheduled in room {surgery[2]

# Output missed surgeries
print("\nMissed Surgeries:")
```

```
for j in missed_surgeries:
    print(f"Missed surgeries of type {j}: {missed_surgeries[j]}")
```

```
Restricted license - for non-production use only - expires 2025-11-24
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 10.0 (19045.2))

CPU model: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 43 rows, 1500 columns and 3375 nonzeros
Model fingerprint: 0x9330af23
Variable types: 0 continuous, 1500 integer (1500 binary)
Coefficient statistics:
  Matrix range     [1e+00, 6e+00]
  Objective range  [2e+01, 3e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+01]
Found heuristic solution: objective 560.0000000
Presolve removed 0 rows and 375 columns
Presolve time: 0.05s
Presolved: 43 rows, 1125 columns, 3375 nonzeros
Variable types: 0 continuous, 1125 integer (1125 binary)
Found heuristic solution: objective 570.0000000

Root relaxation: objective 6.200000e+02, 115 iterations, 0.01 seconds (0.00 work unit
s)

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

     0     0  620.00000    0    9  570.00000  620.00000  8.77%     -    0s
     0     0  620.00000    0   17  570.00000  620.00000  8.77%     -    0s

Cutting planes:
  Cover: 9

Explored 1 nodes (783 simplex iterations) in 0.30 seconds (0.04 work units)
Thread count was 8 (of 8 available processors)

Solution count 2: 570 560

Optimal solution found (tolerance 1.00e-04)
Best objective 5.700000000000e+02, best bound 5.700000000000e+02, gap 0.0000%
Scheduled Surgeries:
Surgery 1 of type 3 is scheduled in room 1 on day 1.
Surgery 2 of type 3 is scheduled in room 1 on day 2.
Surgery 3 of type 1 is scheduled in room 1 on day 3.
Surgery 4 of type 1 is scheduled in room 1 on day 4.
Surgery 5 of type 1 is scheduled in room 1 on day 5.
Surgery 6 of type 1 is scheduled in room 2 on day 1.
Surgery 7 of type 1 is scheduled in room 2 on day 2.
Surgery 8 of type 1 is scheduled in room 2 on day 3.
Surgery 9 of type 1 is scheduled in room 2 on day 4.
Surgery 10 of type 1 is scheduled in room 2 on day 5.
Surgery 11 of type 2 is scheduled in room 1 on day 1.
Surgery 12 of type 2 is scheduled in room 1 on day 2.
Surgery 13 of type 2 is scheduled in room 1 on day 3.
Surgery 14 of type 2 is scheduled in room 1 on day 4.
Surgery 15 of type 2 is scheduled in room 1 on day 5.
Surgery 16 of type 2 is scheduled in room 2 on day 1.
Surgery 17 of type 2 is scheduled in room 2 on day 2.
Surgery 18 of type 2 is scheduled in room 2 on day 3.
Surgery 19 of type 3 is scheduled in room 3 on day 1.
```

```
Surgery 20 of type 3 is scheduled in room 3 on day 2.
Surgery 21 of type 3 is scheduled in room 3 on day 3.
Surgery 22 of type 3 is scheduled in room 3 on day 4.
Surgery 23 of type 3 is scheduled in room 3 on day 5.

Missed Surgeries:
Missed surgeries of type 1: 2
Missed surgeries of type 2: 0
Missed surgeries of type 3: 0
```

In [ ]:
```python
# Define a custom sorting key function
def custom_sort(surgery):
    # Extract relevant information
    surgery_id, type_id, room, day = surgery

    # Get duration and priority of the surgery
    duration = surgery_durations[type_id]
    priority = surgery_priorities[type_id]

    # Return a tuple for sorting
    return (day, duration, priority)

# Sort the scheduled surgeries
scheduled_surgeries_sorted = sorted(scheduled_surgeries, key=custom_sort)

# Scheduled Surgeries Output Transformation with sorting
scheduled_surgeries_data = []
for surgery in scheduled_surgeries_sorted:
    scheduled_surgeries_data.append({
        "Day": surgery[3],
        "Room": surgery[2],
        "Type": surgery[1],
        'Priority': surgery_priorities[ surgery[1]],
        "Surgery ID": surgery[0],
    })

# Missed Surgeries Output Transformation
missed_surgeries_data = [{"Type": j, "Missed Count": missed_surgeries[j]} for j in mis
```

In [ ]:
```python
import pandas as pd

# Convert the data to pandas DataFrame for visualization
df_scheduled = pd.DataFrame(scheduled_surgeries_data)
df_scheduled_summary = df_scheduled.groupby(['Day', 'Room', 'Type']).size().reset_inde

df_missed = pd.DataFrame(missed_surgeries_data)
df_missed["Type"] =df_missed["Type"].astype(str) #for graphing purposes

# Adding a 'Duration' column to df_scheduled based on the surgery type
df_scheduled['Duration'] = df_scheduled['Type'].apply(lambda x: surgery_durations[x])

# Calculate the total duration (utilization) of each OR by day
or_utilization = df_scheduled.groupby(['Room', 'Day'])['Duration'].sum().reset_index(r

# Assuming 8 hours of available operating time per day
or_utilization['Total Hours Available'] = 8
or_utilization['Utilization (%)'] = (or_utilization['Total Hours Used'] / or_utilizati
```

In [ ]:
```python
df_missed
```

Out[ ]:

	Type	Missed Count
0	1	2
1	2	0
2	3	0

In [ ]:
```
df_scheduled
```

Out[ ]:

	Day	Room	Type	Priority	Surgery ID	Duration
0	1	1	2	20	11	2
1	1	2	2	20	16	2
2	1	2	1	25	6	5
3	1	1	3	30	1	6
4	1	3	3	30	19	6
5	2	1	2	20	12	2
6	2	2	2	20	17	2
7	2	2	1	25	7	5
8	2	1	3	30	2	6
9	2	3	3	30	20	6
10	3	1	2	20	13	2
11	3	2	2	20	18	2
12	3	1	1	25	3	5
13	3	2	1	25	8	5
14	3	3	3	30	21	6
15	4	1	2	20	14	2
16	4	1	1	25	4	5
17	4	2	1	25	9	5
18	4	3	3	30	22	6
19	5	1	2	20	15	2
20	5	1	1	25	5	5
21	5	2	1	25	10	5
22	5	3	3	30	23	6

In [ ]:
```
or_utilization
```

Out[ ]:

	Room	Day	Total Hours Used	Total Hours Available	Utilization (%)
0	1	1	8	8	100.0
1	1	2	8	8	100.0
2	1	3	7	8	87.5
3	1	4	7	8	87.5
4	1	5	7	8	87.5
5	2	1	7	8	87.5
6	2	2	7	8	87.5
7	2	3	7	8	87.5
8	2	4	5	8	62.5
9	2	5	5	8	62.5
10	3	1	6	8	75.0
11	3	2	6	8	75.0
12	3	3	6	8	75.0
13	3	4	6	8	75.0
14	3	5	6	8	75.0

In [ ]:

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.offline as pyo

# Create subplots for each operating room and a summary subplot
fig = go.Figure()
# Assuming num_types is the total number of surgery types
num_types = 3

# Add subplots for each operating room
for room in sorted(df_scheduled['Room'].unique()):
    room_data = df_scheduled[df_scheduled['Room'] == room].groupby('Day').agg({'Durati

    # Calculate text for the bar labels
    text = [f"Total: {count}" for count in room_data['Type']]

    hover_text = []
    for day in room_data.index:
        day_data = df_scheduled[(df_scheduled['Room'] == room) & (df_scheduled['Day']
        type_counts = [day_data[day_data['Type'] == i]['Type'].count() for i in range(
        hover_text.append(", ".join([f"Type {i+1}: {count}" for i, count in enumerate(

    fig.add_trace(go.Bar(x=room_data.index, y=room_data['Duration'], name=f'Room {room
                         hoverinfo='text', text=text, hovertext=hover_text, textpositi


# Add summary subplot
summary_data = df_scheduled.groupby('Day').agg({'Duration': 'sum', 'Type': 'size'})
text_summary = [f"Total: {count}" for count in summary_data['Type']]
hover_text_summary = []
```

```
for day in summary_data.index:
    day_data = df_scheduled[df_scheduled['Day'] == day]
    type_counts = [day_data[day_data['Type'] == i]['Type'].count() for i in range(1, r
    hover_text_summary.append(", ".join([f"Type {i+1}: {count}" for i, count in enumer

fig.add_trace(go.Bar(x=summary_data.index, y=summary_data['Duration'], name='Total Hou
                     marker_color='black', opacity=0.3,
                     hoverinfo='text', text=text_summary, hovertext=hover_text_summary

# Update layout
fig.update_layout(title_text='Operating Room Utilization by Day',
                  xaxis_title="Day",
                  yaxis_title="Hours",
                  legend=dict(yanchor="top", y=0.99, xanchor="left", x=0.01),
                  barmode='group',
                  bargap=0.2)

# Create bar chart for missed surgeries by type
fig_missed = px.bar(df_missed, x='Type', y='Missed Count', text='Missed Count',
                    title='Missed Surgeries by Type',
                    labels={'Missed Count': 'Number of Missed Surgeries', 'Type': 'Sur

# Update x-axis category order
fig_missed.update_xaxes(categoryorder='array', categoryarray=[1, 2, 3])

# Update layout
fig_missed.update_layout(xaxis_title="Surgery Type", yaxis_title="Number of Missed Sur

# Save HTML content of both plots separately
html_content_fig = fig.to_html(full_html=False)
html_content_fig_missed = fig_missed.to_html(full_html=False)

# Combine the HTML content into one file
combined_html_content = html_content_fig + html_content_fig_missed

# Save the combined HTML content to a file
with open('operating_room_utilization_combined.html', 'w') as f:
    f.write(combined_html_content)
```