MOM projekt 1 sprawozdanie Dawid Bartosiak

1. Zadanie 1. Sieć przepływowa (3pkt.)

W centrum dyspozytorskim planuje się dostawy węgla z określonych kopalń do elektrowni. Rozważana jest możliwość dostaw węgla kamiennego z trzech kopalń A, B, C do trzech elektrowni F, G, H za pomocą sieci kolejowej z dwoma stacjami pośrednimi D i E.

- Jednostkowe koszty transportowe i przepustowości na poszczególnych odcinkach wynoszą (koszt, przepustowość; skierowanie łuku jest od wiersza do kolumny):

	D	E	F	G	H
A	3, 8	6, 10	-	-	-
B	6, 10	3, 13	-	-	-
C	4, 10	5, 8	-	-	-
D	-	2, 20	5, 16	7, 6	3, 10
E	-	-	5, 7	4, 4	2, 2

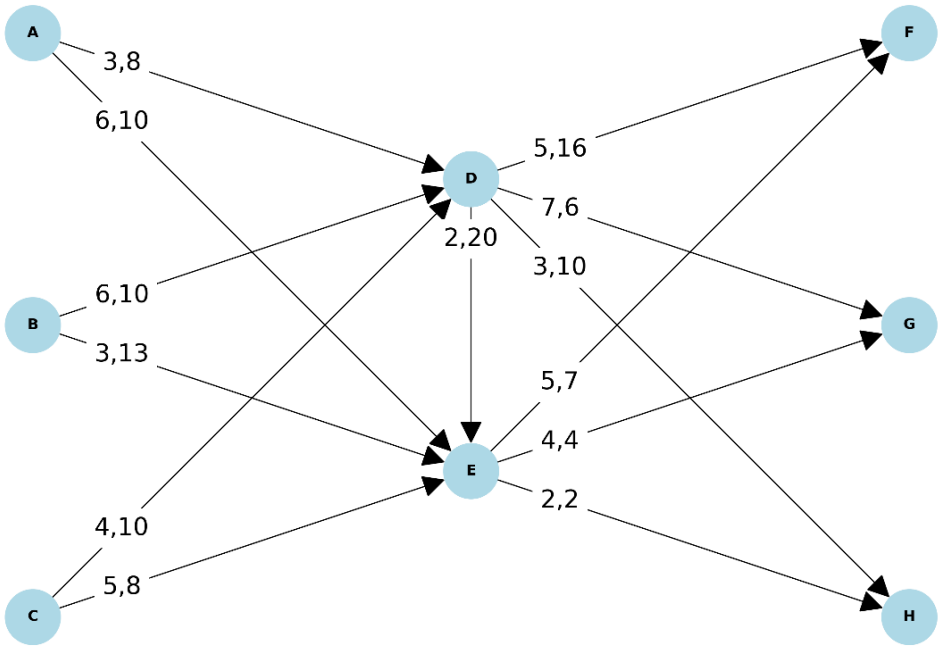
- Zdolności wydobywcze kopalń wynoszą (w tys. ton): $W_A = 10$, $W_B = 13$, $W_C = 22$.
- Średnie zużycie dobowe węgla przez elektrownie wynosi (w tys. ton): $Z_F = 15$, $Z_G = 10$, $Z_H = 10$.

Zadanie polega na wyznaczeniu planu codziennych dostaw węgla zaspokajający zapotrzebowania elektrowni i minimalizujący sumaryczne koszty transportu. W tym celu należy:

- sformułować i narysować model sieciowy (sieć przepływową); Określić jaki problem na tej sieci należy rozwiązać (należy nazwać problem do rozwiązania, nie algorytm)
- znaleźć jak najlepsze rozwiązanie (dowolną metodą ręcznie lub algorytmicznie)
- zapisać odpowiadające zadanie programowania liniowego

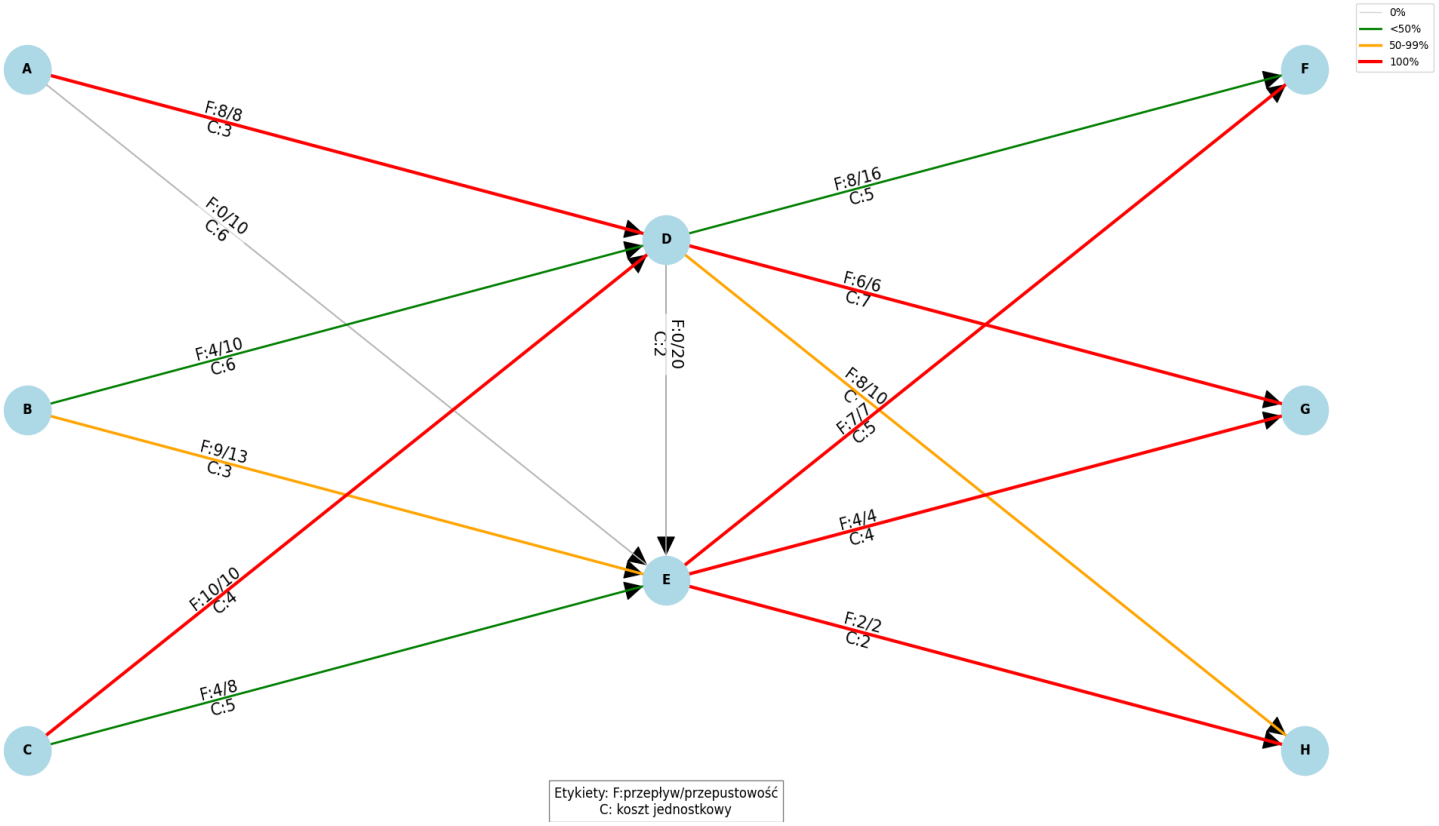
Ponadto, należy sprawdzić, gdzie w sieci transportowej występuje wąskie gardło, które stanowiłoby ograniczenie w przypadku zwiększonego zapotrzebowania ze strony elektrowni (koszty transportu należy pominąć). W tym celu należy znaleźć przekrój o jak najmniejszej przepustowości; jaką informację niesie przepustowość wybranego przekroju?

Model sieciowy z networkx w Pythonie:



Zadanie dotyczy problemu minimalnego kosztu w sieci przepływowej.

Rozwiązanie znalezione ręcznie dało koszt 296 i następujący przepływ węgla w sieci:



Najlepsze rozwiązanie przy przesłaniu pełnego zapotrzebowania opisane jest przez następujące zadanie programowania liniowego:

Dane:

$c_{ij} - koszt\ przesłania\ 1\ tys.ton\ ze\ stacji\ i\ do\ stacji\ j$

$u_{ij} - maksymalna\ przepustowość\ krawędzi\ i,j\ w\ tys.ton$

$p_i - zdolność\ produkcyjna\ kopalni\ i\ w\ tys.ton$

$z_i - zużycie\ dobowe\ elektrowni\ i\ w\ tys.ton$

krawędzie – zbiór wszystkich krawędzi

Dane pomocnicze:

węzły początkowe – zbiór wszystkich kopalni

węzły końcowe – zbiór wszystkich fabryk

węzły transportowe – zbiór wszystkich stacji pośrednich

Zmienne:

$x_{ij} - przepływ\ w\ tys.ton\ ze\ stacji\ i\ do\ stacji\ j$

Ograniczenia:

Ograniczenie maksymalnego przepływu na każdej z krawędzi:

$$\bigvee_{i,j}^{krawędzie} \quad 0 \leq x_{ij} \leq u_{ij}$$

Dla węzłów początkowych – maksymalna produkcja:

$$\bigvee_{i}^{węzły\ początkowe} \quad p_i \geq \sum_{j}^{węzły:(i,j)\in krawędzie} x_{ij}$$

Dla węzłów pośrednich – zachowanie przepływu:

$$\bigvee_{i}^{węzły\ transportowe} \quad \sum_{j}^{węzły:(i,j)\in krawędzie} x_{ij} = \sum_{j}^{węzły:(j,i)\in krawędzie} x_{ji}$$

Dla węzłów końcowych – spełnienie zapotrzebowania:

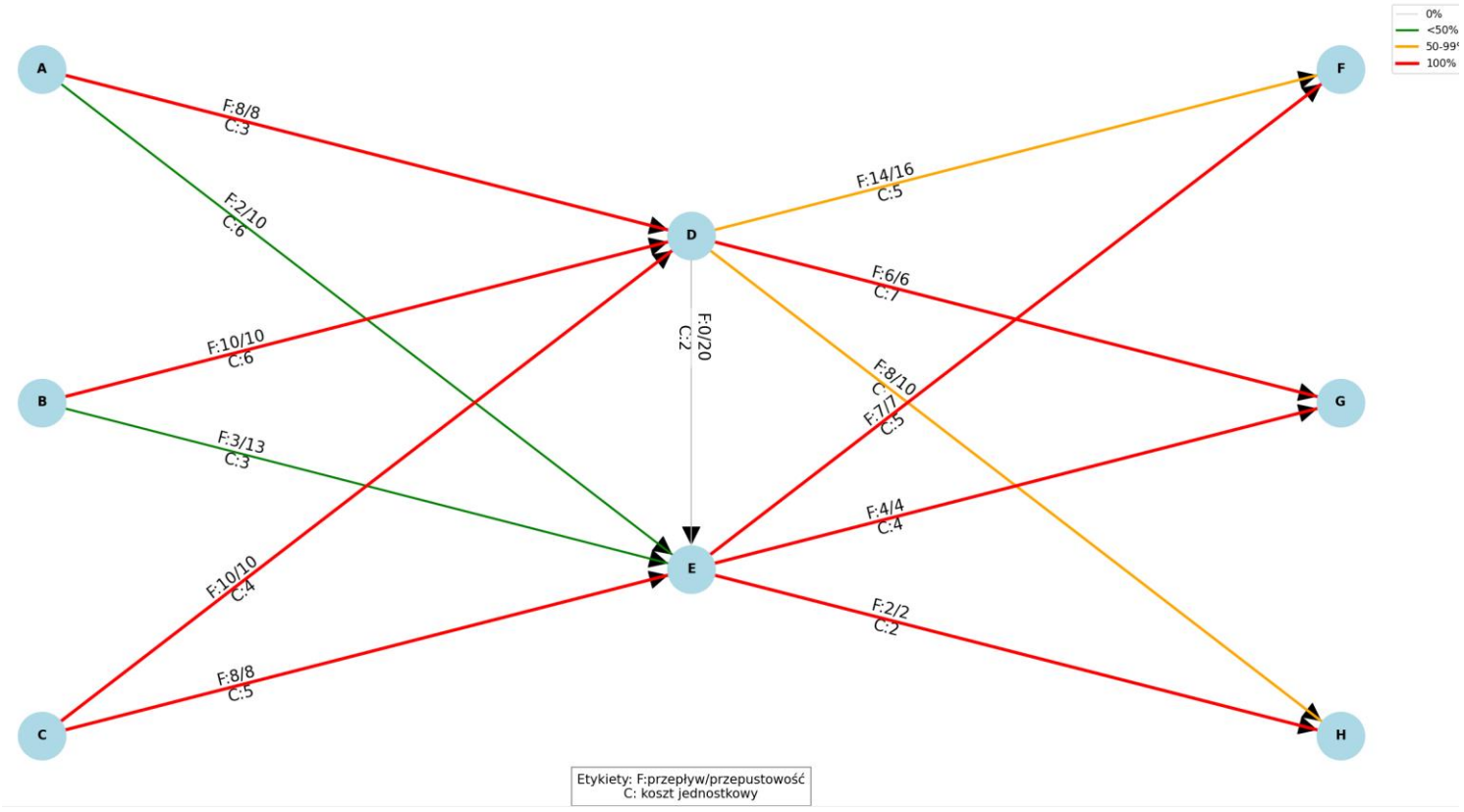
$$\bigvee_{i}^{węzły\ końcowe} \quad z_i = \sum_{j}^{węzły:(j,i)\in krawędzie} x_{ji}$$

Oraz funkcja celu zdefiniowana jest jako:

$$\text{minimize} \left(\sum_{i,j}^{krawędzie} c_{ij} \cdot x_{ij} \right)$$

Rozwiązanie tego zadania w AMPL daje koszt minimalny w wysokości 296:

```
# Zbiory
set NODES;                      # Zbiór wszystkich węzłów
set SOURCES within NODES;       # Zbiór źródeł (kopalnie)
set TRANS within NODES;         # Zbiór węzłów pośrednich (punkty przeładunkowe)
set SINKS within NODES;         # Zbiór ujść (elektrownie)
set ARCS within {NODES, NODES}; # Zbiór krawędzi (połączeń transportowych)

# Parametry
param cost {ARCS} >= 0;         # Koszt jednostkowy transportu
param capacity {ARCS} >= 0;     # Maksymalna przepustowość krawędzi
param supply {SOURCES} >= 0;    # Podaż w źródłach (kopalniach)
param demand {SINKS} >= 0;      # Popyt w ujściach (elektrowniach)

# Zmienne decyzyjne
var flow {(i,j) in ARCS} >= 0, <= capacity[i,j]; # Przepływ na krawędziach

# Funkcja celu - minimalizacja kosztu transportu
minimize Total_Cost: sum {(i,j) in ARCS} cost[i,j] * flow[i,j];

# Ograniczenia
# Zachowanie przepływu w węzłach źródłowych (kopalniach)
subject to Source_Flow {i in SOURCES}:
    sum {j in NODES: (i,j) in ARCS} flow[i,j] <= supply[i];

# Zachowanie przepływu w węzłach pośrednich (punktach przeładunkowych)
subject to Trans_Flow {i in TRANS}:
    sum {j in NODES: (i,j) in ARCS} flow[i,j] = sum {j in NODES: (j,i) in ARCS} flow[j,i];

# Zachowanie przepływu w węzłach ujściowych (elektrowniach)
subject to Sink_Flow {i in SINKS}:
    sum {j in NODES: (j,i) in ARCS} flow[j,i] = demand[i];
```



Potencjalne wąskie gardła przy zwiększaniu całkowitego zapotrzebowania stanowią krawędzie zaznaczone na czerwono na modelu przepływu zamieszonym na poprzedniej stronie, zatem AD, BD, CD, CE, DG, EF, EG, EH. Mając to na świadomości określam jako minimalny przekrój taki w którym dzielę węzły na następujące zbiory: {A, B, C, E}, oraz {D, F, G, H} i jego przepustowość wynosi 41 jednostek. Celem zwiększenia przepustowości potrzebowalibyśmy zwiększyć maksymalny możliwy przepływ do D poprzez poprawę krawędzi AD, BD lub CD, albo zwiększyć maksymalny przepływ z E poprzez poprawę krawędzi EF, EG, EH.

2. Zadanie 2. Zadanie przydziału (6 pkt.)

2.1 Planowanie realizacji portfela przy ograniczonych kompetencjach

Softwarehouse posiada portfel projektów oznaczonych 1-6 oraz zespoły programistyczne oznaczone A-F. Poniża tabela przedstawia kompetencje zespołów, gdzie „-„ oznacza brak kompetencji zespołu do realizacji danego projektu.

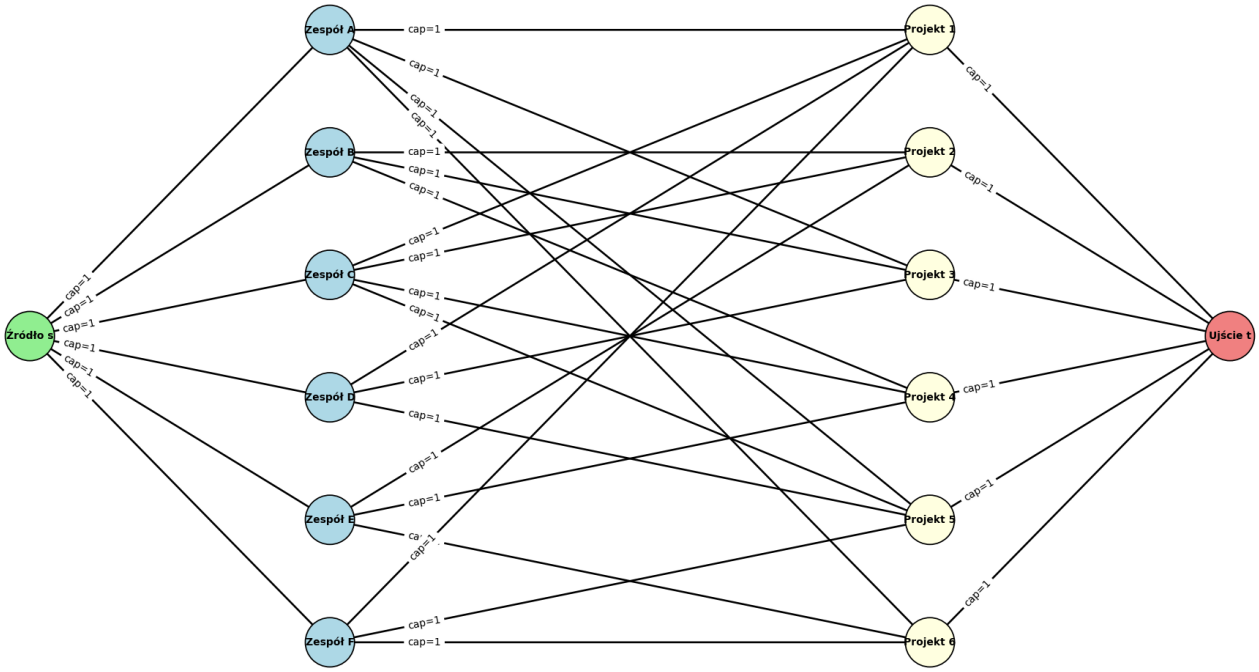
		Zespoły					
		A	**B**	**C**	**D**	**E**	**F**
Projekty	**1**	X	-	X	X	-	X
	2	-	X	X	-	X	-
	3	X	X	-	X	-	-
	4	-	X	X	-	X	-
	5	X	-	X	X	-	X
	6	X	-	-	-	X	X

Tabela 2: Kompetencje zespołów

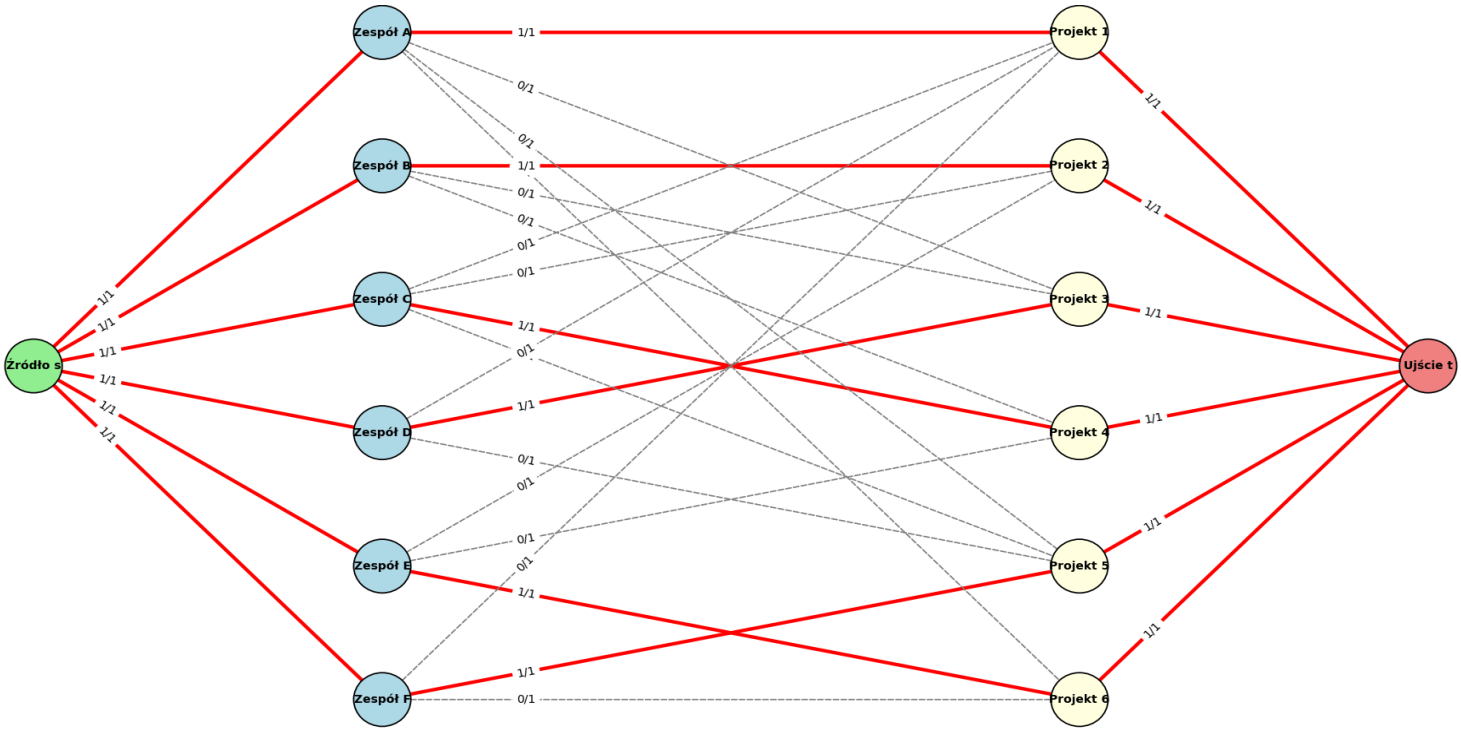
Należy dokonać przydziału zespołów programistycznych do poszczególnych projektów, przy założeniu, że jeden zespół może realizować tylko jeden projekt, a jeden projekt może być realizowany przez tylko jeden zespół. W tym celu:

- narysować model sieciowy problemu
- określić jaki problem należy rozwiązać i znaleźć ręcznie rozwiązanie
- na podstawie rozwiązania modelu sieciowego określić przydział zespołów do projektów

Do rozwiązania jest tutaj zadanie maksymalnego skojarzenia, czy też maksymalnego przydziału. Odpowiedzieć należy zatem na pytanie „Ile zadań da się wykonać?”, a koszty przepływu to 0. Model sieciowy problemu z wykorzystaniem biblioteki networkx w Pythonie:



Przykładowy przydział zespołów do projektów – ręczne rozwiązanie, w którym udało się przydzielić wszystkie zespoły do projektów:



2.2 Minimalizacja kosztów realizacji projektów

Zakładamy, że firma wynajmuje zespoły do realizacji projektów. Koszty wynajmu są podane w poniższej tabeli.

		Zespoły					
		A	**B**	**C**	**D**	**E**	**F**
Projekty	**1**	15	-	14	9	-	12
	2	-	12	16	-	10	-
	3	11	14	-	12	-	-
	4	-	16	11	-	12	-
	5	13	-	17	13	-	15
	6	11	-	-	-	16	18

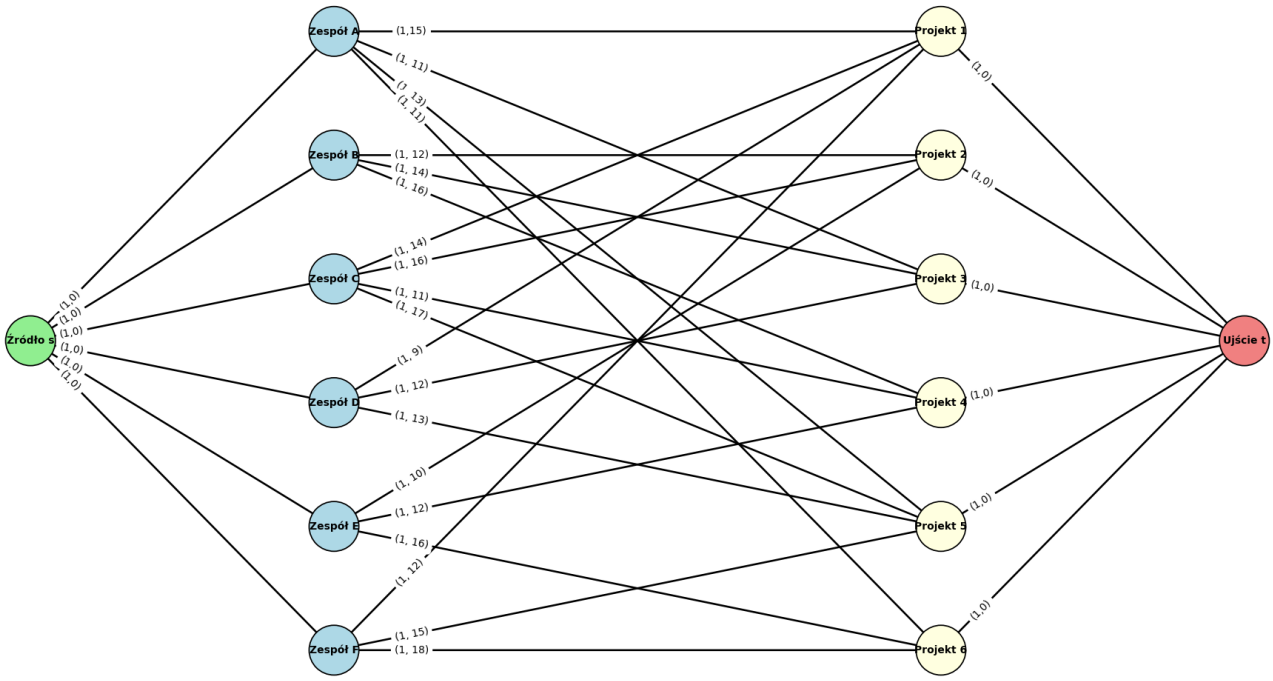
Tabela 2: Koszty wykonywania projektów przez poszczególne zespoły

Należy dokonać przydziału zespołów programistycznych do projektów tak, aby minimalizować koszty najmu zespołów. Ograniczenia dotyczące jednego zespołu i jednego projektu pkt. 2.1 dalej obowiązują. W tym celu:

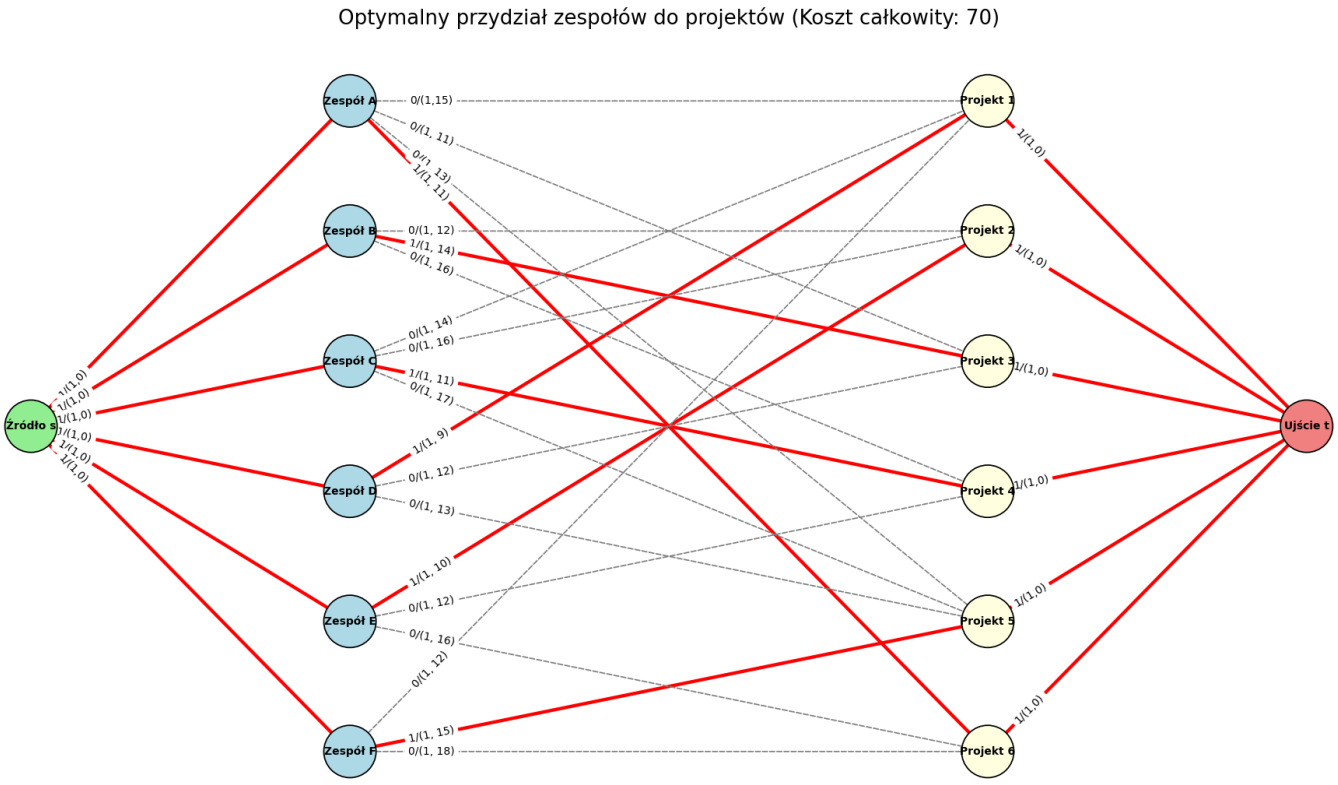
- narysować model sieciowy problemu
- określić jaki problem należy rozwiązać na tym modelu sieciowym
- spróbować znaleźć jak najlepsze rozwiązanie

Jest tutaj do rozwiązania problem najtańszego skojarzenia (najtańszego przydziału) który opisany jest przez poniższy model sieciowy narysowany z użyciem networkx w Pythonie:

Model sieciowy problemu przydziału zespołów do projektów



Najlepszy przydział znaleziony ręcznie został przedstawiony na poniższym modelu a jego koszt wyniósł 70:



Oraz potwierdzony przez rozwiązane zadanie MIPL w AMPL:

```
# Zbiory
set TEAMS;    # Zbiór zespołów
set PROJECTS; # Zbiór projektów

# Parametry
param time {TEAMS, PROJECTS} default Infinity; # Czas realizacji projektu przez zespół (w miesiącach)

# Zmienne decyzyjne
var assign {TEAMS, PROJECTS} binary; # 1 jeśli zespół i realizuje projekt j, 0 w przeciwnym przypadku

# Funkcja celu - minimalizacja sumy czasów realizacji wszystkich projektów
minimize Total_Time: sum {i in TEAMS, j in PROJECTS: time[i,j] < Infinity} time[i,j] * assign[i,j];

# Ograniczenia
# Każdy zespół realizuje co najwyżej jeden projekt
subject to One_Project_Per_Team {i in TEAMS}:
    sum {j in PROJECTS} assign[i,j] <= 1;

# Każdy projekt jest realizowany przez dokładnie jeden zespół
subject to One_Team_Per_Project {j in PROJECTS}:
    sum {i in TEAMS} assign[i,j] = 1;
```

Wynik działania programu:

```
ampl: include 'E:\AMPL\MOM\P1\ampl\main2.run';
CPLEX 22.1.2:                CPLEX 22.1.2: optimal solution; objective 70
3 simplex iterations

--- WYNIKI PRZYDZIAŁU ZESPOŁÓW DO PROJEKTÓW ---

Całkowity czas realizacji całego portfela projektów: 70.00 miesięcy

Optymalny przydział zespołów do projektów:
Zespół     Projekt    Czas realizacji
------     -------    ---------------
A          6          11.00
B          3          14.00
C          4          11.00
D          1          9.00
E          2          10.00
F          5          15.00
```

2.3 Minimalizacja terminu realizacji puli projektów

Załóżmy teraz, że dane podane w tabeli 2 to czasy (w miesiącach) realizacji projektów przez poszczególne zespoły. Ograniczenia dotyczące jednego zespołu i jednego projektu pkt. 2.1 dalej obowiązują. Zaproponować model programowania liniowego minimalizujący termin realizacji całego portfela projektów (jest to termin zdeterminowany przez najdłużej wykonywany projekt).

Dane:

zespoły – zbiór wszystkich zespołów

projekty – zbiór wszystkich projektów

$time_{ij}$ – czas potrzebny zespołowi i na realizację projektu j

Zmienne:

$x_{ij} - przypisanie\ zespołu\ i\ do\ projektu\ j$ (zmienna binarna)

max_time – czas realizacji całego portfela projektów (zarazem czas trwania najdłuższego z wykonywanych projektów)

Ograniczenia:

Nieujemny czas trwania realizacji całego portfela projektów:

$$max\_time \geq 0$$

Każdy zespół realizuje co najwyżej jeden projekt:

$$\bigvee_{i}^{zespoły} \sum_{j}^{projekty} x_{ij} \leq 1$$

Każdy projekt jest realizowany przez dokładnie jeden zespół:

$$\bigvee_{i}^{projekty} \sum_{j}^{zespoły} x_{ji} = 1$$

Zespół może realizować projekt tylko jeśli ma odpowiednie kompetencje – zostało to uzyskane poprzez uzupełnienie brakujących kompetencji kosztem większym niż pozostałe, tutaj 20 miesięcy.

Maksymalny czas realizacji jest nie mniejszy niż czas realizacji każdego przydzielonego projektu:

$$\bigvee_{i}^{zespoły} \bigvee_{j}^{projekty} max\_time \geq time_{ij} \cdot x_{ij}$$

Funkcja celu w tym modelu to minimalizacja maksymalnego czasu realizacji. Ostatecznie minimalny czas realizacji całego portfela projektów to 14 miesięcy:

Zespół	Projekt	Czas realizacji
A	6	11.00
B	3	14.00
C	4	11.00
D	5	13.00
E	2	10.00
F	1	12.00

Kod w AMPL zamieszczony został na następnej stronie:

```
# Zbiory
set TEAMS;    # Zbiór zespołów
set PROJECTS; # Zbiór projektów

# Parametry
param time {TEAMS, PROJECTS} default Infinity; # Czas realizacji projektu przez zespół (w miesiącach)

# Zmienne decyzyjne
var assign {TEAMS, PROJECTS} binary; # 1 jeśli zespół i realizuje projekt j, 0 w przeciwnym przypadku
var max_time >= 0;                   # Maksymalny czas realizacji (do minimalizacji)

# Funkcja celu - minimalizacja maksymalnego czasu realizacji
minimize Total_Time: max_time;

# Ograniczenia
# Każdy zespół realizuje co najwyżej jeden projekt
subject to One_Project_Per_Team {i in TEAMS}:
    sum {j in PROJECTS} assign[i,j] <= 1;

# Każdy projekt jest realizowany przez dokładnie jeden zespół
subject to One_Team_Per_Project {j in PROJECTS}:
    sum {i in TEAMS} assign[i,j] = 1;

# Maksymalny czas realizacji jest nie mniejszy niż czas realizacji każdego przydzielonego projektu
subject to Max_Time_Constraint {i in TEAMS, j in PROJECTS}:
    max_time >= time[i,j] * assign[i,j];
```

3. Zadanie 3 (9 pkt.)

Pewna firma FMCG planuje sprzedaż jednego produktu. Produkt jest dostarczany do 8 punktów sprzedaży. Na podstawie danych historycznych (lub prognozowanych) utworzony tzw. plan bazowy dostaw opisujący ilości produktu, które były (powinny być) dostarczane do każdego punktu. Jest on następujący:

punkt	1	2	3	4	5	6	7	8
ilość	240	385	138	224	144	460	198	200

Jednak ze względu na akcje marketingowe oraz różnego rodzaju umowy/ustalenia z handlowcami tego produktu wprowadzono różnego rodzaju modyfikacje ww. planu bazowego w formie zagregowanych ograniczeń eksperckich:

1. Suma towaru dostarczonego do punktów 1, 3, 8 ma być przynajmniej o 12% większa niż planie bazowym.
2. Suma towaru dostarczonego do punktów 3, 5 ma być przynajmniej o 7% mniejsza niż w planie bazowym.
3. Ilość towaru dostarczonego do punktu 3 ma stanowić przynajmniej 80% towaru dostarczonego do punktu 7.

Zakładając, że sumaryczna wielkość sprzedaży produktu we wszystkich punktach nie może zostać zmieniona, należy zaplanować wielkość sprzedaży w poszczególnych punktach minimalizującą względne odchylenie (upewnij się, że dobrze rozumiesz „względne odchylenie”) od planu bazowego (a dokładnie - wartość bezwzględną względnego odchylenia). Ponieważ jest 8 względnych odchyleń (kryteriów), należy sformułować własną funkcję celu, która jest sumą ważoną dwóch składników: 1) maksymalnego względnego odchylenia pośród 8 odchyleń, 2) sumy wszystkich względnych odchyleń.

Należy zamodelować powyższy problem w postaci zadania programowania liniowego.

Spostrzeżenie: Nie jest konieczna definicja flagi dodatniości gdyż solver PL będzie dążyć do rozwiązania w których jedna ze zmiennych będzie wynosiła 0. Zatem nie są potrzebne dodatkowe definicje i ograniczenia.

Dane:

punkty – zbiór wszystkich punktów sprzedaży

$base\_plan_i$ – plan bazowy dostaw do punktu i

Zmienne:

$delivery_i - ilość\ produktu\ dostarczanego\ do\ punktu\ i \geq 0$

Wprowadzić należy dodatkowe oznaczenia:

$pos\_dev_i/neg\_dev_i\ - dodatnie/ujemne\ odchylenie\ od\ planu\ bazowego\ dla\ punktu\ i$

$rel\_dev_i - względne\ odchylenie\ od\ planu\ bazowego\ dla\ punktu\ i$

Oraz definicje oznaczeń:

$$\bigvee_i^{punkty} delivery_i - base\_plan_i = pos\_dev_i - neg\_dev_i$$

$$\bigvee_i^{punkty} rel\_dev_i = \frac{pos\_dev_i - neg\_dev_i}{base\_plan_i}$$

$$\bigvee_i^{punkty} max\_rel\_dev \geq rel\_dev_i$$

Ograniczenia:

Wszystkie zmienne nieujemne:

$$\bigvee_i^{punkty} pos\_dev_i \geq 0$$

$$\bigvee_i^{punkty} neg\_dev_i \geq 0$$

$$\bigvee_i^{punkty} rel\_dev_i \geq 0$$

$$max\_rel\_dev \geq 0$$

Suma dostaw musi być równa sumie planu bazowego:

$$\sum_i^{punkty} delivery_i = plan\ bazowy_i$$

Suma towaru dostarczonego do punktów 1, 3, 8 ma być przynajmniej o 12% większa niż w planie bazowym:

$$delivery_1 + delivery_3 + delivery_8 \geq 1{,}12 \cdot (base\_plan_1 + base\_plan_3 + base\_plan_8)$$

Suma towaru dostarczonego do punktów 3, 5 ma być przynajmniej o 7% mniejsza niż w planie bazowym:

$$delivery_3 + delivery_5 \leq 0{,}93 \cdot (base\_plan_3 + base\_plan_5)$$

Ilość towaru dostarczonego do punktu 3 ma stanowić przynajmniej 80% towaru dostarczonego do punktu 7:

$$delivery_3 \geq 0{,}8 \cdot base\_plan_7$$

Oraz funkcja celu:

$$minimize\ (max\_rel\_dev + w_{sumy\ odchyleń} \cdot \sum_{i}^{punkty} rel\_dev_i)$$

Ostatecznie otrzymany wynik dla wagi sumy odchyleń 0,1 to około 21,68.

```
--- WYNIKI PLANOWANIA DOSTAW PRODUKTU FMCG ---

Plan bazowy i optymalny plan dostaw:
Punkt      Plan bazowy     Optymalny plan  Odchylenie      Względne odch. (%)
------     ------------    --------------  ----------      ------------------
1          240             277             37              15.26
2          385             385             0               0.00
3          138             140             2               1.62
4          224             224             0               0.00
5          144             122             -22             15.26
6          460             435             -25             5.36
7          198             175             -23             11.47
8          200             231             31              15.26

Suma planu bazowego: 1989
Suma optymalnego planu: 1989

Maksymalne względne odchylenie: 15.26%
Suma wszystkich względnych odchyleń: 64.22%
Wartość funkcji celu: 0.2168

--- WERYFIKACJA OGRANICZEŃ ---

Ograniczenie 1: Suma towaru dostarczonego do punktów 1, 3, 8 ma być przynajmniej o 12% większa niż w planie bazowym
Suma w planie bazowym (punkty 1, 3, 8): 578
Suma w optymalnym planie (punkty 1, 3, 8): 647
Uzyskana suma: 647
Wymagana minimalna suma: 647
Ograniczenie spełnione: TAK

Ograniczenie 2: Suma towaru dostarczonego do punktów 3, 5 ma być przynajmniej o 7% mniejsza niż w planie bazowym
Suma w planie bazowym (punkty 3, 5): 282
Suma w optymalnym planie (punkty 3, 5): 262
Uzyskana suma: 262
Wymagana maksymalna suma: 262
Ograniczenie spełnione: TAK

Ograniczenie 3: Ilość towaru dostarczonego do punktu 3 ma stanowić przynajmniej 80% towaru dostarczonego do punktu 7
Ilość w punkcie 3: 140
Ilość w punkcie 7: 175
Wymagana minimalna ilość w punkcie 3: 140
Ograniczenie spełnione: TAK
```

Kod w AMPL:

```
# Zbiory
set POINTS;  # Zbiór punktów sprzedaży

# Parametry
param base_plan {POINTS} >= 0;  # Plan bazowy dostaw
param total_base_plan := sum {i in POINTS} base_plan[i];  # Suma planu bazowego

# Zmienne decyzyjne
var delivery {i in POINTS} >= 0;  # Ilość produktu dostarczanego do punktu i
var pos_dev {i in POINTS} >= 0;   # Dodatnie odchylenie od planu bazowego
var neg_dev {i in POINTS} >= 0;   # Ujemne odchylenie od planu bazowego
var rel_dev {i in POINTS} >= 0;   # Względne odchylenie od planu bazowego
var max_rel_dev >= 0;             # Maksymalne względne odchylenie

# Funkcja celu - minimalizacja ważonej sumy maksymalnego odchylenia i sumy wszystkich odchyleń
minimize Objective: max_rel_dev + 0.1 * sum {i in POINTS} rel_dev[i];

# Ograniczenia
# Definicja odchyleń
subject to Deviation_Def {i in POINTS}:
    delivery[i] - base_plan[i] = pos_dev[i] - neg_dev[i];

# Definicja względnego odchylenia
subject to Rel_Dev_Def {i in POINTS}:
    rel_dev[i] = (pos_dev[i] + neg_dev[i]) / base_plan[i];

# Maksymalne względne odchylenie
subject to Max_Rel_Dev_Def {i in POINTS}:
    max_rel_dev >= rel_dev[i];

# Suma dostaw musi być równa sumie planu bazowego
subject to Total_Delivery:
    sum {i in POINTS} delivery[i] = total_base_plan;

# Ograniczenie 1: Suma towaru dostarczonego do punktów 1, 3, 8 ma być przynajmniej o 12% większa niż w planie bazowym
subject to Constraint_1:
    delivery[1] + delivery[3] + delivery[8] >= 1.12 * (base_plan[1] + base_plan[3] + base_plan[8]);

# Ograniczenie 2: Suma towaru dostarczonego do punktów 3, 5 ma być przynajmniej o 7% mniejsza niż w planie bazowym
subject to Constraint_2:
    delivery[3] + delivery[5] <= 0.93 * (base_plan[3] + base_plan[5]);

# Ograniczenie 3: Ilość towaru dostarczonego do punktu 3 ma stanowić przynajmniej 80% towaru dostarczonego do punktu 7
subject to Constraint_3:
    delivery[3] >= 0.8 * delivery[7];
```