ESKİŞEHİR TEKNİK ÜNİVERSİTESİ
ESKİŞEHİR TECHNICAL UNIVERSITY

Eskişehir Technical University

Faculty of Engineering

Industrial Engineering

ENM 326 Network Optimization and Algorithms

Project Assignment

Prepared by

Erhan Şimşeker 21355017598

Eren Mete 42742072884

Instructor

Dr. Banu İçmen Erdem, Assistant Professor

May, 2025

TABLE OF CONTENTS

1. PROBLEM

Nurse staff scheduling (Khan and Lewis [1987]). To provide adequate medical service to its constituents at a reasonable cost, hospital administrators must constantly seek ways to hold staff levels as low as possible while maintaining sufficient staffing to provide satisfactory levels of health care. An urban hospital has three departments: the emergency room (department 1), the neonatal intensive care nursery (department 2), and the orthopedics (department 3). The hospital has three work shifts, each with different levels of necessary staffing for nurses. The hospital would like to identify the minimum number of nurses required to meet the following three constraints: (1) the hospital must allocate at least 13, 32, and 22 nurses to the three departments (over all shifts); (2) the hospital must assign at least 26, 24, and 19 nurses to the three shifts (over all departments); and (3) the minimum and maximum number of nurses allocated to each department in a specific shift must satisfy the following limits:

		Department		
		1	2	3
	1	(6, 8)	(11, 12)	(7, 12)
Shift	2	(4, 6)	(11, 12)	(7, 12)
	3	(2, 4)	(10, 12)	(5, 7)

Suggest a method using maximum flows to identify the minimum number of nurses required to satisfy all the constraints. [1]

1.1.What is the question?

a) **Objective**: To determine the minimum number of nurses needed to adequately staff a hospital with 3 departments over 3 work shifts.

b) **Departments**:

   a. Department 1: Emergency Room

   b. Department 2: Neonatal Intensive Care Unit

   c. Department 3: Orthopedics

c) **Shifts**:

   a. Shift 1

   b. Shift 2

   c. Shift 3

d) **Constraints**:

   a. Each department must receive at least a specified number of nurses across all shifts:

      a. Dept 1 $\geq$ 13 nurses

b. Dept 2 $\geq$ 32 nurses

c. Dept 3 $\geq$ 22 nurses

b. Each shift must be staffed with at least a minimum number of nurses (from any department):

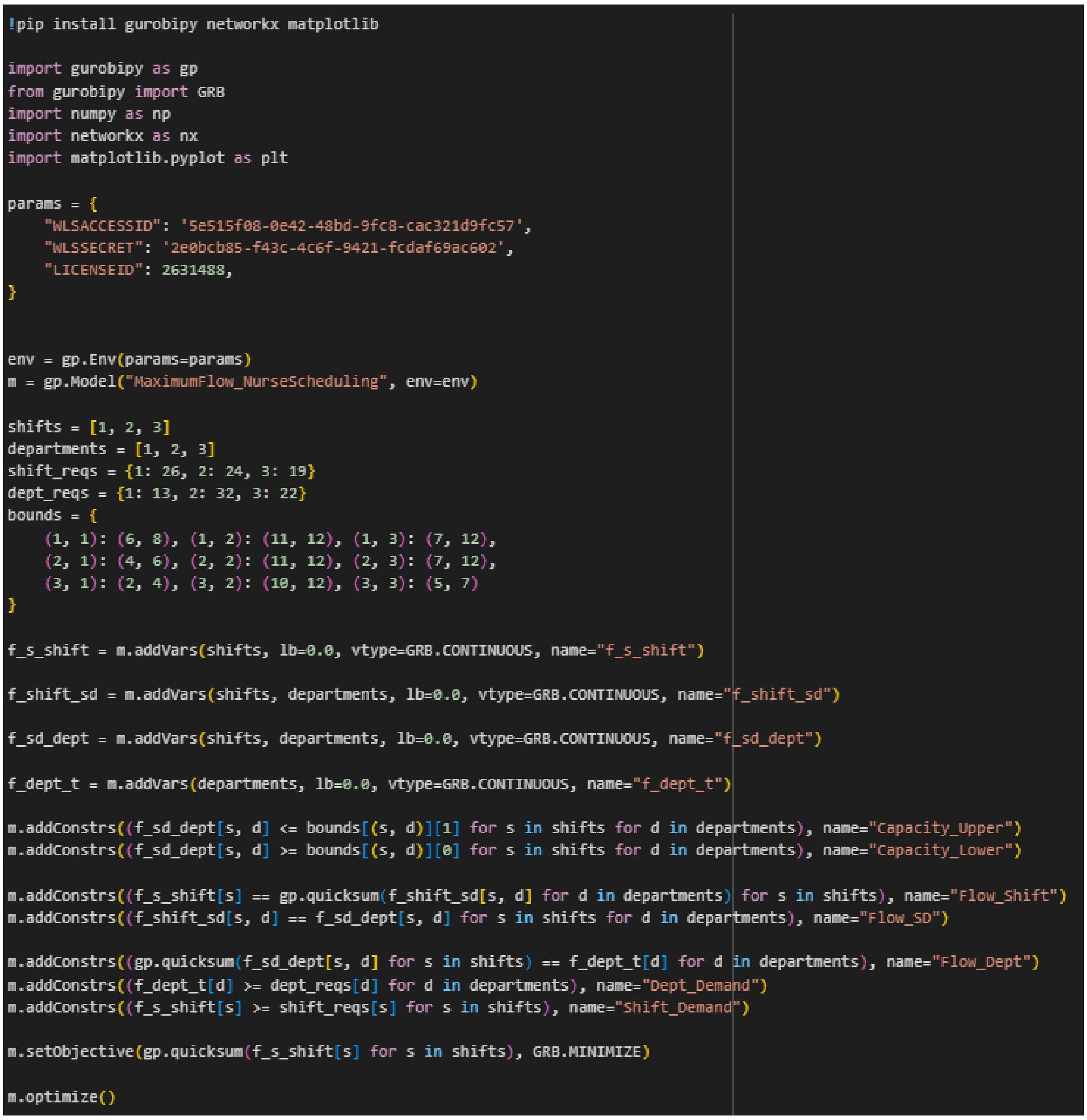
a. Shift 1 $\geq$ 26 nurses

b. Shift 2 $\geq$ 24 nurses

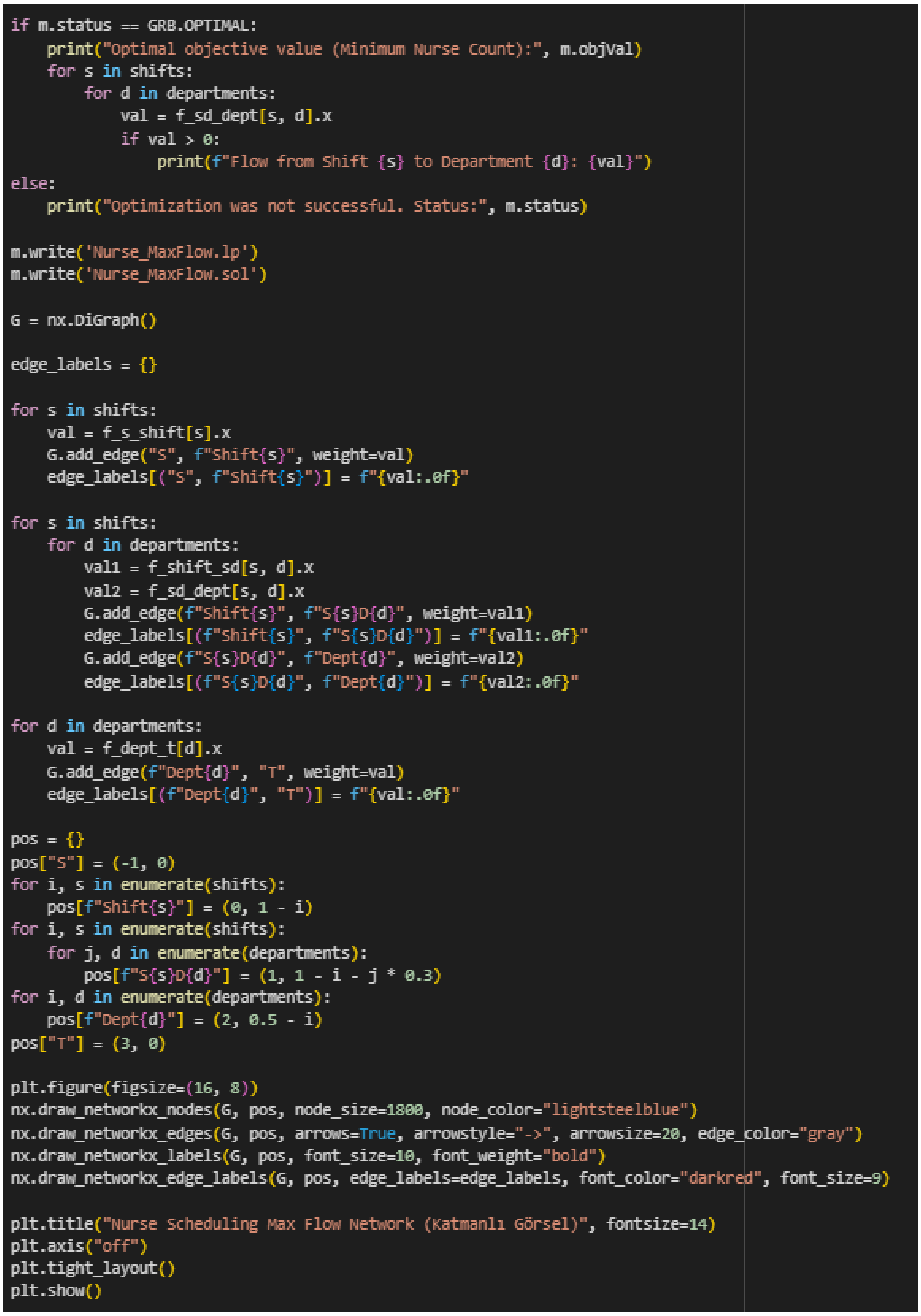
c. Shift 3 $\geq$ 19 nurses

c. Each shift-department combination has a minimum and maximum number of nurses that can be assigned (e.g., Shift 1, Dept 1 must get between 6–8 nurses).

2. COD LINES



```
!pip install gurobipy networkx matplotlib

import gurobipy as gp
from gurobipy import GRB
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

params = {
    "WLSACCESSID": '5e515f08-0e42-48bd-9fc8-cac321d9fc57',
    "WLSSECRET": '2e0bcb85-f43c-4c6f-9421-fcdaf69ac602',
    "LICENSEID": 2631488,
}


env = gp.Env(params=params)
m = gp.Model("MaximumFlow_NurseScheduling", env=env)

shifts = [1, 2, 3]
departments = [1, 2, 3]
shift_reqs = {1: 26, 2: 24, 3: 19}
dept_reqs = {1: 13, 2: 32, 3: 22}
bounds = {
    (1, 1): (6, 8), (1, 2): (11, 12), (1, 3): (7, 12),
    (2, 1): (4, 6), (2, 2): (11, 12), (2, 3): (7, 12),
    (3, 1): (2, 4), (3, 2): (10, 12), (3, 3): (5, 7)
}

f_s_shift = m.addVars(shifts, lb=0.0, vtype=GRB.CONTINUOUS, name="f_s_shift")

f_shift_sd = m.addVars(shifts, departments, lb=0.0, vtype=GRB.CONTINUOUS, name="f_shift_sd")

f_sd_dept = m.addVars(shifts, departments, lb=0.0, vtype=GRB.CONTINUOUS, name="f_sd_dept")

f_dept_t = m.addVars(departments, lb=0.0, vtype=GRB.CONTINUOUS, name="f_dept_t")

m.addConstrs((f_sd_dept[s, d] <= bounds[(s, d)][1] for s in shifts for d in departments), name="Capacity_Upper")
m.addConstrs((f_sd_dept[s, d] >= bounds[(s, d)][0] for s in shifts for d in departments), name="Capacity_Lower")

m.addConstrs((f_s_shift[s] == gp.quicksum(f_shift_sd[s, d] for d in departments) for s in shifts), name="Flow_Shift")
m.addConstrs((f_shift_sd[s, d] == f_sd_dept[s, d] for s in shifts for d in departments), name="Flow_SD")

m.addConstrs((gp.quicksum(f_sd_dept[s, d] for s in shifts) == f_dept_t[d] for d in departments), name="Flow_Dept")
m.addConstrs((f_dept_t[d] >= dept_reqs[d] for d in departments), name="Dept_Demand")
m.addConstrs((f_s_shift[s] >= shift_reqs[s] for s in shifts), name="Shift_Demand")

m.setObjective(gp.quicksum(f_s_shift[s] for s in shifts), GRB.MINIMIZE)

m.optimize()
```



```python
if m.status == GRB.OPTIMAL:
    print("Optimal objective value (Minimum Nurse Count):", m.objVal)
    for s in shifts:
        for d in departments:
            val = f_sd_dept[s, d].x
            if val > 0:
                print(f"Flow from Shift {s} to Department {d}: {val}")
else:
    print("Optimization was not successful. Status:", m.status)

m.write('Nurse_MaxFlow.lp')
m.write('Nurse_MaxFlow.sol')

G = nx.DiGraph()

edge_labels = {}

for s in shifts:
    val = f_s_shift[s].x
    G.add_edge("S", f"Shift{s}", weight=val)
    edge_labels[("S", f"Shift{s}")] = f"{val:.0f}"

for s in shifts:
    for d in departments:
        val1 = f_shift_sd[s, d].x
        val2 = f_sd_dept[s, d].x
        G.add_edge(f"Shift{s}", f"S{s}D{d}", weight=val1)
        edge_labels[(f"Shift{s}", f"S{s}D{d}")] = f"{val1:.0f}"
        G.add_edge(f"S{s}D{d}", f"Dept{d}", weight=val2)
        edge_labels[(f"S{s}D{d}", f"Dept{d}")] = f"{val2:.0f}"

for d in departments:
    val = f_dept_t[d].x
    G.add_edge(f"Dept{d}", "T", weight=val)
    edge_labels[(f"Dept{d}", "T")] = f"{val:.0f}"

pos = {}
pos["S"] = (-1, 0)
for i, s in enumerate(shifts):
    pos[f"Shift{s}"] = (0, 1 - i)
for i, s in enumerate(shifts):
    for j, d in enumerate(departments):
        pos[f"S{s}D{d}"] = (1, 1 - i - j * 0.3)
for i, d in enumerate(departments):
    pos[f"Dept{d}"] = (2, 0.5 - i)
pos["T"] = (3, 0)

plt.figure(figsize=(16, 8))
nx.draw_networkx_nodes(G, pos, node_size=1800, node_color="lightsteelblue")
nx.draw_networkx_edges(G, pos, arrows=True, arrowstyle="->", arrowsize=20, edge_color="gray")
nx.draw_networkx_labels(G, pos, font_size=10, font_weight="bold")
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color="darkred", font_size=9)

plt.title("Nurse Scheduling Max Flow Network (Katmanlı Görsel)", fontsize=14)
plt.axis("off")
plt.tight_layout()
plt.show()
```

2.1. Explanation of the Code

```
!pip install gurobipy networkx matplotlib
```

Installs required Python libraries:

a) gurobipy: For mathematical optimization with Gurobi

b) networkx: For graph structure and flow modeling

c) matplotlib: For plotting the network graph

2.1.1. Module Installation and Import

```
import gurobipy as gp
from gurobipy import GRB
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
```

Imports the necessary modules for optimization, matrix handling, graph modeling, and plotting.

2.1.2. Gurobi License Parameters and Model Initialization

```
params = {
    "WLSACCESSID": '5e515f08-0e42-48bd-9fc8-cac321d9fc57',
    "WLSSECRET": '2e0bcb85-f43c-4c6f-9421-fcdaf69ac602',
    "LICENSEID": 2631488,
}


env = gp.Env(params=params)
m = gp.Model("MaximumFlow_NurseScheduling", env=env)
```

Sets Gurobi cloud license credentials and initializes a new model named "MaximumFlow_NurseScheduling".

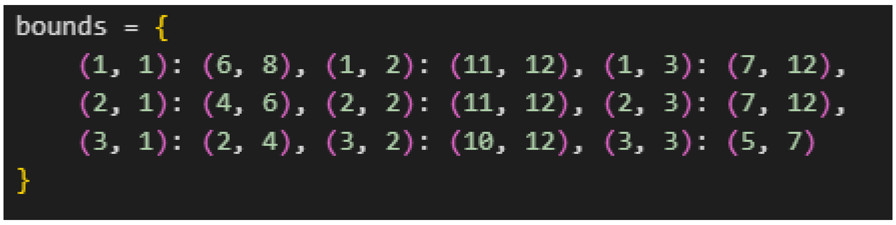
2.1.3. Problem Data

```
shifts = [1, 2, 3]
departments = [1, 2, 3]
```

Defines three work shifts and three hospital departments.

```
shift_reqs = {1: 26, 2: 24, 3: 19}
dept_reqs = {1: 13, 2: 32, 3: 22}
```

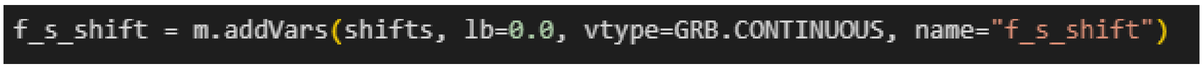
Minimum number of nurses required per shift and department.



```
bounds = {
    (1, 1): (6, 8), (1, 2): (11, 12), (1, 3): (7, 12),
    (2, 1): (4, 6), (2, 2): (11, 12), (2, 3): (7, 12),
    (3, 1): (2, 4), (3, 2): (10, 12), (3, 3): (5, 7)
}
```

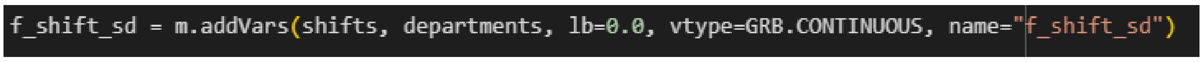
Specifies min and max nurse allocation bounds for each shift-department combination.

2.1.4. Decision Variables (Flow Variables)



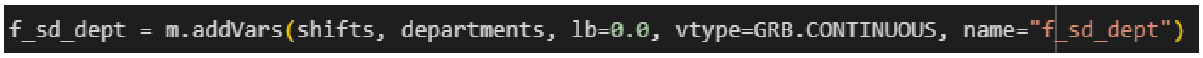
```
f_s_shift = m.addVars(shifts, lb=0.0, vtype=GRB.CONTINUOUS, name="f_s_shift")
```

Flow from source S to each shift node (total nurses per shift).



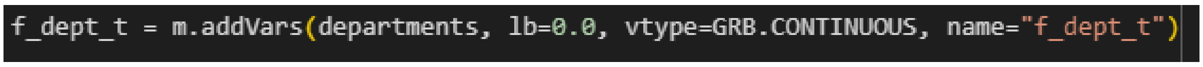
```
f_shift_sd = m.addVars(shifts, departments, lb=0.0, vtype=GRB.CONTINUOUS, name="f_shift_sd")
```

Flow from shift node to intermediate shift-department node (raw allocation).



```
f_sd_dept = m.addVars(shifts, departments, lb=0.0, vtype=GRB.CONTINUOUS, name="f_sd_dept")
```

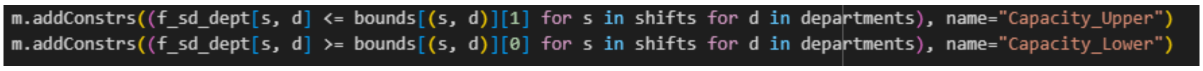
Flow from shift-department node to department (actual allocation, with bounds).



```
f_dept_t = m.addVars(departments, lb=0.0, vtype=GRB.CONTINUOUS, name="f_dept_t")
```

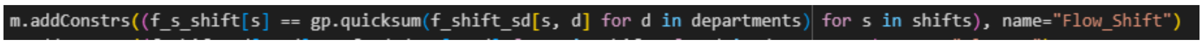
Flow from each department node to the sink node T.

2.1.5. Constraints



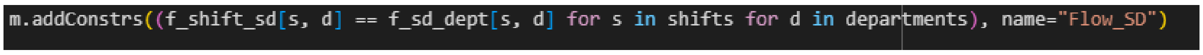
```
m.addConstrs((f_sd_dept[s, d] <= bounds[(s, d)][1] for s in shifts for d in departments), name="Capacity_Upper")
m.addConstrs((f_sd_dept[s, d] >= bounds[(s, d)][0] for s in shifts for d in departments), name="Capacity_Lower")
```

Each shift-department allocation must be within the specified min/max bounds.



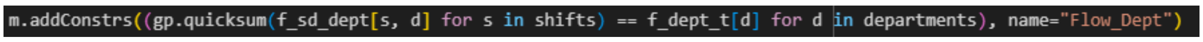
```
m.addConstrs((f_s_shift[s] == gp.quicksum(f_shift_sd[s, d] for d in departments) for s in shifts), name="Flow_Shift")
```

Ensures flow conservation: Total flow from source to a shift equals total flow out from that shift to all departments.



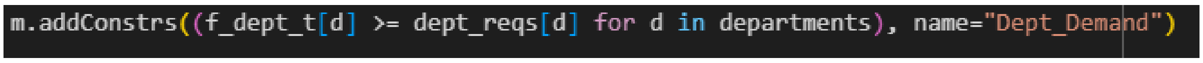
```
m.addConstrs((f_shift_sd[s, d] == f_sd_dept[s, d] for s in shifts for d in departments), name="Flow_SD")
```

Intermediate node just passes flow: output from shift equals input to department.



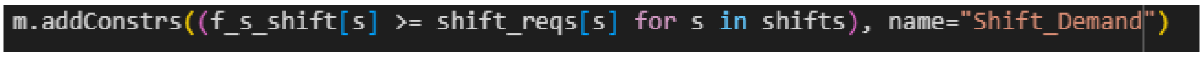
```
m.addConstrs((gp.quicksum(f_sd_dept[s, d] for s in shifts) == f_dept_t[d] for d in departments), name="Flow_Dept")
```

All flow into a department equals its output to the sink.



```
m.addConstrs((f_dept_t[d] >= dept_reqs[d] for d in departments), name="Dept_Demand")
```

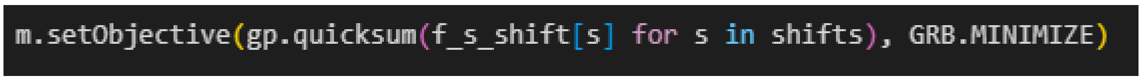
Each department must receive at least its minimum required number of nurses.



```
m.addConstrs((f_s_shift[s] >= shift_reqs[s] for s in shifts), name="Shift_Demand")
```

It’s ensured that each shift receives at least the minimum number of nurses required.

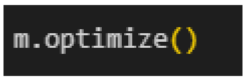
2.1.6. Objective Function



```
m.setObjective(gp.quicksum(f_s_shift[s] for s in shifts), GRB.MINIMIZE)
```

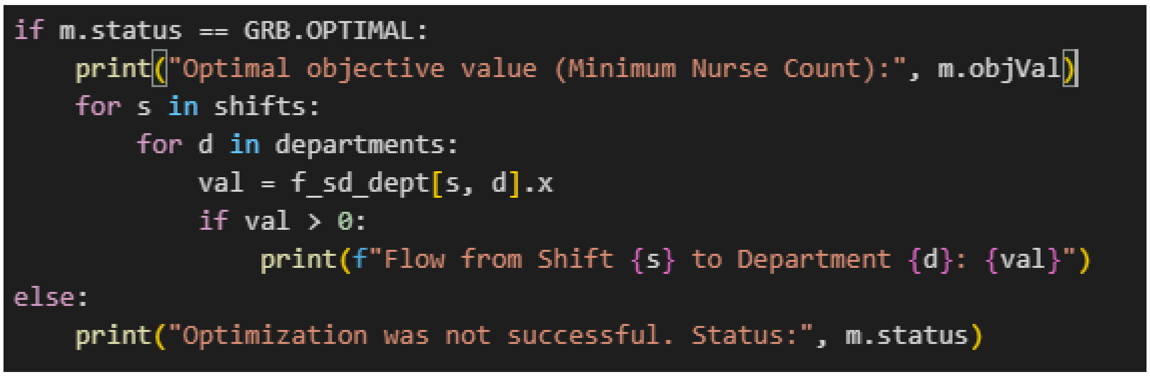
Minimize the total number of nurses sent from the source across all shifts.

2.1.7. Solve and Print Results



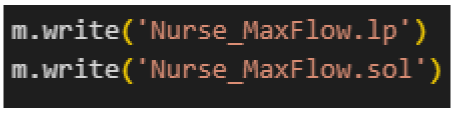
```
m.optimize()
```

Solves the optimization model.



```
if m.status == GRB.OPTIMAL:
    print("Optimal objective value (Minimum Nurse Count):", m.objVal)
    for s in shifts:
        for d in departments:
            val = f_sd_dept[s, d].x
            if val > 0:
                print(f"Flow from Shift {s} to Department {d}: {val}")
else:
    print("Optimization was not successful. Status:", m.status)
```

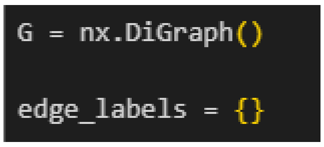
Checks if the optimal solution was found and prints each shift-to-department flow.



```
m.write('Nurse_MaxFlow.lp')
m.write('Nurse_MaxFlow.sol')
```

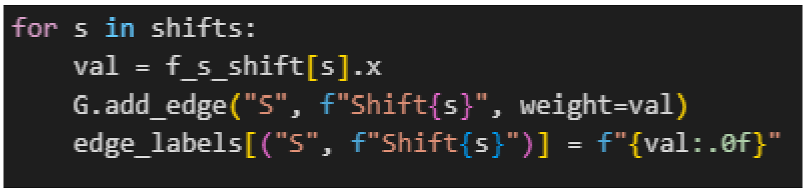
Writes the model and solution to .lp and .sol files.

2.1.8. Network Graph Visualization



```
G = nx.DiGraph()

edge_labels = {}
```

Initializes a directed graph and a dictionary to store edge labels (flow values).



```
for s in shifts:
    val = f_s_shift[s].x
    G.add_edge("S", f"Shift{s}", weight=val)
    edge_labels[("S", f"Shift{s}")] = f"{val:.0f}"
```

Adds edges from S to each Shift{s} with flow values as weights.

```
for s in shifts:
    for d in departments:
        val1 = f_shift_sd[s, d].x
        val2 = f_sd_dept[s, d].x
        G.add_edge(f"Shift{s}", f"S{s}D{d}", weight=val1)
        edge_labels[(f"Shift{s}", f"S{s}D{d}")] = f"{val1:.0f}"
        G.add_edge(f"S{s}D{d}", f"Dept{d}", weight=val2)
        edge_labels[(f"S{s}D{d}", f"Dept{d}")] = f"{val2:.0f}"
```

Adds edges from Shift{s} → S{s}D{d} and from S{s}D{d} → Dept{d}, storing both parts of the flow.

```
for d in departments:
    val = f_dept_t[d].x
    G.add_edge(f"Dept{d}", "T", weight=val)
    edge_labels[(f"Dept{d}", "T")] = f"{val:.0f}"
```

Adds edges from each Dept{d} to terminal node T.

2.1.9. Manual Node Positioning

```
pos = {}
pos["S"] = (-1, 0)
for i, s in enumerate(shifts):
    pos[f"Shift{s}"] = (0, 1 - i)
for i, s in enumerate(shifts):
    for j, d in enumerate(departments):
        pos[f"S{s}D{d}"] = (1, 1 - i - j * 0.3)
for i, d in enumerate(departments):
    pos[f"Dept{d}"] = (2, 0.5 - i)
pos["T"] = (3, 0)
```

Manually sets x/y coordinates for each node to draw the network in layered, readable form.

2.1.10. Drawing the Graph

```
plt.figure(figsize=(16, 8))
nx.draw_networkx_nodes(G, pos, node_size=1800, node_color="lightsteelblue")
nx.draw_networkx_edges(G, pos, arrows=True, arrowstyle="->", arrowsize=20, edge_color="gray")
nx.draw_networkx_labels(G, pos, font_size=10, font_weight="bold")
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color="darkred", font_size=9)
```

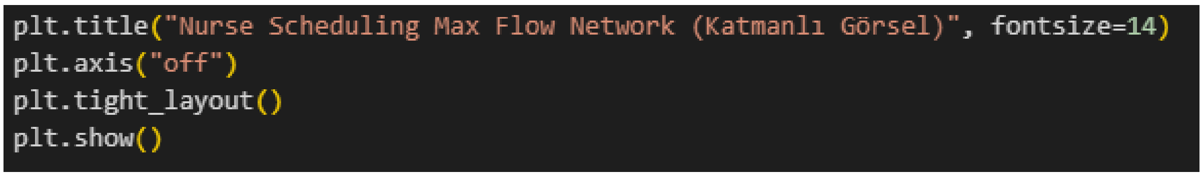
Draws the nodes, edges, labels, and edge weights with stylistic enhancements:

a) Nodes in light blue

b) Edge labels in red

c) Arrows between layers

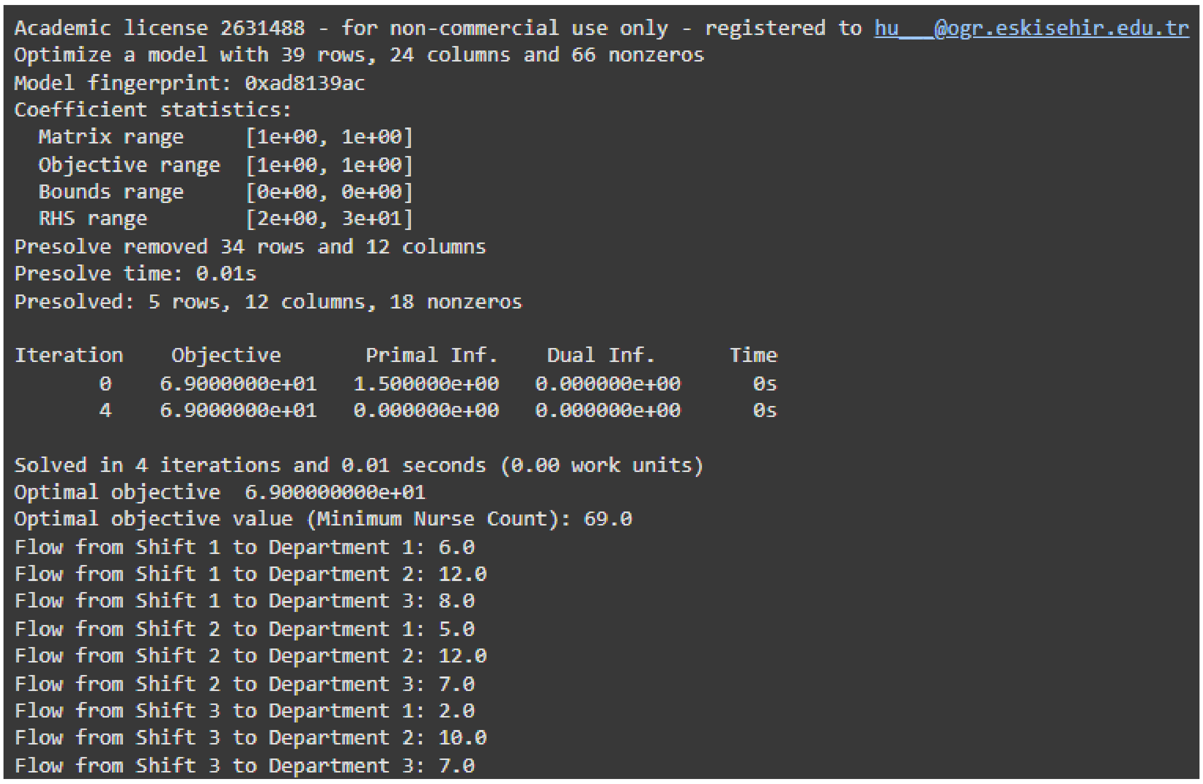
d) Proper node sizes for readability



```
plt.title("Nurse Scheduling Max Flow Network (Katmanlı Görsel)", fontsize=14)
plt.axis("off")
plt.tight_layout()
plt.show()
```

Adds a title, removes axis ticks, and displays the final layout.

3. OUTPUT



```
Academic license 2631488 - for non-commercial use only - registered to hu___@ogr.eskisehir.edu.tr
Optimize a model with 39 rows, 24 columns and 66 nonzeros
Model fingerprint: 0xad8139ac
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+00, 3e+01]
Presolve removed 34 rows and 12 columns
Presolve time: 0.01s
Presolved: 5 rows, 12 columns, 18 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    6.9000000e+01   1.500000e+00   0.000000e+00      0s
       4    6.9000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.01 seconds (0.00 work units)
Optimal objective  6.900000000e+01
Optimal objective value (Minimum Nurse Count): 69.0
Flow from Shift 1 to Department 1: 6.0
Flow from Shift 1 to Department 2: 12.0
Flow from Shift 1 to Department 3: 8.0
Flow from Shift 2 to Department 1: 5.0
Flow from Shift 2 to Department 2: 12.0
Flow from Shift 2 to Department 3: 7.0
Flow from Shift 3 to Department 1: 2.0
Flow from Shift 3 to Department 2: 10.0
Flow from Shift 3 to Department 3: 7.0
```

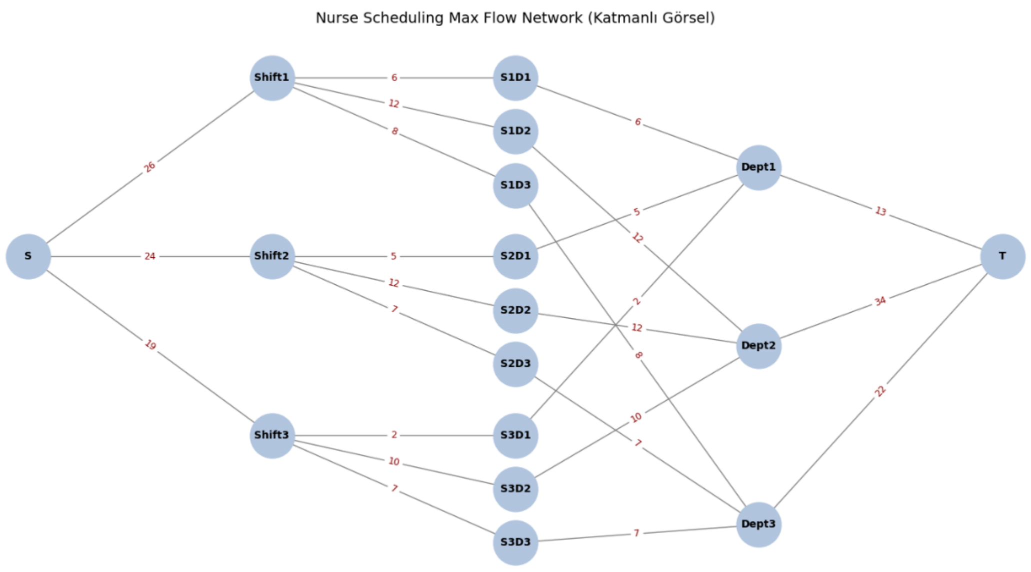
The Gurobi optimization model successfully solved the nurse scheduling problem formulated as a maximum flow network. The objective was to minimize the total number of nurses allocated across all shifts while satisfying both shift-specific and department-specific requirements. The optimal solution found indicates that a total of **69 nurses** are required to meet all constraints, achieving the most efficient staffing configuration.

Each shift's total nurse allocation matches its respective lower bound requirement, ensuring compliance with minimum staffing needs. Specifically, **Shift 1** is assigned 26 nurses, **Shift 2** receives 24 nurses, and **Shift 3** is assigned 19 nurses. These values align exactly with the predefined lower bounds of 26, 24, and 19, respectively.

The nurses from each shift are distributed among three departments. For example, Shift 1 sends 6 nurses to Department 1, 12 to Department 2, and 8 to Department 3. Shift 2 sends 5, 12, and 7 nurses to Departments 1, 2, and 3, respectively. Similarly, Shift 3 distributes 2 nurses to Department 1, 10 to Department 2, and 7 to Department 3. All flows respect the defined upper and lower capacity bounds for each shift-to-department pair.

All departments receive at least the minimum number of nurses required. Department 1 receives exactly 13 nurses, which meets its requirement. Department 2 receives 34 nurses, exceeding the minimum requirement of 32. Department 3 receives 22 nurses, matching its exact requirement. This confirms that department-level constraints are satisfied in the optimal solution.

4. MAX FLOW NETWORK OF THE SOLUTION



The diagram shows a structured nurse assignment from source to departments through shifts, ensuring that each shift receives exactly the minimum number of required nurses: 26 for Shift1, 24 for Shift2, and 19 for Shift3. Nurses are then distributed from each shift to three departments through intermediate layers, maintaining flow constraints between shift-department pairs. Department demands are fully satisfied, with Dept1 receiving 13 nurses, Dept2 receiving 34 (slightly above its requirement), and Dept3 receiving 22 as needed. The network reflects a balanced and optimized allocation that minimizes total nurse usage while meeting all staffing requirements.

5. REFERENCES

[1] Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Theory, algorithms, and applications*. Prentice Hall. Retrieved from https://www.dl.behinehyab.com/Ebooks/NETWORK/NET005_354338_www.behinehyab.com.pdf