Cuprins

1 Secvenţe de grade

Materialul din această secţiune urmează cartea [2].

Fie $s_0 = \{d_1, \ldots, d_n\}$ o secvenţă de numere naturale.

Problemă. Să se construiască, dacă se poate, un (multi)graf neorientat G cu $s(G) = s_0$.

Observaţie 1.1. *Deoarece suma gradelor vârfurilor într-un (multi)graf este egală cu dublul numărului de muchii, o condiţie necesară pentru existenţa unui (multi)graf G cu $s(G) = s_0$ este ca suma*

$$d_1 + \ldots + d_n$$

să fie număr par.

? Este condiţia din Observaţia 1.1 şi suficientă?

1.1 Construcţia unui multigraf neorientat cu secvenţa gradelor dată

Teorema 1.2. *O secvenţă de $n \geq 2$ numere naturale $s_0 = \{d_1, \ldots, d_n\}$ este secvenţa gradelor unui multigraf neorientat dacă şi numai dacă suma $d_1 + \ldots + d_n$ este număr par.*

Demonstraţie. " $\Longrightarrow$ " Presupunem că există un multigraf neorientat G cu $s(G) = s_0$. Atunci

$$d_1 + \ldots + d_n = 2|E(G)| \text{ este număr par.}$$

" $\Longleftarrow$ " Presupunem că $d_1 + \ldots + d_n$ este număr par.

Rezultă că există un număr par de numere impare în secvenţa (multisetul) s_0

Construim un multigraf G cu $V(G) = \{x_1, \ldots, x_n\}$ având $s(G) = s_0$ (mai exact cu $d_G(x_i) = d_i$) după următorul algoritm:

1. Adăugăm în fiecare vârf $x_i \in V(G)$ $\left\lfloor \frac{d_i}{2} \right\rfloor$ bucle.
2. Formăm perechi disjuncte cu vârfurile care trebuie să aibă gradul impar şi unim vârfurile din aceste perechi cu câte o muchie.

Formalizând, dacă renotăm numerele din secvenţa s_0 astfel încât primele $2k$ numere din secvenţă: $d_1, \ldots, d_{2k}$ să fie impare şi celelalte pare, definim

$$E(G) = \left\{ x_i x_i^{\left\lfloor \frac{d_i}{2} \right\rfloor} | i \in \{1, \ldots, n\}, d_i > 0 \right\} \cup \{x_i x_{i+1} | i \in \{1, \ldots, 2k-1\}\}.$$

Atunci, pentru i cu $1 \leq i \leq 2k$ avem d_i impar şi

$$d_G(x_i) = 2\left\lfloor \frac{d_i}{2} \right\rfloor + 1 = 2\frac{d_i - 1}{2} + 1 = d_i,$$

iar pentru $2k + 1 \leq i \leq n$ avem d_i par şi

$$d_G(x_i) = 2\left\lfloor \frac{d_i}{2} \right\rfloor = d_i,$$

deci $s(G) = s_0$. □

1.2 Construcţia unui graf neorientat cu secvenţa gradelor dată

Dat un graf neorientat G, pentru a obţine grafuri neorientate cu aceeaşi secvenţă de grade ca şi G se poate folosi următoarea transformare t (pe care o vom numi de interschimbare pe pătrat). Fie x, y, u, v patru vârfuri distincte ale lui G astfel încât $xy, uv \in E(G)$, dar $xu, yv \notin E(G)$. Considerăm graful notat $t(G, xy, uv)$ definit astfel:

$$t(G, xy, uv) = G - \{xy, uv\} \cup \{xu, yv\}$$

Spunem că $t(G, xy, uv)$ este graful obţinut din G prin aplicarea transformării t de interschimbare pe pătratul $xyvu$ - figura 1.

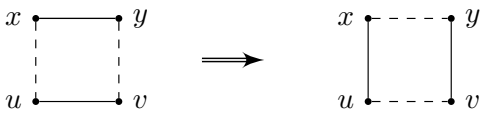


Figura 1: Transformarea t de interschimbare pe pătrat

Observaţie 1.3. *Graful $t(G, xy, uv)$ are aceeaşi secvenţă de grade ca şi G.*

Teorema 1.4. *(Havel-Hakimi) O secvenţă de $n \geq 2$ numere naturale $s_0 = \{d_1 \geq \ldots \geq d_n\}$ cu $d_1 \leq n-1$ este secvenţa gradelor unui graf neorientat (cu n vârfuri) dacă şi numai dacă secvenţa $s_0' = \{d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n\}$ este secvenţa gradelor unui graf neorientat (cu $n-1$ vârfuri).*

Demonstraţie. " $\Longleftarrow$ " Presupunem că s_0' este secvenţa gradelor unui graf neorientat. Fie $G' = (V', E')$ un graf neorientat cu $V' = \{x_2, \ldots, x_n\}$ având secvenţa gradelor $s(G') = s_0'$, mai precis cu

$$d_{G'}(x_i) = \begin{cases} d_i - 1, & \text{dacă } i \in \{2, \ldots, d_1 + 1\} \\ d_i, & \text{dacă } i \in \{d_1 + 2, \ldots, n\}. \end{cases}$$

Construim pornind de la G' un nou graf $G = (V, E)$ adăugând un vârf nou x_1 pe care îl unim cu vârfurile $x_2, \ldots, x_{d_1+1}$:

- $V = V' \cup \{x_1\}$
- $E = E' \cup \{x_1 x_i | i \in \{2, \ldots, d_1 + 1\}\}$.

Pentru un $i \in \{1, \ldots, n\}$ avem atunci

$$d_G(x_i) = \begin{cases} d_{G'}(x_i) + 1 = d_i - 1 + 1 = d_i, & \text{dacă } i \in \{2, \ldots, d_1 + 1\} \\ d_{G'}(x_i) = d_i, & \text{dacă } i \in \{d_1 + 2, \ldots, n\} \\ d_1, & \text{dacă } i = 1. \end{cases}$$

Rezultă că $s(G) = s_0$, deci s_0 este secvenţa gradelor unui graf neorientat.

" $\Longrightarrow$ " Presupunem că s_0 este secvenţa gradelor unui graf neorientat.

Fie $G = (V, E)$ un graf neorientat cu $V = \{x_1, \ldots, x_n\}$ astfel încât $d_G(x_i) = d_i$ pentru orice $i \in \{1 \ldots, n\}$. Vom construi un graf G' cu $s(G') = s'_0$ pornind de la G.

Pentru aceasta, construim întâi din G un graf G^* având secvenţa gradelor tot s_0, dar în care vârful x_1 are mulţimea vecinilor $N_{G^*}(x_1) = \{x_2, \ldots, x_{d_1+1}\}$.

Cazul 1**.** Dacă $N_G(x_1) = \{x_2, \ldots, x_{d_1+1}\}$, atunci considerăm $G^* = G$.

Cazul 2**.** Există cel puţin un indice $i \in \{2, \ldots, d_1 + 1\}$ cu $x_1x_i \notin E$ (i.e $x_i \notin N_G(x_1)$). Fie i minim cu această proprietate.

Deoarece $d_G(x_1) = d_1$, rezultă că există $j \in \{d_1 + 2, \ldots, n\}$ cu $x_1x_j \in E$. Mai mult, deoarece $j > d_1 + 1 \geq i$, avem $d_i = d_G(x_i) \geq d_G(x_j) = d_j$. În plus, x_1 este adiacent cu x_j, dar nu şi cu x_i. Rezultă că există un alt vârf x_k cu $k \in \{2, \ldots, n\} - \{i, j\}$ care este adiacent cu x_i ($x_ix_k \in E$), dar care nu este adiacent cu x_j ($x_jx_k \notin E$) - figura 2.

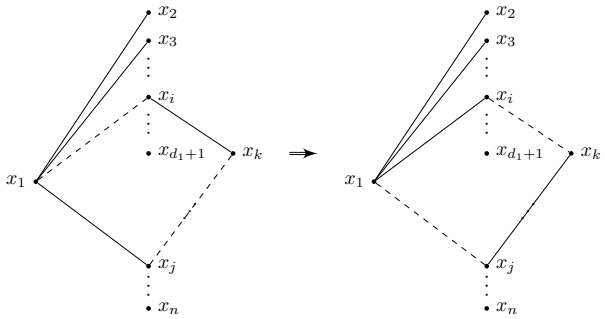


Figura 2: Transformare din demonstraţia teoremei Havel-Hakimi

Considerăm graful G_i obţiunt din G prin aplicarea transformării de interschimbare t pentru pătratul $x_ix_kx_jx_1$:

$$G_i = t(G, x_ix_k, x_1, x_j) = G - \{x_ix_k, x_1x_j\} \cup \{x_1x_i, x_kx_j\}$$

Avem $N_{G_i}(x_1) \cap \{x_2, \ldots, x_{d_1+1}\} = (N_G(x_1) \cap \{x_2, \ldots, x_{d_1+1}\}) \cup \{x_i\}$ (x_1 are un vecin în plus în $\{x_2, \ldots, x_{d_1+1}\}$) şi, conform Observaţiei 1.3, $s(G_i) = s(G) = s_0$.

Aplicând succesiv transformări de tip t pentru fiecare indice $i \in \{2, 3, \ldots, d_1 + 1\}$ pentru care x_1 şi x_i nu sunt adiacente obţinem în final un graf G^* cu $s(G^*) = s_0$ şi $N_{G^*}(x_1) = \{x_2, \ldots, x_{d_1+1}\}$.

Fie $G' = G^* - x_1$. Atunci $V(G') = \{x_2, \ldots, x_n\}$ şi pentru orice $i \in \{2, \ldots, n\}$:

$$d_{G'}(x_i) = \begin{cases} d_{G^*}(x_i) - 1 = d_i - 1, & \text{dacă } i \leq d_1 + 1 \\ d_{G^*}(x_i) = d_i, & \text{dacă } i \geq d_1 + 2, \end{cases}$$

deci $s(G') = s'_0$. Rezultă că $s'_0 = \{d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n\}$ este secvenţa gradelor unui graf neorientat. □

Din Teorema Havel-Hakimi se obţine următorul algoritm de determinare a unui graf neorientat cu secvenţa gradelor dată.

Algoritmul Havel-Hakimi

Intrare: o secvenţă de n numere naturale $d_1, \ldots, d_n$

Ieşire: un graf G cu $V(G) = \{x_1, \ldots, x_n\}$ cu $s(G) = s_0$ dacă s_0 este secvenţa gradelor unui graf, sau mesajul NU altfel.

Idee: La un pas unim un vârf de grad maxim d din secvenţa s_0 cu vârfurile corespunzătoare următoarelor cele mai mari d elemente din s_0 diferite de d şi actualizăm secvenţa s_0 (s_0=s_0'). Se repetă pasul până când secvenţa conţine numai 0 sau conţine elemente negative.

Pseudocod:

Pasul 1. Dacă $d_1 + \ldots + d_n$ este număr impar sau există în s_0 un număr $d_i > n - 1$, atunci scrie NU, STOP.

Pasul 2.

cât timp s_0 conţine valori nenule execută

alege d_k cel mai mare număr din secvenţa s_0

elimină d_k din s_0

fie $d_{i_1}, \ldots, d_{i_{d_k}}$ cele mai mari d_k numere din s_0

pentru $j \in \{i_1, \ldots, i_{d_k}\}$:

adaugă la G muchia $x_k x_j$

înlocuieşte d_j în secvenţa s_0 cu $d_j - 1$

dacă $d_j - 1 < 0$, atunci scrie NU, STOP.

Observaţie. Pentru a determina uşor care este cel mai mare număr din secvenţă şi care sunt cele mai mari valori care îi urmează, este util ca pe parcursul algoritmului secvenţa s_0 să fie ordonată descrescător.

Exemplu. - vezi curs + laborator

Teorema 1.5. *(Extindere a teoremei Havel-Hakimi) Fie $s_0 = \{d_1 \geq \ldots \geq d_n\}$, o secvenţă de $n \geq 2$ numere naturale cu $d_1 \leq n - 1$ şi fie $i \in \{1, \ldots, n\}$ fixat. Fie $s_0^{(i)}$ secvenţa obţinută din s_0 prin următoarele operaţii:*

- eliminăm elementul d_i

- scădem o unitate din primele d_i componente în ordine descrescătoare ale secvenţei rămase.

Are loc echivalenţa:

s_0 este secvenţa gradelor unui graf neorientat $\Longleftrightarrow$

$s_0^{(i)}$ este secvenţa gradelor unui graf neorientat

Demonstraţie. Demonstraţia este similară cu cea a Teoremei Havel-Hakimi ([3] - exerciţiul 2.11). □

Exerciţiu ([3] - exerciţiul 2.12) Fie G_1 şi G_2 două grafuri neorientate cu mulţimea vârfurilor $V = \{1, \ldots, n\}$. Atunci $s(G_1) = s(G_2)$ dacă şi numai dacă există un şir de transformări t de interschimbare pe pătrat prin care se poate obţine graful G_2 din G_1.

2 Arbori

Definiţie 2.1. *Un arbore este un graf neorientat conex fară cicluri.*

Lema 2.2. *Orice arbore cu $n \geq 2$ vârfuri are cel puţin două vârfuri terminale (de grad 1).*

Demonstraţie. Fie T un arbore cu $n \geq 2$ vârfuri. Fie P cel mai lung lanţ elementar în T. Extremităţile lui P sunt vârfuri terminale, altfel am putea extinde lanţul P cu o muchie sau se formează un ciclu elementar în T. Într-adevăr, fie x, y extremităţile lui P. Presupunem prin absurd că $d_T(x) \geq 2$. Atunci există un vârf v adiacent cu x astfel încât $xv \notin E(P)$.

- Dacă $v \in V(P)$, atunci există în T ciclul $[x \overset{P}{-} v, x]$, contradicţie (T este aciclic).

- Dacă $v \notin V(P)$, atunci lanţul $P' = [v, x \overset{P}{-} y]$ este elementar şi $l(P') = l(P) + 1$ contradicţie (P este lanţ elementar maxim în T). □

Lema 2.3. *Fie T un arbore cu $n \geq 2$ vârfuri şi v un vârf terminal în T. Atunci $T - v$ este arbore.*

Demonstraţie. Afirmaţia rezultă din faptul că un vârf terminal nu poate fi vârf intern al unui lanţ elementar, deci pentru orice $x, y \neq v$ un $x - y$lanţ elementar din T este lanţ şi în $T - v$. □

Propozitia 2.4. *Un arbore cu n vârfuri are $n - 1$ muchii.*

Demonstraţie. Demonstrăm afirmaţia prin inducţie.

Pentru $n = 1$, un arbore cu n vârfuri are un vârf şi zero muchii. Pentru $n = 2$, un arbore cu 2 vârfuri are o singură muchie (este izomorf cu P_2).

"$n - 1 \Longrightarrow n$" Presupunem afirmaţia adevărată pentru un arbore cu $n - 1$ vârfuri.

Fie T un arbore cu n vârfuri. Conform Lemei 2.2, există un vârf terminal v în T. Atunci $T' = T - v$ este arbore cu $n - 1$ vârfuri (Lema 2.3) şi $|E(T')| = |E(T)| - 1$. Din ipoteza de inducţie, T' are $|E(T')| = |V(T')| - 1 = n - 2$ muchii. Rezultă $|E(T)| = |E(T')| + 1 = n - 1$. □

Lema 2.5. *Fie G un graf neorientat conex şi C un ciclu in G. Fie $e \in E(C)$ o muchie din ciclul C. Atunci $G - e$ este tot un graf conex.*

Demonstraţie. Afirmaţia rezultă din definiţia unui graf conex şi din următoarea observaţie: dintr-un $x - y$lanţ care conţine muchia e în G se poate obţine un $x - y$lanţ în $G - e$ înlocuind muchia e cu $C - e$. □

Propozitia 2.6. *Fie T un graf neorientat cu $n \geq 1$ vârfuri. Următoarele afirmaţii sunt echivalente.*

1. *T este arbore (conex şi aciclic)*
2. *T este conex muchie-minimal (prin eliminarea unei muchii din T se obţine un graf care nu mai este conex)*
3. *T este aciclic muchie-maximal*
4. *T este conex şi are $n - 1$ muchii*
5. *T este aciclic şi are $n - 1$ muchii*

6. Între oricare două vârfuri din T există un unic lanţ elementar.

Demonstraţie. "$1 \iff 2$": Pentru $n = 1$ este evident. Presupunem $n \geq 2$.

$1 \implies 2$: Presupunem că T este arbore (conex şi aciclic).

Fie $e = xy \in E(T)$. Arătăm că $T - e$ nu este conex (deci T este conex muchie-minimal). Presupunem prin absurd că $T - e$ este conex. Atunci există un lanţ elementar P în $T - e$ de la x la y. Atunci $P + xy = [x \overset{P}{-} y, x]$ este ciclu în T, contradicţie.

$2 \impliedby 1$: Presupunem că T este conex muchie-minimal. Demonstrăm că T este aciclic.

Presupunem prin absurd că T conţine un ciclu C. Fie $e \in E(C)$. Din Lema 2.5 rezultă că $T - e$ este conex, contradicţie (T este conex muchie-minimal).

"$1 \iff i$" pentru $i \in \{3, 4, 5, 6\}$ - Exerciţii ([4] - v. Secţiunea 2.1, [2] - v. Secţiunea 4.1) □

Corolar 2.7. *Orice graf conex conţine un arbore parţial (un graf parţial care este arbore).*

2.1 Construcţia unui arbore cu secvenţa gradelor dată

Teorema 2.8. *O secvenţă de $n \geq 2$ numere naturale* **strict pozitive** $s_0 = \{d_1, \ldots, d_n\}$ *este secvenţa gradelor unui arbore dacă şi numai dacă $d_1 + \ldots + d_n = 2(n-1)$.*

Demonstraţie. "$\Longrightarrow$" Presupunem că există un arbore T cu $s(T) = s_0$. Atunci

$$d_1 + \ldots + d_n = 2|E(T)| = 2(n-1) \text{ (conform Propoziţiei 2.6)}$$

"$\Longleftarrow$" **Varianta 1.** Demonstrăm prin inducţie după n că o secvenţă de n numere naturale (strict) pozitive $s_0 = \{d_1, \ldots, d_n\}$ cu proprietatea că $d_1 + \ldots + d_n = 2(n-1)$ este secvenţa gradelor unui arbore.

Pentru $n = 2$ avem $d_1, d_2 > 0$ şi $d_1 + d_2 = 2(2-1) = 2$, deci $d_1 = d_2 = 1$. Există un arbore cu secvenţa gradelor $s_0 = \{1, 1\}$, izomorf cu graful P_2 (lanţ cu două vârfuri).

Presupunem afirmaţia adevărată pentru $n - 1$. Fie $s_0 = \{d_1, \ldots, d_n\}$ o secvenţă de $n \geq 3$ numere naturale (strict) pozitive cu proprietatea că $d_1 + \ldots + d_n = 2(n-1)$. Arătăm că există un arbore cu secvenţa gradelor s_0.

Presupunem (fără a restrânge generalitatea) că $d_1 \geq \ldots \geq d_n$. Demonstrăm că $d_1 > 1$ şi $d_n = 1$.

- Presupunem prin absurd că $d_1 = 1$. Atunci $d_1 = \ldots = d_n = 1$. Rezultă $d_1 + \ldots + d_n = n$, de unde $2(n-1) = n$, deci $n = 2$, contradicţie ($n \geq 3$).
- Presupunem prin absurd că $d_n \geq 2$. Atunci $d_1 \geq \ldots \geq d_n \geq 2$. Rezultă $d_1 + \ldots + d_n \geq 2n$, de unde $2(n-1) \geq 2n$, contradicţie.

Considerăm secvenţa $s_0' = \{d_1 - 1, d_2, \ldots, d_{n-1}\}$. Numerele din secvenţă sunt pozitive şi avem

$$d_1 - 1 + d_2 + \ldots + d_{n-1} = d_1 + \ldots + d_n - (1 + d_n) = 2(n-1) - (1+1) = 2((n-1)-1).$$

Atunci putem aplica ipoteza de inducţie pentru secvenţa s_0'. Rezultă că există un arbore T' cu mulţimea vârfurilor $V' = \{x_1, \ldots, x_{n-1}\}$ având secvenţa gradelor $s(T') = s_0'$, mai exact cu:

$$d_{T'}(x_i) = \begin{cases} d_i - 1, & \text{dacă } i = 1 \\ d_i, & \text{dacă } i \in \{2, \ldots, n-1\}. \end{cases}$$

Construim pornind de la T' un nou arbore $T = (V, E)$ adăugând un vârf nou x_n pe care îl unim cu vârful x_1: $V = V' \cup \{x_n\}$, $E = E' \cup \{x_1x_n\}$. Pentru un $i \in \{1, \ldots, n\}$ avem atunci

$$d_T(x_i) = \begin{cases} d_{T'}(x_i) + 1 = d_i - 1 + 1 = d_i, & \text{dacă } i = 1 \\ d_{T'}(x_i) = d_i, & \text{dacă } i \in \{2, \ldots, n-1\} \\ 1 = d_n, & \text{dacă } i = n. \end{cases}$$

Rezultă că $s(T) = s_0$, deci s_0 este secvenţa gradelor unui arbore.

Varianta 2**.** Presupunem (fără a restrânge generalitatea) că $d_1 \geq \ldots \geq d_n$. Avem $d_1 \geq 1$ şi $d_n = 1$ (ca în varianta 1). Fie k astfel încât $d_k > 1$ şi $d_{k+1} = 1$.

Construim un arbore T de tip omidă cu mulţimea vârfurilor $\{x_1, \ldots, x_n\}$ având secvenţa gradelor s_0 astfel.

1. Unim printr-un lanţ vârfurile $x_1, \ldots, x_k$ (care trebuie să fie neterminale - formează corpul omizii), în această ordine
2. Considerăm mulţimea de vârfuri $\{x_{k+1}, \ldots, x_n\}$ (care trebuie să fie terminale) şi unim
   - x_1 cu primele $d_1 - 1$ vârfuri din această mulţime
   - x_2 cu următoarele $d_2 - 2$ vârfuri
   - ...
   - x_{k-1} cu următoarele $d_{k-1} - 2$ vârfuri
   - x_k cu ultimele $d_k - 1$ vârfuri din această mulţime.

Deoarece $d_1 - 1 + \sum_{i=2}^{k-1}(d_i - 2) + d_k - 1 = \sum_{i=1}^{n} d_i - 2k + 2 - (n - k) = n - k = |\{x_{k+1}, \ldots, x_n\}|$, construcţia este corectă.

Avem $s(T) = s_0$. □

Din demonstraţia Teoremei 2.8 se desprind următorii algoritmi de determinare a unui arbore cu secvenţa gradelor dată.

Algoritm de construţie a unui arbore cu secvenţa de grade dată

Intrare: o secvenţă de n numere naturale pozitive $d_1, \ldots, d_n$

Ieşire: un arbore T cu $V(T) = \{x_1, \ldots, x_n\}$ cu $s(T) = s_0$ dacă s_0 este secvenţa gradelor unui arbore, sau mesajul NU altfel.

Idee: La un pas unim un vârf de grad 1 cu un vârf de grad mai mare decât 1 şi actualizăm secvenţa s_0. Se repetă de $n - 2$ ori, în final rămânând în secvenţă două vârfuri de grad 1, care se unesc printr-o muchie.

Pseudocod:

Varianta 1**.**

Pasul 1. Dacă $d_1 + \ldots + d_n \neq 2(n-1)$, atunci scrie NU, STOP.

Pasul 2.

Cât timp s_0 conţine valori mai mari decât 1 execută *//sau pentru* $i = 1, n-2$

alege un număr $d_k > 1$ şi un număr $d_t = 1$ din secvenţă s_0 şi adaugă la T muchia x_kx_t.

elimină d_t din s_0

înlocuieşte d_k în secvenţa s_0 cu $d_k - 1$

Pasul 3.

fie d_k, d_t unicele elemente nenule (egale cu 1) din s_0; adaugă la T muchia $x_k x_t$.

Varianta 2**.** Corespunde variantei a doua de demonstrare a teoremei anterioare - construim un arbore de tip omidă.

3 Arbori parţiali de cost minim (apcm)

Pentru această secţiune se poate consulta cartea [1].

3.1 Algoritmul lui Kruskal

Intrare: $G = (V, E, w)$ - graf conex ponderat
Ieşire: un apcm T al lui G
Idee: La un pas este selectată o muchie de cost minim din G care nu formează cicluri cu muchiile deja selectate în T (care uneşte două componente conexe din graful deja construit)
Pseudocod:

$T = (V, \emptyset)$: iniţial fiecare vârf formează o componentă conexă în T
pentru $i = 1, n-1$ execută
    alege o muchie $e_i = uv$ de cost minim din G astfel încât u şi v sunt în componente conexe diferite în T
    $E(T) = E(T) \cup \{uv\}$: reuneşte compunenta lui u şi componenta lui v

Exemplu. Complexitate. Detalii implementare: v slide-uri+laborator.

3.1.1 Clustering

Algoritm de determinare a unui k-clustering cu grad de separare minim [1]
Intrare:
- o mulţime de n obiecte $S = \{o_1, \ldots, o_n\}$
- o funcţie de distanţă $d : S \times S \longrightarrow \mathbb{R}_+$
- k un număr natural (numărul de clase)

Ieşire: un k-clustering (o partiţionare $\mathcal{C}$ a lui S în k clase) cu grad de separare maxim ($\text{sep}(\mathcal{C}) = \min\{d(o, o') | o, o' \in S$, o şi o' sunt în clase diferite ale lui $\mathcal{C}\}$)
V. slide-uri pentru detalii si exemplu
Pseudocod:

Iniţial fiecare obiect formează o clasă
pentru $i = 1, n-k$ execută
    alege două obiecte o_r, o_t din clase diferite cu $d(o_r, o_t)$ minimă
    reuneşte clasa lui o_r şi clasa lui o_t
returnează cele k clase obţinute

Asociem problemei un graf complet $G \sim K_n$, în care vârfurile corespund obiectelor, iar o muchie între două vârfuri o_i, o_j are ponderea $w(o_i o_j) = d(o_i, o_j)$.

Atunci algoritmul este echivalent cu **algoritmul lui Kruskal**, oprit când numărul de muchii selectate este $n-k$, mai exact are următorul pseudocod:

Pseudocod:

Iniţial fiecare vârf formează o componentă conexă (clasă): $T' = (V, \emptyset)$

pentru $i = 1, n-k$ execută

alege o muchie $e_i = uv$ de cost minim din G astfel încât u şi v sunt în componente conexe diferite în T'

reuneşte componenta lui u şi componenta lui v: $E(T') = E(T') \cup \{uv\}$

returnează cele k mulţimi formate cu vârfurile celor k componente conexe ale lui T'

Observaţie 3.1. *Algoritmul este echivalent cu următorul, mai general:*

Pseudocod - variantă:

- determină un arbore parţial de cost minim T
- consideră mulţimea $\{e_{n-k+1}, \ldots, e_{n-1}\}$ formată cu $k-1$ muchii cu cele mai mari ponderi în T
- fie pădurea $T' = T - \{e_{n-k+1}, \ldots, e_{n-1}\}$
- defineşte clasele k-clustering-ului ca fiind mulţimile vârfurilor celor k componente conexe ale pădurii astfel obţinute

Propozitia 3.2. Corectitudine Algoritm Clustering

Algoritmul descris anterior determină un k-clustering cu grad de separare maxim.

Demonstraţie. La finalul algoritmului $E(T') = \{e_1, \ldots, e_{n-k}\}$, cu $w(e_1) \leq \ldots \leq w(e_{n-k})$, şi T' este o pădure cu k componente conexe, vârfurile componentelor determinând clasele lui $\mathcal{C}$.

Avem

$sep(\mathcal{C}) = \min\{w(e) |\ e = uv \in E(G)$ muchie din G ce uneşte două componente conexe din $T'\}$

Fie e_{n-k+1} următoarea muchie care ar fi fost selectată de algoritm dacă ar fi continuat cu $i = n-k+1$, adică muchia de cost minim care uneşte două componente conexe din T'. Atunci $sep(\mathcal{C}) = w(e_{n-k+1})$.

Presupunem că există un alt k-clustering $\mathcal{C}'$ cu sep($\mathcal{C}'$) > sep($\mathcal{C}$). Atunci există două obiecte p, q care sunt în aceeaşi clasă în $\mathcal{C}$, dar sunt în două clase diferite C'_i şi C'_j în $\mathcal{C}'$.

Deoarece p, q sunt în aceeaşi clasă a lui $\mathcal{C}$, atunci vârfurile p şi q sunt în aceeaşi componentă conexă a lui T', deci există P un lanţ de la p la q în T'. Mai mult, deoarece p şi q sunt în clase diferite ale lui $\mathcal{C}'$, există o muchie e a lui P cu o extremitate în C'_i şi cealaltă în C'_j. Rezultă că sep($\mathcal{C}'$)$\leq w(e)$. Dar $e \in E(P) \subseteq E(T')$, deci $w(e) \leq w(e_{n-k}) \leq w(e_{n-k+1}) = sep(\mathcal{C})$, de unde obţinem sep($\mathcal{C}'$)$\leq$sep($\mathcal{C}$), contradicţie.

□

3.2 Algoritmul lui Prim

Intrare: $G = (V, E, w)$ - graf conex ponderat
Ieşire: un apcm T al lui G
Idee: Se porneşte de la un vârf s (care formează arborele iniţial). La un pas este selectată o muchie de cost minim de la un vârf deja adăugat la arbore la unul neadăugat; se adaugă acest vârf la arbore împreună cu muchia selectată.
Pseudocod:

Fie s- vârful de start; $T = (\{s\}, \emptyset)$
pentru $i = 1, n-1$ execută
  alege o muchie uv cu cost minim astfel încât $u \in V(T)$ şi $v \notin V(T)$
  $E(T) = E(T) \cup \{uv\}$
  $V(T) = V(T) \cup \{v\}$: adaugă v la arbore

Exemplu. Complexitate. Detalii implementare: v slide-uri+laborator.

Corectitudinea algoritmilor lui Kruskal şi Prim

Notaţie. Pentru o mulţime de muchii $A \subseteq E(G)$, scriem $A \subseteq apcm$ dacă există un apcm T al lui G astfel încât $A \subseteq E(T)$ (A poate fi extinsă până devine mulţimea muchiilor unui apcm).

Atât algoritmul lui Kruskal, cât şi cel al lui Prim funcţionează după următoarea schemă:
Schemă:

$A = \emptyset$ (mulţimea muchiilor selectate în arborele construit)
pentru $i = 1, n-1$ execută
  alege o muchie e astfel încât $A \cup \{e\} \subseteq$ apcm
  $A = A \cup \{e\}$
returnează $T = (V, A)$

Vom demonstra un criteriu de alegere a muchiei e la un pas astfel încât dacă $A \subseteq$apcm, atunci şi $A \cup \{e\} \subseteq$ apcm şi vom demonstra că algoritmii lui Kruskal şi Prim aplică acest criteriu.

Propozitia 3.3. *Fie $G = (V, E, w)$ un graf conex ponderat.*
Fie $A \subseteq E$ astfel încât $A \subseteq$ apcm.
Fie $S \subset V$ astfel încât orice muchie din A are ambele extremităţii în S sau ambele extremităţi în $V - S$.
Fie e o muchie de cost minim cu o extremitate în S şi cealaltă în $V - S$ (v.fig. 3).
Atunci $A \cup \{e\} \subseteq$ apcm.

Demonstraţie. Fie T_{min} un apcm astfel încât $A \subseteq E(T_{min})$.
Dacă $e \in E(T_{min})$, atunci $A \cup \{e\} \subseteq E(T_{min})$.

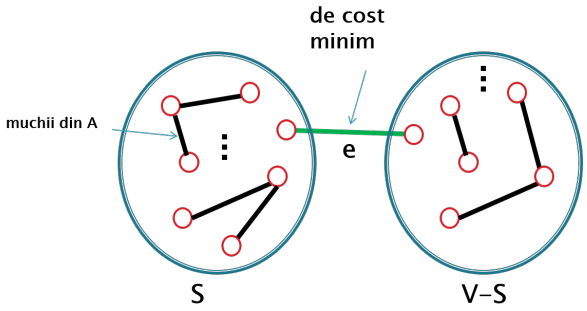


Figura 3

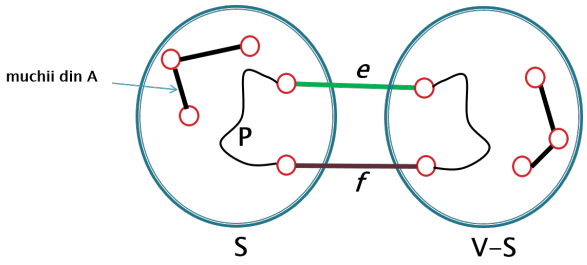


Figura 4

Dacă $e \notin E(T_{min})$, construim pornind de la T_{min} un alt apcm care să conţină şi muchia e (şi pe cele din A).

Notăm cu u şi v extremităţile muchiei e, astfel încât $u \in S$ şi $v \in V - S$.

Deoarece T_{min} este arbore, există un lanţ elementar P de la u la v în T_{min}. Lanţul P are o extremitate în mulţimea S (extremitatea u) şi cealaltă extremitate în mulţimea $V - S$ (extremitatea v). Rezultă că P conţine o muchie f cu o extremitate în S şi una în $V - S$ (v. fig. 4). Deoarece, conform ipotezei, A nu conţine nicio muchie de la S la $V - S$, avem $f \notin A$.

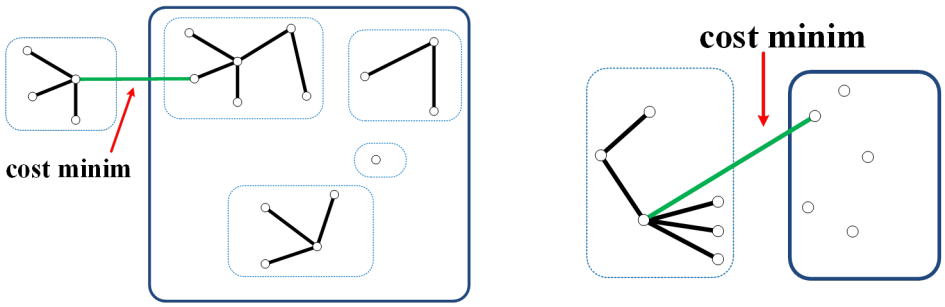
Considerăm graful $T' = T_{min} + e - f$. Deoarece $T_{min} + e$ conţine un unic ciclu elementar, format din lanţul P şi muchia e, iar f aparţine acestui ciclu, graful $T' = T_{min} + e - f$ este un graf conex şi aciclic, deci un arbore.

Avem:

- $A \subseteq E(T')$, deoarece $f \notin A$
- $w(T') = w(T_{min}) + w(e) - w(f) \leq w(T_{min})$, deoarece, conform ipotezei muchia e are cel mai mic cost dintre muchiile de la S la $V - S$, deci $w(e) \leq w(f)$. Rezultă $w(T') = w(T_{min})$, deci T' este tot apcm.

Am obţiunt astfel un apcm T' cu $A \cup \{e\} \subseteq E(T')$, deci $A \cup \{e\} \subseteq$ apcm.

□



(a) Kruskal (b) Prim

Figura 5

Teorema 3.4. - Corectitudinea algoritmului lui Kruskal. *Fie $G = (V, E, w)$ un graf conex ponderat. Algoritmul lui Kruskal determină un arbore parţial de cost minim al lui G.*

Demonstraţie. Fie $n = |V(G)|$. Algoritmul construieşte un graf T conex cu $n - 1$ muchii (şi aciclic), deci un arbore (la fiecare dintre cele $n - 1$ etape există o muchie în G care uneşte două componente din T, deoarece G este conex şi T este graf parţial al lui G).

Folosim Propoziţia 3.3 pentru a demonstra că după fiecare pas $i = \{1, \ldots, n - 1\}$, avem $E(T) \subseteq$ apcm. În final avem chiar egalitate, deoarece $|E(T)| = n - 1$, deci T este apcm.

Iniţial $E(T) = \emptyset \subseteq$ apcm.

Presupunem că înainte de etapa i avem $E(T) \subseteq$ apcm.

Fie $e = uv$ muchia aleasă de algoritm la pasul i şi $S =$ componenta conexă a lui u în pădurea curentă T - v. fig. 5 (a). Atunci $E(T)$ şi e verifică proprietăţile din Propoziţia 3.3 (toate muchiile din $E(T)$ sunt în interiorul componentelor conexe ale lui T, deci nu există muchie în $E(T)$ de la S la $V - S$).

Rezultă $E(T) \cup \{e\} \subseteq$ apcm. □

Teorema 3.5. - Corectitudinea algoritmului lui Prim. *Fie $G = (V, E, w)$ un graf conex ponderat. Algoritmul lui Prim determină un arbore parţial de cost minim al lui G.*

Demonstraţie. Algoritmul construieşte un graf T conex şi aciclic, deci un arbore.

Ca şi în demonstraţia Teoremei 3.4, faptul că T este un apcm rezultă aplicând la fiecare pas Propozitia 3.3 pentru mulţimea $S = V(T) =$ mulţimea vârfurilor deja adăugate la T până la pasul curent - v. fig. 5 (b). □

4 Drumuri minime în grafuri orientate ponderate

4.1 Drumuri minime de sursă unică (de la un vârf dat la celelalte)

4.1.1 Algoritmul lui Dijkstra

Intrare: $G = (V, E, w)$ - graf orientat ponderat cu ponderi **pozitive** ($w : E \longrightarrow \mathbb{R}_+^*$), s - vârf de start
Ieşire:

- vector d de distanţe, cu semnificaţia $d[x] =$ distanţa de la s la vârful x

- vectorul $tata$, reprezentând un arbore al distanţelor faţă de s (din care se poate determina câte un drum minim de la s la fiecare vârf x)

Idee: Fiecare vârf u are asociată o etichetă de distanţă $d[u]$ = costul minim al unui drum de la s la u descoperit până la acel pas = estimare superioară pentru distanţa de la s la u.

La un pas este ales ca vârf curent vârful u care estimat a fi cel mai apropiat de s şi se descoperă noi drumuri către vecinii lui u, actualizându-se etichetele de distanţă ale acestora.

Pseudocod:

iniţializează mulţimea vârfurilor nevizitate Q cu V
pentru fiecare $u \in V$ executa
    $d[u] = \infty$; $tata[u] = 0$
$d[s] = 0$
cat timp $Q \neq \emptyset$ executa
    u = extrage un vârf cu eticheta d minimă din Q
    pentru fiecare $uv \in E$ executa **//relaxarea arcului uv**
        daca $d[u] + w(u, v) < d[v]$ atunci
            $d[v] = d[u] + w(u, v)$
            $tata[v] = u$
scrie d, $tata$
//Variantă: scrie drum minim de la s la t un vârf t dat folosind $tata$

Exemplu. Complexitate. Detalii implementare: v slide-uri+laborator.

Corectitudinea algoritmului lui Dijkstra

Următoarele observaţii generale privind un drum minim între două vârfuri sunt evidente, dar importante în demonstrarea corectitudinii algoritmilor de drumuri minime.

Observaţie 4.1. *1. Dacă P este un drum minim de la s la u, atunci P este drum elementar.*

2. Dacă P este un drum minim de la s la u şi z este un vârf al lui P, atunci subdrumul lui P de la s la z este drum minim de la s la z.

Mai general, dacă x şi y sunt două vârfuri din P, atunci subdrumul lui P de la x la y este drum minim de la x la y.

Lema 4.2. *Pentru orice $u \in V$, la orice pas al algoritmului lui Dijkstra avem:*

(a) dacă $d[u] < \infty$, atunci există un drum de la s la u în G de cost $d[u]$ şi acesta se poate determina din vectorul tata, mai exact $tata[u]$= predecesorul lui u pe un drum de la s la u de cost $d[u]$.

(b) $d[u] \geq \delta(s, u)$

Demonstraţie. a) Demonstrăm afirmaţia prin inducţie după numărul de iteraţii (execuţii ale ciclului "cat timp").

Iniţial singura etichetă finită este $d[s] = 0$, corespunzătoare drumului $[s]$. La prima iteraţie este extras din Q vârful s şi pentru el afirmaţia se verifică.

Presupunem afirmaţia adevărată până la un pas.

Fie $u \in Q$ vârful extras la iteraţia curentă. Fie v cu $uv \in E$ un vecin al lui u căruia i se modifică eticheta la acest pas prin relaxarea arcului uv. Avem:

- după relaxarea arcului uv: $d[v] = d[u] + w(uv)$; $tata[v] = u$
- din ipoteza de inducţie există P un drum de la s la u de cost $d[u]$.

Atunci drumul $R = [s \overset{P}{-} u, v]$ este un drum de la s la v de cost

$$w(R) = w(P) + w(uv) = d[u] + w(uv) = d[v]$$

şi $u = tata[v]$ este predecesorul lui v pe acest drum.

□

Corolar 4.3. *Dacă la un pas al algoritmului pentru un vârf u avem relaţia $d[u] = \delta(s, u)$, atunci $d[u]$ nu se mai modifică până la final.*

Demonstraţie. Afirmaţia rezultă din Lema anterioară (nu există drum de la s la u cu cost mai mic decât cel minim, adică decât $\delta(s, u)$). □

Teorema 4.4. *Fie $G = (V, E, w)$ un graf orientat ponderat cu $w : E \longrightarrow \mathbb{R}_+^*$ şi $s \in V$ fixat.*

La finalul algoritmul lui Dijkstra avem:

$$d[u] = \delta(s, u) \text{ pentru orice } u \in V$$

şi $tata$ memorează un arbore al distanţelor faţă de s.

Demonstraţie. Vom demonstra prin inducţie după numărul de iteraţii (execuţii ale ciclului "cat timp") că după fiecare iteraţie avem $d[x] = \delta(s, x)$ pentru orice $x \in V - Q$ (eticheta vârfurilor deja selectate este corect calculaltă).

Iniţial avem $d[s] = \delta(s, s)$ şi primul vârf extras din Q este s, deci afirmaţia se verifică după prima iteraţie.

Presupunem că afirmaţia este adevărată până la iteraţia curentă: $d[x] = \delta(s, x)$ pentru orice $x \in V - Q$.

Fie $u \in Q$ vârful extras la această iteraţie. Demonstrăm că $d[u] = \delta(s, u)$ (adică eticheta lui u este corect calculată, fiind egală cu distanţa de la s la u).

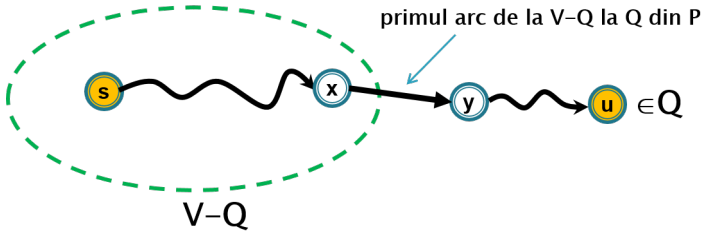


Figura 6

Dacă u nu este accesibil din s atunci, din Lema 4.2, $d[u] = \infty = \delta(s, u)$.

Altfel, fie P un drum minim de la s la u (avem $w(P) = \delta(s, u)$). Deoarece P are o extremitate $s \in V - Q$ şi cealaltă extremitate $u \in Q$, rezultă că P conţine un arc cu o extremitate în $V - Q$ şi cealaltă în Q. Fie xy primul astfel de arc din P (cu $x \in V - Q$ şi $y \in Q$) - v. fig. 6.

Din ipoteza de inducţie şi Observaţia 4.1 avem

$$d[x] = \delta(s, x) = w([s \overset{P}{-} x]).$$

La iteraţia la care a fost extras din Q vârful x, după relaxarea arcului xy avem:

$$d[y] \leq d[x] + w(xy) = w([s \overset{P}{-} x]) + w(xy) = w([s \overset{P}{-} y]).$$

Deoarece arcele au capacităţi pozitive, avem $w([s \overset{P}{-} y]) \leq w(P)$ şi deci

$$d[y] \leq w(P) = \delta(s, u) \leq d[u].$$

Dar, deoarece la pasul curent şi u şi y sunt în Q, din modul în care algoritmul alege vârful u avem

$$d[u] \leq d[y].$$

Rezultă că $d[u] = d[y] = w(P) = \delta(s, u)$ (şi P are vârfuri interne doar din mulţimea vârfurilor deja selectate).

Din Lema 4.2, pentru fiecare u, $d[u]$ este ponderea drumului de la s la u memorat în vectorul $tata$, deci acesta corespunde unui arbore al distanţelor faţă de s.

□

4.1.2 Drumuri minime în grafuri fără circuite (DAGs= Directed Acyclic Graphs)

Intrare: $G = (V, E, w)$ - graf orientat ponderat (ponderi reale) **fără circuite**, s - vârf de start
Ieşire:

- vector d de distanţe, cu semnificaţia $d[x]$ = distanţa de la s la vârful x

- vectorul $tata$, reprezentând un arbore al distanţelor faţă de s (din care se poate reconstrui câte un drum minim de la s la fiecare vârf x)

Idee: Fiecare vârf u are asociată o etichetă de distanţă $d[u]$ = costul minim al unui drum de la s la u descoperit până la acel pas (ca şi la Dijkstra).

Etichetele vârfurilor sunt calculate în ordinea dată de sortarea topologică. La un pas sunt relaxate toate arcele care ies din vârful curent.

Pseudocod:

pentru fiecare $u \in V$ executa
    $d[u] = \infty$; $tata[u] = 0$
$d[s] = 0$
$SortTop =$ sortare-topologica(G)
pentru fiecare $u \in SortTop$
    pentru fiecare $uv \in E$ executa
        daca $d[u] + w(u, v) < d[v]$ atunci //relaxam uv
            $d[v] = d[u] + w(u, v)$
            $tata[v] = u$
scrie d, tata

Exemplu. Complexitate. Detalii implementare: v slide-uri+laborator.

Corectitudine.

Are loc un rezultat similar celui pentru algoritmul lui Dijkstra

Lema 4.5. *Pentru orice $u \in V$, la orice pas al algoritmului de drumuri minime în grafuri fără circuite are loc proprietatea:*

(a) dacă $d[u] < \infty$, atunci există un drum de la s la u în G de cost $d[u]$ şi acesta se poate determina din vectorul tata, mai exact $tata[u]$= predecesorul lui u pe un drum de la s la u de cost $d[u]$.

(b) $d[u] \geq \delta(s, u)$

Teorema 4.6. *Fie $G = (V, E, w)$ un graf orientat ponderat fără circuite (DAG) şi $s \in V$ fixat.*

La finalul algoritmul de drumuri minime în grafuri fără circuite avem:

$$d[u] = \delta(s, u) \text{ pentru orice } u \in V$$

şi $tata$ memorează un arbore al distanţelor faţă de s.

Demonstraţie. Presupunem că toate vârfurile sunt accesibile din s. Altfel, eticheta lor va rămâne ∞ pe tot parcursul algoritmului, deci este corect calculată.

Vom demonstra prin inducţie după numărul de iteraţii (execuţii ale ciclului "pentru fiecare $u \in SortTop$") că la fiecare iteraţie avem $d[x] = \delta(s, x)$ pentru orice x deja considerat (aflat în ordonare topologică înaintea lui u).

Iniţial avem $d[s] = 0 = \delta(s, s)$, deci afirmaţia se verifică pentru prima iteraţie.

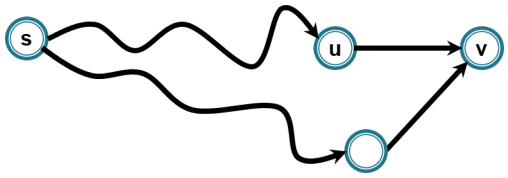


Figura 7

Presupunem că afirmaţia este adevărată până la iteraţia curentă. Fie $u \in SortTop$ vârful considerat la această iteraţie. Demonstrăm că $d[u] = \delta(s, u)$ (adică eticheta lui u este corect calculată, fiind egală cu distanţa de la s la u).

Variantă 1. Are loc relaţia (v. fig. 7) :

$$\delta(s, u) = \min\{\delta(s, x) + w(xu) | xu \in E\}, \forall u \neq s.$$

La pasul curent eticheta $d[u]$ calculată de algoritm prin relaxări de arce este

$$d[u] = \min\{d[x] + w(xu) | x \text{ înaintea lui } u \text{ în ordonarea topologică}\}.$$

Deoarece dacă $xu \in E$, x este înaintea lui u în ordonarea topologică, atunci rezultă:

$$d[u] = \min\{d[x] + w(xu) | xu \in E\}$$

şi, din ipoteza de inducţie, $d[x] = \delta(s, x)$. Obţinem

$$d[u] = \min\{d[x] + w(xu) | xu \in E\} = \min\{\delta(s, x) + w(xu) | xu \in E\} = \delta(s, u).$$

Varianta 2 (similar Dijkstra).

Fie P un drum minim de la s la u (avem $w(P) = \delta(s, u)$) şi x predecesorul lui u pe acest drum. Atunci x este înaintea lui u în ordonarea topologică deci, din ipoteza de inducţie şi Observaţia 4.1 avem

$$d[x] = \delta(s, x) = w([s \overset{P}{-} x]).$$

La iteraţia la care a fost considerat x, după relaxarea arcului xu avem:

$$d[u] \leq d[x] + w(xu) = w([s \overset{P}{-} x]) + w(xu) = w([s \overset{P}{-} u]) = \delta(s, u).$$

Din Lema 4.5, pentru fiecare u $d[u]$ este ponderea drumului de la s la u memorat în vectorul $tata$, deci acesta corespunde unui arbore al distanţelor faţă de s.

□

4.2 Drumuri minime între oricare două vârfuri

V. slide

5 Fluxuri în reţele de transport

5.1 Noţiuni introductive

Definiţie 5.1. *O* **reţea de transport** *este un cvintuplu $N = (G, S, T, I, c)$ unde*

- *$G = (V, E)$ este un graf orientat cu $V = S \cup I \cup T$, unde S, I, T sunt mulţimi disjuncte, nevide, cu semnificaţia:*
  - *S - mulţimea surselor (intrărilor)*
  - *T - mulţimea destinaţiilor (ieşirilor)*
  - *I - mulţimea vârfurilor intermediare*
- *$c : E \longrightarrow \mathbb{N}$ este funcţia capacitate, cu semnificaţia: $c(e)$ = cantitatea maximă care poate fi transportată prin arcul e.*

În cele ce urmează vom considera doar reţele cu următoarele proprietăţi

- $S = \{s\}$ - sursă unică (notată s)
- $T = \{t\}$ - destinaţie unică (notată t)
- $d^-(s) = 0$ - în sursă nu intră arce
- $d^+(t) = 0$ - din destinaţie nu ies arce

(oricărei reţele de transport i se poate asocia o reţea de transport care verifică aceste ipoteze, echivalentă din punct de vedere al valorii fluxului, prin adăugarea unei surse noi şi a unei destinaţii noi - v. fig 8).

În plus, vom presupune că orice vârf al lui G este accesibil din s.

În continuare vom nota o reţeau de transport $N = (G, s, t, I, c)$.

Notaţii. Fie reţeaua $N = (G, s, t, I, c)$, X şi Y două mulţimi de vârfuri şi o funcţie $f : E \longrightarrow \mathbb{N}$. Notăm:

- $f(X, Y) = \sum_{xy \in E, x \in X, y \in Y} f(xy)$
- $f^+(X) = f(X, V - X)$
- $f^-(X) = f(V - X, X)$.
- Pentru un vârf v notăm $f^+(v) = f^+(\{v\})$ şi $f^-(v) = f^-(\{v\})$.

Definiţie 5.2. *Fie o reţea de transport $N = (G, s, t, I, c)$ (cu proprietăţile amintite: $d^-(s) = d^+(t) = 0$). Un $s - t$* **flux** *în N (numit în continuare şi doar* **flux***) este o funcţie $f : E \longrightarrow \mathbb{N}$ cu proprietăţile:*

1. *$0 \leq f(e) \leq c(e), \forall e \in E(G)$ -* **condiţia de mărginire**
2. *pentru orice vârf intermediar $v \in I$, fluxul (total) care intră în vârful v este egal cu fluxul (total) care iese din vârful v -* **condiţia de conservare a fluxului***:*

$$\sum_{uv \in E} f(uv) = \sum_{vu \in E} f(vu),$$

sau, cu notaţiile anterioare,

$$f^-(v) = f^+(v), \forall v \in I.$$

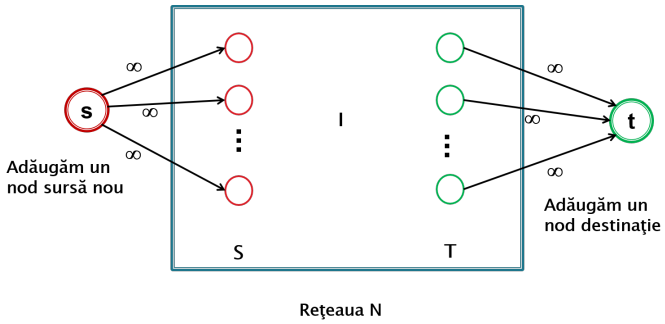


Figura 8

Valoarea fluxului f*, notată* $val(f)$*, se defineşte ca fiind suma fluxului pe arcele care ies din sursa* s*:*

$$val(f) = \sum_{sv \in E} f(sv) = f^+(s).$$

Vom demonstra ulterior că are loc relaţia

$$val(f) = f^+(s) = f^-(t).$$

Observaţie 5.3. *În orice reţea există cel puţin un flux, şi anume fluxul vid* $f = 0$*.*

Definiţie 5.4. *Un flux* f^* *în reţeaua* N *se numeşte* **flux maxim** *sau* **flux de valoare maximă***, dacă:*

$$val(f^*) = \max\{val(f) | f \textit{ flux în } N\}.$$

Problema fluxului maxim MAX-FLOW. Dată o reţea de transport $N = (G, s, t, I, c)$, să se determine un flux maxim (de valoare maximă) în N.

O problemă legată de problema fluxului maxim este problema determinării unei tăieturi minime.

Definiţie 5.5. *O* $s - t$ **tăietură** K *în reţeaua* N *(numită în cele ce urmează doar* **tăietură***) este o partiţionare a mulţimii vârfurilor* V *în două submulţimi* X *şi* $Y = V - X$ *(disjuncte) astfel încât* $s \in X$ *şi* $t \in Y$ *şi se notează* $K = (X, Y)$*. Un arc cu o extremitate în* X *şi cealaltă în* Y *(de la* X *la* Y*) se numeşte* **arc** *(sau* **arc direct***) al tăieturii* $K = (X, Y)$*, iar un arc de la* Y *la* X *se numeşte* **arc invers** *pentru tăietura* $K = (X, Y)$*.*

Capacitatea unei tăieturi $K = (X, Y)$*, notată cu* $c(K)$*, se defineşte ca fiind suma capacităţilor arcelor directe ale tăieturii (=suma capacităţilor arcelor care ies din* X*):*

$$c(K) = c^+(X) = \sum_{xy \in E, x \in X, y \in Y} c(xy).$$

O tăietură $\widetilde{K}$ *se numeşte* **tăietură minimă** *(***de capacitate minimă***) în* N *dacă*

$$c(\widetilde{K}) = \min\{c(K) | K \textit{ este } s - t \textit{ tăietură în } N\}$$

Problema tăieturii minime MIN-CUT. Dată o reţea de transport $N = (G, s, t, I, c)$, să se determine o tăietură minimă (de capacitate minimă) în N.

Vom demonstra că pentru orice flux f şi orice tăietură K în N are loc relaţia:

$$val(f) \leq c(K).$$

Va rezulta că, dacă are loc egalitate în această relaţie, f este flux maxim şi K este tăietură minimă. Astfel, algoritmul Ford-Fulkerson pentru problema fluxului maxim va găsi şi o soluţie pentru problema tăieturii minime.

Pentru a prezenta algoritmul Ford-Fulkerson de determinare a unui flux maxim, vom defini noţiunile necesare.

Fie N o reţea de transport şi f un flux în N.

- Un lanţ într-un graf orientat G corespunde unui lanţ în graful neorientat asociat acestuia (ignorând orientarea arcelor), mai exact este o succesiune de vârfuri şi arce

  $$P = [v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k]$$

  în care $e_i \in E(G)$ şi fie $e_i = v_{i-1}v_i$, fie $e_i = v_iv_{i-1}$ pentru fiecare $i \in \{1, \ldots, k\}$. Dacă $e_i = v_{i-1}v_i$, arcul e_i se numeşte **arc direct** sau **arc înaninte** în P, altfel se numeşte **arc invers** sau **înapoi**.
- Fie P un $s - v$ lanţ în N (în G), cu $v \in V$:

  $$P = [s = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k = v]$$

  Asociem fiecărui arc e din P o capacitate $i_P(e)$, numită **capacitate reziduală în raport cu** P, astfel:

  $$i_P(e) = \begin{cases} c(e) - f(e), & \text{dacă } e \text{ este arc direct în } P \\ f(e), & \text{dacă } e \text{ este arc invers în } P. \end{cases}$$

  ($i_P(e)$ care reprezintă cantitatea maximă cu care se poate modifica fluxul f pe arcul e dacă folosim pentru revizuirea fluxului lanţul P).
- **Capacitatea reziduală** $i(P)$ a unui $s - v$ lanţ P se defineşte ca fiind

  $$i(P) = \min\{i_P(e) | e \in E(P)\}$$

  ($i(P)$ reprezintă cantitatea maximă cu care se poate revizui fluxul f de-a lungul lanţului P).
  Prin convenţie, dacă P are un singur vârf , $i(P) = \min(\emptyset) = \infty$.

5.2 Revizuirea fluxului

Fie f un flux în reţeaua N.

Un $s - t$ lanţ cu $i(P) > 0$ se numeşte lanţ f-nesaturat sau **f-drum de creştere** (*augmenting path*).

Fie P un $s-t$ lanţ f-nesaturat. **Fluxul revizuit de-a lungul lanţului** P se defineşte ca fiind funcţia $f_P : E \longrightarrow \mathbb{N}$,

$$f_P(e) = \begin{cases} f(e) + i(P), & \text{dacă } e \text{ este arc direct al lui } P \\ f(e) - i(P), & \text{dacă } e \text{ este arc invers al lui } P \\ f(e), & \text{altfel} \end{cases}$$

Propozitia 5.6. *Fluxul revizuit f_P este flux în N şi*

$$val(f_P) = val(f) + i(P).$$

În plus, lanţul P este f_P-saturat.

Pentru $e = uv$ notăm cu $e^{-1} = vu$ inversul lui e (obţinut din e inversând orientarea).

În loc de a considera lanţuri cu arce inverse în reţeaua N (în graful G), putem gestiona capacităţile reziduale ale arcelor asociind reţelei şi fluxului f un graf numit graf rezidual, care conţine pe lângă arce din G şi arce obţinute din cele din G inversând orientarea (inverse ale arcelor din G).

Mai exact, **(multi)graful rezidual** G_f asociat fluxului f din N este un graf orientat ponderat, cu funcţia de pondere (de capacitate) c_f, construit astfel:

- $V(G_f) = V(G)$
- Pentru fiecare arc $e = uv$ cu $c(e) - f(e) > 0$, adăugăm arcul e la G_f cu ponderea asociată $c_f(e) = c(e) - f(e)$
- Pentru fiecare arc $e = uv$ cu $f(e) > 0$, adăugăm la G_f arcul $e^{-1} = vu$ (inversul arcului e) cu ponderea asociată $c_f(e^{-1}) = f(e)$.

Capacitatea unui drum P în G_f se defineşte ca fiind

$$c_f(P) = \min\{c_f(e) | e \in E(P)\}.$$

Notăm $E^{-1}(G_f) = \{e \in E(G) | e^{-1} \in E(G_f)\}$.

Observaţie 5.7. *O succesiune de vârfuri $P = [v_1, \ldots, v_k]$ este un $s-t$ lanţ f-nesaturat în G dacă şi numai dacă este $s-t$ drum în G_f; în plus, capacitatea reziduală $i(P)$ a lanţului P în G este egală cu capacitatea $c_f(P)$ a drumului P în G_f.*

Un arc e este arc invers în lanţul P în $G \Longleftrightarrow e^{-1}$ este arc în drumul P din G_f.

Vom demonstra că un flux este maxim dacă şi numai dacă nu există un $s-t$ lanţ f-nesaturat în G ($\Longleftrightarrow$ nu există $s-t$ drum în G_f), de aici rezultând şi corectiudinea algoritmului Ford-Fulkerson de determinare a unui flux maxim într-o reţea de transport.

5.3 Algoritmul Ford-Fulkerson

Intrare: $N = (G, s, t, I, c)$ - reţea de transport cu capacităţi numere naturale
Ieşire: un flux maxim f în G (+ o $s-t$ tăietură minimă în G)

Idee: La un pas este revizuit fluxul curent pe un $s-t$ lanţ nesaturat (drum de creştere). Se repetă revizuriea până când nu mai există un astfel de lanţ.

Pseudocod:

1. Fie f un flux în N (spre exemplu $f = 0$ fluxul vid: $f(e) = 0, \forall e \in E$)
2. Cât timp există un $s - t$ lanţ f-nesaturat P în G ($\Longleftrightarrow$ există un $s - t$ drum în G_f)
   determină un astfel de lanţ P în G ($\Longleftrightarrow$ un $s - t$ drum în G_f)
   revizuieşte fluxul f de-a lungul lanţului P
3. Fie X= mulţimea vârfurilor accesibile din s prin lanţuri f-nesaturate
   returnează f, $K = (X, V - X)$

Complexitate Determinarea unui lanţ - complexitate $O(m)$ (parcurgere)

La fiecare pas $val(f)$ creşte cu $i(P) \geq 1$.

Deoarece $val(f)$ este mai mică sau egală cu capacitatea oricărei tăieturi, putem determina ordinul de complexitate în funcţie de capacitatea tăieturii minime, notată $L \Longrightarrow O(mL)$.

Putem determina limite superioare pentru L: $L \leq c^+(s)$ sau $L \leq nU$ unde C este capacitatea maximă a unui arc ($O(nmC)$)

5.4 Corectitudinea Algoritmului Ford-Fulkerson

Lema 5.8. *Fie $N = (G, s, t, I, c)$ o reţea de transport, f flux în N şi $K = (X, Y = V - X)$ o tăietură în N. Are loc relaţia*

$$val(f) = f^+(X) - f^-(X) = f^-(Y) - f^+(Y).$$

Demonstraţie. Pentru $x \in X - \{s\}$, din condiţia de conservare a fluxului avem

$$f^+(x) - f^-(x) = 0.$$

Din definiţia valorii fluxului şi deoarece $f^-(s) = 0$ (deoarece în s nu intră arce), avem

$$f^+(s) - f^-(s) = val(f).$$

Adunând relaţiile obţinem

$$val(f) = \sum_{x \in X} (f^+(x) - f^-(x)).$$

Dar

$$\sum_{x \in X} f^+(x) = f(X, V) = f(X, V - X) + f(X, X) = f^+(X) + f(X, X).$$

Analog,

$$\sum_{x \in X} f^-(x) = f(V, X) = f^-(X) + f(X, X).$$

Rezultă

$$val(f) = f^+(X) + f(X, X) - (f^-(X) + f(X, X)) = f^+(X) - f^-(X).$$

În plus, deoarece $Y = V - X$, avem

$$f^+(X) = f(X, V - X) = f(X, Y) = f(V - Y, Y) = f^-(Y)$$

şi, analog,

$$f^-(X) = f^+(Y)$$

Rezultă astfel că

$$val(f) = f^+(X) - f^-(X) = f^-(Y) - f^+(Y)$$

□

Corolar 5.9. *Fie $N = (G, s, t, I, c)$ o reţea de transport şi f flux în N. Are loc relaţia*

$$val(f) = f^+(s) = f^-(t).$$

Demonstraţie. Considerăm în Lema 5.8 tăietura $K = (X, Y)$ cu $X = V - \{t\}$, $Y = \{t\}$. Rezultă $val(f) = f^-(t) - f^+(t) = f^-(t)$ (din t nu ies arce). □

Propozitia 5.10. *Fie $N = (G, s, t, I, c)$ o reţea de transport, f flux în N şi $K = (X, Y = V - X)$ o tăietură în N.*

a) Are loc relaţia

$$val(f) \leq c(K)$$

cu egalitate dacă şi numai dacă $f^+(X) = c^+(X)$ (toate arcele directe din K au fluxul egal cu capacitatea) şi $f^-(X) = 0$ (toate arcele inverse pentru K au fluxul 0).

b) Dacă are loc egalitatea $val(f) = c(K)$, atunci f este flux maxim şi K este tăietură minimă.

Demonstraţie. a) Din Lema 5.8 şi condiţia de mărginire pentru f avem:

$$val(f) = f^+(X) - f^-(X).$$

Dar $f^+(X) \leq c^+(X) = c(K)$ şi $f^-(X) \geq 0$. Rezultă

$$val(f) \leq c(K) - 0 = c(K)$$

cu egalitate doar dacă $f^+(X) = c^+(X)$ şi $f^-(X) = 0$.

b) Presupunem $val(f) = c(K)$.

Fie f^* flux maxim şi $\tilde{K}$ tăietură minimă. Din definiţiile fluxului maxim/ tăieturii minime rezultă relaţiile

$$val(f) \leq val(f^*) \leq c(\tilde{K}) \leq c(K) = val(f),$$

deci toate relaţiile sunt de egalitate: $val(f) = val(f^*)$ şi $c(\tilde{K}) = c(K)$. Rezultă că f are valoare maximă şi K are capacitate minimă. □

Propozitia 5.11. (Tăietura minimă asociată unui flux maxim) *Fie $N = (G, s, t, I, c)$ o reţea de transport, f flux în N.*

Dacă nu există un $s - t$ lanţ f-nesaturat în N, atunci există în N o tăietură K_f cu $val(f) = c(K_f)$ şi, mai mult, f este flux maxim şi K_f este tăietură minimă.

Demonstraţie. Fie $X = \{x \in v | \exists\ s - x$ lanţ f-nesaturat în $G\}$
(sau, echivalent, $X = \{x \in v | \exists\ s - x$ drum în $G_f\}$. Fie $K = (X, V - X)$.

Avem $s \in X$ şi $t \notin X$ (deoarece, conform ipotezei, nu există $s - t$ lanţ f-nesaturat în G), deci K este o $s - t$ tăietură.

Din Lema 5.8 avem

$$val(f) = f^+(X) - f^-(X).$$

Arătăm că avem $f^+(X) = c^+(X)$ (adică $f(e) = c(e)$ pentru orice arc e de la X la $Y = V - X$) şi $f^-(X) = 0$ (adică $f(e) = 0$ pentru orice arc e de la $Y = V - X$ la X). Rezultă atunci că

$$val(f) = c^+(X) = c(K).$$

Fie $e = xy$ cu $x \in X$ şi $y \in V - X$ un arc direct în K. Din definiţia lui X, rezultă că există Q un $s - x$ lanţ f-nesaturat în G: $i(Q) > 0$. Considerăm $s - y$ lanţul $P = [s \overset{Q}{-} x, e, y]$. Deoarece $y \notin X$, avem $i(P) = 0 = \min\{i(Q), i_P(e)\}$. Dar $i(Q) > 0$ şi e este arc direct în P, deci $i_P(e) = c(e) - f(e) = 0$. Rezultă $f(e) = c(e)$.

Fie $e = yx$ cu $x \in X$ şi $y \in V - X$ un arc invers în K. Raţionând analog, avem lanţul $P = [s \overset{Q}{-} x, e, y]$ în care e este acum arc invers. Obţinem $i_P(e) = f(e) = 0$

□

Din demonstraţia Propoziţiei anterioare rezultă următorul algoritm de determinare a unei tăieturi minime dat un flux maxim.

Algoritm de determinare a unei tăieturi minime dat un flux maxim

Intrare: $N = (G, s, t, I, c)$, f - flux maxim

Ieşire: o tăietură minimă $K = (X, Y)$ în N

Idee: X = mulţimea vârfurilor x accesibile din s prin $s - x$ lanţuri f-nesaturate. Aceste vârfuri se determină prin parcurgerea grafului G pornind din vârful s şi considerând doar arce cu capacitatea reziduală pozitivă (în raport cu lanţurile construite prin parcurgere, memorate cu vectorul tata) - v. alg Ford-Fulkeson.

Observaţie 5.12. *În finalul algoritmului Ford-Fulkeson mulţimea X care defineşte tăietura minimă $K = (X, Y)$ este chiar mulţimea vârfurilor vizitate la ultima parcurgere a grafului (în care nu s-a mai gasit un $s - t$ lanţ f-nesaturat).*

Teorema 5.13. (Caracterizarea unui flux maxim) *Fie $N = (G, s, t, I, c)$ o reţea de transport, f flux în N. Are loc echivalenţa*

f este flux maxim $\Longleftrightarrow$ nu există $s - t$ lanţ f-nesaturat în G.

Demonstraţie. " $\Longrightarrow$ " Presupunem că f este flux maxim.

Presupunem prin absurd că există P un $s-t$ lanţ f-nesaturat în G. Fie f_P fluxul obţinut prin revizuirea lui f de-a lungul lui P. Conform Propoziţiei 5.6 avem:

$$val(f_P) = val(f) + i(P) > val(f),$$

ceea ce contrazice faptul că f este flux maxim.

" $\Longleftarrow$ " Presupunem că nu există $s-t$ lanţ f-nesaturat în G. Atunci, conform Propoziţiei 5.11, există o tăietură K cu $val(f) = c(K)$ şi f este flux maxim. □

Teorema 5.14. (Corectitudinea Algoritmului Ford-Fulkerson) *Algoritmului Ford-Fulkerson se termină într-un număr finit de paşi şi fluxul f determinat de algoritm este flux maxim. În plus, mulţimea X a vârfurilor accesibile din s prin lanţuri f-nesaturate determină o tăietură $K = (X, V - X)$ de capacitate minimă.*

Demonstraţie. Fie L capacitatea minimă a unei $s-t$ tăieturi în N. Fie f un flux în N. Avem

$$val(f) \leq L.$$

Conform Propoziţiei 5.6, dacă P este un $s-t$ lanţ f-nesaturat, avem $val(f_P) = val(f) + i(P)$. Dar, deoarece $i(P) > 0$ şi $i(P) \in \mathbb{N}$, avem $i(P) \geq 1$ şi deci

$$val(f_P) \geq val(f) + 1.$$

Rezultă că la fiecare iteraţie a algoritmului Ford-Fulkerson valoarea fluxului creşte cu cel puţin 1. Deoarece această valoare este mai mică sau egală cu L, rezultă că algoritmul are cel mult L etape.

În final fluxul f obţinut verifică proprietatea: nu există $s-t$ lanţ f-nesaturat, deci, conform Propoziţiei 5.13, este flux maxim. □

Teorema 5.15. (Teorema Ford-Fulkerson / MAX-FLOW, MIN-CUT Theorem) *Într-o reţea de transport $N = (G, s, t, I, c)$ valoarea unui $s-t$ flux maxim este egală cu capacitatea unei $s-t$ tăieturi minime.*

Demonstraţie. Fie f^* flux maxim. Din Propoziţia 5.13 rezultă că nu există $s-t$ lanţ f^*-nesaturat. Atunci, conform Propoziţiei 5.11, există o tăietură minimă $\tilde{K}$ cu $val(f^*) = c(\tilde{K})$. □

5.5 Aplicaţii - probleme care se reduc la determinarea unui flux maxim

5.5.1 Cuplaje în grafuri bipartite

Fie $G = (V, E)$ un graf neorientat. Un cuplaj în G este o mulţime de muchii $M \subseteq E$ neadiacente două câte două.

Problema cuplajului maxim (Maximum-Matching). Dat un graf neorientat G, să se determine un cuplaj de cardinal maxim în G.

Vom descrie un algoritm care determină un cuplaj maxim într-un graf **bipartit** reducând această problemă la problema determinării unui flux maxim într-o reţea asociată lui N.

Algoritm de determinare a unei cuplaj maxim într-un graf bipartit

Intrare: $G = (V = X \cup Y, E)$ un graf bipartit

Ieşire: un cuplaj de cardinal maxim în G

Pseudocod:

1. Asociem grafului G o reţea $N_G = (G', s, t, X \cup Y, c)$ astfel:
   - $V(G') = V(G) \cup \{s, t\}$
   - adaugăm la $E(G')$ arce xy cu $x \in X$, $y \in Y$ (orientate de la x la y) corespunzătoare muchiilor xy din G
   - adaugăm la $E(G')$ arcele sx, $x \in X$
   - adaugăm la $E(G')$ arcele yt, $y \in Y$
   - asociem arcelor capacitatea 1 ($c = 1$)
2. Determinăm f un flux maxim în N.
3. Fie $M = \{xy \in E(G) | f(xy) = 1, x, y \in X \cup Y\}$.
4. returnează M

Complexitate - În N_G capacitatea arcelor este 1. Algoritmul Ford-Fulkerson pentru o astfel de reţea are complexitatea $O(nmC) = O(nm)$.

Corectitudine

Propozitia 5.16. *Fie $G = (V = X \cup Y, E)$ un graf bipartit şi N_G reţeaua de transport asociată în algoritm.*

a) Fie M un cuplaj în G. Atunci există în N_G un flux f cu valoarea $val(f) = |M|$ definit astfel:
- *pentru fiecare $xy \in M$, $x \in X$, $y \in Y$: $f(xy) = 1$*
- *pentru fiecare $x \in X \cap V(M)$: $f(sx) = 1$*
- *pentru fiecare $y \in Y \cap V(M)$: $f(yt) = 1$*
- *pentru toate celelalte arce $f(e) = 0$*

b) Fie f un flux în N_G. Atunci există în G un cuplaj de cardinal $val(f)$, şi anume

$$M = \{xy \in E(G) | x \in X, y \in Y, f(xy) = 1\}$$

Demonstraţie. a) Arătăm că f definit ca în enunţ este flux. Din definiţia lui f avem $f(e) \leq 1 \leq c(e)$ pentru orice arc e, deci este satisfacută condiţia de mărginire.

Fie $x \in X$. Dacă $x \notin V(M)$ atunci $f^+(x) = f^-(x) = 0$. Altfel, deoarece M este cuplaj, există un unic y cu $xy \in M$, deci un unic arc $e = xy$ care iese din x cu flux pozitiv. Avem atunci $f(sx) = f(xy) = 1$ şi deci $f^+(x) = f^-(x) = 1$. Analog, pentru $y \in Y$, avem

$$f^+(y) = f^-(y) = \begin{cases} 1, & \text{dacă } y \in V(M) \\ 0, & \text{dacă } y \notin V(M) \end{cases}$$

În plus

$$val(f) = \sum_{sx \in N_G} f(sx) = \sum_{x \in X \cap V(M)} 1 = |X \cap V(M)| = |M|.$$

b) Fluxul f pe fiecare arc este 0 sau 1. Pentru orice $x \in X$ există un unic arc care intră în x, anume sx. Deoarece $c(sx) = 1$, rezultă că $f^+(x) = f^-(x) \in \{0, 1\}$. Analog, pentru orice $y \in Y$, deoarece $c(yt) = 1$, avem $f^+(y) = f^-(y) \in \{0, 1\}$.

Arătăm că M este cuplaj.

Fie $xy \in M$. Avem $f(xy) > 0$, deci $f^+(x) = f^-(x) = 1$. Rezultă că xy este unicul arc care iese din x cu flux pozitiv. Din definiţia lui M rezultă că xy este unica muchie din M incidentă în x. Analog se demonstrează că xy este unica muchie din M incidentă în y, deci M este cuplaj.

Ca şi la a) avem

$$val(f) = \sum_{sx \in N_G} f(sx) = \sum_{x \in X \cap V(M)} 1 = |X \cap V(M)| = |M|.$$

(sau, folosind tăieturi, avem:

$$val(f) = f^+(X \cup \{s\}, Y \cup \{t\}) - f^-(X \cup \{s\}, Y \cup \{t\}) = |M| - 0 = |M|.)$$

□

Corolar 5.17. *Fie f^* un flux maxim în N_G. Atunci cuplajul M^* corespunzător, definit prin*

$$M^* = \{xy \in E(G) | x \in X, y \in Y, f^*(xy) = 1\}$$

este cuplaj maxim în G.

Demonstraţie. Din Propoziţia 5.16 avem $|M^*| = val(f^*)$.

Dacă prin absurd ar exista un cuplaj M' cu $|M'| > |M^*|$, atunci fluxul f' corespunzător lui M' (definit ca în Propoziţia 5.16) are valoarea

$$val(f') = |M'| > |M^*| = val(f^*),$$

ceea ce contrazice faptul că f^* este maxim. □

5.5.2 Construcţia unui graf orientat cu secvenţele de grade interne şi externe date

Algoritm de determinare a unui graf orientat cu secvenţele de grade interne şi externe date
Intrare: Două secvenţe de n numere naturale: $s_0^+ = \{d_1^+, \ldots, d_n^+\}$ şi $s_0^- = \{d_1^-, \ldots, d_n^-\}$ cu

$$d_1^+ + \ldots + d_n^+ = d_1^- + \ldots + d_n^-$$

Ieşire: Un graf orientat G cu $s^+(G) = s_0^+$ şi $s^-(G) = s_0^-$ dacă există, mesajul NU altfel.
Pseudocod:

1. Construim o reţea $N = (G', s, t, X \cup Y, c)$ astfel:

- $G' = (X \cup Y \cup \{s,t\}, E')$ unde
  - $X = \{x_1, \ldots, x_n\}$
  - $Y = \{y_1, \ldots, y_n\}$
  - $E' = \{x_i y_j | i, j\{1, \ldots, n\}, i \neq j\} \cup \{sx_i | x_i \in X\} \cup \{y_j t | y_j \in Y\}$
- Definim capacităţile arcelor astfel:
  - $c(sx_i) = d_i^+$, pentru $sx_i \in E$
  - $c(y_j t) = d_j^-$, pentru $y_j t \in E$
  - $c(x_i y_j) = 1$, pentru $x_i y_j \in E$

2. Determinăm f un flux maxim în N.
3. Dacă $val(f) \neq d_1^+ + \ldots + d_n^+$, atunci scrie "NU", STOP.
4. Definim graful orientat $G = (V, E)$ cu $V = \{1, \ldots, n\}$ şi $E = \{ij | f(x_i y_j) = 1\}$
5. returnează G

Complexitate - Capacitatea minimă a unei tăieturi $L \leq c^+(s) = d_1^+ + \ldots + d_n^+ = |E(G)| \Longrightarrow O(m^2)$.
Corectitudine

Propozitia 5.18. *Fie $s_0^+ = \{d_1^+, \ldots, d_n^+\}$ şi $s_0^- = \{d_1^-, \ldots, d_n^-\}$ două secvenţe de n numere naturale cu $d_1^+ + \ldots + d_n^+ = d_1^- + \ldots + d_n^-$.*

Fie N reţeaua construită în algoritm.

Atunci există un graf orientat G cu $s^+(G) = s_0^+$ c si $s^-(G) = s_0^-$ dacă şi numai dacă valoarea fluxului maxim în N este $d_1^+ + \ldots + d_n^+$.

Demonstraţie. "$\Longrightarrow$" Presupunem că există un graf orientat G cu $V(G) = \{1, \ldots, n\}$ având $s^+(G) = s_0^+$ şi $s^-(G) = s_0^-$. Definim un flux f în N astfel:

- $f(x_i y_j) = 1$ pentru $ij \in E(G)$
- $f(sx_i) = d_i^+$, pentru $i \in \{1, \ldots, n\}$
- $f(y_j t) = d_j^-$, pentru $j \in \{1, \ldots, n\}$

Arătăm că f este corect definit. Evident f satisface condiţia de mărginire.

Fie $i \in \{1, \ldots, n\}$ fixat. Avem

$$f^+(x_i) = \sum_{ij \in E(G)} f(x_i y_j) = |\{ij | j \in V, ij \in E(G)\}| = d_G^+(i) = d_i^+.$$

Dar şi $f^-(x_i) = f(sx_i) = d_i^+$, deci f satisface condiţia de conservare în x_i. Analog, pentru $j \in \{1, \ldots, n\}$ avem $f^-(y_j) = d_j^- = f(y_j t) = f^+(y_j)$.

Rezultă că f este flux cu $val(f) = f^+(s) = f(sx_1) + \ldots + f(sx_n) = d_1^+ + \ldots + d_n^+ = c^+(s)$, deci este şi un flux maxim.

"$\Longleftarrow$" Presupunem că există un flux f în N cu $val(f) = d_1^+ + \ldots + d_n^+$. Avem atunci

$$\begin{aligned} &d_1^+ + \ldots + d_n^+ = val(f) = f^+(s) = \\ &= f(sx_1) + \ldots + f(sx_n) \leq c(sx_1) + \ldots + c(sx_n) = \\ &= d_1^+ + \ldots + d_n^+, \end{aligned}$$

deci au loc egalităţile $f(sx_i) = d_i^+$ pentru orice $i \in \{1, \ldots, n\}$. Analog,

$$\begin{aligned} &d_1^- + \ldots + d_n^- = d_1^+ + \ldots + d_n^+ = val(f) = f^-(t) = \\ &= f(y_1 t) + \ldots + f(y_n t) \leq c(y_1 t) + \ldots + c(y_n t) = \\ &= d_1^- + \ldots + d_n^-, \end{aligned}$$

deci $f(y_j t) = d_j^-$ pentru orice $j \in \{1, \ldots, n\}$.

Definim graful orientat $G = (V, E)$ astfel: $V = \{1, \ldots, n\}$ şi

$$E = \{ij | f(x_i y_j) > 0\} = \{ij | f(x_i y_j) = 1\}.$$

Pentru $i \in \{1, \ldots, n\}$ fixat avem

$$\begin{aligned} &d_G^+(i) = |\{ij | j \in \{1, \ldots, n\}, ij \in E\}| = |\{x_i y_j | j \in \{1, \ldots, n\}, f(x_i y_j) = 1\}| = \\ &= \sum_{x_i y_j \in N, j \in \{1, \ldots, n\}} f(x_i y_j) = f^+(x_i) = f^-(x_i) = f(sx_i) = d_i^+, \end{aligned}$$

deci $s^+(G) = s_0^+$.

Analog $d_G^-(j) = f^-(y_j) = f^+(y_j) = f(y_j t) = d_j^-$, deci $s^-(G) = s_0^-$. □

5.6 Conectivitate

- Suplimentar, v. slide.

Bibliografie

[1] Jon Kleinberg, Eva Tardos, Algorithm Design, Addison-Wesley 2005. http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

[2] Dragoş-Radu Popescu, *Combinatorică şi teoria grafurilor*. Editura Societatea de Ştiinţe Matematice din România, Bucureşti, 2005.

[3] Dragoş-Radu Popescu, R. Marinescu-Ghemeci, *Combinatorică şi teoria grafurilor prin exerciţii şi probleme*. Editura Matrixrom, 2014.

[4] Douglas B. West, *Introduction to Graph Theory*. Prentice Hall 1996, 2001.