ESKİŞEHİR TEKNİK ÜNİVERSİTESİ

ESKİŞEHİR TECHNICAL UNIVERSITY

Eskişehir Teknik Üniversitesi

Mühendislik Fakültesi

Endüstri Mühendisliği

ENM 326 Network Optimization and Algorithms

(Ağ Optimizasyonu ve Algoritmaları)

Proje Ödevi

Hazırlayanlar

Erhan Şimşeker

Eren Mete

Öğretim Görevlisi

Dr. Öğr. Üyesi Banu İçmen Erdem,

Mayıs, 2025

İÇİNDEKİLER

1. PROBLEM

Hemşire personel planlaması (Khan ve Lewis [1987]). Hastane yöneticileri, uygun maliyetle yeterli sağlık hizmeti sunabilmek için, personel seviyelerini mümkün olduğunca düşük tutmanın yollarını sürekli olarak aramak zorundadır; bunu yaparken aynı zamanda sağlık hizmetinin tatmin edici düzeyde sürdürülebilmesi için yeterli personel bulundurulmalıdır. Kentsel bir hastanede üç bölüm bulunmaktadır: acil servis (1. bölüm), yenidoğan yoğun bakım ünitesi (2. bölüm) ve ortopedi (3. bölüm). Hastanede hemşireler için gerekli personel seviyeleri farklı olan üç çalışma vardiyası vardır. Hastane, aşağıdaki üç kısıtı karşılayacak şekilde gerekli minimum hemşire sayısını belirlemek istemektedir: (1) hastane, üç bölüme (tüm vardiyalar boyunca) en az 13, 32 ve 22 hemşire tahsis etmelidir; (2) hastane, üç vardiyaya (tüm bölümler boyunca) en az 26, 24 ve 19 hemşire atamalıdır; ve (3) belirli bir vardiyada her bir bölüme tahsis edilen hemşire sayısı, aşağıdaki sınırları sağlamalıdır.

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

Tüm kısıtlamaları karşılamak için gereken minimum hemşire sayısını belirlemek amacıyla maksimum akışları kullanarak bir yöntem önerin. \[1]

1.1. Soru Nedir?

a) **Amaç**: Üç çalışma vardiyası boyunca üç departmana yeterli sayıda hemşire atamak için gereken minimum hemşire sayısını belirlemek.

b) **Departmanlar**:

   a. Departman 1: Acil Servis

   b. Departman 2: Yenidoğan Yoğun Bakım Ünitesi

   c. Departman 3: Ortopedi

c) **Vardiyalar**:

   a. Vardiya 1

   b. Vardiya 2

   c. Vardiya 3

d) **Kısıtlar**:

   a. Her departmana, tüm vardiyalar boyunca en az belirli sayıda hemşire tahsis edilmelidir:

      a. Dep 1 ≥ 13 hemşire

b. Dep 2 ≥ 32 hemşire

c. Dep 3 ≥ 22 hemşire

b. Her vardiya, (herhangi bir departmandan) en az belirli sayıda hemşire ile doantılmalıdır:

a. Vardiya 1 ≥ 26 hemşire

b. Vardiya 2 ≥ 24 hemşire

c. Vardiya 3 ≥ 19 hemşire

c. Her vardiya-departman kombinasyonuna atanabilecek minimum ve maksimum hemşire sayısı vardır (örneğin, 1. vardiya, 1. departmana 6–8 hemşire atanmalıdır).

2. KOD SATIRLARI

```
!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. Kodun Açıklaması

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

Gerekli Python kütüphanelerinin kurulumu:

a) gurobipy: Gurobi ile matematiksel optimizasyon için

b) networkx: Grafik yapısı ve akış modellemesi için

c) matplotlib: Ağ grafiğinin çizimi için

2.1.1. Modül Kurulumu ve İçe Aktarma

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

Optimizasyon, matris işlemleri, grafik modelleme ve çizim için gerekli modüllerin içe aktartılması.

2.1.2. Gurobi Lisans Parametreleri ve Model Başlatma

```
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)
```

Gurobi bulut lisans kimlik bilgilerini ayarlar ve “MaximumFlow_NurseScheduling” adlı yeni bir model başlatır.

2.1.3. Problem Verileri

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

Üç çalışma vardiyasını ve üç hastane departmanını tanımlanır.

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

Her vardiya ve departman için gereken minimum hemşire sayısı tanımlanır.

```
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)
}
```

Her vardiya-departman kombinasyonu için minimum ve maksimum hemşire tahsis sınırlarını belirtir

2.1.4. Karar Değişkenleri (Akış Değişkenleri)

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

Kaynak S'den her vardiya düğümüne olan akış (her vardiyadaki toplam hemşire sayısı).

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

Vardiya düğümünden ara vardiya-bölüm düğümüne olan akış (ham dağılım).

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

Vardiya-bölüm düğümünden bölüme olan akış (gerçek dağılım, sınırlar dahilinde).

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

Her bölüm düğümünden hedef düğüm T'ye olan akış.

2.1.5. Kısıtlar

```
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")
```

Her vardiya-bölüm tahsisi belirtilen minimum/maksimum sınırlar içinde olmalıdır.

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

Akış korunumu sağlanır: Kaynaktan bir vardiyaya olan toplam akış, o vardiyadan tüm bölümlere olan toplam akışa eşittir.

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

Ara düğüm sadece akışı iletir: vardiyadan çıkan akış, bölüme giren akışa eşittir.

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

Bir bölüme giren tüm akış, o bölümden çıkış olarak hedef düğüme (sink) eşittir.

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

Her bölüm, kendisi için belirlenmiş minimum hemşire sayısını en az almalıdır.

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

Her vardiyanın, gerekli olan minimum hemşire sayısını en az alması sağlanır.

2.1.6. Amaç Fonksiyonu

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

Kaynaktan tüm vardiyalar üzerinden gönderilen toplam hemşire sayısını en aza indirir.he total number of nurses sent from the source across all shifts.

2.1.7. Çözüm ve Sonuçların Yazdırılması

```
m.optimize()
```

Optimazsyon modelini çözer.

```
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)
```

Optimal çözüm bulunup bulunmadığını kontrol eder ve her vardiya ile departman arasındaki akışı yazdırır.

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

Modeli ve çözümü .lp ve .sol dosyaların yazar.

2.1.8. Ağ Grafiği Görselleştirmesi

```
G = nx.DiGraph()

edge_labels = {}
```

Yönlendirilmiş bir grafik ve kenar etiketlerini (akış değerlerini) saklamak için bir sözlük başlatır.

```
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}"
```

S'den her Shift{s}'e akış değerleri ağırlık olarak eklenir.

```
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}"
```

Shift{s} → S{s}D{d} ve S{s}D{d} → Dept{d} kenarları eklenir, akışın her iki kısmı da saklanır.

```
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}"
```

Her Dept{d} düğümünden terminal düğüm T'ye kenarlar eklenir.

2.1.9. Düğümlerin Manuel Konumlandırılması

```
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)
```

Ağı katmanlı ve okunabilir bir şekilde çizmek için her düğümün x/y koordinatları elle belirlenir.

2.1.10. Grafiği Çizme

```
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)
```

Düğümleri, kenarları, etiketleri ve kenar ağırlıklarını stilize ederek çizer:

a) Düğümler açık mavi renkte

b) Kenar etiketleri kırmızı renkte

c) Katmanlar arası oklar

d) Okunabilirlik için uygun düğüm boyutları

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

Başlık ekler, eksen işaretlerini kaldırır ve son düzenlemeyi gösterir.

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
```

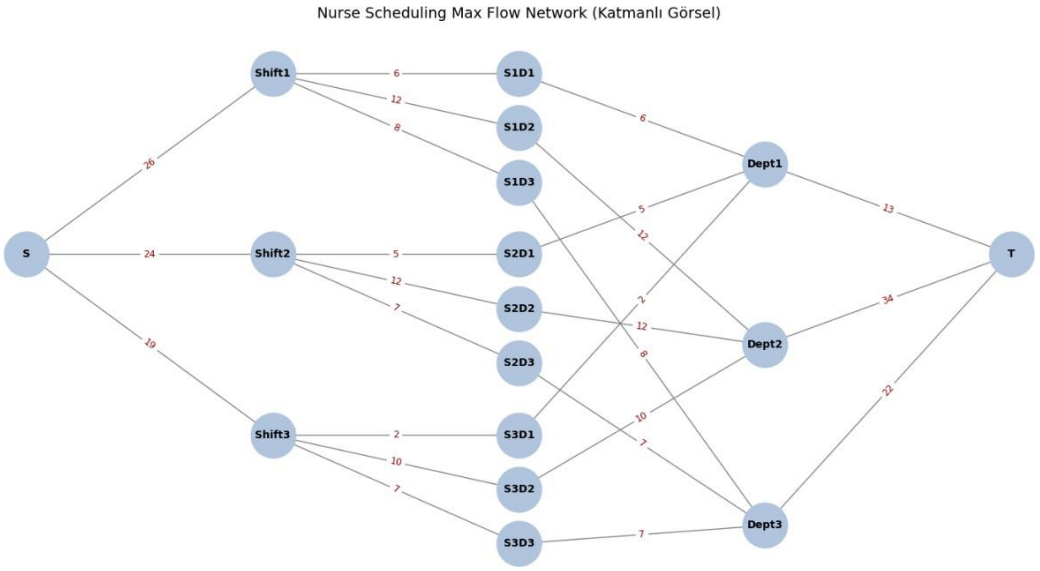
Gurobi optimizasyon modeli, maksimum akış ağı olarak formüle edilen hemşire planlama problemini başarıyla çözdü. Amaç, vardiyalar arasında toplam atanan hemşire sayısını minimize ederken, hem vardiya hem de departman bazındaki gereksinimleri karşılamaktı. Bulunan optimal çözüm, tüm kısıtlamaları karşılamak için toplam 69 hemşirenin gerektiğini ve en verimli personel dağılımının bu şekilde olduğunu göstermektedir.

Her vardiyadaki toplam hemşire sayısı, ilgili minimum gereksinimle tam olarak eşleşmekte ve asgari personel ihtiyacının karşılandığını garanti etmektedir. Özellikle, 1. Vardiyaya 26 hemşire, 2. Vardiyaya 24 hemşire ve 3. Vardiyaya 19 hemşire atanmıştır. Bu değerler sırasıyla önceden belirlenmiş olan 26, 24 ve 19 sayılarına tam olarak uyumludur.

Her vardiyedeki hemşireler üç departmana dağıtılmıştır. Örneğin, 1. Vardiya 6 hemşireyi 1. Departmana, 12 hemşireyi 2. Departmana ve 8 hemşireyi 3. Departmana göndermektedir. 2. Vardiya sırasıyla 1., 2. ve 3. Departmanlara 5, 12 ve 7 hemşire ayırmıştır. Benzer şekilde, 3. Vardiya 1. Departmana 2, 2. Departmana 10 ve 3. Departmana 7 hemşire dağıtmaktadır. Tüm dağılımlar, vardiya-departman çiftleri için belirlenmiş üst ve alt kapasite sınırlarına tam olarak uyum sağlamaktadır.

Tüm departmanlar en az gerekli minimum hemşire sayısını almaktadır. 1. Departman tam olarak 13 hemşire alarak gereksinimini karşılamaktadır. 2. Departman 32 olan minimum gereksinimin üzerinde 34 hemşire almaktadır. 3. Departman ise tam olarak 22 hemşire alarak gereksinimiyle eşleşmektedir. Bu, departman düzeyindeki kısıtlamaların optimal çözümde sağlandığını doğrulamaktadır.

4. ÇÖZÜMÜN MAKSİMUM AKIŞ AĞI



Şema, kaynaktan departmanlara doğru yapılandırılmış bir hemşire atamasını göstermektedir; bu atama sırasında her vardiyanın tam olarak ihtiyaç duyulan minimum hemşire sayısını aldığı garanti edilir: Vardiya 1 için 26, Vardiya 2 için 24 ve Vardiya 3 için 19. Hemşireler, her vardiyadan üç departmana ara katmanlar aracılığıyla dağıtılırken, vardiya-departman çiftleri arasındaki akış kısıtlamaları korunur. Departman talepleri tamamen karşılanmıştır; Departman 1 tam olarak 13 hemşire alırken, Departman 2 gereksinimin biraz üzerinde 34 hemşireye sahiptir ve Departman 3 ise ihtiyacı olan 22 hemşireyi alır. Ağ, tüm personel gereksinimlerini karşılayarak toplam hemşire kullanımını en aza indiren dengeli ve optimize edilmiş bir dağılımı yansıtır.

5. KAYNAKÇA

[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