ANALIZA CZASOWO – KOSZTOWA REALIZACJI PRZEDSIĘWZIĘĆ

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

W rozważaniach dotyczących sieciowej analizy przedsięwzięć z funkcją czasu (metoda CPM) konstruując optymalny harmonogram realizacji przedsięwzięcia przyjętym **kryterium** optymalności była **minimalizacja czasu** realizacji całego przedsięwzięcia.

W analizie ekonomicznej realizowanych przedsięwzięć niezmiernie ważne jest także **podejście kosztowe** – mające na celu oszacowanie kosztów realizacji badanego przedsięwzięcia.

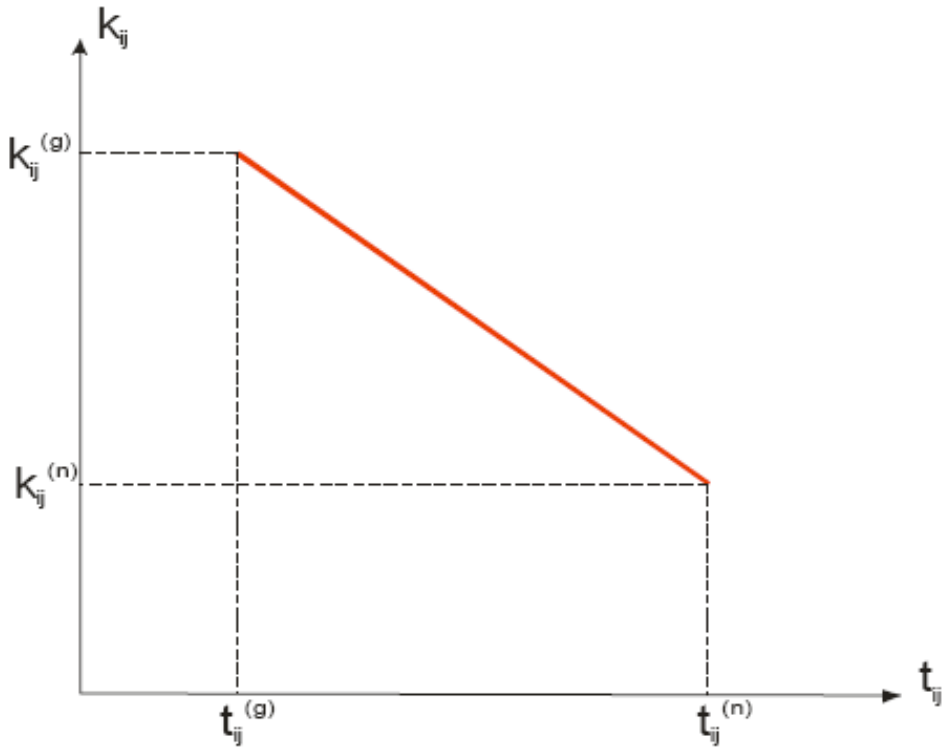
Analiza harmonogramu realizacji przedsięwzięć z uwzględnieniem kosztów ich realizacji nosi nazwę **analizy czasowo – kosztowej** i posiada w planowaniu sieciowym bardzo ważną rolę.

Dotychczas przyjmowaliśmy, że czas trwania poszczególnych czynności jest ustalony i związany jest on z wykonaniem czynności w zwykłych (normalnych warunkach). Taki czas wykonania czynności będziemy nazywać – **normalnym czasem trwania czynności :** $t_{ij}^{(n)}$, a koszt związany z realizacją czynności w tym czasie nazywać będziemy – kosztem normalnym: $k_{ij}^{(n)}$.

W niektórych przypadkach **czasy** trwania czynności mogą **ulec skróceniu**. Wiąże się to jednak za koniecznym zaangażowaniem dodatkowych środków, co pociąga za sobą **wzrost kosztów** wykonania tej czynności.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Najkrótszy możliwy czas, w którym wykonana dana czynność z wykorzystaniem wszystkich zaangażowanych środków – nazywamy **granicznym czasem** trwania czynności: $t_{ij}^{(g)}$. Koszt realizacji czynności w czasie granicznym nazywamy **kosztem granicznym**: $k_{ij}^{(g)}$.



Uwaga:
Zakładamy, że funkcję kosztów w przedziale $\left[t_{ij}^{(g)}, t_{ij}^{(n)}\right]$ można aproksymować funkcją liniową, a skrócenie czasu trwania jednej czynności nie wpływa na czas trwania pozostałych.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Przyjmujemy oznaczenia:

$\Delta t_{ij} = t_{ij}^{(n)} - t_{ij}^{(g)}$ - odcinek czasu o jaki możemy maksymalnie skrócić czynności przedsięwzięcia (wykonywane w czasie normalnym) do czasów granicznych;

$\Delta k_{ij} = k_{ij}^{(g)} - k_{ij}^{(n)}$ - wzrost kosztów realizacji czynności wynikający ze skrócenia czasów trwania czynności do czasów granicznych;

$S_{ij} = \frac{k_{ij}^{(g)} - k_{ij}^{(n)}}{t_{ij}^{(n)} - t_{ij}^{(g)}}$ - **średni gradient kosztu** – określa przyrost kosztów wykonania czynności spowodowany skróceniem czasu trwania czynności o jednostkę.

W tak pojmowanym planowaniu sieciowym bardzo ważnym zagadnieniem programowania sieciowego jest wszechstronna analiza przedsięwzięć w aspekcie ekonomicznym oraz możliwość modyfikacji modelu, poprzez kompresję sieci – wynikającą ze **zbyt długiego** dla inwestora lub odbiorcy okresu realizacji przedsięwzięcia.

Względy ekonomiczne powodują, że należy rozpatrzyć techniczne możliwości **skrócenia** wykonania całego przedsięwzięcia, aby **koszty** jego realizacji były **jak najniższe**.

Wiązać się to będzie z ułożeniem takiego programu przyspieszenia, aby największa akceleracja przypadała na te czynności krytyczne, których koszty przyspieszenia będą najniższe.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

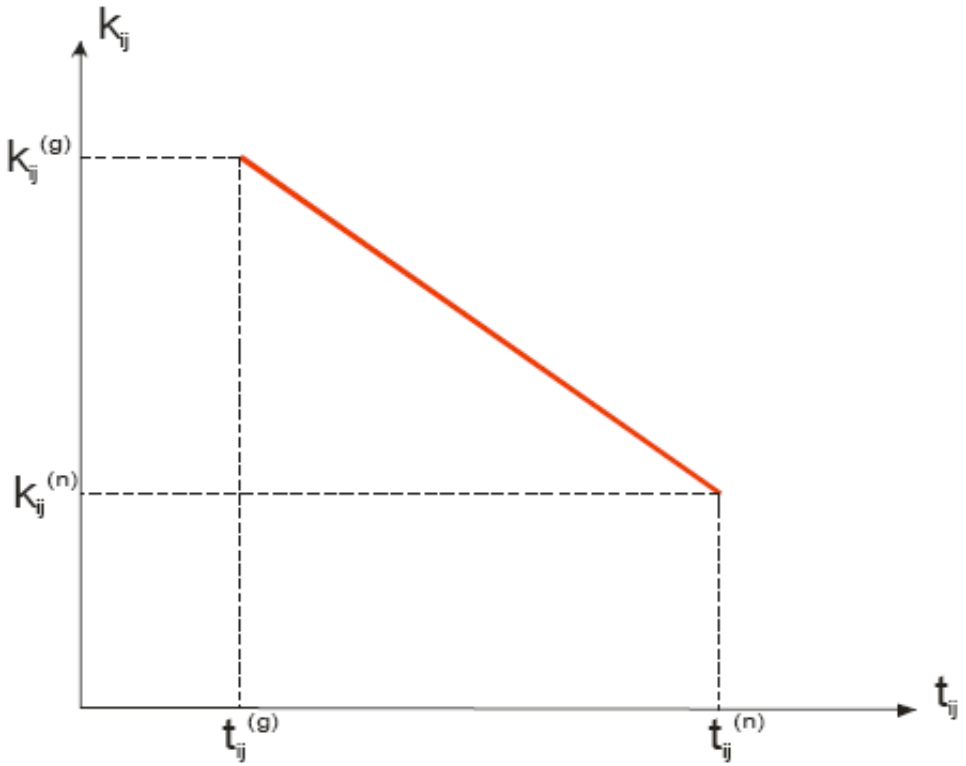
W tego typu analizie rozpatruje się różnego rodzaju koszty składające się na sumaryczny koszt realizacji całego przedsięwzięcia. Uwzględnia się zazwyczaj następujące rodzaje kosztów:

- **Koszty stałe** (dotyczące sporządzenia dokumentacji, pozyskania środków wytwarzania itp.);
- **Koszty bezpośrednie** (dotyczące robocizny, materiałów, maszyn itp. – występują dla każdej czynności przedsięwzięcia);
- **Koszty pośrednie** (związane głównie z czasem realizacji przedsięwzięcia i dotyczą dla przykładu przestojów na stanowiskach pracy, kar za nieterminowość wykonania itp.);
- **Koszty zamrożenia kapitału** (dotyczące nakładów inwestycyjnych, zasobów itp.)

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

W rozważaniach ograniczymy się tylko do kosztów bezpośrednich.

Koszty bezpośrednie wykonania czynności dla dowolnego czasu jej trwania $t_{ij}^{(g)} \le t_{ij} \le t_{ij}^{(n)}$ można obliczyć ze wzoru: $k_{ij} = -a \cdot t_{ij} + b = -s_{ij} t_{ij} + \left(k_{ij}^{(n)} + s_{ij} t_{ij}^{(n)}\right)$



❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Celem analizy czasowo kosztowej jest wyznaczenie takich czasów trwania poszczególnych czynności, dla których całkowity koszt bezpośredni realizacji przedsięwzięcia jest minimalny. Zadanie to można przedstawić za pomocą następującego zadania – ZPL:

$$K = \sum_{(i,j)\in U} \left[\left(k_{ij}^{(n)} + s_{ij} t_{ij}^{(n)} \right) - s_{ij} t_{ij} \right] \rightarrow \min$$

Przy warunkach dla każdej czynności sieci $(i, j) \in U$:

$$\begin{cases} t_i + t_{ij} \le t_j, \\ t_{ij}^{(g)} \le t_{ij} \le t_{ij}^{(n)}, \\ t_1 = 0, t_n \le T^{(g)} \end{cases}$$

gdzie: $T^{(g)}$ - czas graniczny zakończenia przedsięwzięcia.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Uwaga :

Można również sformułować inaczej problem, tzn. minimalizować łączny czas realizacji całego przedsięwzięcia, tak aby łączne koszty bezpośrednie tego przedsięwzięcia nie przekroczyły pewnego - z góry ustalonego limitu środków. Tak sformułowane zagadnienie prowadzi do następującego zadania ZPL:

$$\begin{cases} t_n \to \min, \\ t_i + t_{ij} \le t_j,\ t_{ij}^{(g)} \le t_{ij} \le t_{ij}^{(n)} \ \text{ dla } (\mathrm{i},\mathrm{j}) \in U, \\ \sum\limits_{(i,j)\in U} \left[\left(k_{ij}^{(n)} + s_{ij} t_{ij}^{(n)} \right) - s_{ij} t_{ij} \right] \le K, \end{cases}$$

W ułożeniu programu przyspieszającego wykonanie przedsięwzięcia pomaga wykorzystanie specjalnego algorytmu analizy czasowo-kosztowej znanego pod nazwą: **CPM - COST**.

Zastosowanie algorytmu kompensacji sieci zostanie omówione na następującym przykładzie.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Przykład:

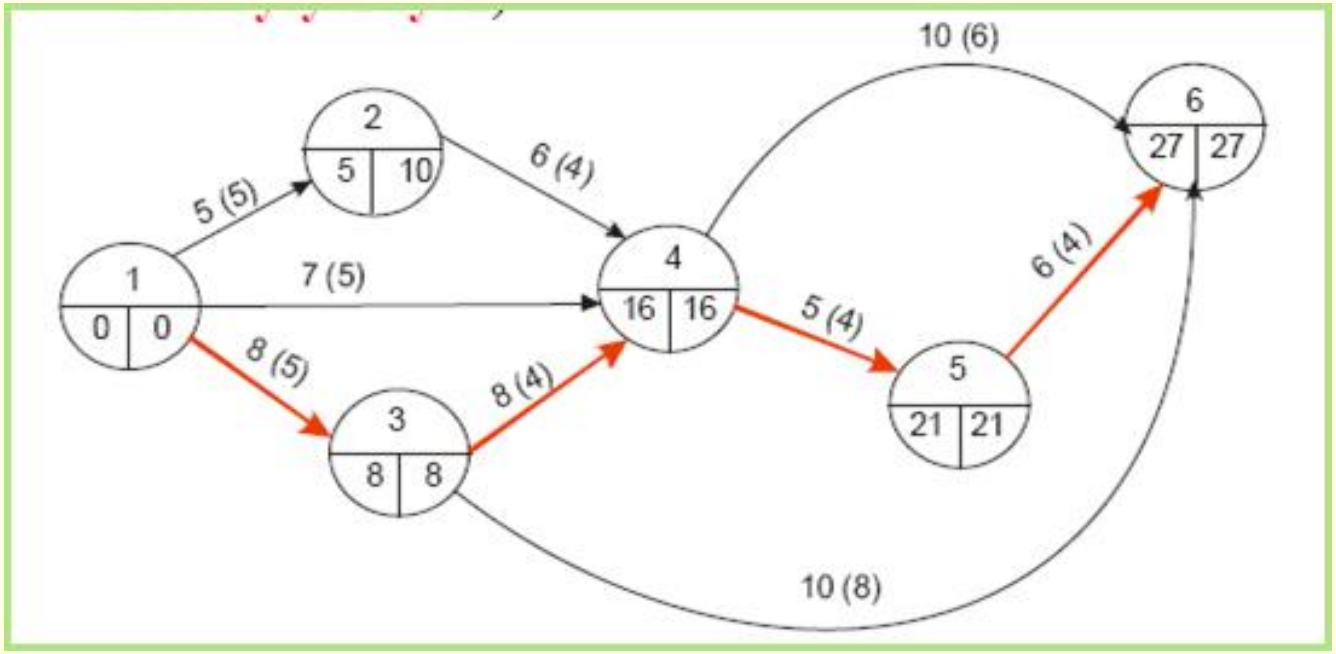
Pewien etap większego przedsięwzięcia składa się z 9 czynności których czasy normalne, czasy graniczne, koszty normalne i koszty graniczne podane są w tabeli.

Czynność (i,j)	t_n	t_{gr}	K_n	K_{gr}
(1,2)	5	5	30	30
(1,3)	8	5	44	50
(1,4)	7	5	30	35
(2,4)	6	4	25	30
(3,4)	8	4	35	40
(3,6)	10	8	44	50
(4,5)	5	4	10	12
(4,6)	10	6	24	28
(5,6)	6	4	20	26

- Wykreślić sieć czynności oraz wyznaczyć ścieżkę krytyczną;
- O ile tygodni można maksymalnie zredukować czas realizacji przedsięwzięcia ? Jak w tym wypadku kształtować będą się koszty.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Sieć czynności oraz ścieżka krytyczna:



❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

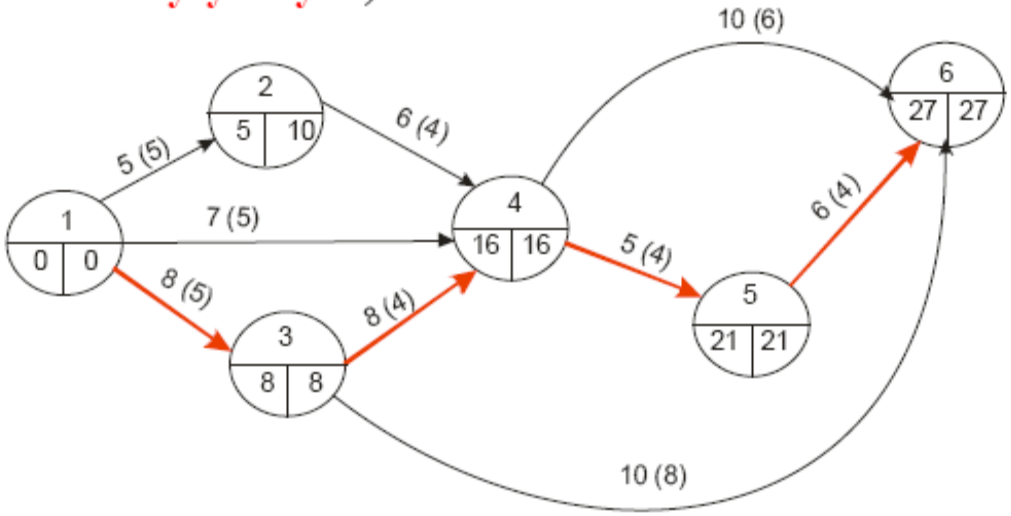
2. Wyznaczyć dla czynności krytycznych gradienty kosztów: $S_{ij} = \frac{K_{ij}^{(g)} - K_{ij}^{(n)}}{t_{ij}^{(n)} - t_{ij}^{(g)}}$

Czynność (i,j)	t_n	t_{gr}	K_n	K_{gr}	S_{ij}	$Z_c(i,j)$
(1,2)	5	5	30	30	-	5
(1,3) (*)	8	5	44	50	2	0 (*)
(1,4)	7	5	30	35	2,5	9
(2,4)	6	4	25	30	2,5	5
(3,4) (*)	8	4	35	40	1,25	0 (*)
(3,6)	10	8	44	50	3	9
(4,5) (*)	5	4	10	12	2	0 (*)
(4,6)	10	6	24	28	1	1
(5,6) (*)	6	4	20	26	3	0 (*)

3. Wyeliminować te czynności krytyczne dla których średni gradient nie istnieje ($t_{ij}^{(n)} = t_{ij}^{(g)}$) – **w przykładzie: (1,2) – ale ona jest niekrytyczna**
4. Proces skracania czasów wykonania czynności rozpoczyna się od czynności krytycznej o najniższym gradiencie kosztów
   Dla naszego przykładu od czynności **(3,4)** – **gradient** wynosi: **1,25**

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

5. Skracając czasy trwania czynności krytycznych należy starać się skrócić jej czas o jak największą możliwą liczbę jednostek.
   Należy przestrzegać jednak dwa ograniczenia:
   - skracamy **do czasu granicznego**
   - skracamy **dotąd** aż pojawi się **nowa ścieżka krytyczna** (pojawia się wówczas, gdy **zniknie zapas czasu całkowitego** w ciągu **czynności niekrytycznych**)

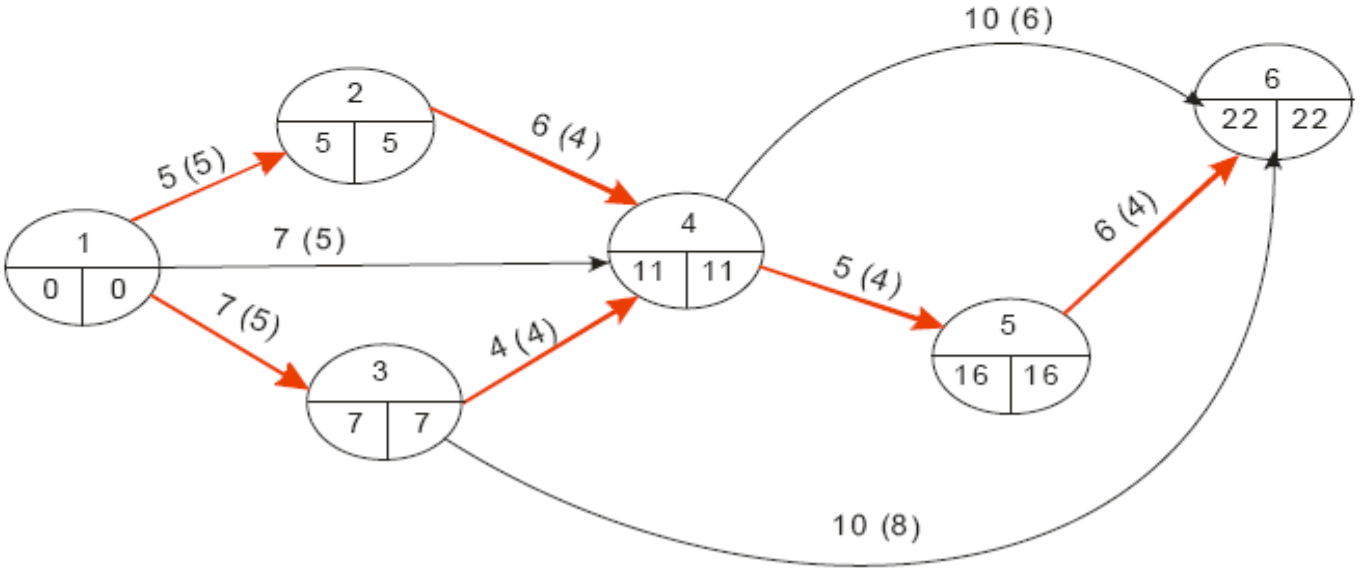


Skracamy czynność (3,4) o 4 jednostki – koszt skrócenia czynności (przyspieszenia realizacji przedsięwzięcia) wynosi: $K_1 = S_{34} \cdot 4 = 1{,}25 \cdot 4 = 5$.
Czas trwania przedsięwzięcia wynosi: $t_6 = 27 - 4 = 23$.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Kolejną czynnością którą będziemy skracać jest np. (1,3) – nie możemy jednak skrócić jej o 3 dni, bo i tak termin wykonania wyniesie 22 dni - nowa ścieżka krytyczna: **1-2-4-5-6** i poniesiemy niepotrzebne koszty. Skracamy ją zatem tylko o 1 dzień. Koszt przyspieszenia realizacji przedsięwzięcia wynosi: $K_2 = S_{13} \cdot 1 = 2 \cdot 1 = 2$.

Nowa sieć po 2-krotnym skróceniu czasów trwania czynności przedsięwzięcia. Czas wykonania przedsięwzięcia wynosi: $t_6 = 22$.

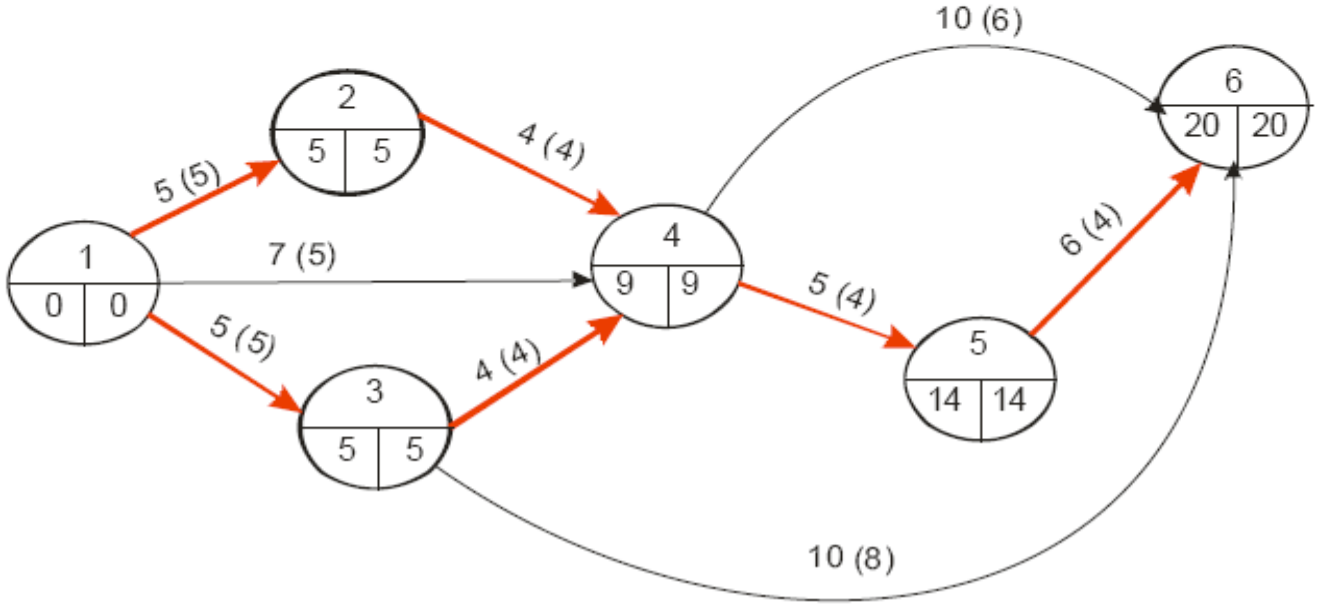


❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

6. Jeżeli występuje dwie lub więcej ścieżek krytycznych, należy skracać czasy o tę samą wielkość na wszystkich równoległych ścieżkach krytycznych. Możemy teraz w dalszym ciągu skrócić czynność (1,3) o pozostałe 2 jednostki. Tym samym należy skrócić o 2 jednostki także czynność (2,4), leżącą na drugiej równoległej ścieżce krytycznej. Koszt przyspieszenia realizacji przedsięwzięcia wynosi zatem:

$K_3 = S_{13} \cdot 2 + S_{24} \cdot 2 = 2 \cdot 2 + 2{,}5 \cdot 2 = 4 + 5 = 9$.

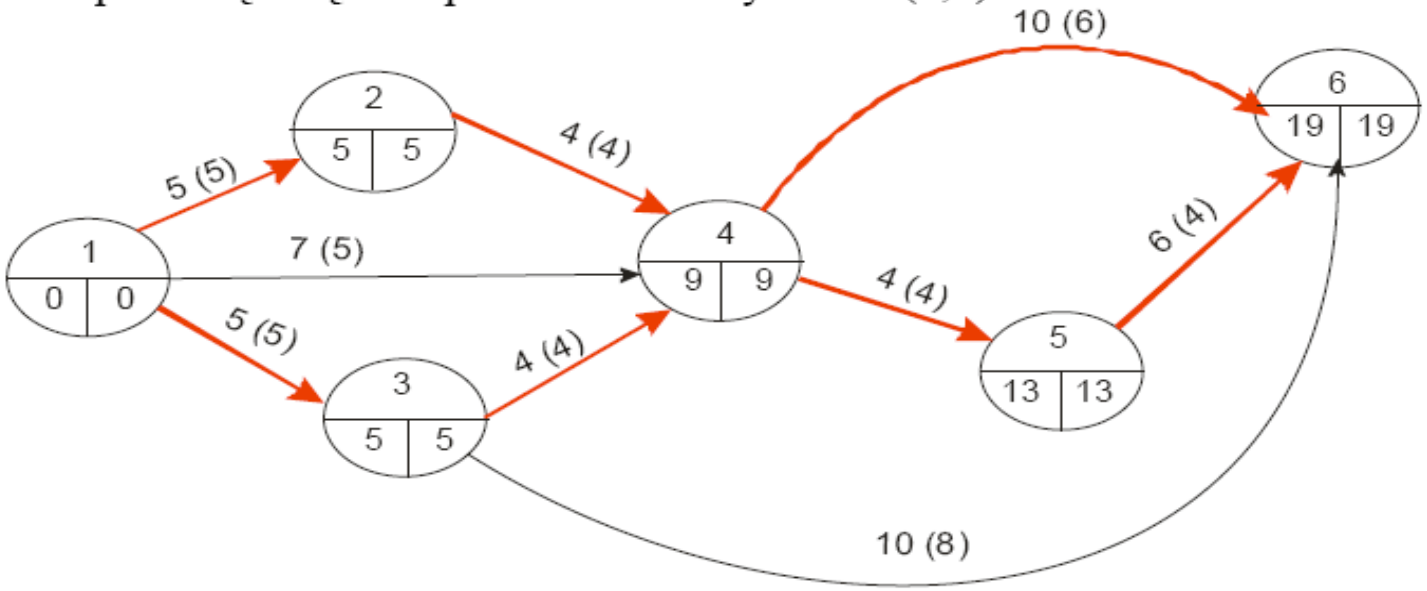
Czas wykonania przedsięwzięcia wynosi: $t_6 = 22 - 2 = 20$.



❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Kolejną skracaną czynnością krytyczną – która posiada najmniejszy gradient kosztów, a nie została jeszcze skrócona – jest czynność: (4,5), którą skracamy o 1 jednostkę. Koszt przyspieszenia realizacji przedsięwzięcia wynosi zatem: $K_4 = S_{45} \cdot 1 = 2 \cdot 1 = 2$. Czas wykonania przedsięwzięcia wynosi: $t_6 = 20 - 1 = 19$.

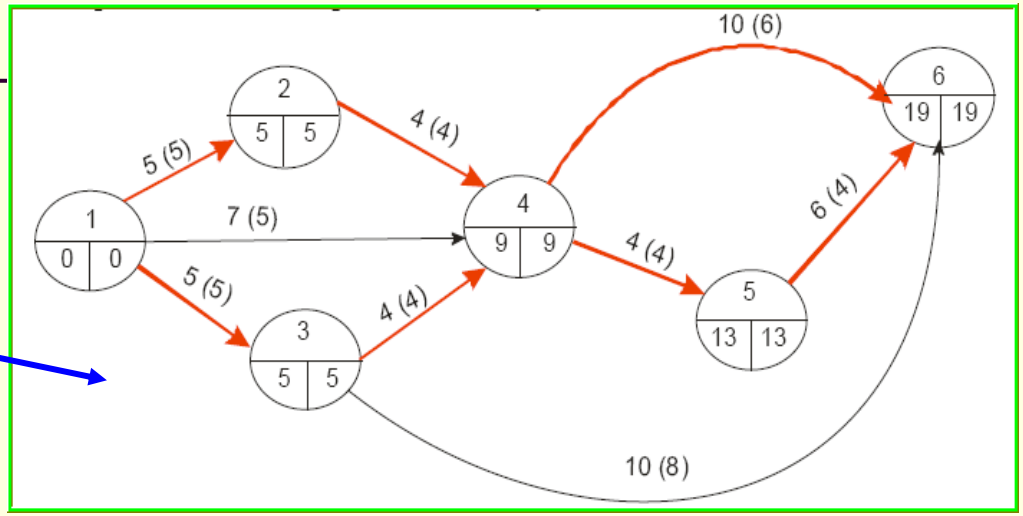
Sieć przedsięwzięcia – po skróceniu czynności (4,5):



Powstają dwie kolejne równoległe ścieżki krytyczne: 1-2-4-6 oraz 1-3-4-5-6.

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

aktualna sieć czynności

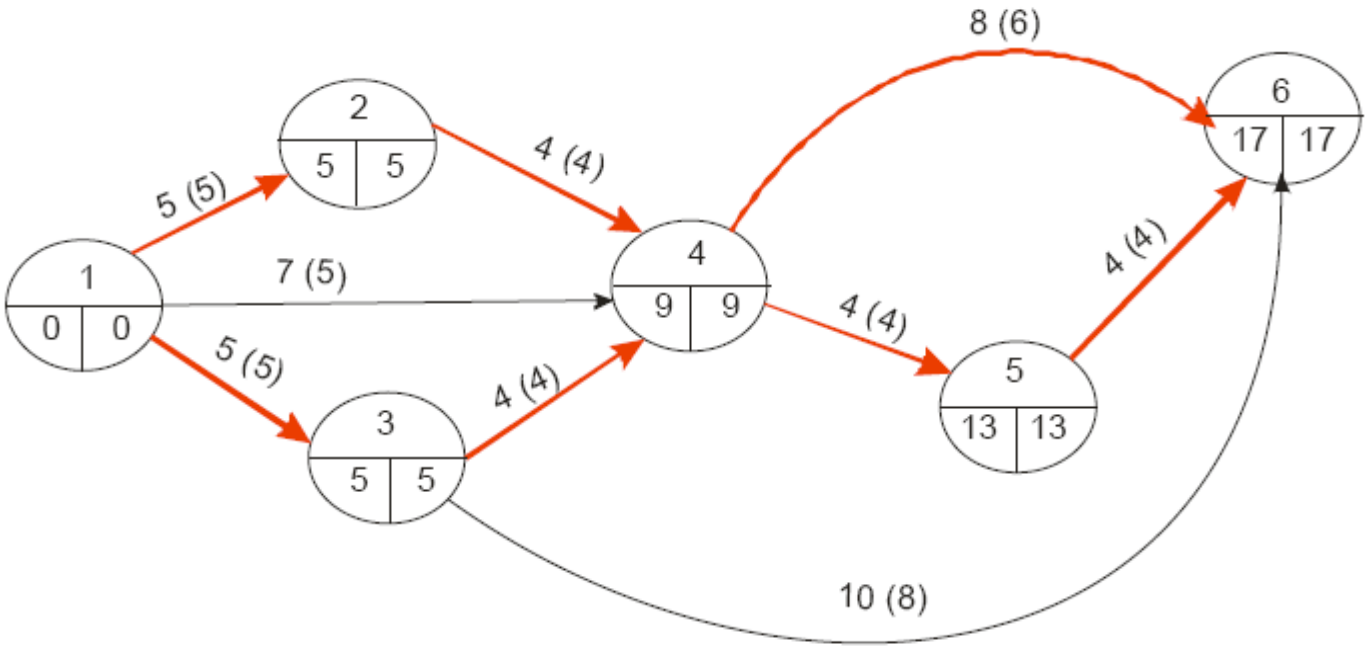


Kolejną skracaną czynnością krytyczną – która posiada najmniejszy gradient kosztów, a nie została jeszcze skrócona – jest czynność: (5,6), którą skracamy o 2 jednostki. Tym samym czynność (4,6) należy również skrócić o 2 jednostki. Koszt przyspieszenia realizacji przedsięwzięcia wynosi zatem:
$K_5 = S_{56} \cdot 2 + S_{46} \cdot 2 = 3 \cdot 2 + 1 \cdot 2 = 6 + 2 = 8$.
Czas wykonania przedsięwzięcia zostaje skrócony zatem do wartości:
$t_6 = 19 - 2 = 17$.

7. Najkrótszy termin wykonania przedsięwzięcia uzyskuje się, gdy wszystkie czynności leżące na jakiejkolwiek drodze krytycznej osiągną czasy graniczne: $t_{ij}^{(g)}$ (wtedy dalsze skracanie jest niemożliwe).

❑ SIECIOWA ANALIZA PRZEDSIĘWZIĘĆ – analiza czasowo - kosztowa

Sieć przedsięwzięcia – po skróceniu czynności (5,6) oraz (4,6):



8. Łączne koszty przyspieszenia czasu wykonania przedsięwzięcia są sumą kosztów poniesionych na poszczególnych etapach:
$K_C = K_1 + K_2 + K_3 + K_4 + K_5 = 5 + 2 + 9 + 2 + 8 = 26$ (jednostek).

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:

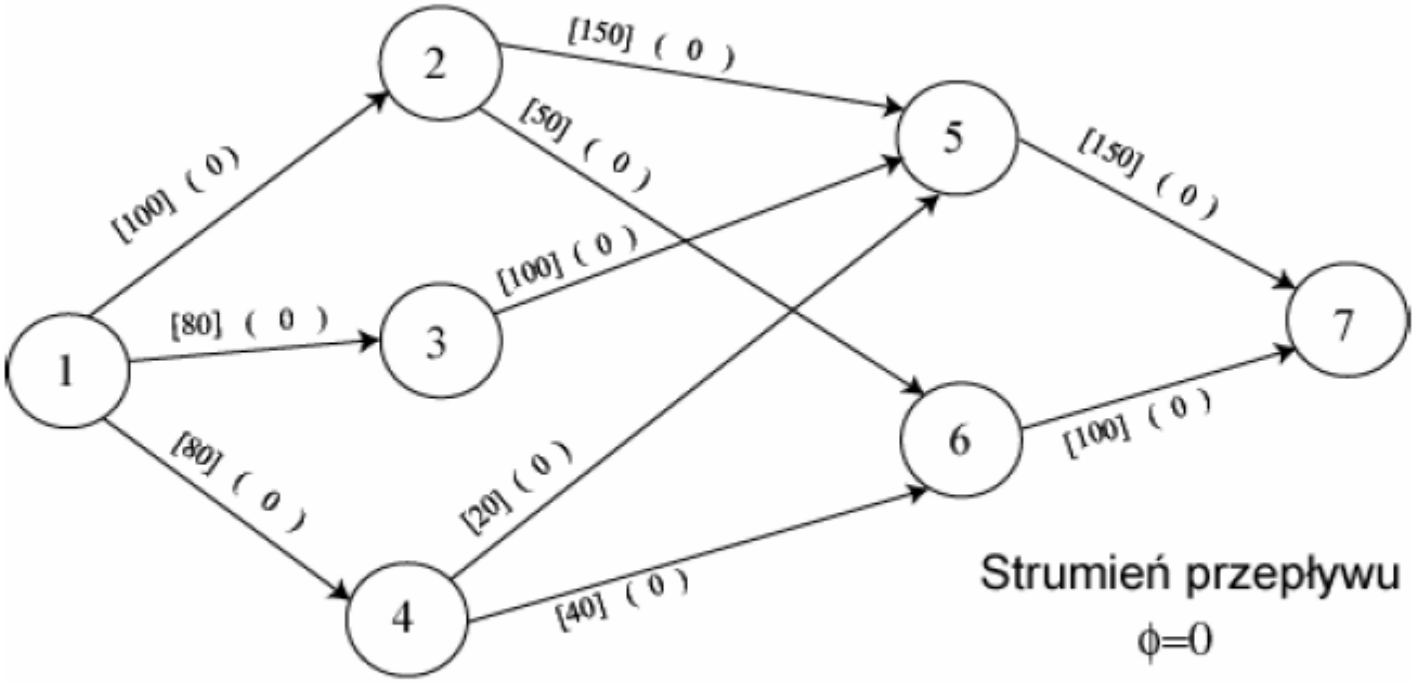
Kanał transportowy $\langle i, j \rangle$	Maksymalna ładowność C_{ij} (w tonach)
⟨1,2⟩	100
⟨1,3⟩	80
⟨1,4⟩	80
⟨2,5⟩	150
⟨2,6⟩	50
⟨3,5⟩	100
⟨4,5⟩	20
⟨4,6⟩	40
⟨5,7⟩	150
⟨6,7⟩	100

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.

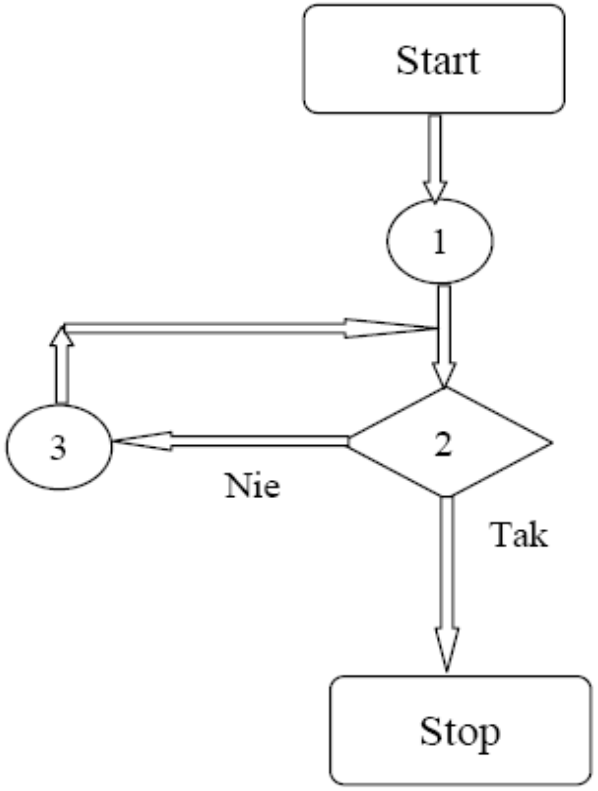
Idea algorytmu jest następująca:
Najpierw poszukuje się jakiegokolwiek dopuszczalnego **strumienia** Φ przepływu (spełniającego tzw. **Prawo Kirchoffa** dla sieci), taki strumień początkowy zawsze istnieje (w najgorszym wypadku jest to strumień zerowy). W dalszych krokach algorytmu strumień ten odpowiednio 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).

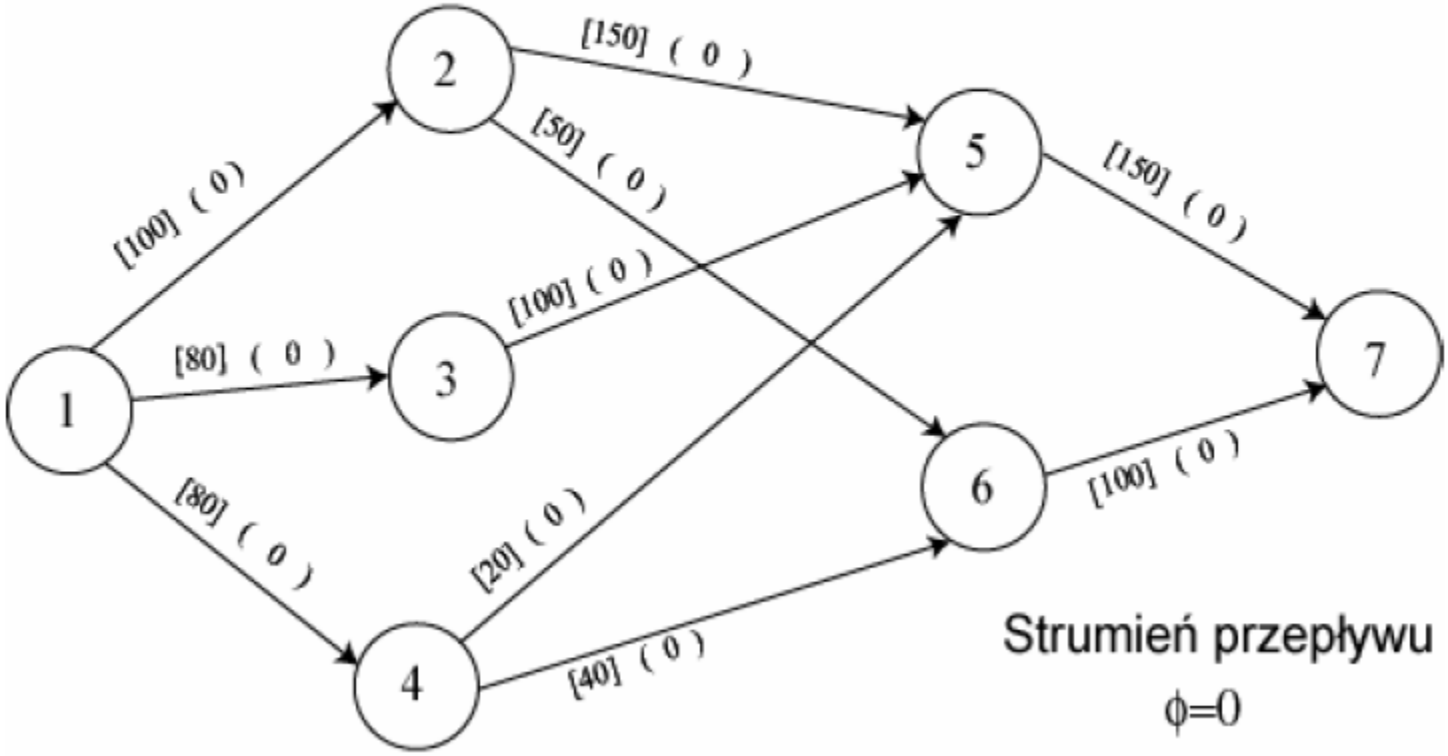
Krok 2 – sprawdzenie warunku czy wyznaczony strumień przepływu jest maksymalny (**wykorzystanie procedury cechowania wierzchołków – kryterium max przepływu**).

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.

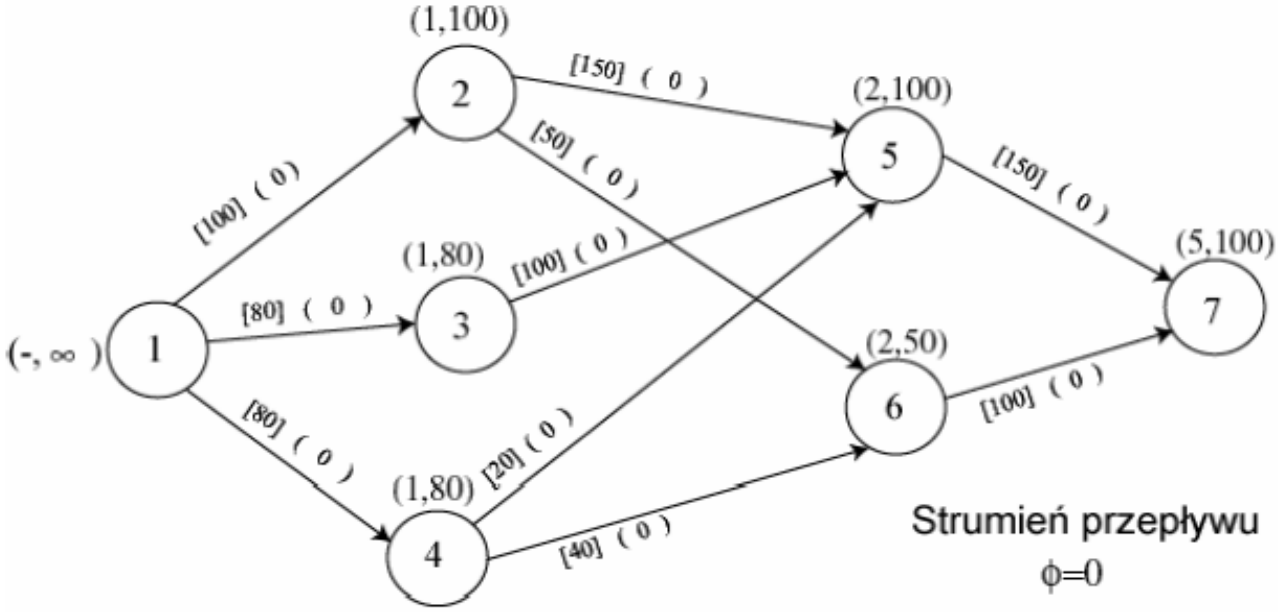
- Jeżeli $(j,l) \in U$ oraz $x_{j,l} < c_{j,l}$ (**nie jest to łuk maksymalnie nasycony**), to wierzchołkowi: **l** - nadajemy cechę $(j, \varepsilon_l = \min\{\varepsilon_j, c_{j,l} - x_{j,l}\}$.
- Jeżeli $(l,j) \in U$ oraz $x_{l,j} > 0$ (**istnieje już jakiś przepływ przez ten łuk**), to wierzchołkowi: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

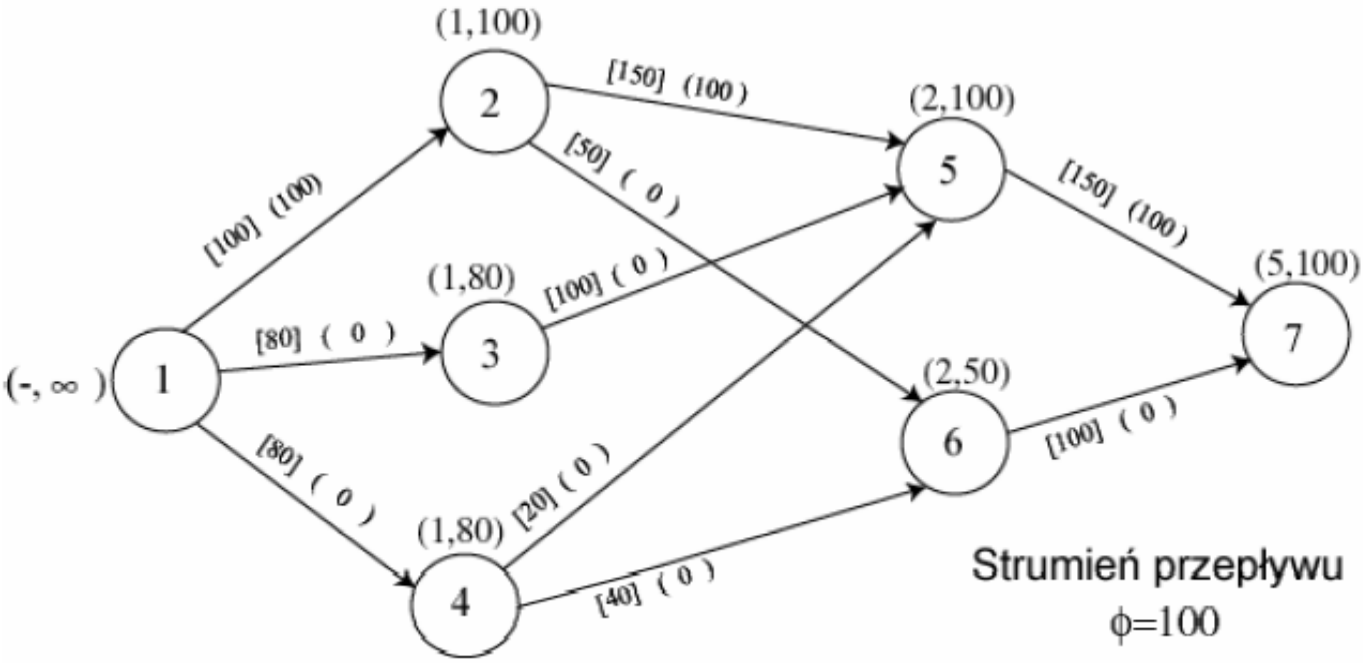


❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

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

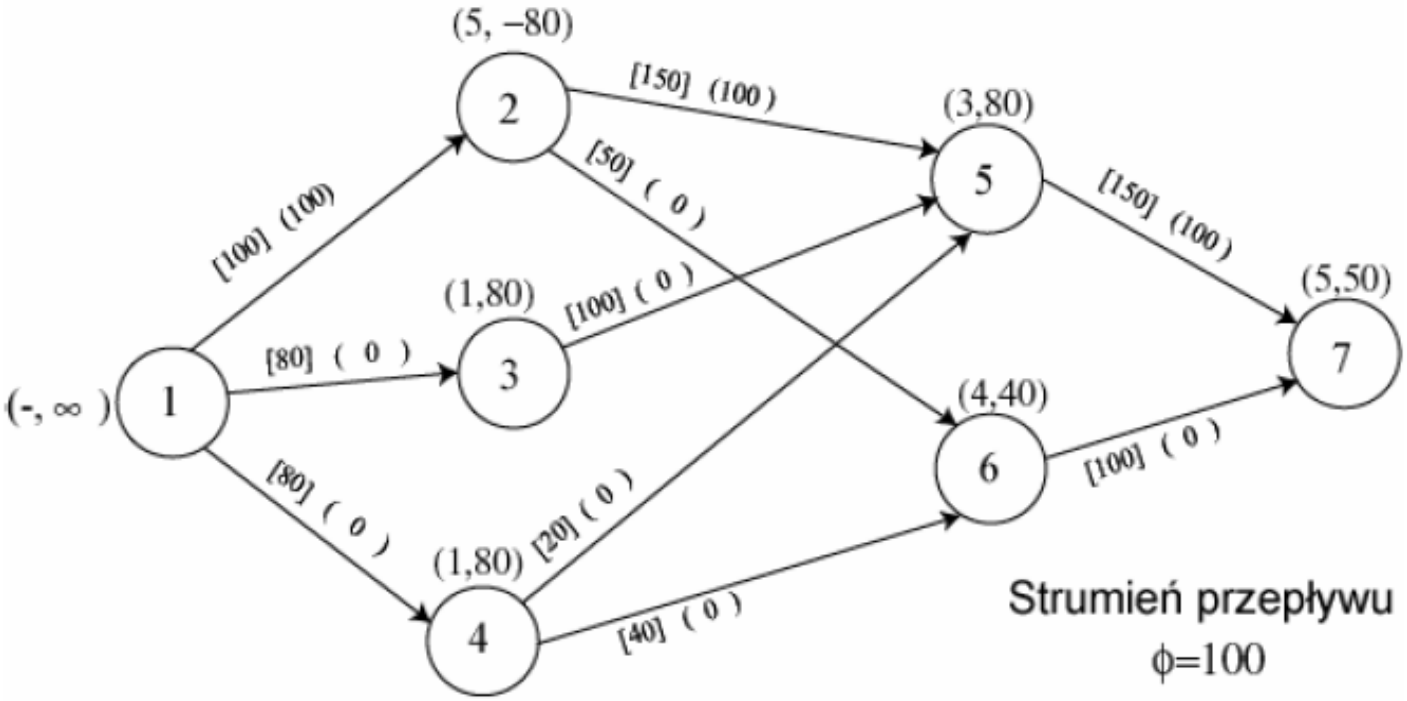
Realizacja kroku 3:

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**.



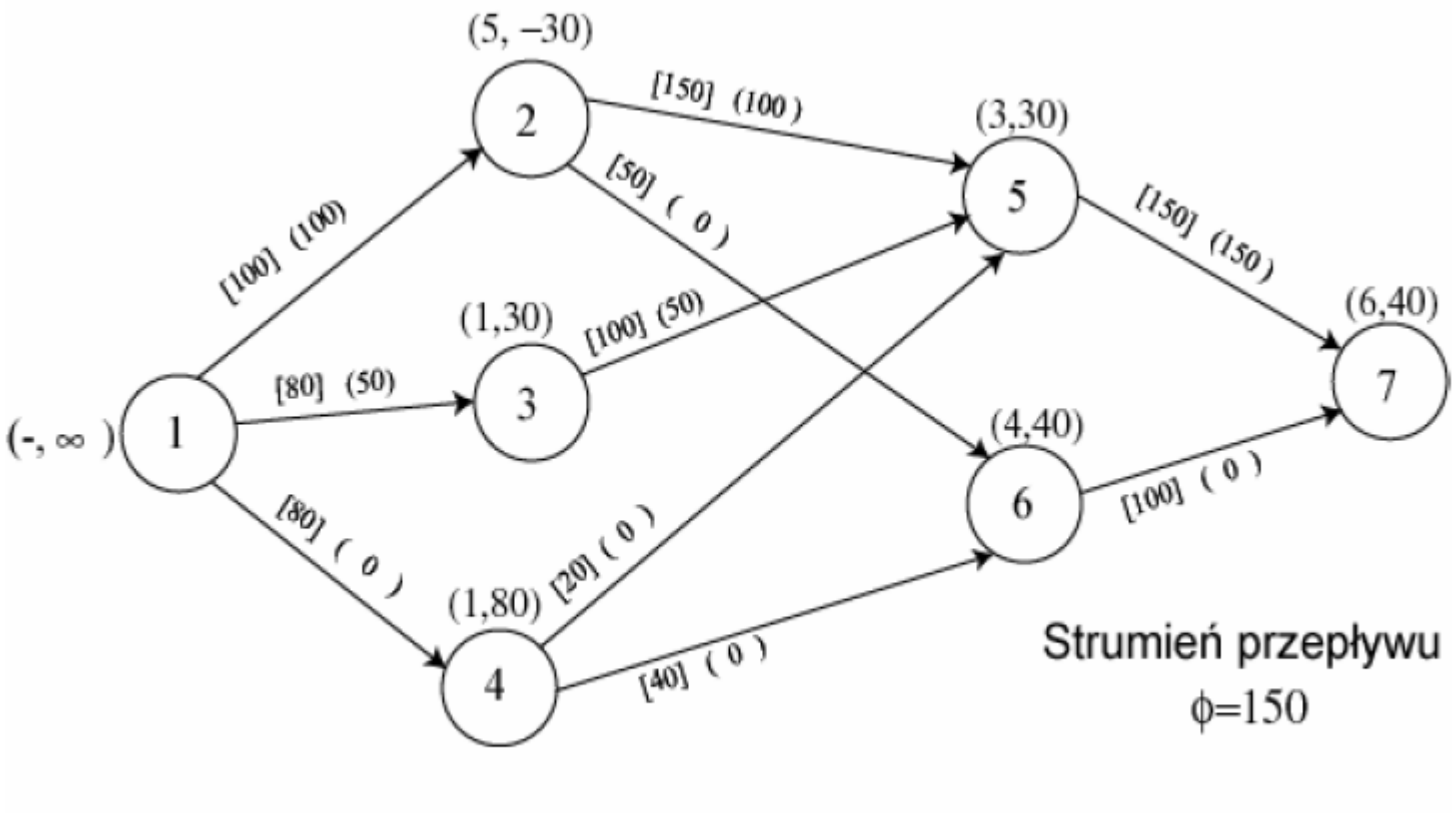
❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona

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



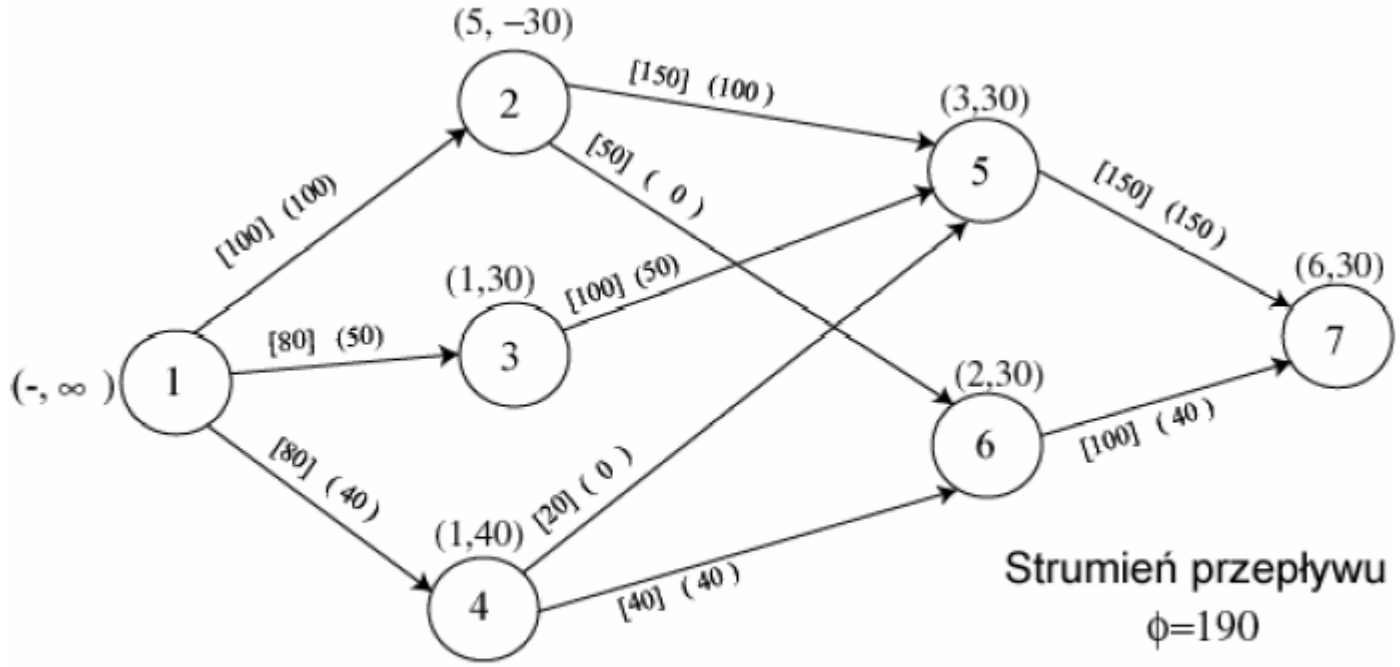
Ścieżka zwiększająca przepływ **o 50** jest postaci: **1 – 3 – 5 – 7**.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



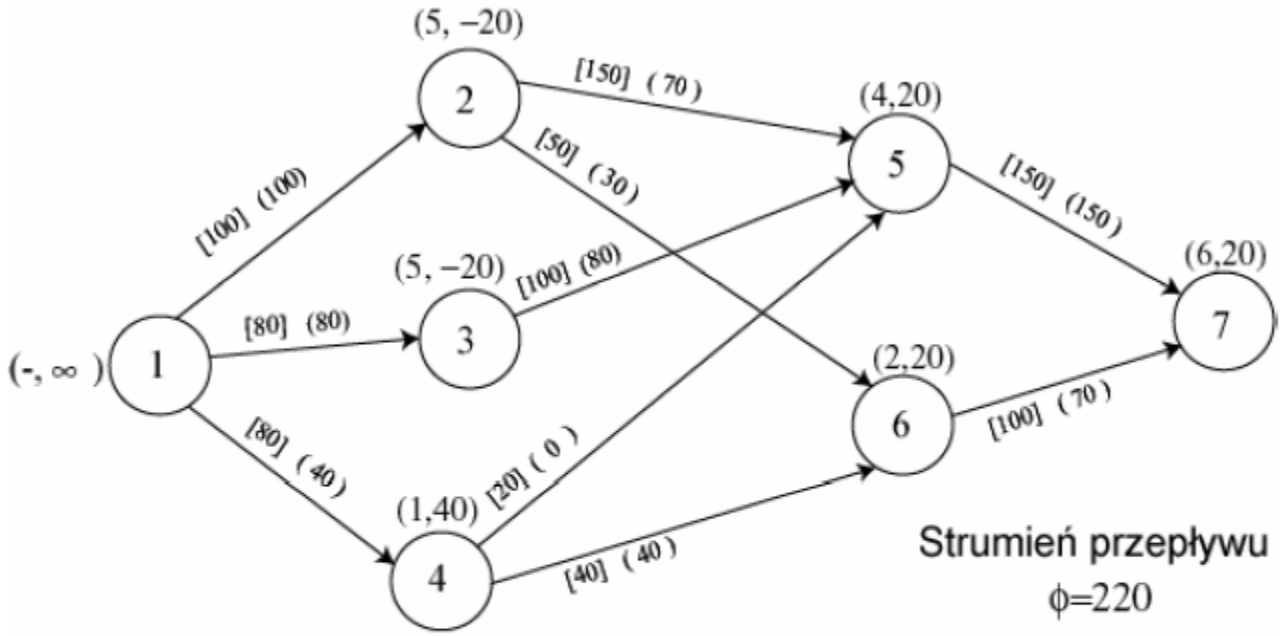
Ścieżka zwiększająca przepływ o 40 jest postaci: 1 – 4 – 6 – 7.

❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



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

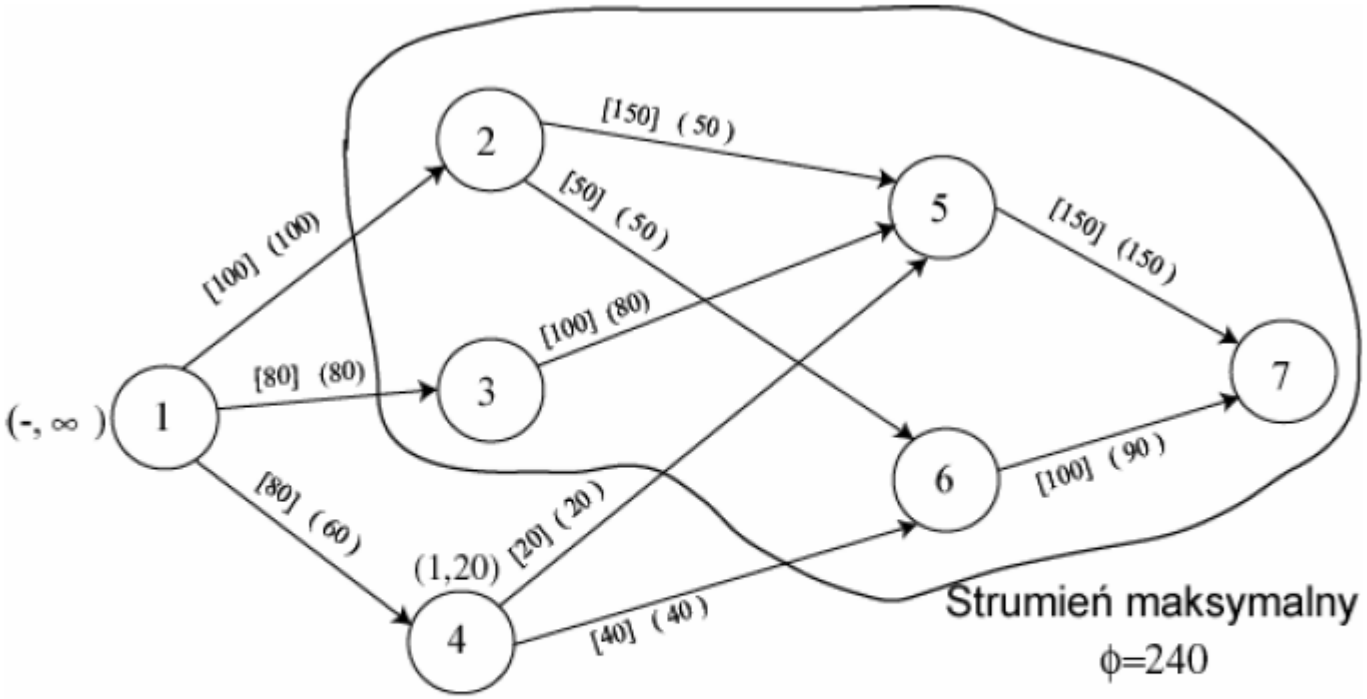
❑ PROGRAMOWANIE SIECIOWE - algorytm maksymalnego przepływu Forda - Fulkersona



(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 $i \in P^1, j \in P^2$.

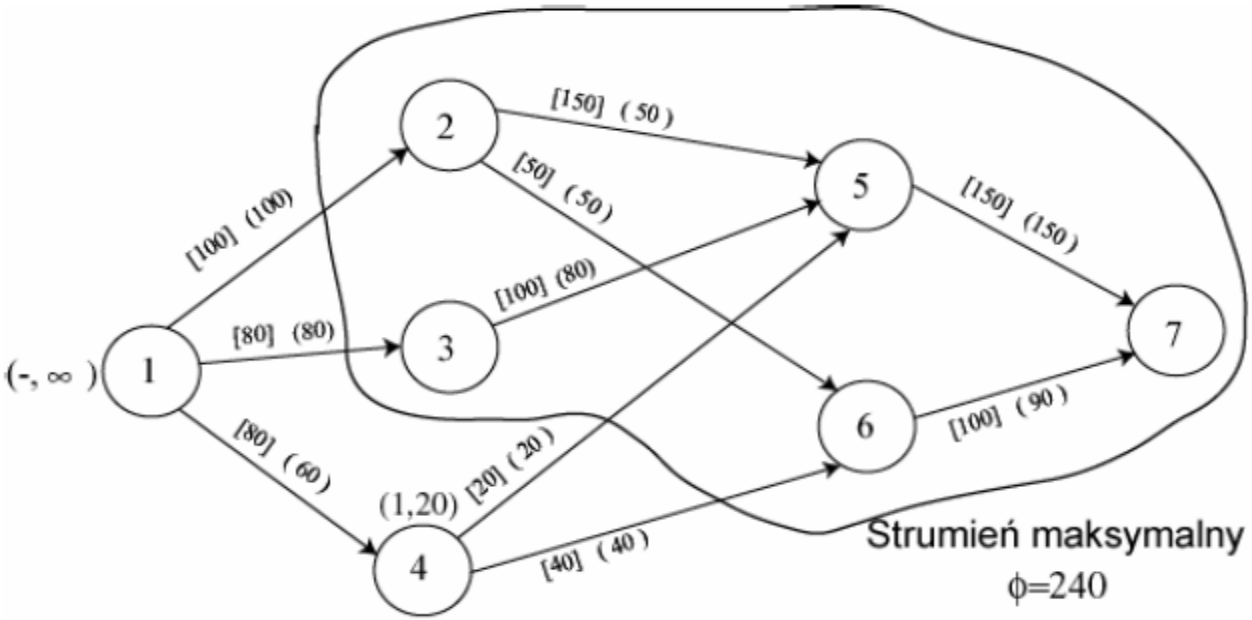
  Jest to każdy **minimalny zbiór łuków**, których usunięcie spowoduje **utratę spójności sieci** (nie będzie istnieć ścieżka prowadząca od v_1 do v_n).

  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: $\Phi_{max} = \Psi_{min} = 240$.