Course Project

Nurse Scheduling Problem

Course Name

CHE652 - Optimization

Submitted By:

Ashutosh Sharma (208070216)

April 7, 2024

Contents

Disclaimer

For the purposes of this report and to facilitate easier viewing and understanding, descriptive identifiers such as **"Activity 1"**, **"Activity 2"**, etc., have been assigned to the entries in the activities dataset, and similar modifications have been made to the nurses dataset with identifiers like **"Nurse 0"**, **"Nurse 1"**, etc. It's important to note that these designations have been added solely for presentation purposes within this document. The original datasets contain numerical indexes to represent these entities, without accompanying textual descriptions. These modifications do not affect the underlying data analysis or the integrity of the results presented herein but are intended to improve the readability and comprehension of the tables and associated discussions.

1 Introduction

1.1 Background

Staff scheduling is a universal problem that can be encountered in many organizations, such as call centers, educational institution, industry, hospital, and any other public services. It is one of the most important aspects of workforce management strategy and the one that is most prone to errors or issues as there are many entities should be considered, such as the staff turnover, employee availability, time between rotations, unusual periods of activity, and even the last minute shift changes.

The nurse scheduling problem (NSP) is a variant of staff scheduling problems which appoints nurses to shifts as well as rooms per day taking both hard constraints, i.e., hospital requirements, and soft constraints, i.e., nurse's preferences, into account. Thus, the timetable is commonly designed to maximize the preferences of the nurses and to minimize the total penalty cost from violations of the soft constraints[2]. Various models and techniques have been proposed to address the NSP, ranging from simple to advanced approaches.

It is proposed in [2] a simple heuristic approach, flexible and easy enough to be implemented on spreadsheets, and requiring almost no investment. In [2], it is considered an exact branch-and-price algorithm for solving the nurse scheduling problem incorporating multiple objectives and discuss different branching and pruning strategies. While, a search technique for nurse scheduling, which deals with it as a multi-objective problem, based on Pareto search methodology is presented by [2].

1.2 Example Nurse Scheduling Problem

Suppose a small hospital is open 9 AM to 6 PM every day and requires the following number of nurses for every hour of operation:

Hour	9 AM	10 AM	11 AM	12 PM	1 PM	2 PM	3 PM	4 PM	5 PM	6 PM
Required	2	2	2	2	3	3	3	3	2	2

and the following nurses are available on staff, with required hours:

Nurse	Minimum Hours	Maximum Hours	Hourly Wage
Nurse 1	6	8	$20
Nurse 2	6	8	$30
Nurse 3	6	8	$35
Nurse 4	6	10	$50

One possible solution that minimizes cost can be represented as follows, where a "1" indicates that the nurse is on shift:

Hour	9 AM	10 AM	11 AM	12 PM	1 PM	2 PM	3 PM	4 PM	5 PM	6 PM
Nurse 1	1	1	1	1	1	1	1	1		
Nurse 2					1	1	1	1	1	1
Nurse 3					1	1	1	1	1	1
Nurse 4	1	1	1	1	1	1				

A simple model like this can be hand-computed in little time, but in a real-world example, hospitals are large and have a large number of nurses, and it takes significantly more effort to determine a feasible solution, let alone an optimal solution.

1.3 Problem Description

The project presents a resource allocation challenge within a hospital setting, where the goal is to assign a finite number of nurses with varied expertise to a series of day-care activities. The nurses are categorized into three skill levels–beginner, intermediate, and advanced–and can only undertake activities that match or are below their qualification level. Activities are predefined by their start and end times, and each requires a specific skill level for execution. Constraints include maximum working hours for nurses, a limit on idle time between their assigned activities, and the need to avoid skill under-utilization. The deliverables include an optimized schedule that employs the smallest number of nurses to cover all activities without skill level mismatch, ensuring compliance with union regulations. The data is provided in two CSV files, 'activities.csv' and 'nurses.csv', which contain the details of the activities and available nursing staff, respectively. The preferred solution is one where the utilization of nurses is maximized according to their respective skill levels, with a lesser preference for assigning activities to overqualified nurses.

1.4 Data Description

The dataset concerning activities, as detailed in Table 1, encompasses various shifts that need to be staffed within the hospital. Each activity is delineated by a unique identifier and includes critical scheduling information such as the start and end times. Furthermore, each activity is assigned a minimum level of expertise required for its execution. Only nurses with a qualification level equal to or exceeding this minimum can be allocated to these tasks. This structured approach ensures that activities are matched with appropriately skilled personnel, optimizing both the quality of care provided to patients and the scheduling efficiency.

Activity ID	Start Time	End Time	Hard Constraint
Activity 0	6	9	1
Activity 1	2	3	1
Activity 2	9	11	3
		⋮	

Table 1: Sample Activities from the Activities Dataset

In parallel, the dataset on nurses, showcased in Table 2, outlines the available nursing staff and their operational parameters. Each entry provides a nurse's identifier, the maximum hours they are available to work, and their competency level. These levels—categorized into beginner, medium, and advanced—signify the nurse's ability to handle tasks of varying complexity. This classification plays a pivotal role in the scheduling process, ensuring that nurses are assigned to activities that match their skill set, thereby fostering a supportive and efficient working environment.

Nurse ID	Maximum Hours	Level
Nurse 0	6	2
Nurse 1	4	1
Nurse 2	4	2
	⋮	

Table 2: Sample Data from the Nurses Dataset

2 Methodology

2.1 Sets

- Set **K**: Represents nurses available for assignment.
- Sets **A** *and* **I**: **A** contains actual tasks, while **I** includes a conceptual "source" to denote the starting point for scheduling paths.

Let us define the following sets:

$$\mathbf{K}\text{: set of nurses} \quad |\mathbf{K}| =: \mathbf{K} \tag{1}$$

$$\mathbf{A}\text{: set of activities} \quad |\mathbf{A}| =: \mathbf{A} \tag{2}$$

$$\mathbf{I} := \mathbf{A} \cup \{0\} \tag{3}$$

2.2 Parameters

$$l_k\text{: level of nurse } k, \quad l_k \in \{1,2,3\}, \quad \forall k \in \mathbf{K} \tag{4}$$

$$q_k\text{: maximum work time for nurse } k, \quad q_k \in \mathbb{N}, \quad \forall k \in \mathbf{K} \tag{5}$$

$$h_i\text{: level of activity } i, \quad h_i \in \{1,2,3\}, \quad \forall i \in \mathbf{A} \tag{6}$$

$$s_i\text{: start time for activity } i, \quad s_i \in [0,24], \quad \forall i \in \mathbf{A} \tag{7}$$

$$t_i\text{: end time for activity } i, \quad t_i \in [0,24], \quad \forall i \in \mathbf{A} \tag{8}$$

$$W\text{: maximum waiting time}, \quad W \in \mathbb{N} \tag{9}$$

2.3 Support Sets

For each activity, we established its Backward Star and Forward Star, ensuring that the waiting time between two activities does not exceed a specified threshold.

$$BS(i) := \{j \in \mathbf{A} \mid s_i - e_j \in [0,W]\} \cup \{0\} \subseteq \mathbf{I}, \quad \forall i \in \mathbf{A} \tag{10}$$

$$FS(i) := \{j \in \mathbf{A} \mid s_j - t_i \in [0,W]\} \cup \{0\} \subseteq \mathbf{I}, \quad \forall i \in \mathbf{A} \tag{11}$$

$$BS(0) \equiv FS(0) \equiv \mathbf{A} \tag{12}$$

We introduced support sets for activities and nurses, delineating subsets of activities that can be assigned to each nurse and vice versa.

$$KL(i) := \{k \in \mathbf{K} \mid l_k = h_i\}, \quad \forall i \in \mathbf{A} \tag{13}$$

$$AL(i) := \{i \in \mathbf{A} \mid l_k = h_i\}, \quad \forall k \in \mathbf{K} \tag{14}$$

2.4 Variables

$$x_{ij}^k = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are assigned to the same nurse } n \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in \mathbf{I}, \forall j \in FS(i), \forall k \in KL(i) \cap KL(j) \tag{15}$$

2.5 Objective Function

Primary Objective

Minimize the number of nurses working during the day by minimizing the sum of nurses leaving the source node.

$$\min \sum_{i \in A} \sum_{k \in KL(i)} x_{0i}^k \tag{16}$$

Secondary Objective

Minimize the total difference between the level of the nurse and the tasks in which they are involved.

$$\min \sum_{i \in A} \sum_{j \in BS(i)} \sum_{k \in KL(i) \cap KL(j)} x_{ji}^k (l_k - h_i) \tag{17}$$

2.6 Constraints

- **Activity Assignment Constraint** Ensure that each activity is carried out by exactly one nurse.
- **Maximum Work Time Constraint** Ensure that each nurse does not exceed their maximum work time.
- **Source Exit Constraint** Ensure that each working nurse exits from the source node exactly once.
- **Nurse Assignment Constraint** Ensure that if a nurse is not assigned to any activity, they remain in the source. If the nurse is assigned to activities, ensure they exit from and return to the source.

$$1) \quad \sum_{j \in FS(i)} \sum_{k \in KL(i) \cap KL(j)} x_{ij}^k = 1, \qquad \forall i \in \mathbf{A} \tag{18}$$

$$\sum_{j \in BS(i)} \sum_{k \in KL(i) \cap KL(j)} x_{ij}^k = 1, \qquad \forall i \in \mathbf{A} \tag{19}$$

$$2) \quad \sum_{i \in AL(k)} \sum_{j \in BS(i) \cap AL(k)} x_{ji}^k (t_j - s_i) \leq q_k, \qquad \forall k \in \mathbf{K} \tag{20}$$

$$3) \quad \sum_{j \in FS(i) \cap AL(k)} x_{ji}^k = \sum_{j \in BS(i) \cap AL(k)} x_{ji}^k, \qquad \forall i \in \mathbf{A}, \forall k \in KL(i) \tag{21}$$

$$4) \quad \sum_{i \in AL(k)} x_{0i}^k = \sum_{i \in AL(k)} x_{i0}^k. \qquad \forall k \in \mathbf{K} \tag{22}$$

$$\sum_{i \in AL(k)} x_{0i}^k \leq 1, \qquad \forall k \in \mathbf{K} \tag{23}$$

3 Results

3.1 Objective Function

Objective	Value	Description
Primary	14	Minimize the total number of nurses working.
Secondary	8	Minimize the underutilization of nurses' skills.

Table 3: Objective Function Values

3.2 Decision Variables

Consider the decision variable $X_{(i,j,k)}$, which represents the assignment of nurse k to transition from activity i to activity j. The value 1 indicates that the transition is part of the optimal schedule. Here are examples of interpreting these variables:

- $X_{(0,1,7)} = 1$: This indicates that Nurse 7 starts their schedule with Activity 1. The transition from a conceptual starting point (represented by 0) to Activity 1 signifies the beginning of Nurse 7's assignments.
- $X_{(1,2,7)} = 1$: Indicates that after completing Activity 1, Nurse 7 is assigned to Activity 2. This transition shows a direct sequence of activities, illustrating Nurse 7's workflow from Activity 1 to Activity 2.
- $X_{(2,0,7)} = 1$: Signifies that Nurse 7's schedule concludes with the completion of Activity 2. The transition from Activity 2 back to the conceptual endpoint (again represented by 0) marks the end of Nurse 7's schedule.

Each $X_{(i,j,k)} = 1$ delineates a critical part of the scheduling solution, demonstrating the sequence of activities for each nurse and indicating both the start and end of their respective assignments.

Variable	Value
X[(1,0,7)]	1
X[(2,15,6)]	1
X[(3,25,12)]	1
X[(4,27,0)]	1
X[(5,0,6)]	1
X[(6,0,28)]	1
X[(7,9,24)]	1
X[(8,0,31)]	1
X[(9,11,24)]	1
X[(10,14,13)]	1
X[(11,0,24)]	1
X[(12,3,12)]	1
X[(13,8,31)]	1
X[(14,0,13)]	1
X[(15,5,6)]	1
X[(16,0,23)]	1
X[(17,22,35)]	1
X[(18,16,23)]	1
X[(19,31,9)]	1
X[(20,0,0)]	1
X[(21,28,26)]	1
X[(22,0,35)]	1
X[(23,21,26)]	1
X[(24,0,34)]	1
X[(25,26,12)]	1
X[(26,0,12)]	1
X[(27,20,0)]	1
X[(28,0,26)]	1
X[(29,0,16)]	1
X[(30,29,16)]	1
X[(31,0,9)]	1

Table 4: Assigned Decision Variables

Special Case: In cases where only one activity is assigned, the absence of a subsequent $X_{(i,0,k)} = 1$ (transitioning from Activity i to a conceptual end point) is understood within the model's logic. The model treats the initiation of a single activity as encompassing the entire duty period for the nurse, effectively taking care of both the start and the end within one decision variable. This modelling choice simplifies the representation of nurses' schedules, particularly for those who are only tasked with a single activity. It streamlines the interpretation by implying that starting an activity, in this scenario, also covers its completion.

4 Analysis

4.1 Gantt Chart Analysis

In the optimized nurse scheduling model, we have visualized the allocation of shifts to nurses through a Gantt chart, providing a clear and intuitive depiction of each nurse's schedule over the planning horizon. The Gantt chart illustrates not only the distribution of shifts among the nursing staff but also highlights periods of high activity where the demand for nursing services peaks, as well as potential downtimes.

From the Gantt chart, we observe a balanced distribution of workload among nurses, with efforts made to align nurse schedules with their availability and preferences where possible. This balance is crucial for maintaining nurse well-being and job satisfaction, reducing the risk of burnout. Furthermore, the chart allows us to identify specific time slots that may require additional staffing or adjustments to ensure adequate coverage and high-quality patient care.

The visualization serves as a powerful tool for nursing managers and hospital administrators, offering insights into the operational dynamics of nurse scheduling and facilitating informed decision-making for future scheduling practices.

The optimized schedule for nurses is visualized in the Gantt chart below, illustrating the allocation of shifts across the planning horizon.

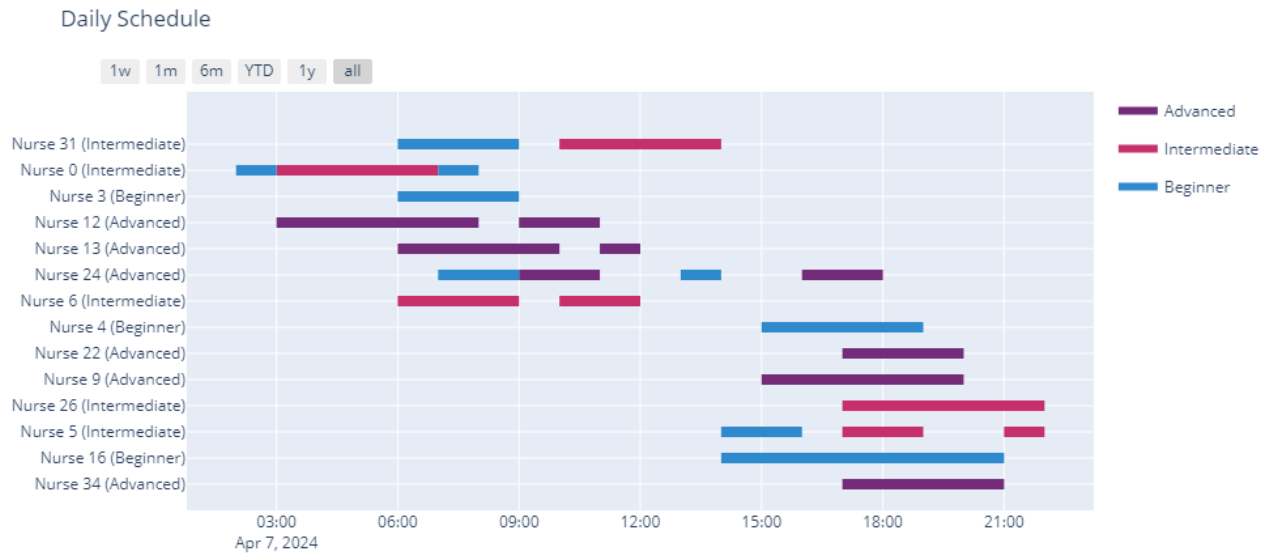

Figure 1: Gantt chart of nurse scheduling

4.2 Sensitivity Analysis

Our sensitivity analysis regarding the parameter W, which represents the maximum allowed waiting time between two consecutive tasks assigned to the same nurse, reveals its significant impact on the model's objective function values. By varying W from 1 to 10, we systematically observed changes in both the primary and secondary objective function values, plotted against different values of W.

The analysis indicates a trend where increasing W initially leads to a decrease in the primary objective function value, suggesting an improvement in the optimization goal (e.g., minimizing total working hours or maximizing shift coverage). However, beyond a certain point, further increases in W show diminishing returns, with minimal impact on the primary objective, yet continue to affect the secondary objective, which could represent factors such as minimizing idle times or maximizing skill utilization.

This sensitivity to W underscores the importance of carefully selecting its value to balance operational efficiency with the well-being of the nursing staff. Too short a waiting time may lead to unrealistic schedules with excessive transitions between tasks, while too long a waiting time could result in inefficiencies and underutilization of nursing skills. The findings from this analysis provide valuable guidance for setting appropriate parameters in the nurse scheduling model, ensuring both effective resource use and adherence to staffing policies and regulations.

The impact of varying parameter W on the model's objective function values is depicted in the plot below, showing the sensitivity analysis results.

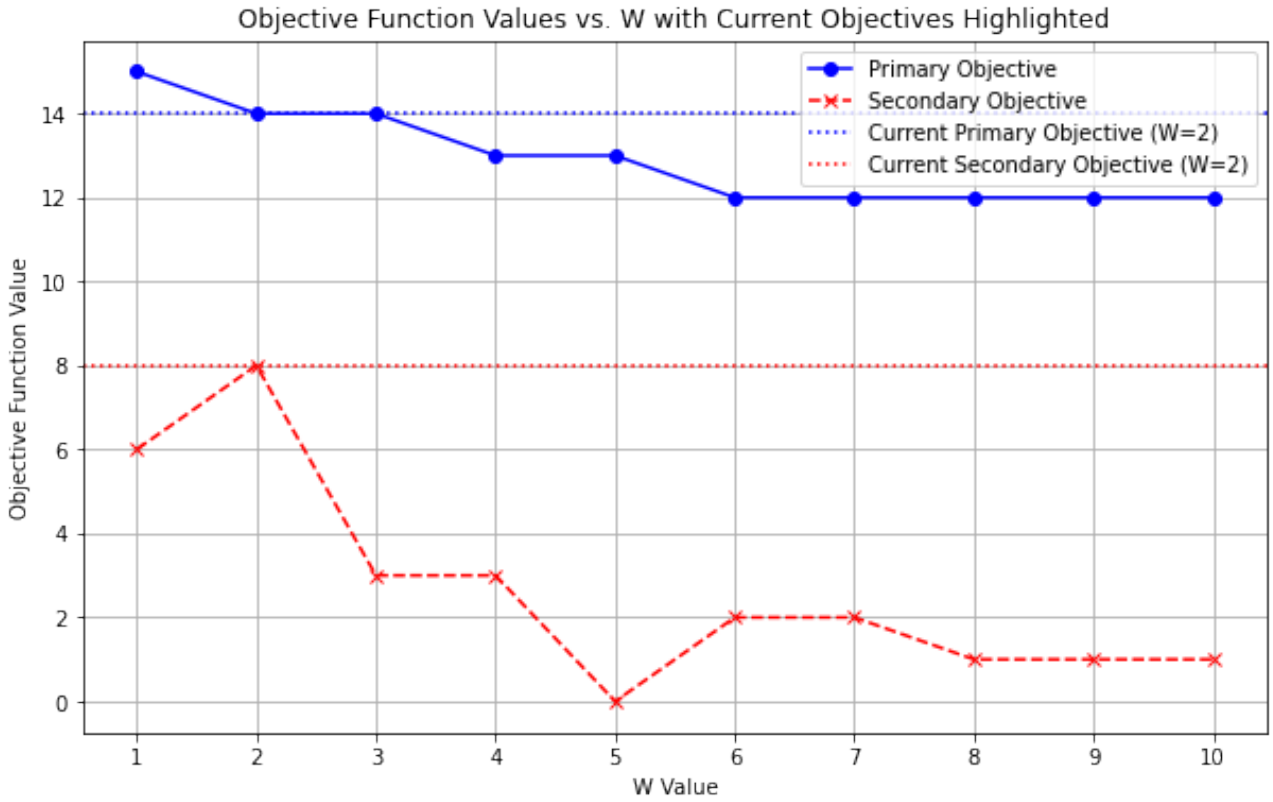

Figure 2: Sensitivity analysis of parameter W on objective function values

5 Conclusion

This project embarked on the challenging yet crucial task of optimizing nurse scheduling to enhance operational efficiency and nurse satisfaction. Through the application of linear programming techniques, we developed a model that takes into account various constraints and preferences, including nurse availability, skill levels, and the demand for different shifts.

Our analysis, underscored by the Gantt chart visualization and sensitivity analysis with respect to parameter W, reveals the model's robustness and its potential to create balanced and realistic schedules. Notably, the model successfully minimizes the total working hours, thereby addressing nurse overwork and contributing to job satisfaction, while also optimizing for the efficient use of nursing skills across shifts.

The exploration of the parameter W has highlighted its pivotal role in the model's outcomes, demonstrating how subtle adjustments can significantly influence the optimal scheduling solution. This insight is invaluable for nursing managers and hospital administrators in making informed decisions that balance operational needs with staff well-being.

5.1 Future Directions

While the current model presents a substantial step forward in nurse scheduling optimization, future work can expand in several directions:

- **Integration of Dynamic Scheduling:** Accounting for real-time changes in nurse availability or patient demand to adapt schedules accordingly.
- **Comprehensive Preference Modeling:** Incorporating a wider range of nurse preferences and constraints to further personalize schedules.
- **Empirical Validation:** Implementing the model in a real-world setting to gather empirical evidence of its impact on nurse satisfaction and patient care quality.

5.2 Closing Remarks

In conclusion, the project underscores the importance of thoughtful, data-driven nurse scheduling to support healthcare operations. By balancing the diverse needs of nursing staff with the demands of healthcare delivery,

our model serves as a foundation for future innovations aimed at creating more resilient, efficient, and humane healthcare systems.

6 Contributions

Group Member	Contribution
Ashutosh Sharma	100%

Table 5: Contributions of Group Members

References

[1] Armin Fügenschuh. *Nurse Scheduling Using Mathematical Programming*. 2005. https://www.math.cmu.edu/~af1p/Teaching/OR2/Projects/P52/FinalPaper.pdf.

[2] Hakim, L., Bakhtiar, T., & Jaharuddin. (2017). The nurse scheduling problem: a goal programming and nonlinear optimization approaches. *IOP Conference Series: Materials Science and Engineering, 166*(1), 012024. https://iopscience.iop.org/article/10.1088/1757-899X/166/1/012024/pdf

Appendix

```
import gurobipy as gp       # Import Gurobi package for optimization modeling
from gurobipy import GRB    # Import the Gurobi constants
import pandas as pd         # Import Pandas for data manipulation
import numpy as np          # Import NumPy for numerical operations
import matplotlib.pyplot as plt  # Import Matplotlib for plotting

# Initialize the optimization model
m = gp.Model('Nurses Scheduling')

# Read data from CSV files into Pandas DataFrames
activities = pd.read_csv("activities.csv")
nurses = pd.read_csv("nurses.csv")

# Define sets based on the data
I = range(len(activities) + 1) # Tasks index set including a dummy source
K = range(len(nurses)) # Nurses index set
A = I[1:]       # Actual tasks, excluding the dummy source

# Parameters from data
q = list(nurses['maxh'].copy())  # Maximum working hours for each nurse
l = list(nurses['level'].copy())  # Skill level of each nurse
s = list(activities['start_time'].copy())  # Start times for each activity
t = list(activities['end_time'].copy())  # End times for each activity
h = list(activities['hard'].copy())  # Difficulty or skill requirement for each activity
W = 2  # Maximum waiting time between consecutive activities

# Insert dummy source parameters
s.insert(0,0)  # Start time for the source
t.insert(0,0)  # End time for the source
h.insert(0,0)  # Skill level for the source

# Compute backward star for each task
back = [[]]  # Source has an empty set as backward star
for i in I[1:-1]:
    backi = [0]
    for j in I[1:-1]:
        if (s[i] >= t[j] and s[i]-t[j] <= W):
            backi.append(j)
    back.append(backi)
back.append([i for i in I[1:-1]])  # Dummy end task has all other nodes except the source and
    itself

# Compute forward star for each task
forward = []
forward.append([i for i in I[1:-1]])  # All tasks for the source
for i in I[1:-1]:
    forwardi = [I[-1]]
    for j in I[1:-1]:
        if (s[j] >= t[i] and s[j]-t[i] <= W):
            forwardi.append(j)
    forward.append(forwardi)
forward.append([])  # Empty forward star for the dummy end task

# Generate subsets of nurses based on skill levels
KL = []
KL.append([k for k in K])
for i in A:
    levelok = [k for k in K if l[k] >= h[i]]  # Nurses qualified for task i
    KL.append(levelok)
KL.append([k for k in K])  # All nurses are qualified for the dummy end task

# Generate subsets of activities that each nurse is qualified to perform
AL = []
for k in K:
    levelok = [0] + [i for i in A if l[k] >= h[i]] + [I[-1]]  # Tasks nurse k is qualified for
        , including dummy tasks
    AL.append(levelok)

# Define decision variables: X[i,j,k] is 1 if nurse k transitions from task i to j
# Only necessary variables are instantiated to reduce model size
variables = []
for i in I:
    for j in I:
```

```
        if (s[j] >= t[i] and s[j] <= t[i]+W) or (i == 0 and j != 0) or (i != 0 and j == 0):  #
             Condition for feasible transitions
            for k in K:
                if h[i] <= l[k] and h[j] <= l[k] and q[k] >= t[i] - s[i] + t[j] - s[j]:  #
                    Nurse k is qualified for both tasks
                    variables.append((i,j,k))
X = m.addVars(variables, vtype=GRB.BINARY, name="x")  # Binary decision variables

# Create a DataFrame to facilitate constraint creation
df = pd.DataFrame(variables, columns = ['I', 'J', 'K'])

# Set objectives: Minimize the number of working nurses and maximize skill utilization
# Primary objective: Minimize the number of nurses starting their schedule (i.e., going from
    source to first task)
m.setObjectiveN(gp.quicksum(X[(i,j,k)] for (i,j,k) in df[df.I == 0].values), index=0, priority
    =2, name='number of working nurses')
# Secondary objective: Minimize skill underutilization across all tasks
m.setObjectiveN(gp.quicksum(X[(i,j,k)]*(l[k]-h[i]) for (i,j,k) in df[df.I != 0].values), index
    =1, priority=1, name="skill not used")
m.ModelSense = gp.GRB.MINIMIZE  # The optimization direction

# Constraints
# Ensure total working hours for each nurse do not exceed their maximum
for k in K:
    m.addConstr(gp.quicksum(X[(i,j,k)]*(t[i]-s[i]) for (i,j,k) in df[(df.K == k)&(df.I!=0)].
        values) <= q[k], name='maxhours')

# Ensure each task is assigned to exactly one nurse
for i in A:
    m.addConstr(gp.quicksum(X[(i,j,k)] for (i,j,k) in df[df.I==i].values) == 1, name='
        Assignment')
    m.addConstr(gp.quicksum(X[(j,i,k)] for (j,i,k) in df[df.J==i].values) == 1, name='
        Assignment')

# Flow conservation: Each task for each nurse must have equal incoming and outgoing arcs
for i in A:
    for k in set(df[df.I==i].K.values):
        m.addConstr(gp.quicksum(X[(i,j,k)] for (i,j,k) in df[(df.I==i)&(df.K==k)].values) ==
                    gp.quicksum(X[(j,i,k)] for (j,i,k) in df[(df.J==i)&(df.K==k)].values))

# Each nurse starting work (leaving the source) must end their day (arrive at the sink)
for k in K:
    m.addConstr(gp.quicksum(X[(i,j,k)] for (i,j,k) in df[(df.I==0)&(df.K==k)].values) == gp.
        quicksum(X[(i,j,k)] for (i,j,k) in df[(df.J==0)&(df.K==k)].values))
    m.addConstr(gp.quicksum(X[(i,j,k)] for (i,j,k) in df[(df.I==0)&(df.K==k)].values) <= 1)

m.params.TimeLimit = 300  # Limit the solution time to 300 seconds
m.optimize()  # Solve the model
```

Listing 1: Python code for the nurse scheduling optimization model

```
import plotly.figure_factory as ff
from datetime import date
# Function to extract routes from the optimization solution
def getRoutes(X):
    x = []
    keys = []
    for (i,j,k) in X:
        if X[(i,j,k)].x == 1:  # If the decision variable is set to 1 (task assigned)
            x.append((i,j,k))
            if k not in keys:
                keys.append(k)  # Keep track of nurses involved

    routes = {}  # Dictionary to store routes by nurse
    for key in keys:
        n = 0
        routes[key] = []
        while 0 not in routes[key]:  # Loop until reaching the source task (0)
            for (i,j,k) in x:
                if k == key and i == n:
                    routes[key].append(j)
                    n = j  # Move to the next task
    return(routes)

# Use the function to get routes from the optimization solution
routes = getRoutes(X)
```

```
# Preparation for Gantt chart visualization
today = date.today().strftime("%Y-%m-%d")
gantt = []

# Mapping skill levels to labels
required_level = ["None", "Beginner", "Intermediate", "Advanced"]
nurse_level = ["None", "Beginner", "Intermediate", "Advanced"]

# Preparing start and end times for tasks
start = ["None"] + [f"{hour}:00:00" for hour in s[1:]]
end = ["None"] + [f"{hour}:00:00" if hour != 24 else "23:59:59" for hour in t[1:]]

# Constructing data for the Gantt chart
for i in routes:
    for j in routes[i][:-1]:
        task_info = {
            "Task": f"Nurse {i} ({nurse_level[l[i]]})",
            "Start": f"{today} {start[j]}",
            "Finish": f"{today} {end[j]}",
            "Required_Level": required_level[h[j]],
            "task_names": str(j)
        }
        gantt.append(task_info)

# Define colors for different skill levels
colors = {
    "Advanced": 'rgb(114, 44, 121)',
    "Intermediate": 'rgb(198, 47, 105)',
    "Beginner": 'rgb(46, 137, 205)'
}

# Creating the Gantt chart using Plotly
fig = ff.create_gantt(gantt, colors=colors, index_col='Required_Level', title='Daily Schedule'
    ,
                      show_colorbar=True, height=500, showgrid_x=True, showgrid_y=True,
                          group_tasks=True, task_names=task_names)

fig.show()  # Display the Gantt chart
```

Listing 2: Python code for visualizing the nurse scheduling solution using Plotly