# Sprawozdanie-Podstawy podejmowania decyzji

## 1) Wstęp

Tematem mojego projektu jest znalezienie takiej trasy w terenie między wybranymi punktami A-B, aby znaleźć optymalny (minimalny) koszt budowy wodociągu. Jest to problem NP-trudny.

Mapę terenu przedstawiłam za pomocą grafu nieskierowanego. Wierzchołki są punktami, w których może być zakończona trasa. Krawędzie grafu przedstawiają możliwe ścieżki, po których można poprowadzić trasę budowy wodociągu. Między każdym z węzłów mogą być inne wartości przepływu (Q) oraz średnic (D).

W celu znalezienia optymalnego rozwiązania posłużę się oprogramowaniem CPLEX, a do obliczenia kosztów budowy wodociągu między konkretnymi punktami posłużę się językiem programowania Python.

## 2) Sformułowanie matematyczne- obliczanie kosztów

Koszt budowy przewodu wodociągowego zbudowanego z rur z tworzyw sztucznych o długości L, składającego się z m odcinków o różnych średnicach, to koszt całkowity będzie stanowił sumę kosztów każdego odcinka.
Jest przedstawiony wzorem:

$$I = \sum_{i=1}^{i=m} (A + B * D_i^N) * L_i \ [zł],$$

gdzie
A,B,n - parametry zależne od materiału przewodu, rodzaju gruntu i innych warunków budowy
A=310
B=4400
n=1.7
Wzór na wartość średnicy:

$$D = (\frac{1}{\gamma * R})^{\frac{1}{n+5.33}} * Q^{\frac{3}{n+5.33}} \ [m],$$

gdzie

Q-przepływ wody $[\frac{m^3}{s}]$

R, C, n, $\alpha$-stałe

$\gamma = \frac{\alpha * \mu * n * C}{796.5 * e}$

$\mu - współczynnik\ zespołów\ pompowych$

$e - cena\ kWh\ energii$ [zł]

$\mu = 0.72$

$\alpha = 0.175$

$R = 1$

$e = 0.5\ zł$

Q przyjmuje wartości: 0.15, 0.1, 0.11, 0.05, 0.08, 0.12, 0.16 [$\frac{m^3}{s}$]

### 3) Implementacja w Python- obliczanie kosztów

```
import numpy as np
import csv
import networkx as nx
import matplotlib.pyplot as plt
# przykładowe wartości Q
Q_values = [0.15, 0.10, 0.11, 0.05, 0.08, 0.12, 0.16]  # m^3/s
# Przykładowe wartości L
values_of_L = [10, 20, 13, 9, 7, 8, 15]  # m
# Krawędzie grafu
edges = [(1, 2), (1, 3), (1, 4), (3, 4), (3, 5), (3, 6), (4, 7)]

A = 310
B = 4400
n = 1.5
N = 1.7
R = 1
ni = 0.72
alfa = 0.175
e = 0.5
gamma = (ni * n * alfa * B) / (796.5 * e)

#średnica
def D(Q):
    cz1 = pow((1 / (gamma * R)), (1 / (n + 5.33)))
    cz2 = pow(Q, (3 / (n + 5.33)))
    D = cz1 * cz2  # m
    return D

srednice = []
price = []
for i in range(len(Q_values)):
    sr = round(D(Q_values[i]), 2)
    srednice.append(sr)

#koszta budowy
def cost(D, L):
    I = (A + B * pow(D, N)) * L
    return I

for i in range(len(srednice)):
    c = round(cost(srednice[i], values_of_L[i]), 2)
    price.append(c)

# Tworzenie macierzy kosztów
matrix = np.zeros((len(edges), len(edges)))

# Przypisywanie wag krawędzi z listy price
for i, (u, v) in enumerate(edges):
    cost_val = price[i]
    matrix[u - 1, v - 1] = cost_val
    matrix[v - 1, u - 1] = cost_val

# Zamiana 0 na 9999 w macierzy
matrix[matrix == 0] = 999999
```

```
# Wyświetlanie macierzy kosztów
print("Macierz kosztów:")
print(matrix)


G = nx.Graph()

G.add_nodes_from(range(1, len(edges) + 1))

edge_labels = {}
for i, (u, v) in enumerate(edges):
    G.add_edge(u, v)
    edge_labels[(u, v)] = f"Cost: {matrix[u - 1, v - 1]}, Q: {Q_values[i]}"
    edge_labels[(v, u)] = f"Cost: {matrix[v - 1, u - 1]}, Q: {Q_values[i]}"

pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue',
font_size=10, edge_color='black', linewidths=1,
        alpha=0.7)
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels,
font_color='red')

plt.savefig("graph.png")
plt.show()
```

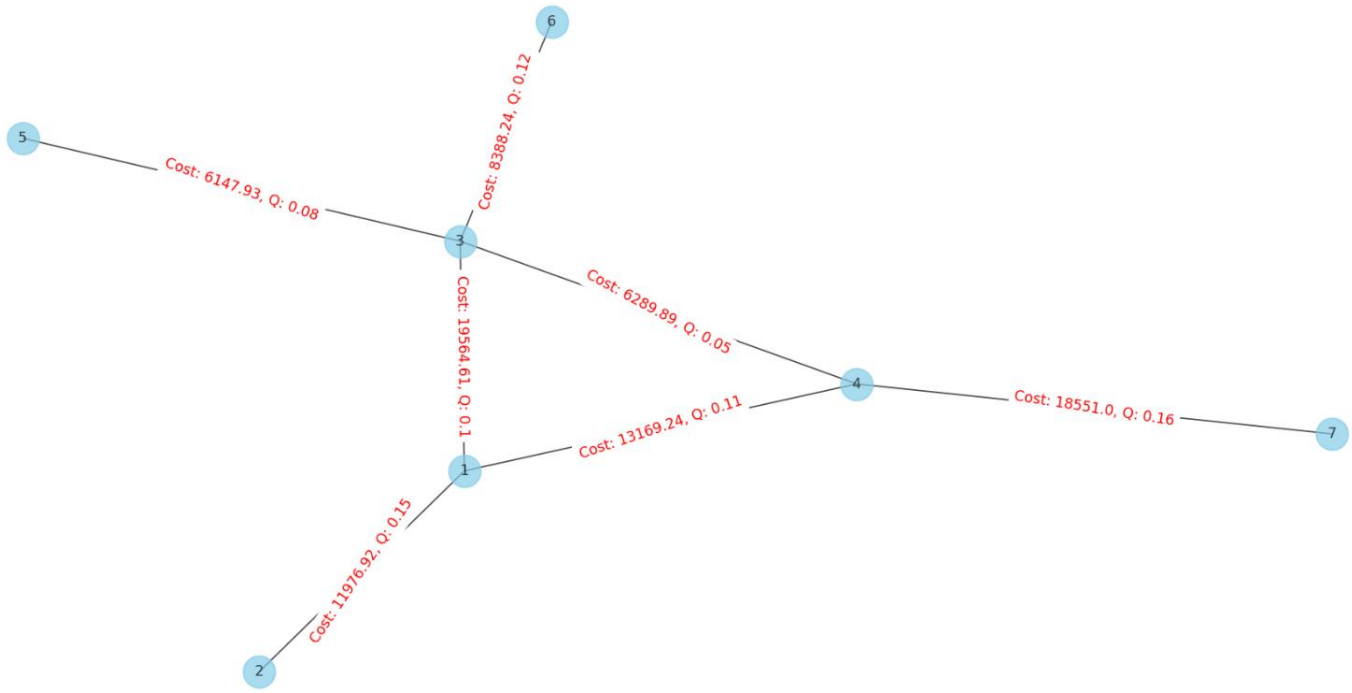
Powyższy program na podstawie wzorów w punkcie 2) oblicza macierz kosztów. Połączenia między węzłami w grafie, zostały dobrane przeze mnie.

Wynikiem jest macierz kosztów, która zawiera jednostkowe koszty(I) między punktami:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 999999 | 11976.92 | 19564.61 | 13169.24 | 999999 | 999999 | 999999 |
| 2 | 11976.92 | 999999 | 999999 | 999999 | 999999 | 999999 | 999999 |
| 3 | 19564.61 | 999999 | 999999 | 6289.89 | 6147.93 | 8388.24 | 999999 |
| 4 | 13169.24 | 999999 | 6289.89 | 999999 | 999999 | 999999 | 18551 |
| 5 | 999999 | 999999 | 6147.93 | 999999 | 999999 | 999999 | 999999 |
| 6 | 999999 | 999999 | 8388.24 | 999999 | 999999 | 999999 | 999999 |
| 7 | 999999 | 999999 | 999999 | 18551 | 999999 | 999999 | 999999 |

gdzie duża liczba-999999 oznacza brak krawędzi między danymi wierzchołkami.

Graficzne przedstawienie:




## 4) Sformułowanie matematyczne problemu optymalizacyjnego

*Dane:*
Macierz kosztów-C = $[c_{i,j}]_{7x7}$
$c_{i,j} - pojedyczny\ koszt\ budowy\ odcinka\ i-j)$
Nodes=7 (liczba wierzchołków)
Max_flow=0.3 (maksymalny przepływ) $[\frac{m^3}{s}]$
Max_cost=50000 [zł] (pieniądze przeznaczone na inwestycję)
Start=6 (węzeł startowy)
Stop=7 (węzeł końcowy)

Wartości przepływów na węzłach: Q_values=0.15, 0.10, 0.11, 0.05, 0.08, 0.12, 0.16 $[\frac{m^3}{s}]$

*Zmienne decyzyjne:*

X-macierz zmiennych decyzyjnych

$$X = [x_{i,j}]_{7x7}$$

$x_{i,j} = 1$, gdy dana krawędź została wybrana
$x_{i,j} = 0$, w przeciwnym przypadku

*Funkcja celu:*

$$\mathrm{F(X)} = \sum_{i=1}^{7}\sum_{j=1}^{7} c_{i,j} * x_{i,j}$$

*Cel:*
Minimalizacja funkcji celu F(X)

*Ograniczenia:*

$$i, j \epsilon \{1,2,3,4,5,6,7\}$$

$$\sum_{j=1}^{7} x_{start,j} = 1$$

$$\sum_{i=1}^{7} x_{i,stop} = 1$$

$$\sum_{i=1}^{7}\sum_{j=1}^{7} c_{i,j} * x_{i,j} \leq \text{Max\_cost}$$

$$\sum_{i=1}^{7}\sum_{j=1}^{7} c_{i,j} * x_{i,j} \leq \text{Q\_values}$$

$$\sum_{i=1}^{7}\sum_{j=1}^{7} x_{i,j} + \sum_{i=1}^{7}\sum_{j=1}^{7} x_{j,i} \leq 1, \text{dla i} \neq \text{j}$$

$$\sum_{i=1}^{7} x_{i,j} \leq 1$$

$$\sum_{j=1}^{7} x_{i,j} \leq 1$$

$$\sum_{j=1}^{7} x_{i,j} = \sum_{j=1}^{7} x_{j,i} \; dla \; i \neq start, j \neq stop$$

$$x_{2,j} = 0 \; i \; x_{i,2} = 0$$

$$x_{i,j} \epsilon \{0,1\}$$

## 5) Implementacja w CPLEX

```
int start = 6;
int stop = 7;
int nodes = 7;
float Q_values[1..nodes] = [0.15, 0.10, 0.11, 0.05, 0.08, 0.12, 0.16]; //wartości przepływu
float max_flow = 35.0;
//macierz kosztów
float C[1..nodes][1..nodes] =
[[999999,11976.92,19564.61,13169.24,999999,999999,999999],
 [ 11976.92,999999,999999,999999,999999,999999,999999],
 [ 19564.61, 999999,999999,6289.89,6147.93,8388.24,999999],
 [ 13169.24, 999999,6289.89,999999,999999,999999,18551],
 [999999,999999,6147.93, 999999,999999,999999,999999],
 [999999,999999,8388.24,999999,999999,999999,999999],
 [999999,999999,999999,18551,999999,999999,999999]];
int max_cost = 50000; //kwota przeznaczona na inwestycje
dvar boolean X[1..nodes][1..nodes]; //macierz zmiennych decyzyjnych
dexpr float price = sum(i in 1..nodes, j in 1..nodes) X[i][j] * C[i][j]; //funkcja celu
minimize price;
subject to {
  //suma w kolumnach i wierszach równa max 1
    forall(j in 1..nodes)
    sum(i in 1..nodes) X[i][j]<=1;
    forall(i in 1..nodes)
    sum(j in 1..nodes) X[i][j]<=1;
    // Ograniczenia dotyczące startu i końca budowy
    sum(i in 1..nodes) X[start][i] == 1 && sum(i in 1..nodes) X[i][stop] == 1;
    // Ograniczenie maksymalnej przepustowości
    forall(i in 1..nodes)
        sum(i in 1..nodes,j in 1..nodes) X[i][j] * Q_values[i] <= max_flow;
    // Ograniczenie kosztu budowy
    forall(i in 1..nodes, j in 1..nodes)
        sum(i in 1..nodes, j in 1..nodes) X[i][j] * C[i][j] <= max_cost;
    // Eliminacja cykli
    forall(i in 1..nodes, j in 1..nodes: i != j)
        X[i][j] + X[j][i] <= 1;
    //wymuszenie równowagi przepływu w wierzchołkach.
    forall(i in 1..nodes: i != start && i != stop)
    sum(j in 1..nodes) X[i][j] == sum(j in 1..nodes) X[j][i];
    //przeszkody np. awaria, węzeł 2 wyłączony z użytku
    forall(i in 1..nodes)
    X[2][i] == 0 && X[i][2] == 0;}
```

Wyniki:
Macierz zmiennych decyzyjnych:

used_edges =[[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 1 0 0 0]
[0 0 0 0 0 0 1]
[0 0 0 0 0 0 0]
[0 0 1 0 0 0 0]
[0 0 0 0 0 0 0]]

Wartość funkcji celu:

price=33229.13 [zł]

CPLEX wybrał następujące przejście po krawędziach grafu:

6→3→4→7

## 6) Heurystyka

Do rozwiązania powyżej opisanego problemu użyłam również metaheurystyki- algorytmu Tabu search.
Algorytm składa się z następujących kroków:

1) Inicjalizacja początkowego rozwiązania.
2) Inicjalizacja listy tabu, która przechowuje ruchy, których nie można powtórzyć przez określony czas.
3) Obliczanie wartości funkcji celu dla aktualnego rozwiązania, która określa jakość rozwiązania.
4) Generowanie nowych rozwiązań przez wykonanie ruchów w przestrzeni sąsiednich rozwiązań.
5) Wybór najlepszego ruchu.
6) Aktualizacja listy tabu.
7) Powtarzanie kroków 3-6 przez1000 iteracji (warunek stopu).
8) Zwrócenie najlepszego rozwiązania po spełnieniu warunku stopu.

Próbowałam dwukrotnej implementacji tego algorytmu w zastosowaniu do mojego problemu. W pierwszym podejściu wynik, czyli macierz binarna zmiennych decyzyjnych nie spełniała ograniczeń, to samo dotyczyło wartości funkcji celu, która była znacznie większa niż maksymalna kwota przeznaczona na inwestycję. W drugim podejściu natomiast, wartości zmiennych decyzyjnych wynosiły 0, a co za tym idzie wartość funkcji celu również wynosiła 0.

## 7) Wnioski

Powyżej opisany problem okazał się być ciekawym powiązaniem rzeczywistych technicznych aspektów z metodami optymalizacji. Oczywiście problem został uproszczony, budowa sieci wodociągowych jest trudnym zagadnieniem pełnym ograniczeń dotyczących specyfiki tego problemu.
Dzięki CPLEX udało się znaleźć optymalną cenę budowy wodociągu między podanymi punktami.
Heurystyka okazała się nie być dobrym narzędziem do znalezienia optymalnego rozwiązania, wystąpiły problemy przy implementacji. Wyniki nie były przybliżone do tych uzyskanych za pomocą narzędzia CPLEX, wartość funkcji celu przekraczała budżet, a krawędzie wybierane przez program nie spełniały ograniczeń.