LINIOWE MODELE OPTYMALIZACJI DYSKRETNEJ

❑ Liniowe Modele Optymalizacji Dyskretnej – wprowadzenie w tematykę zagadnień

Istnieje bardzo wiele sytuacji decyzyjnych, których nie możemy opisać używając tylko wyłącznie zmiennych ciągłych.

Wynika to z nieciągłości pewnych rozważanych procesów ekonomicznych:

- pracownika można przydzielić tylko do jednego z kilku dostępnych stanowisk pracy;
- projekt inwestycyjny będzie przyjmowany do realizacji lub nie;
- zakład produkcyjny będzie lokalizowany w jednym z możliwych punktów lokalizacji lub też nie;

We wszystkich przytoczonych sytuacjach decyzyjnych wymagamy, aby wszystkie (lub choć jedna zmienna decyzyjna), spośród tych które mamy wyznaczyć przyjmowały wartości tzw. ***dyskretne*** (np. ze zbioru liczb całkowitych: $x \in Z$, lub ze zbioru liczb binarnych: $x \in \{0,1\}$).

Zagadnienia decyzyjne, w których przynajmniej jedna zmienna decyzyjna przyjmuje wartości dyskretne nazywamy – ***dyskretnym zagadnieniem decyzyjnym***, a ich matematyczne modele – ***dyskretnym zadaniem decyzyjnym*** (DZD).

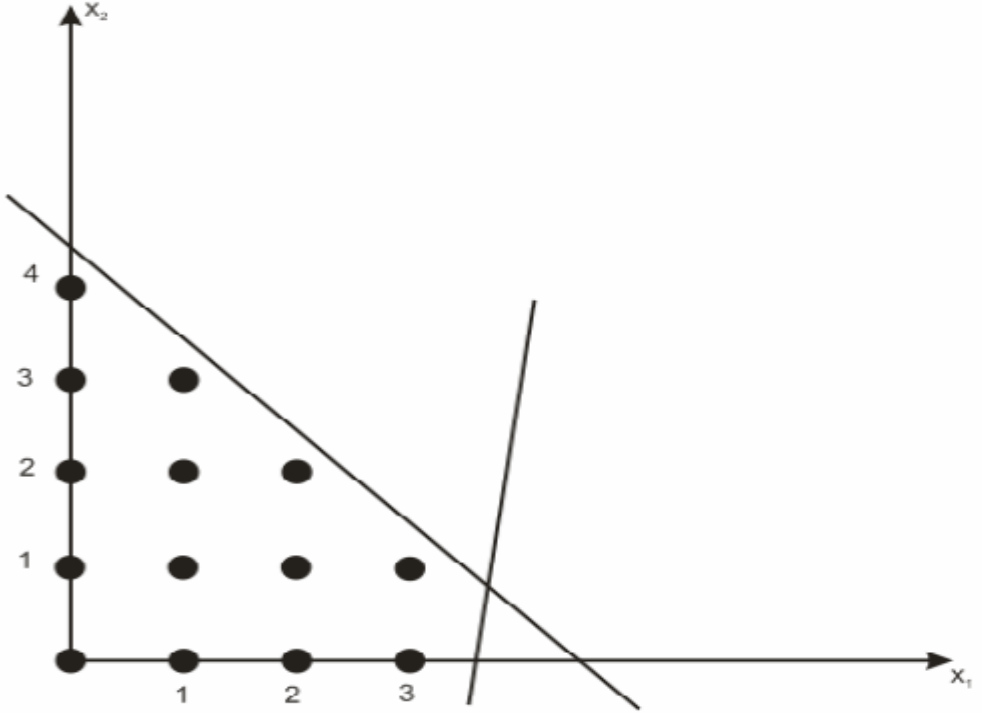
❑ Liniowe Modele Optymalizacji Dyskretnej – wprowadzenie w tematykę zagadnień

Interesować nas będą obecnie tylko takie dyskretne problemy decyzyjne, w których zarówno funkcja celu jak i warunki ograniczające są postaci liniowej – *zadania programowania dyskretnego - liniowego* (PDL). Wśród tego typu zadań wyróżnia się trzy podstawowe grupy:

- zadania *programowania całkowitoliczbowego - liniowego* (PCL)
- zadania *programowania binarnego – liniowego* (PBL)
- zadania *programowania mieszanego – liniowego* (PML)

❑ Liniowe Modele Optymalizacji Dyskretnej – wprowadzenie w tematykę zagadnień

Zbiór rozwiązań dopuszczalnych „*D*" zadania programowania dyskretnego – liniowego jest zawsze zbiorem niespójnym (np. dla zadania programowania całkowitoliczbowego z dwoma zmiennymi - będzie to zbiór punktów o współrzędnych całkowitych znajdujących się w pewnym wieloboku). Nieciągłość zmiennych decyzyjnych powoduje, że zadania tego typu są trudniejsze do rozwiązania, niż zwykłe zadania programowania liniowego.



❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

- **Zagadnienie optymalnego przydziału**

Istnieje możliwość obsadzenia „n" – stanowisk roboczych (i = 1,2,...,n) przez „n" – osób (pracowników) (j = 1,2,...,n). Znane są efekty pracy j – tego robotnika na i – tym stanowisku pracy (macierz efektów pracy - $W_{i,j} = [w_{i,j}]_{i,j=1,2,...,n}$).

Efekty te mogą być oceniane pozytywnie (wydajność pracy, wartość produkcji w przeliczeniu na jednostkę czasu) lub negatywnie (liczba braków, czas wykonania pracy, koszty związane z pracą).

Należy dokonać takiego przydziału pracowników do poszczególnych stanowisk pracy, tak aby zminimalizować negatywne lub zmaksymalizować pozytywne efekty pracy dla całego zespołu (zakładu pracy).

Zakłada się ponadto, że każde stanowisko pracy może być obsadzone tylko przez jednego pracownika, a tym samym każdy pracownik może pracować tylko na jednym stanowisku.

Oznaczenia:

Oznaczmy przez $X_{i,j} = [x_{i,j}]_{i,j} = 1,2,...,n$ - macierz zmiennych decyzyjnych, która jest postaci:

$$x_{i,j} = \begin{cases} 1, & \text{gdy j - ty pracownik jest przydzielony do i - tego stanowiska} \\ 0, & \text{w przeciwnym wypadku} \end{cases}$$

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Model matematyczny:

Problem ten można przedstawić za pomocą następującego liniowego zadania programowania binarnego (PBL):

(funkcja celu)

$$f(x_{i,j}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \cdot x_{i,j} \rightarrow \max(\min)$$

(warunki ograniczające)

$$\sum_{j=1}^{n} x_{i,j} = 1, \quad i = 1,2,\ldots,n$$

(każde stanowisko jest obsadzone tylko przez 1 pracownika)

$$\sum_{i=1}^{n} x_{i,j} = 1, \quad j = 1,2,\ldots,n$$

(każdy pracownik jest przydzielony tylko do 1 stanowiska)

$$x_{i,j} = \begin{cases} 1 \\ 0 \end{cases} \quad \text{i, j=1,2,...,n.}$$

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Każde rozwiązanie bazowe dopuszczalne (tym samym optymalne) to macierze postaci (dla n=5):

$$X = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(w każdym wierszu oraz w każdej kolumnie jest tylko jedna jedynka).

Zadanie optymalnego przydziału, mimo że jest klasycznym problemem **programowania dyskretnego**, to może być rozwiązane metodami programowania liniowego – **algorytmem simpleks** (co jest bardzo pracochłonne). Istnieje jednak stosunkowo prosty i skuteczny algorytm postępowania – **algorytm węgierski** (oparty na twierdzeniu węgierskiego matematyka - **Denesa Königa**), który można zastosować do rozwiązywania zadań optymalnego przydziału.

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Zagadnienie rozwózki:

Często mamy do czynienia z sytuacją, gdy pewien jednorodny produkt musi zostać przewieziony od producenta do wielu jego odbiorców. Dla przykładu z cukrowni rozwozi sie wyprodukowany cukier, z mleczarni np. masło i mleko, z browaru piwo itd.

Niekiedy mamy sytuację odwrotną, zwłaszcza w przemyśle spożywczym zakupiony surowiec od wielu jego producentów (np. mleko) należy przewieźć do zakładu (np. zakładów mleczarskich) w którym odbywa się dalsza jego przeróbka.

Tego typu zagadnienia nazywają się ogólnie zagadnieniami rozwózkowo-przywozowymi. Dla uproszczenia w dalszym ciągu będziemy rozważać tylko zagadnienie rozwózki.

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Zakładamy, że dane są:

- baza będąca miejscem produkcji jednorodnego towaru oraz postoju parku transportowego;

- dana jest liczba pojazdów o jednakowej ładowności;

- znamy popyt każdego odbiorcy;

- oraz macierz odległości (lub kosztu przewozu, czasu przewozu) pomiędzy wszystkimi punktami odbioru;

- popyt każdego odbiorcy jest mniejszy od ładowności pojazdów, a łączne zapotrzebowanie wszystkich punktów odbioru jest mniejsze od ładowności całego parku transportowego;

- towar jest dostarczany do odbiorcy w okresie planowanym (w dniu, tygodniu) przez jeden pojazd.

Należy ustalić taki zbiór marszrut (trasę dostaw towaru) aby:

1. popyt każdego odbiorcy był zrealizowany przez jeden pojazd.

2. ładowność każdego pojazdu nie była przekroczona

3. długość wszystkich marszrut była minimalna

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Wprowadzamy oznaczenia:

n - liczba odbiorców towaru (punktów odbioru)

P - zbiór indeksów wszystkich punktów odbioru $P = \{1, 2, ..., n\}$

$\overline{P}$ - zbiór indeksów wszystkich punktów odbioru i dostawy $\overline{P} = \{0, 1, 2, ..., n\}$

m - liczba pojazdów

V - zbiór wszystkich połączeń pomiędzy punktami (możliwych marszrut)

$V = \{\langle i, j \rangle : i, j \in \overline{P} \wedge i \neq j\}$

w - jednakowa ładowność wszystkich pojazdów

b_j - popyt j-tego odbiorcy

c_{ij} - odległość od punktu i do punktu j (długość trasy $\langle i, j \rangle$)

Zgodnie z przyjętymi założeniami dane te muszą spełniać warunki:

$$\sum_{j=1}^{n} b_j \leq m \cdot w \text{ oraz } b_j < w, j \in P$$

Dla każdego pojazdu wyznaczamy jedną marszrutę (łącznie będzie ich m).

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Przyjmiemy następujące zmienne decyzyjne:

x_{ij} - ilość towaru (dobra) przewożona na trasie $\langle i, j \rangle$

$$y_{ij} = \begin{cases} 1, & \text{gdy pojazd pokonuje trasę } \langle i, j \rangle \\ 0, & \text{w przypadku przeciwnym} \end{cases}$$

Zadanie decyzyjne (PML) będzie miało postać: Znaleźć takie wartości zmiennych x_{ij} oraz y_{ij}, aby:

Funkcja celu: $\sum_{\langle i,j \rangle \in V} c_{ij} \cdot y_{ij} \rightarrow min$

przy warunkach ograniczających:

(1) $\sum_{i \in P} y_{ij} = 1$, dla $(j \in P)$

(2) $\sum_{i \in P} y_{ji} = 1$, dla $(j \in P)$

warunki (1) i (2) oznaczają, że dla każdego odbiorcy wjeżdża i z każdego wyjeżdża jeden pojazd

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

(3) $\sum_{i \in P} y_{i0} = m$

(4) $\sum_{i \in P} y_{0i} = m$

warunki (3) i (4) wymuszają, aby z bazy wyjechało i do niej wróciło dokładnie m - pojazdów

(5) $\sum_{i \in P} x_{ij} - \sum_{i \in P} x_{ji} = b_j, \ (j \in P)$

warunek (5) oznacza, że w każdym punkcie zostawiamy tyle ile wynosi jego popyt

(6) $\sum_{j \in P} x_{0j} = \sum_{j \in P} b_j$

warunek (6) pozwala wywieźć z bazy tyle towaru ile wynosi łączny popyt odbiorców

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

(7) $x_{ij} \le w \cdot y_{ij}, \ (\langle i,j \rangle \in V)$

warunek (7) zapewnia, że na każdej trasie przewieziemy towaru nie więcej niż wynosi ładowność pojazdu. Jeśli danej trasy pojazd nie pokonuje, to przewóz towaru na tej trasie jest zerowy.

(8) $x_{ij} \ge 0, \ (\langle i,j \rangle \in V)$

(9) $y_{ij} \in \{0,1\}, \ (\langle i,j \rangle \in V)$

Zadanie rozwózki jest zadaniem o dużych wymiarach.

liczba zmiennych, to: $L(z) = 2n(n+1)$

liczba warunków: $L(w) = 3n + n(n+1) + 3$

Dla n=30 odbiorców mamy 8450 zmiennych i 483 warunki.

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Zagadnienie komiwojażera - klasyczny problem optymalizacji dyskretnej

Komiwojażer (dawny sprzedawca objeżdżający domy i oferujący produkty) wyrusza z pewnego miasta (z bazy), ma odwiedzić kilka miejscowości i wrócić do punktu startu. każde z miast może być odwiedzone tylko raz i w dowolnej kolejności.

Dany jest zbiór miast (i=1,2,....,n) oraz nieujemna, kwadratowa macierz odległości (kosztu lub czasu przejazdu) $C = \left[c_{ij} \right] i = 1,...,n; j = 1,...,n$. Należy znaleźć taką drogę zamkniętą, przechodzącą przez wszystkie miejscowości, która jest minimalna.

Droga zamknięta jest zwana dalej marszrutą i składa się z n odcinków, które będziemy nazywać trasami. Ponieważ marszruta nie może zawierać trasy $\langle i,i \rangle$, więc przyjmujemy, że $c_{ii} = \infty$ dla i=1,2,...,n. Łączna liczba marszrut w problemie komiwojażera jest równa $(n-1)!$. Dla n=10 mamy $9! = 362800$ różnych rozwiązań. Przegląd zupełny zbioru rozwiązań w celu znalezienia optymalnego jest efektywny tylko dla małych $n(n \leq 8)$.

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Oznaczmy przez:

$V = \{\langle i, j\rangle : i \neq j; i, j = 1,2,...,n\}$ - niech będzie zbiorem wszystkich możliwych tras

Zmienne decyzyjne:

$$x_{ij} = \begin{cases} 1, & \text{gdy marszruta zawiera trasę } \langle i, j\rangle \\ 0, & \text{w przypadku przeciwnym} \end{cases}$$ - zmienna binarna

z_j - zmienna całkowita, która każdemu miastu j-temu przyporządkowuje cechę - kolejność odwiedzenia tego miasta (przy założeni że dla punktu startu - bazy $z_{j_B} = 0$)

Jeżeli n=5 miast oraz $j_B = 1$ (miasto o numerze 1-baza), to przykładowa marszruta może być postaci: $(1,4,5,3,2,1)$, a zmienne kolejności odwiedzeń: $z_1 = 0, z_4 = 1, z_5 = 2, z_3 = 3, z_2 = 4$ (oczywiście miasto startu posiadające cechę $z_1 = 0$ nie może mieć drugiej cechy równej 5)

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

Problem komiwojażera sprowadza się do następującego zadania decyzyjnego (PML):

Wyznaczyć takie wartości zmiennych: x_{ij} oraz z_j, aby:

funkcja celu: $\sum_{\langle i,j\rangle \in V} c_{ij} x_{ij} \rightarrow \min$

przy warunkach ograniczających:

(1) $\sum_{i=1}^{n} x_{ij} = 1,$ dla $(j = 1, 2, ..., n)$

(2) $\sum_{j=1}^{n} x_{ij} = 1,$ dla $(i = 1, 2, ..., n)$

warunki (1) i (2) oznaczają, że komiwojażer przez każdy punkt przejeżdża tylko jeden raz

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłady dyskretnych problemów decyzyjnych

(3) $z_i - z_j + nx_{ij} \le n-1$, dla $(i=1,2,...,n;\ j=2,3,...,n;\ i \ne j)$

niestety (1) i (2) nie gwarantują, że z wybranych n tras tworzymy tylko jedną marszrutę zamkniętą. Warunek (3) wyklucza możliwość powstawania tzw. podcykli w tworzonej marszrucie.

$z_j \ge 0, z_j \in C,\ (j=1,2,...,n)$

$x_{ij} \in \{0,1\},\ (\langle i,j \rangle \in V)$

Zadanie komiwojażera jest zadaniem o dużych rozmiarach.

Liczba zmiennych to: $L(z) = n + n(n-1) = n^2$

Liczba warunków to: $L(w) = 2n + n(n-1) - (n-1) = 2n + (n-1)^2$

Dla n=10 mamy 100 zmiennych oraz 101 warunków.

❑ Liniowe Modele Optymalizacji Dyskretnej – przykład zastosowania algorytmu węgierskiego

Przykład: W pewnym magazynie pracuje 3 pracowników magazynowych: P1, P2, P3, którzy mogą wykonywać 4 rodzaje zadań: Z1, Z2, Z3, Z4, z różną wydajnością. W tabeli poniżej podana jest wydajność pracowników przy wykonywaniu poszczególnych zadań:

Pracownicy	Wydajność pracowników (szt./godz.) przy wykonywaniu zadań magazynowych			
	Z1	Z2	Z3	Z4
P1	15	4	5	2
P2	3	6	3	10
P3	12	4	6	3

Zakładając specjalizację w ciągu dnia pracowników przy wykonywaniu tylko jednego zadania, przydzielić zadania poszczególnym pracownikom, tak aby zmaksymalizować łączną wydajność ich pracy.

Ponieważ w problemie optymalnego przydziału zakłada się, że liczba stanowisk pracy jest taka sama jak liczba pracowników, to w naszym przykładzie musimy wprowadzić czwartego fikcyjnego pracownika. Oczywiście wydajność jego pracy dla poszczególnych zadań będzie równa 0.

Macierz wydajności pracy (współczynników funkcji celu) jest więc postaci:

$$W = \begin{bmatrix} 15 & 4 & 5 & 2 \\ 3 & 6 & 3 & 10 \\ 12 & 4 & 6 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłada zastosowania algorytmu węgierskiego

Matematyczny model zadania:

$$F(x_{i,j}) = 15x_{1,1} + 4x_{1,2} + 5x_{1,3} + 2x_{1,4} + 3x_{2,1} + 6x_{2,2} + 3x_{2,3} + 10x_{2,4} +$$
$$+ 12x_{3,1} + 4x_{3,2} + 6x_{3,3} + 3x_{3,4} \rightarrow \max$$

$$\begin{cases} x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1 \\ x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1 \\ x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 1 \\ x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} = 1 \\ x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} = 1 \\ x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} = 1 \\ x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} = 1 \\ x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} = 1 \end{cases} \qquad x_{i,j} = 0 \text{ lub } x_{i,j} = 1; \quad i, j = 1,\ldots,4;$$

Rozwiążemy zadanie korzystając z wersji algorytmu węgierskiego, która zakłada, że funkcja celu jest postaci - minimum. Dlatego w rozwiązaniu będziemy minimalizować funkcję przeciwną do funkcji celu: $-F(x_{i,j})$, dla której macierz współczynników jest postaci:

$$W = \begin{bmatrix} -15 & -4 & -5 & -2 \\ -3 & -6 & -3 & -10 \\ -12 & -4 & -6 & -3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłada zastosowania algorytmu węgierskiego

Krok 1: Przekształcenie macierzy: **W** – tak, aby w każdym wierszu i każdej kolumnie znalazło się co najmniej jedno zero;

$$W = \begin{bmatrix} -15 & -4 & -5 & -2 \\ -3 & -6 & -3 & -10 \\ -12 & -4 & -6 & -3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

element najmniejszy

$$W = \begin{bmatrix} 0 & 11 & 10 & 13 \\ 7 & 4 & 7 & 0 \\ 0 & 8 & 6 & 9 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

-2 - (-15) = 13

Krok 2: Skreślenie w przekształconej macierzy współczynników funkcji celu wierszy oraz kolumn zawierających zero możliwie najmniejszą liczbą linii;

$$W = \begin{bmatrix} 0 & 11 & 10 & 13 \\ 7 & 4 & 7 & 0 \\ 0 & 8 & 6 & 9 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

trzy linie – zatem przechodzimy do kroku 4

Jeżeli najmniejsza liczba linii konieczna do pokrycia wszystkich zer jest równa wymiarowi macierzy (**czyli - n**), to rozwiązanie, które otrzymamy na podstawie tak przekształconej macierzy współczynników będzie optymalne – przechodzimy do **kroku 3**. Jeżeli jest ona mniejsza niż wymiar macierzy – **W**, to przechodzimy do **kroku 4**.

❑ Liniowe Modele Optymalizacji Dyskretnej – przykłada zastosowania algorytmu węgierskiego

Krok 3: Ustalić tak rozwiązanie optymalne, aby w macierzy $[x^*_{i,j}]$ jedynki znalazły się tylko na tych miejscach, gdzie są zera w przekształconej macierzy – **W** (musimy dbać także, aby w każdym wierszu i każdej kolumnie była tylko jedna jedynka).

Krok 4: Gdy liczba linii pokrywających zera jest mniejsza od wymiaru macierzy współczynników, to w bieżącej (przekształconej) macierzy współczynników należy znaleźć element najmniejszy oraz:

- odjąć go od elementów nieskreślonych;
- dodać go do elementów podwójnie skreślonych;
- elementy skreślone jedną linią (raz) pozostawiamy bez zmian;

$$W = \begin{bmatrix} 0 & 11 & 10 & 13 \\ 7 & 4 & 7 & 0 \\ 0 & 8 & 6 & 9 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

element minimalny

$$W = \begin{bmatrix} 0 & 5 & 4 & 7 \\ 13 & 4 & 7 & 0 \\ 0 & 2 & 0 & 3 \\ 6 & 0 & 0 & 0 \end{bmatrix}$$

elementy odjęte

Powrót do **kroku 2** i powtórzenie procedury, aż do uzyskania rozwiązania optymalnego.

elementy dodane

$$W = \begin{bmatrix} 0 & 5 & 4 & 7 \\ 13 & 4 & 7 & 0 \\ 0 & 2 & 0 & 3 \\ 6 & 0 & 0 & 0 \end{bmatrix}$$

cztery linie zatem rozwiązanie optymalne (krok 3)

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$F(x_{i,j}) = 15 + 10 + 6 = 31 \text{ [szt./godz.]}$$

MAKSYMALIZACJA PRZEPŁYWU SIECIOWEGO ALGORYTM: FORDA - FULKERSONA

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Bardzo ważną z punktu widzenia zastosowań praktycznych grupę zagadnień sieciowych, stanowią zagadnienia dotyczące optymalizacji (maksymalizacji) przepływu sieciowego w sieciach transportowych typu **Forda – Fulkersona** (nazwa pochodzi od nazwisk twórców algorytmu optymalizacyjnego rozwiązującego tego typu zagadnienia).

Będziemy rozważać **zagadnienie sterowania przepływem** jednorodnego produktu **od dostawcy** (dostawców) poprzez **punkty pośrednie**, aż do **odbiorcy** (odbiorców).

Tego typu zagadnienia są bardzo często spotykane w praktyce:

- wysyłka gotowego produktu z fabryki do hurtowni, a później do sklepów;
- przepływ prądu z elektrowni po liniach wysokiego napięcia do stacji transformatorowych, a później przez linie średniego i niskiego napięcia do odbiorców finalnych;

Powiązania pomiędzy dostawcami, punktami pośrednimi a odbiorcami wygodnie jest przedstawiać w postaci pewnej sieci transportowej, która posiada następujące własności.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Charakterystyczne cechy sieci Forda – Fulkersona:

Występujące we wzorach symbole oznaczają odpowiednio: $|V|$ - ilość elementów zbioru **V**, $\delta : V^2 \to \{0,1\}$ - jest funkcją incydencji (połączenia) za pomocą której określany jest zbiór łuków **U** w grafie **G**. Zbiór ten jest definiowany następująco:

$$U = \left\{ u : u = \langle v_\alpha, v_\beta \rangle \wedge v_\alpha, v_\beta \in V \wedge \delta_{\alpha\beta} = \delta(v_\alpha, v_\beta) = 1 \right\}.$$

(i) $V = V_0 \cup V_1 \cup \ldots \cup V_m \cup V_{m+1}, |V_0| = |V_{m+1}| = 1 \wedge |V_j| > 1; j = 1,2,\ldots,m; m \geq 2$

Wierzchołki sieci należące do zbioru **V** są rozbite na warstwy, które zawierają **nie mniej niż dwa wierzchołki** (za wyjątkiem warstwy pierwszej oraz ostatniej – w których znajduje się po **jednym wierzchołku** – **tzw. węzeł wejścia i wyjścia**).

(ii) Jeżeli $\delta_{\alpha\beta} = \delta(v_\alpha, v_\beta) = 1 \wedge v_\alpha \in V_j (j = 0,1,\ldots,m) \Rightarrow v_\beta \in V_{j+1}$.

Połączenie łukiem dwóch dowolnych wierzchołków sieci jest możliwe tylko pomiędzy wierzchołkami należącymi do dwóch sąsiadujących warstw (początek łuku – warstwa poprzednia, koniec łuku warstwa następna).

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

(iii) Jeżeli $\left[v_\alpha \in V_i \wedge v_\beta \in V_j \wedge \left(j = i \vee j < i \vee j - i > 1\right)\right] \Rightarrow \delta_{\alpha\beta} = 0$.

Nie jest możliwe połączenie łukiem dwóch wierzchołków należących do tej samej warstwy, lub gdy wierzchołek końcowy (łuku) należy do warstwy wcześniejszej, lub gdy wierzchołek końcowy należy do kolejnej, ale nie bezpośrednio następnej warstwy.

(iv) $$\forall_{j=1,2,\ldots,m;\, v_\beta \in V_j} \left[\sum_{v_\alpha \in V_{j-1}} \delta_{\alpha\beta} = \delta(v_\alpha, v_\beta) \geq 1 \wedge \sum_{v_\lambda \in V_{j+1}} \delta_{\beta\lambda} = \delta(v_\beta, v_\lambda) \geq 1 \right].$$

Istnieje co najmniej jeden łuk łączący dowolny wierzchołek danej warstwy (poza warstwą pierwszą i ostatnią) z pewnym wierzchołkiem warstwy następnej oraz analogicznie co najmniej jeden łuk łączący pewien wierzchołek warstwy poprzedniej z rozważanym wierzchołkiem tej warstwy).

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Zbiór łuków **U** może mieć w praktyce bardzo różnorodną interpretację. Łuki możemy na przykład uważać jako skierowane drogi, kanały (połączeń informatycznych), linie energetyczne lub technologiczne, trasy zaopatrzeń (trasy transportowe), a także jako relacje lub pewne zależności finansowe, administracyjne, strukturalne lub organizacyjne.

Na zbiorze łuków określamy dwie funkcje:

- funkcja $c : V^2 \to R^+$, która określa **przepustowość** poszczególnych łuków w sieci typu Forda – Fulkersona (**uwaga:** funkcja **C** na zbiorze: $V^2 - U$ przyjmuje zawsze wartość zero);
- funkcja $x : U \to R^+$, która określa aktualny przepływ $x_{i,j}$ po łukach (i, j) zgodnie z ich orientacją;

Ponieważ wartości funkcji **C** dla poszczególnych elementów ze zbioru łuków „U" możemy uzależnić od dwóch indeksów: $C(u = \langle \alpha, \beta \rangle \in U) = c_{\alpha\beta}$, gdzie **alfa** – indeks wierzchołka początkowego, zaś **beta** – indeks wierzchołka końcowego łuku, to wartości funkcji **C** tworzą macierz kwadratową zwaną **macierzą przepustowości** postaci:

$$C = \left\lfloor c_{\alpha\beta} \right\rfloor_{\alpha,\beta=1,2,\ldots,n}$$

Sieć (transportowa) w sensie Forda – Fulkersona jest grafem $G = \langle V, U, c, x \rangle$ z funkcjami obciążającymi łuki, spełniającym warunki (i)-(iv).

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Wykorzystanie sieci typu Forda – Fulkersona do problemu optymalizacji przewozów w sieci transportowej:

Rozważmy dla przykładu zadanie optymalizacyjne efektywności transportu przez pewną **firmę spedycyjną** - polegające na przetransportowaniu jak największej ilości towarów od wierzchołka wejściowego (w naszym przykładzie wierzchołek v_1 - **nadawca towarów**) sieci transportowej **G** do jej wierzchołka wyjściowego (w przykładzie v_7 - **odbiorca towarów**) po skierowanych łukach (kanałach spedycyjnych) przy uwzględnieniu ograniczeń na ich przepustowość (**ograniczenia ładowności posiadanych środków transportowych**).

Graficzna interpretacja zadania:

Rozważana sieć transportowa w sensie Forda - Fulkersona przedstawia się następująco:

$$V_0 = \{1\}, V_1 = \{2,3,4\}, V_2 = \{5,6\}, V_3 = \{7\}.$$

Pomiędzy warstwami sieci **G** zachodzi oczywista równość: $V = V_0 \cup V_1 \cup V_2 \cup V_3$.

Łuki generuje macierz incydencji $\left[\delta_{\alpha\beta}\right]_{\alpha,\beta\in V}$, gdzie poszczególne jej składowe są postaci:

$\delta_{12} = \delta_{13} = \delta_{14} = 1$, $\delta_{25} = \delta_{26} = 1$, $\delta_{35} = 1$, $\delta_{45} = \delta_{46} = 1$, $\delta_{57} = 1$, $\delta_{67} = 1$, a na pozostałych parach wierzchołków (węzłów) zachodzi: $\delta_{ij} = 0$.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Maksymalna ładowność środków transportowych na poszczególnych etapach transportu (zgodnie z macierzą incydencji) podaje tabela:

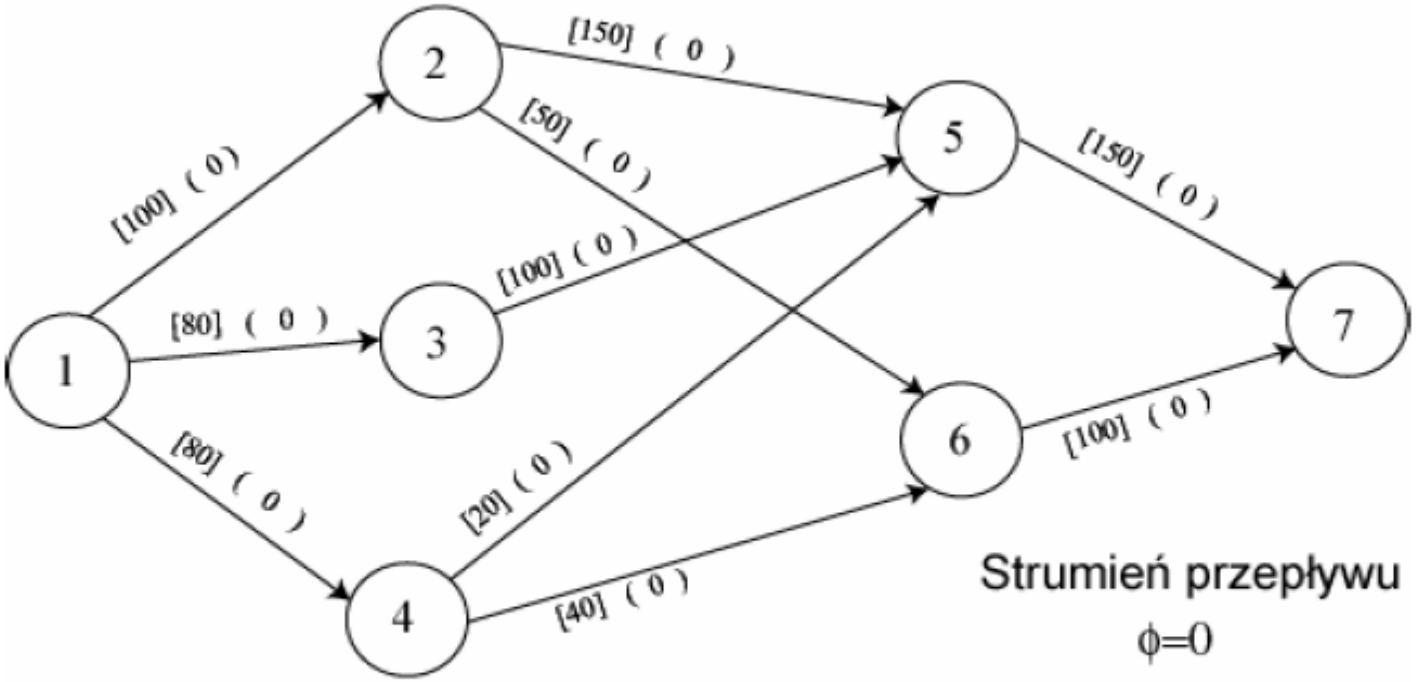
Kanał transportowy $\langle i, j \rangle$	Maksymalna ładowność C_{ij} (w tonach)
⟨1,2⟩	100
⟨1,3⟩	80
⟨1,4⟩	80
⟨2,5⟩	150
⟨2,6⟩	50
⟨3,5⟩	100
⟨4,5⟩	20
⟨4,6⟩	40
⟨5,7⟩	150
⟨6,7⟩	100

Rysunek przedstawia graficzną interpretację zadania za pomocą sieci transportowej Forda – Fulkersona.

Graf ten jest siecią z **przepustowościami** $c_{\alpha\beta}$ naniesionymi na łuki zgodnie z danymi zaczerpniętymi z tabeli. W nawiasach okrągłych przedstawiony jest **aktualny transport** $(x_{ij} \geq 0)$ przez dany kanał transportowy

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

(obecnie wszędzie 0). **Aktualny strumień przepływu** (liczony jako suma $\sum_{i \in \Gamma_7} x_{i7}$ wynosi 0).



Aby najlepiej zrealizować swój cel firma spedycyjna rozwiązuje zadanie transportowe algorytmem Forda – Fulkersona – opublikowanym w pracy:
Ford L. R., Fulkerson D. R., Przepływy w sieciach, PWN, W-wa, 1969.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

W algorytmie tym poszukuje się tzw. **strumienia maksymalnego** Φ_{max} określającego jaką maksymalną ilość towarów z węzła **początkowego** możemy łukami (drogami dystrybucji) przetransportować do węzła **końcowego** nie przekraczając jednocześnie przepustowości (dopuszczalnej ładowności) na poszczególnych łukach sieci.

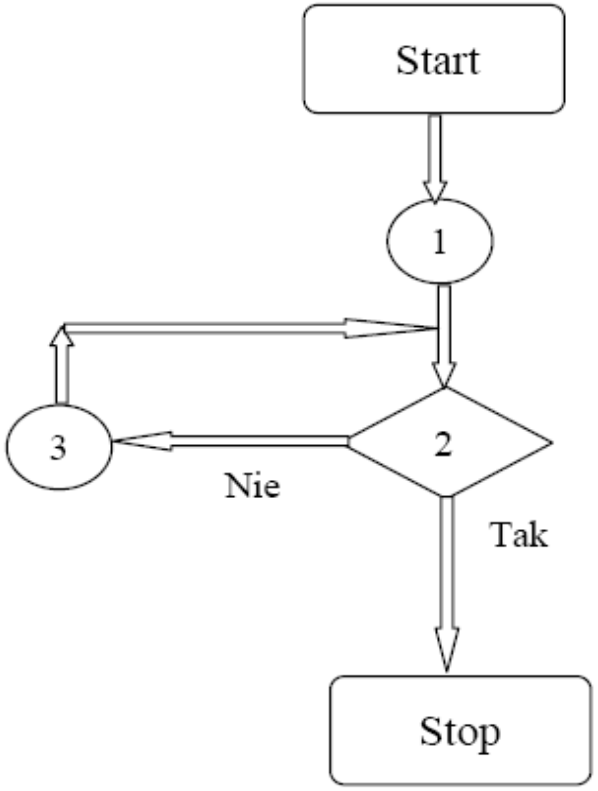
Idea algorytmu jest następująca:
Najpierw poszukuje się jakiegokolwiek dopuszczalnego **strumienia** Φ przepływu (spełniającego tzw. **Prawo Kirchoffa** dla sieci), taki strumień początkowy zawsze istnieje (w najgorszym wypadku jest to strumień zerowy). W dalszych krokach algorytmu strumień ten odpowiednio się poprawia, nasycając kolejne łuki, tak aby uzyskać wreszcie wartość maksymalną.

W sieci forda – fulkersona dla każdego wierzchołka grafu zachowane jest tzw. **I prawo Kirchoffa dla przepływu** (odpowiednik prawa Kirchoffa dla prądu elektrycznego): **Sumaryczny wpływ do każdego wierzchołka sieci za wyjątkiem wierzchołka wejściowego i wyjściowego jest równy sumarycznemu wypływowi z tego węzła (wpływ bilansuje wypływ).**
Zatem dla każdego $j \in V - \{1, n\}$ zachodzi $\sum_{i \in \Gamma_j^-} x_{i,j} = \sum_{k \in \Gamma_j^+} x_{j,k}$.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Powiemy, że łuk $\langle i, j \rangle$ jest **nasycony**, jeśli jego przepustowość jest całkowicie wykorzystana (realizacja dystrybucji towarów wymaga na tym etapie wykorzystania transportu o maksymalnie dostępnej ładowności), tzn. $x_{i,j} = c_{i,j}$.

Schemat algorytmu Forda – Fulkersona:



Krok 1 – wyznaczenie jakiegokolwiek początkowego strumienia przepływu (spełniającego prawo Kirchoffa).

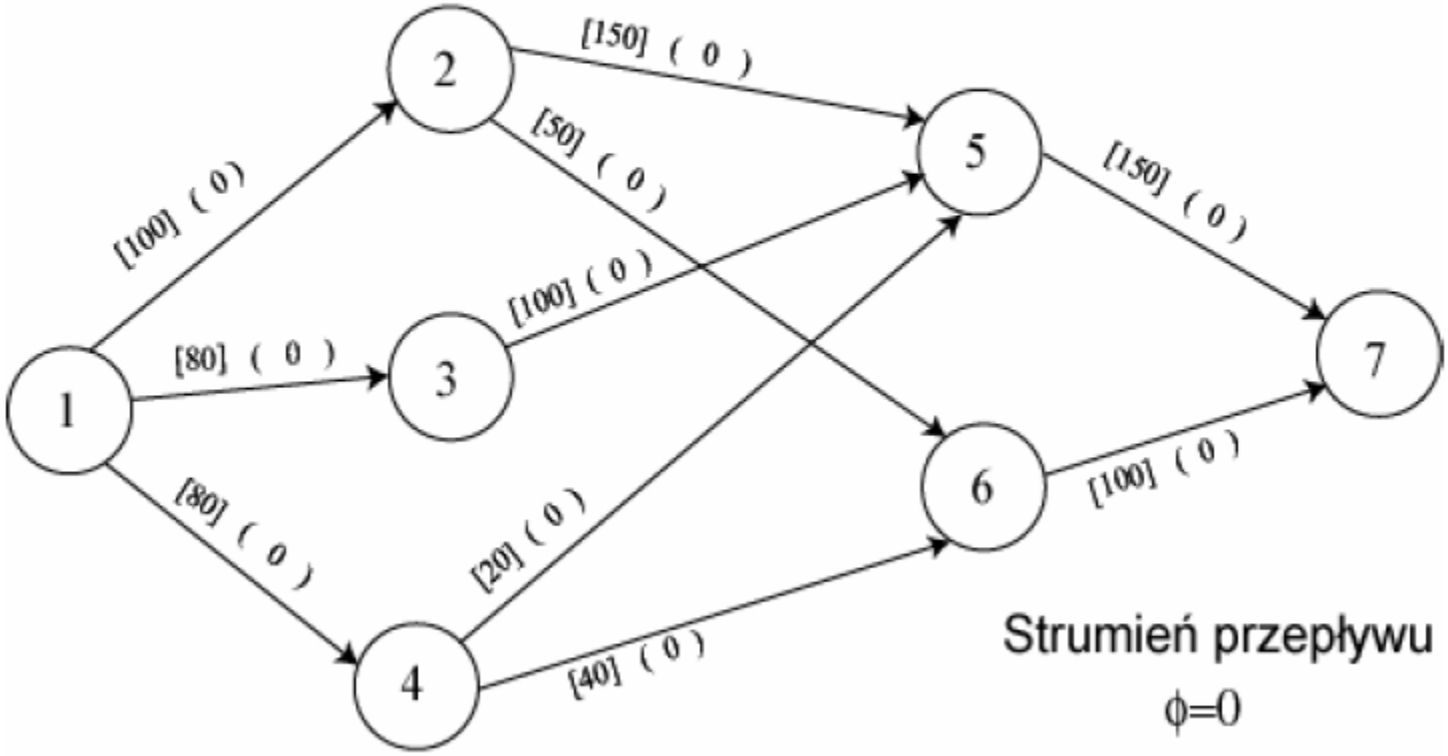
Krok 2 – sprawdzenie warunku czy wyznaczony strumień przepływu jest maksymalny (**wykorzystanie procedury cechowania wierzchołków – kryterium max przepływu**).

Krok 3 – polepszenie wartości strumienia poprzez lepsze nasycenie jego łuków.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Realizacja kroku 1:

W kroku tym staramy się wyznaczyć jakikolwiek strumień początkowy. Początkową dystrybucję towarów (przy najgorszym zerowym strumieniu przepływu) przedstawia rysunek.



❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Realizacja kroku 2:

Aby zbadać czy strumień zupełny jest maksymalny należy zastosować procedurę cechowania wierzchołków zaproponowaną przez **Forda i Fulkersona**.

Cechowanie wierzchołków zawsze zaczynamy od wierzchołka wejściowego v_1, któremu nadajemy cechę $(-,\infty)$. Wierzchołek ten staje się **ocechowany** ale **niezbadany**.

Następnie bierzemy pod uwagę kolejny wierzchołek ocechowany, ale jeszcze niezbadany i rozpatrujemy wszystkie wierzchołki **nieocechowane** do niego **sąsiednie** (tzn. z niego wychodzące lub do niego wchodzące) i staramy się je ocechować zgodnie z **procedurą**:

Oznaczenia: **j** – numer wierzchołka ocechowanego badanego, **l** - numer wierzchołka, który cechujemy.

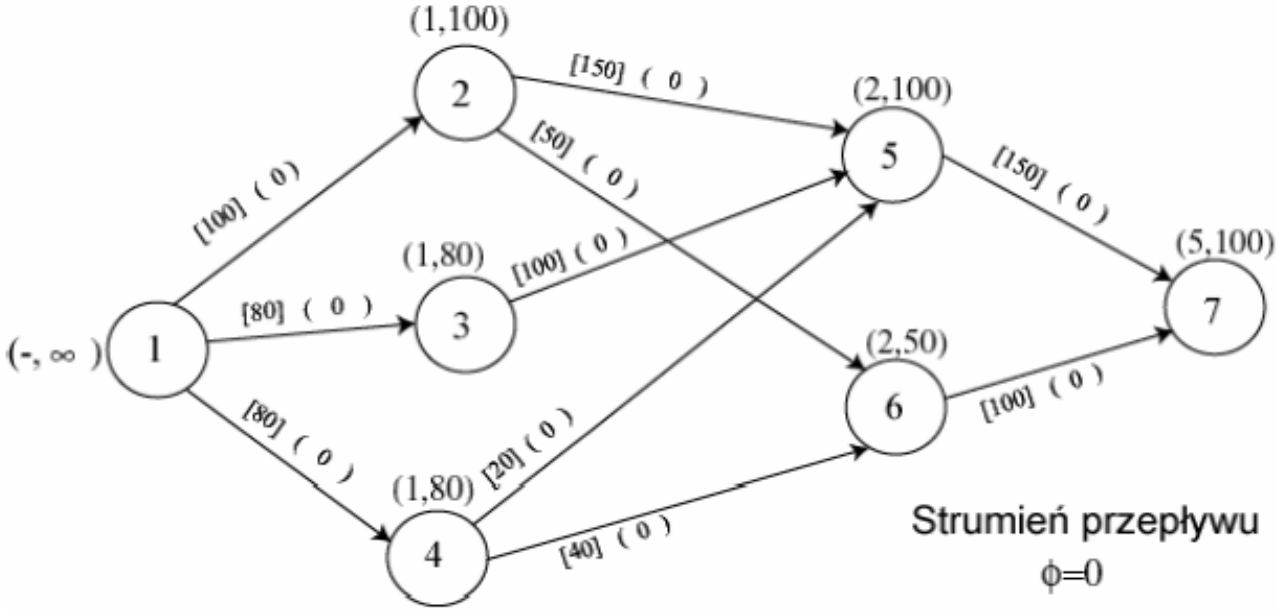
- Jeżeli $(j,l) \in U$ oraz $x_{j,l} < c_{j,l}$ (**nie jest to łuk maksymalnie nasycony**), to wierzchołkowi: **l** - nadajemy cechę $(j, \varepsilon_l = \min\{\varepsilon_j, c_{j,l} - x_{j,l}\}$.
- Jeżeli $(l,j) \in U$ oraz $x_{l,j} > 0$ (**istnieje już jakiś przepływ przez ten łuk**), to wierzchołkowi: **l** - nadajemy cechę $(j, -\varepsilon_l = \min\{\varepsilon_j, x_{l,j}\}$.

Wierzchołek: **j** - badany staje się wtedy **ocechowany i zbadany**. Przechodzimy do kolejnego wierzchołka ocechowanego i niezbadanego i procedurę cechowania powtarzamy, aż uda się ocechować **wierzchołek wyjściowy** (v_n).

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Kryterium optymalności (maksymalności) strumienia przepływu:
Aktualny strumień przepływu jest maksymalny, jeżeli stosując procedurę cechowania wierzchołków nie da się ocechować węzła wyjścia.

Stosując procedurę cechowania wierzchołków do naszego przykładu otrzymujemy:

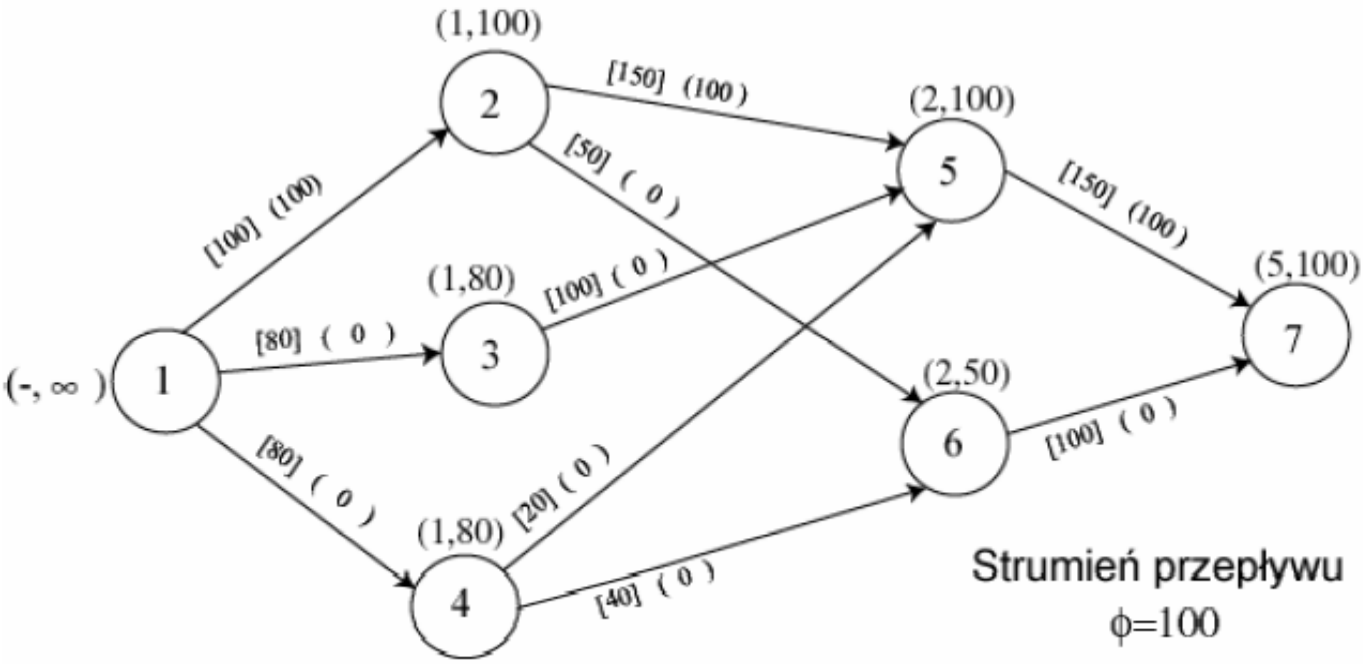


❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Możliwe było ocechowanie wierzchołka wyjściowego zatem (co było do przewidzenia) strumień zerowy nie jest maksymalny.

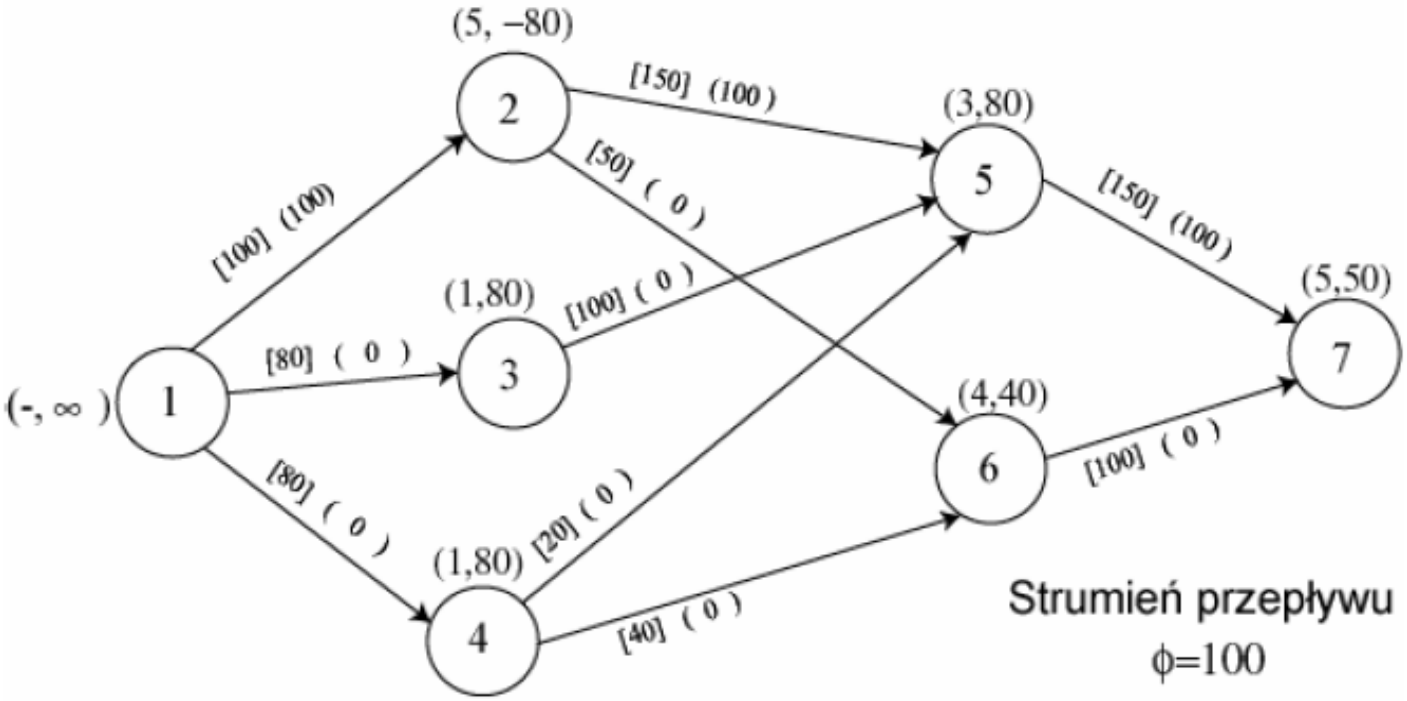
Realizacja kroku 3:

Procedura cechowania wierzchołków podaje ponadto o ile możemy poprawić wartość aktualnego strumienia. Mówi o tym wartość: $\varepsilon_7 = 100$ **przypisana wierzchołkowi wyjścia**. Pierwsze składowe cech podają nam również **ścieżkę zwiększającą ten przepływ: 1 – 2 – 5 – 7**.



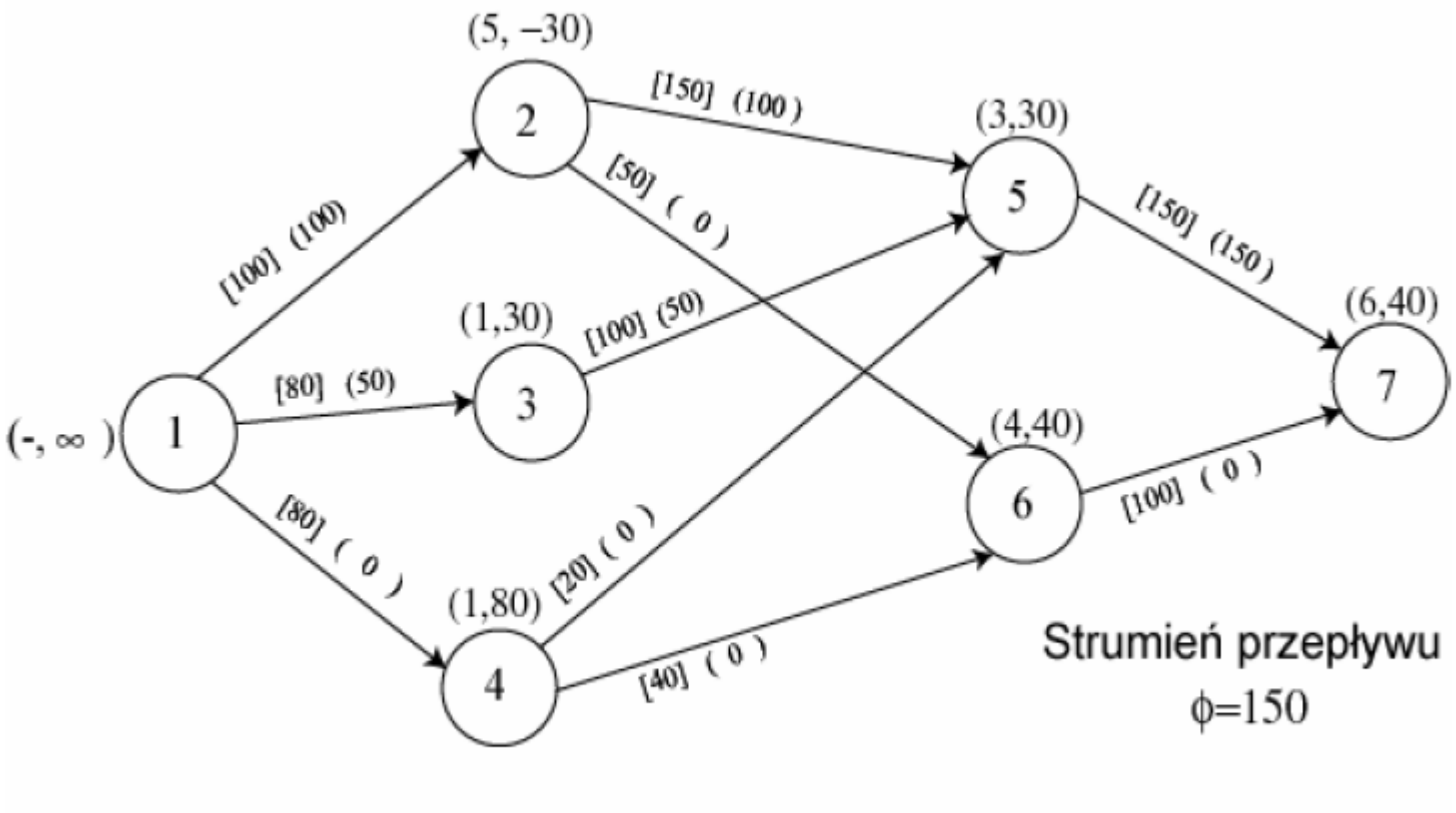
❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Ponownie **powracamy do kroku 2** (sprawdzając czy strumień aktualny $\phi = 100$) jest maksymalny, **itd**. Kolejne etapy przebiegu algorytmu dla naszego przykładu przedstawiają poniższe rysunki.



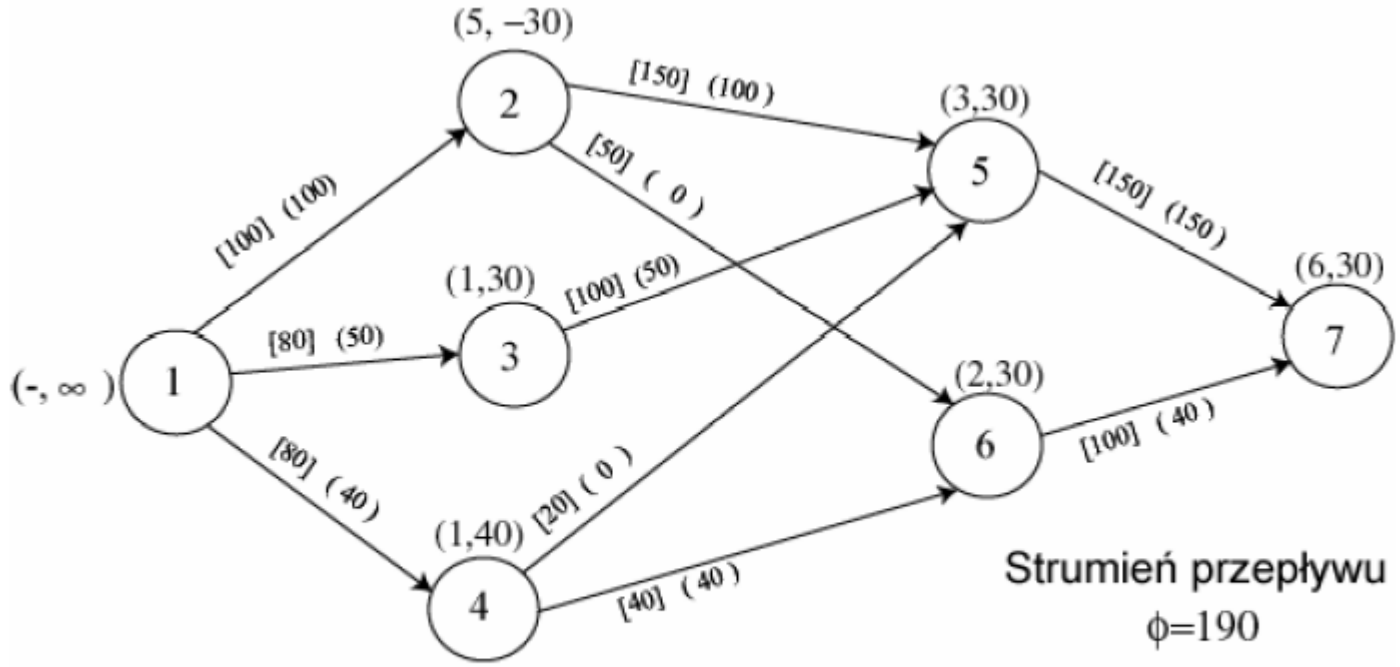
Ścieżka zwiększająca przepływ **o 50** jest postaci: **1 – 3 – 5 – 7**.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



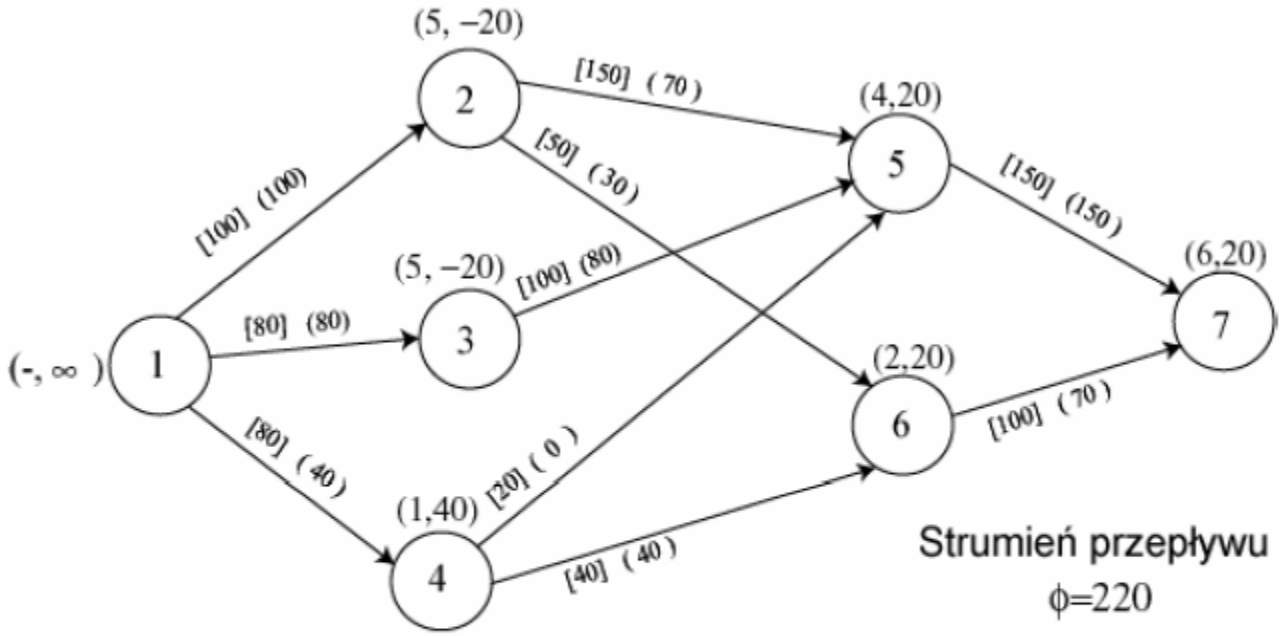
Ścieżka zwiększająca przepływ o 40 jest postaci: 1 – 4 – 6 – 7.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



Ścieżka zwiększająca przepływ o 30 jest postaci: 1 – 3 – 5 – 2 – 6 – 7.

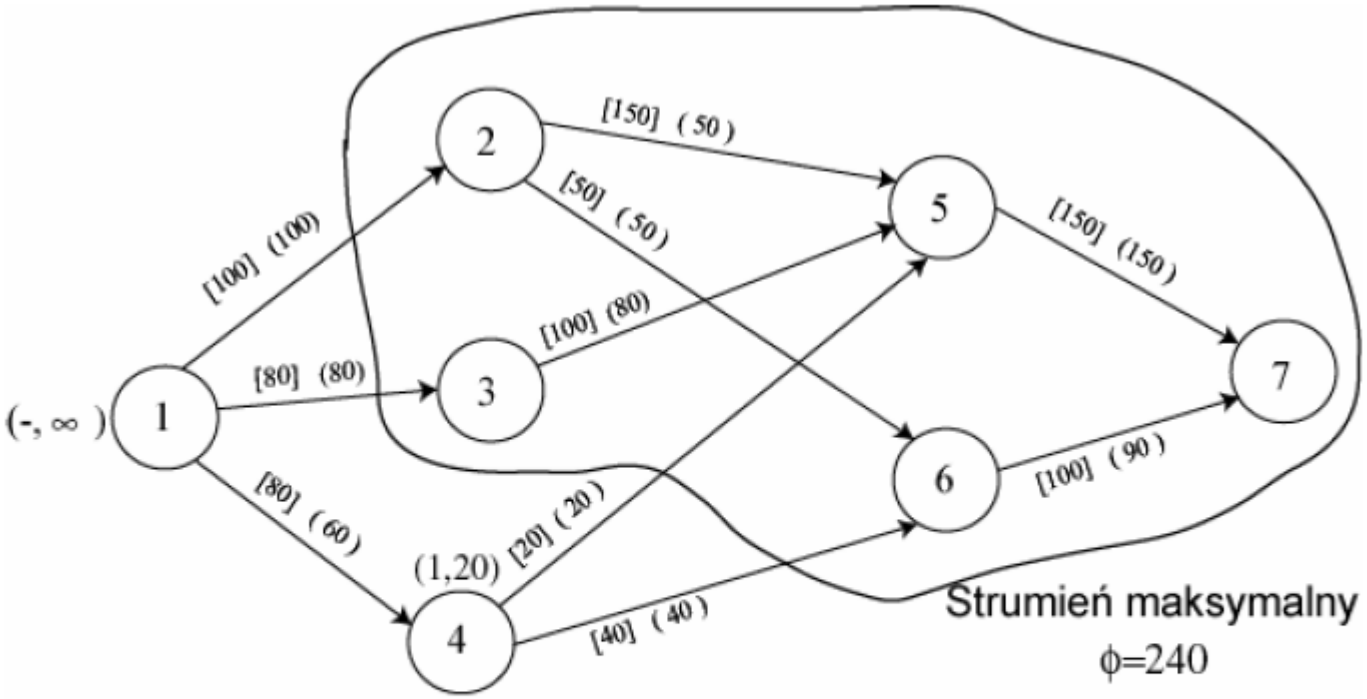
❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



(poprzednia dystrybucja)

Ścieżka zwiększająca przepływ **o 20** jest postaci: **1 – 4 – 5 – 2 – 6 – 7**.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



Otrzymany strumień przepływu jest zarazem strumieniem maksymalnym $\Phi = 240 = \Phi_{max}$, gdyż nie jest możliwe zgodnie z kryterium optymalności strumienia - stosując procedurę cechowania oznaczenie wierzchołka wyjściowego v_7.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

W ostatnim kroku algorytmu otrzymaliśmy rozbicie zbioru wierzchołków **V** sieci na dwa zbiory: wierzchołków **ocechowanych**: $V^O = \{1,4\}$ - zawierający zawsze wierzchołek wejścia i wierzchołków **nieocechowanych**: $V^N = \{2,3,5,6,7\}$ - zawierający zawsze wierzchołek wyjścia.

Zauważmy, że wszystkie łuki prowadzące do wierzchołków, których nie można ocechować są nasycone.

- Przekrojem **S** w sieci **G** nazywamy zbiór łuków: $\{(i,j) \in U : i \in P^1 \wedge j \in P^2\}$, gdzie zbiory P^1, P^2 spełniają następujące warunki: $P^1 \cap P^2 = \emptyset$, $P^1 \cup P^2 = V$; $1 \in P^1, n \in P^2$ oraz dla każdego $(i,j) \in U$ zachodzi: $i,j \in P^1$ lub $i,j \in P^2$ albo $i \in P^1, j \in P^2$.

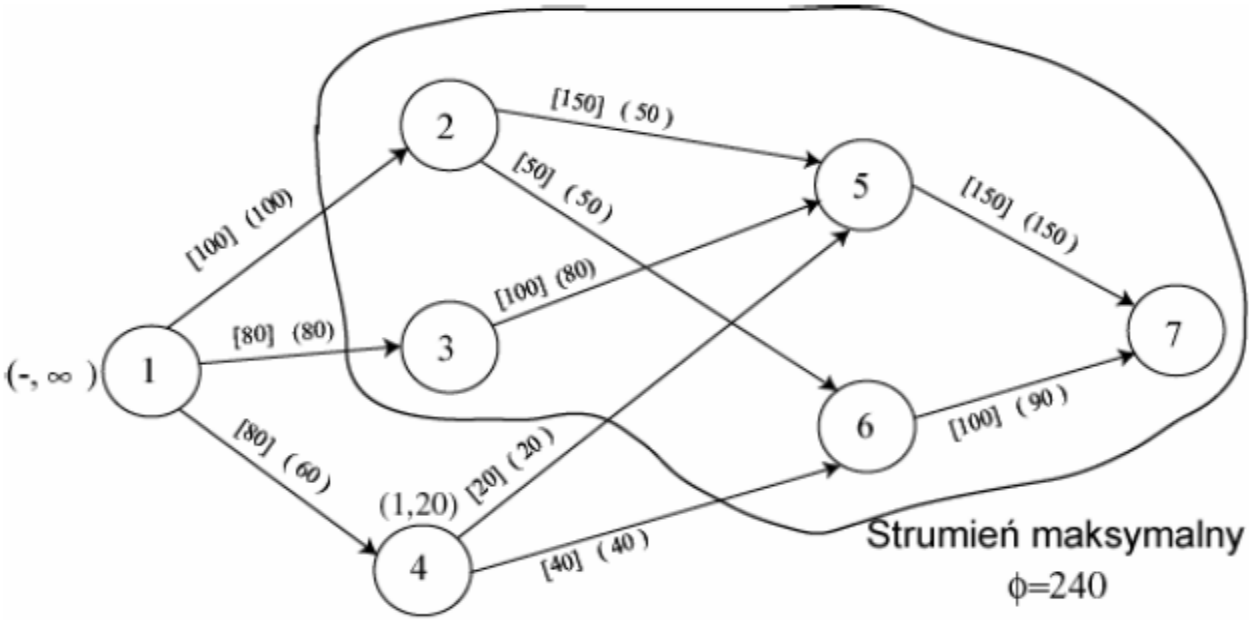
  Jest to każdy **minimalny zbiór łuków**, których usunięcie spowoduje **utratę spójności sieci** (nie będzie istnieć ścieżka prowadząca od v_1 do v_n).

  Do każdej ścieżki $(v_1 \to v_n)$ musi istnieć **co najmniej jeden** z łuków przekroju;

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

- Liczbę $\psi = \psi(S) = \sum_{(i,j)\in S} x_{i,j}$ równą sumie przepustowości łuków przekroju nazywamy **przepustowością przekroju**;

- Przekrojem minimalnym będziemy nazywać przekrój S^* o **minimalnej przepustowości** $\Psi_{\min}$. Jest to przekrój generowany przez zbiór V^N, czyli przekrój postaci: $\{(i,j) \in U : i \in V^O \wedge j \in V^N\}$;



❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

Twierdzenie Forda - Fulkersona:

$$\Phi_{max} = \Psi_{min}$$

Wartość maksymalnego przepływu w sieci równa się wartości minimalnego jej przekroju.

W naszym przykładzie mamy:

$$\Phi_{max} = 150 + 90 = 240 \text{ ton}$$

(liczone na łukach wchodzących do v_7)

Przepustowość przekroju na łukach wchodzących do $V^N = \{2,3,5,6,7\}$ ma wartość: $\Psi_{min} = 100 + 80 + 20 + 40 = 240$ ton

Zatem z twierdzenia Forda – Fulkersona mamy: $\Phi_{max} = \Psi_{min} = 240$.

WIELOKRYTERIALNE ZAGADNIENIA OPTYMALIZACYJNE

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Podstawowe pojęcia

Zjawisko złożone – pewien abstrakcyjny twór związany ze stanem jakościowym rzeczywistych obiektów opisany przez co najmniej dwie cechy (diagnostyczne).

Przykładowe zjawiska złożone:

- poziom rozwoju społeczno-gospodarczego,
- atrakcyjność rynkowa produktów,
- konkurencyjność techniczno-ekonomiczna wyrobów,
- jakość kadry zarządzającej, itp.

Porównanie różnych obiektów w zakresie zjawisk złożonych umożliwia konstrukcja ich rankingu.

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Ranking obiektów – układ, w którym obiekty są uporządkowane nierosnąco ze względu na wartości zmiennej syntetycznej (agregatowej).

Zmienna agregatowa – charakteryzuje obiekty ze względu na oceniane zjawisko złożone; powstaje w wyniku agregacji (najczęściej sumowania) unormowanych zmiennych diagnostycznych.

Unormowane zmienne diagnostyczne – powstają w wyniku przekształcenia (unormowania) oryginalnych cech diagnostycznych (pozbawienie mian, porównywalność rzędu wielkości).

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Przykład: Budowa rankingu obiektów. Agregacja kryteriów decyzyjnych przy pomocy **metody unitaryzacji zerowej (MUZ).**

Możliwych jest 5 różnych wariantów budowy pewnego zakładu produkcyjnego. Warianty te zostały scharakteryzowane za pomocą czterech cech:

X_1 – koszt inwestycji [tys. zł],
X_2 – docelowa roczna zdolność produkcyjna [tys. sztuk],
X_3 – czas wykonania zadania inwestycyjnego [miesiące],
X_4 – przewidywany roczny poziom emisji zanieczyszczeń ze zrealizowanego obiektu [tony].

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Oszacowane wartości wymienionych charakterystyk wariantów przedsięwzięcia podano w tabeli.

Zbudować ranking wariantów (wybrać najlepszy wariant).

Warianty	Cechy (zmienne diagnostyczne)			
	X_1	X_2	X_3	X_4
W1	253	18,3	24	13,1
W2	178	16,8	18	12,2
W3	244	24,6	26	18,4
W4	174	16,4	25	15,8
W5	196	17,7	20	16,7
max x_{ij}	253	24,6	26	18,4
min x_{ij}	174	16,4	18	12,2

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Rozwiązanie:

1. Rozpoznanie typu zmiennych diagnostycznych.

Stymulanta – większe wartości wskazują na bardziej korzystny wariant.

Destymulanta – mniejsze wartości oznaczają bardziej korzystny wariant.

Nominanta – ma określoną, najkorzystniejszą wartość (nominalną) lub przedział takich wartości.

Stymulanta: X_2 **Destymulanty: X_1, X_3, X_4**

Nominant - brak

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

2. Normowanie zmiennych diagnostycznych (MUZ).

Z_j – unormowana j-ta zmienna diagnostyczna.
z_{ij} – i-ta wartość unormowanej j-ej zmiennej diagnostycznej.

Dla stymulant:

$$z_{ij} = \frac{x_{ij} - \min\limits_{i} x_{ij}}{\max\limits_{i} x_{ij} - \min\limits_{i} x_{ij}} \qquad (1)$$

Dla destymulant:

$$z_{ij} = \frac{\max\limits_{i} x_{ij} - x_{ij}}{\max\limits_{i} x_{ij} - \min\limits_{i} x_{ij}} \qquad (2)$$

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Dla nominant:

$$z_{ij} = \begin{cases} \dfrac{x_{ij} - \min\limits_i x_{ij}}{c_{1j} - \min\limits_i x_{ij}}, & \text{gdy} \quad x_{ij} < c_{1j}, \\ 1, & \text{gdy} \quad c_{1j} \le x_{ij} \le c_{2j}, \\ \dfrac{x_{ij} - \max\limits_i x_{ij}}{c_{2j} - \max\limits_i x_{ij}}, & \text{gdy} \quad x_{ij} > c_{2j}, \end{cases} \qquad (3)$$

gdzie: $[c_{1i}, c_{2i}]$ – przedział wartości nominalnych

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Normowanie zmiennej X_1 – destymulanty (wzór 2). Podobnie normujemy zmienne X_3, X_4

$$z_{11} = \frac{253-253}{253-174} = \frac{0}{79} = 0;$$

$$z_{21} = \frac{253-178}{253-174} = \frac{75}{79} = 0{,}949;$$

$$z_{31} = \frac{253-244}{253-174} = \frac{9}{79} = 0{,}114;$$

$$z_{41} = \frac{253-174}{253-174} = \frac{79}{79} = 1;$$

$$z_{51} = \frac{253-196}{253-174} = \frac{57}{79} = 0{,}722.$$

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Normowanie zmiennej X_2 – stymulanty (wzór 1)

$$z_{12} = \frac{18{,}3 - 16{,}4}{24{,}6 - 16{,}4} = \frac{1{,}9}{8{,}2} = 0{,}232;$$

$$z_{22} = \frac{16{,}8 - 16{,}4}{24{,}6 - 16{,}4} = \frac{0{,}4}{8{,}2} = 0{,}049;$$

$$z_{32} = \frac{24{,}6 - 16{,}4}{24{,}6 - 16{,}4} = \frac{8{,}2}{8{,}2} = 1;$$

$$z_{42} = \frac{16{,}4 - 16{,}4}{24{,}6 - 16{,}4} = \frac{0}{8{,}2} = 0;$$

$$z_{52} = \frac{17{,}7 - 16{,}4}{24{,}6 - 16{,}4} = \frac{1{,}3}{8{,}2} = 0{,}159.$$

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Wyniki normowania przedstawione są w tabeli.

Warianty	Unormowane zmienne diagnostyczne				Q_i
	Z_1	Z_2	Z_3	Z_4	
W1	0	0,232	0,250	0,855	1,337
W2	0,949	0,049	1	1	2,998
W3	0,114	1	0	0	1,114
W4	1	0	0,125	0,419	1,544
W5	0,722	0,159	0,750	0,274	1,905

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

3. Obliczenie wartości zmiennej agregatowej (syntetycznej) dla każdego obiektu (wariantu) i budowa rankingu obiektów.

Q – zmienna agregatowa (syntetyczna), będąca łączną wielokryterialną oceną każdego z obiektów.
Q_i – wartość zmiennej agregatowej przypisana i-temu obiektowi.

$$Q_i = \sum_j z_{ij} \qquad (4)$$

❑ BUDOWA RANKINGU OBIEKTÓW W ŚWIETLE OCEN WIELOKRYTERIALNYCH

Ranking wariantów budowy:

Miejsce w rankingu	Wariant	Q_i
1	W_2	2,998
2	W_5	1,905
3	W_4	1,544
4	W_1	1,337
5	W_3	1,114

Najlepszym wariantem jest W_2.