



1

## Cuplaje

- Cuplaje perfecte - Teorema lui Tutte
- Cuplaje de cardinal maxim - Teorema lui Berge
- Cuplaje de cardinal maxim - Algoritmul Hopcroft-Karp

2

## Fluxuri în rețele

- Rețele de transport
- Flux, valoarea fluxului, problema fluxului maxim

3

## Exerciții pentru seminarul 8 (săptămâna 24-28 noiembrie)

Fie  $G$  be un graf. Evident,  $|M| \leq |G|/2, \forall M \in \mathcal{M}_G$ . Un **cuplaj perfect** (sau **1-factor**) în  $G$  este un cuplaj  $M$  cu proprietatea  $|M| = |G|/2$  (i. e.,  $S(M) = V(G)$ ).

O componentă conexă a grafului  $G$  este pară (impară) dacă numărul nodurilor sale este par (impar). **Notăm cu  $q(G)$  numărul componentelor conexe impare ale lui  $G$ .**

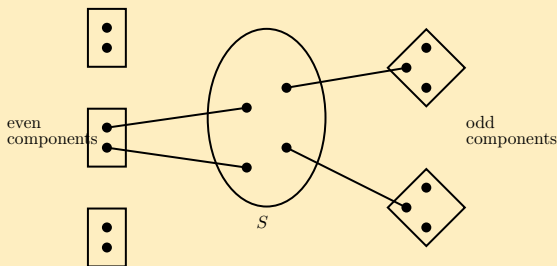
### Remarcă

Fie  $G$  un graf care are un cuplaj perfect. Evident, fiecare componentă conexă a lui  $G$  este pară. Astfel,  $q(G) = 0$ . Mai mult, dacă  $S \subseteq V(G)$  atunci pentru fiecare componentă conexă impară din graful  $G - S$  trebuie să existe o muchie în cuplajul perfect al grafului cu o extremitate în această componentă conexă și cealaltă în  $S$ . Deoarece extremitățile muchiilor diferite sunt distincte, urmează că  $|S| \geq q(G - S)$ .

Pentru  $S = \emptyset$  în (T), obținem  $q(G - \emptyset) \leq 0$ , deci  $q(G) = 0$ .

# Cuplaje perfecte - Teorema lui Tutte

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph



## Teorema 1

(Tutte, 1947) Un graf  $G$  admite cuplaj perfect dacă și numai dacă

$$(T) \quad q(G - S) \leq |S|, \forall S \subseteq V(G).$$

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

**Demonstrație.** Necesitatea condiției (T) a fost dovedită în discuția de mai sus. Demonstrăm prin inducție după  $n = |G|$  că dacă un graf  $G = (V, E)$  satisface (T), atunci  $G$  are un cuplaj perfect.

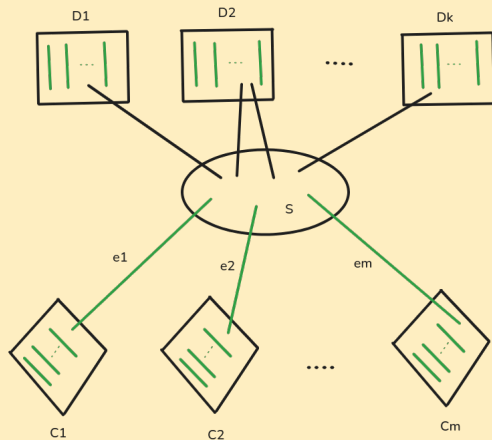
Pentru  $n = 1, 2$  teorema are loc evident. În pasul inductiv, fie  $G$  un graf cu  $n \geq 3$  noduri care satisface (T) și să presupunem că orice graf  $G'$  cu  $|G'| < n$  și care satisface (T) are un cuplaj perfect.

Fie  $S_0 \subseteq V(G)$  astfel încât  $q(G - S_0) = |S_0|$  și maximală cu proprietatea că avem egalitate în (T) (i. e., pentru orice supramulțime  $S$  a lui  $S_0$  avem  $q(G - S) < |S|$ ).

Observăm că familia de submulțimi ale lui  $V(G)$  pentru care (T) este satisfăcută cu egalitate este nevidă, și, deci, există un  $S_0$  (pentru fiecare nod  $v_0$  care nu este punct de articulație în componența lui din  $G$ , avem  $q(G - v_0) = 1 = |\{v_0\}|$ , deoarece fiecare componentă conexă este pară și în fiecare componentă conexă cu cel puțin un nod, există un astfel de nod  $v_0$ ).

## Cuplaje perfecte - Teorema lui Tutte

Fie  $m = |S_0| > 0$ ,  $C_1, C_2, \dots, C_m$  componentele impare ale lui  $G - S_0$  și  $D_1, D_2, \dots, D_k$  componentele pare ale lui  $G - S_0$  ( $k \geq 0$ ):



Vom construi un cuplaj perfect în  $G$  compus din

## Cuplaje perfecte - Teorema lui Tutte

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

- câte un cuplaj perfect în fiecare componentă conexă pară  $D_i$ ;
- un cuplaj cu  $m$  muchii,  $\{e_1, \dots, e_m\}$ , muchia  $e_i$  având un capăt  $s_i \in S_0$  și celălalt  $v_i \in C_i$  ( $i = 1, \overline{1, m}$ );
- cuplaj perfect în fiecare subgraph  $C_i - v_i$  ( $i = 1, \overline{1, m}$ ).

a) Pentru orice  $1 \leq i \leq k$  graful  $[D_i]_G$  admite cuplaj perfect. Într-adevăr, deoarece  $m > 0$ , urmează că  $|D_i| < n$  și din ipoteza inductivă este suficient să arătăm că  $G' = [D_i]_G$  satisface (T).

Fie  $S' \subseteq D_i$ . Dacă  $q(G' - S') > |S'|$ , atunci obținem următoarea contradicție:

$$q(G - (S_0 \cup S')) = q(G - S_0) + q(G' - S') = |S_0| + q(G' - S') > |S_0 \cup S'|.$$

Astfel,  $q(G' - S') \leq |S'|$ ,  $\forall S' \subseteq D_i$ , i. e.,  $G'$  satisface (T).

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

## Cuplaje perfecte - Teorema lui Tutte

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

b) Fie  $H = (S_0, \{C_1, \dots, C_m\}; E')$  graful bipartit cu o clasă a bipartiției  $S_0$ , cealaltă clasă mulțimea componentelor conexe impare ale lui  $G - S_0$ , și cu muchiile de forma  $\{s, C_i\}$ , unde  $s \in S_0$  astfel că există  $v \in C_i$  cu  $sv \in E(G)$ .

$H$  are un cuplaj perfect. Într-adevăr, arătăm că  $H$  satisface condiția din teorema lui Hall pentru existența unui cuplaj  $M_0$  care saturează  $\{C_1, \dots, C_m\}$ .

Fie  $A \subseteq \{C_1, \dots, C_m\}$ . Atunci  $B = N_H(A) \subseteq S_0$ , și din construcția lui  $H$ , în graful  $G$  nu avem nicio muchie de la un nod  $v \in S_0 - B$  la un nod  $w \in C_i \in A$ . Astfel componentele conexe impare din  $A$  rămân componente conexe impare și în  $G - B$ ; astfel  $q(G - B) \geq |A|$ . Deoarece  $G$  satisface condiția lui Tutte (T), avem  $|B| \geq q(G - B)$ . Am obținut că  $|N_H(A)| = |B| \geq |A|$ .

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

Din teorema lui Hall  $H$  are un cuplaj  $M_0$ , care saturează  $\{C_1, \dots, C_m\}$ ; deoarece  $|S_0| = m$ ,  $M_0$  este perfect.

$$M_0 = \{s_1 v_1, s_2 v_2, \dots, s_m v_m\}, S_0 = \{s_1, \dots, s_m\}, v_i \in C_i, \forall i = \overline{1, m}.$$

c)  $\forall i \in \{1, \dots, m\}$  graful  $G' = [C_i - v_i]_G$  admite cuplaj perfect. Folosind ipoteza inductivă, este suficient să dovedim că  $G'$  satisface (T).

Fie  $S' \subseteq C_i - v_i$ . Dacă  $q(G' - S') > |S'|$ , atunci, deoarece  $q(G' - S') + |S'| \equiv 0 \pmod{2}$  (pentru că  $|G'|$  este par), urmează că  $q(G' - S') \geq |S'| + 2$ .

Dacă  $S'' = S_0 \cup \{v_i\} \cup S'$ , avem

$$\begin{aligned} |S''| &\geq q(G - S'') = q(G - S_0) - 1 + q(G' - S') = |S_0| - 1 + q(G' - S') \geq \\ &\geq |S_0| - 1 + |S'| + 2 = |S''|, \end{aligned}$$

i. e.,  $q(G - S'') = |S''|$ , în contradicție cu alegerea lui  $S_0$  (deoarece  $S_0 \subsetneq S''$ ).



Fie  $G = (V, E)$  un graf și  $M \in \mathcal{M}_G$  un cuplaj al lui  $G$ .

## Definiție

Un drum alternat în  $G$  relativ la cuplajul  $M$  este un drum

$$P : v_0, v_0 v_1, v_1, \dots, v_{k-1}, v_{k-1} v_k, v_k$$

astfel încât  $\{v_{i-1} v_i, v_i v_{i+1}\} \cap M \neq \emptyset, \forall i = \overline{1, k-1}$ .

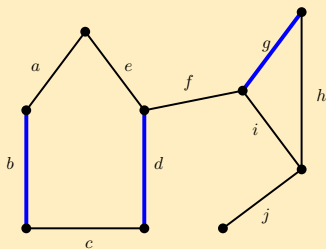
Observăm că, deoarece  $M$  este un cuplaj, dacă  $P$  este un drum alternat relativ la  $M$ , atunci dintre orice două muchii consecutive ale lui  $P$  exact una aparține lui  $M$  (muchii aparțin alternativ lui  $M$  și  $E \setminus M$ ).

În cele ce urmează, când ne vom referi la un drum  $P$  vom înțelege mulțimea muchiilor sale.

## Definiție

Un drum de creștere al lui  $G$  relativ la cuplajul  $M$  este un drum alternat care unește două noduri distincte expuse relativ la  $M$ .

Observăm că, din definiția de mai sus, urmează că dacă  $P$  este un drum de creștere relativ la  $M$ , atunci  $|P \setminus M| = |P \cap M| + 1$ .



$a, b, c, d$  - alternating even path

$f$  - alternating odd path

$j$  - augmenting path

$g, f, d$  - alternating odd path

$a, b, c, d, e$  - closed alternating path

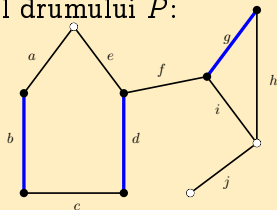
$a, b, c, d, f, g, h$  - augmenting path

## Teorema 2

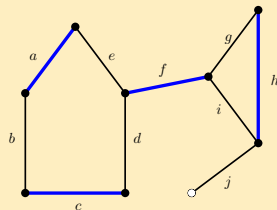
(Berge, 1959)  $M$  este un cuplaj de cardinal maxim în graf  $G$  dacă și numai dacă nu există drum de creștere în  $G$  relativ la  $M$ .

**Demonstrație.** " $\Rightarrow$ " Fie  $M$  un cuplaj de cardinal maxim în  $G$ . Să presupunem că  $P$  este un drum de creștere în  $G$  relativ la  $M$ .

Atunci,  $M' = M \Delta P = (P \setminus M) \cup (M \setminus P) \in \mathcal{M}_G$ . Într-adevăr,  $M'$  poate fi obținut prin interschimbarea muchiilor din  $M$  cu cele din afara lui  $M$  de-a lungul drumului  $P$ :



$P = a, b, c, d, f, g, h$  - augmenting path



$M \Delta P$

## Cuplaje de cardinal maxim - Teorema lui Berge

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

Mai mult,  $|M'| = |P \cap M| + 1 + |M \setminus P| = |M| + 1$ , în contradicție cu alegerea lui  $M$ .

" $\Leftarrow$ " Fie  $M$  un cuplaj în  $G$  cu proprietatea că nu există drumuri de creștere în  $G$  relativ la  $M$ .

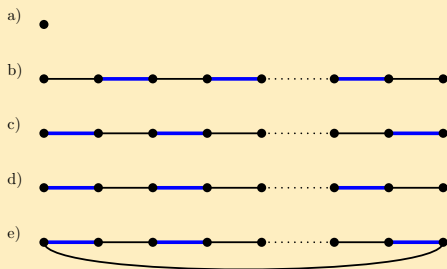
Dacă  $M^*$  este un cuplaj de cardinal maxim în  $G$ , vom arăta că  $|M^*| = |M|$ . Fie  $G'$  subgraful generat de  $M \Delta M^*$  în  $G$  ( $G' = (V, M \Delta M^*)$ ).

Să observăm că  $d_{G'}(v) \leq 2, \forall v \in V$  și deci componentele conexe ale lui  $G'$  pot fi noduri izolate, drumuri de lungime cel puțin unu, sau circuite. Avem cinci posibilități (se văd mai jos; muchiile albastre sunt din  $M^*$ , muchiile negre sunt din  $M$ ).

Cazul b) nu apare deoarece este un drum de creștere relativ la  $M^*$ , care este un cuplaj de cardinal maxim. Cazul c) nu apare deoarece este un drum de creștere relativ la  $M$ .

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

## Cuplaje de cardinal maxim - Teorema lui Berge



Dacă notăm cu  $m_M(C)$  numărul de muchii din  $M$  din componenta conexă  $C$  a lui  $G'$  și cu  $m_{M^*}(C)$  numărul de muchii din  $M^*$  din aceeași componentă conexă a lui  $G'$ , obținem că  $m_M(C) = m_{M^*}(C)$ . Astfel,

$$\begin{aligned} |M \setminus M^*| &= \sum_{C \text{ comp. a lui } G'} m_M(C) = \\ &= \sum_{C \text{ comp. a lui } G'} m_{M^*}(C) = |M^* \setminus M| \end{aligned}$$

## Cuplaje de cardinal maxim - Teorema lui Berge

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

Deci  $|M| = |M^*|$ .  $\square$

Obținem o strategie pentru a determina un cuplaj de cardinal maxim:

```
fie  $M$  un cuplaj în  $G$  (e. g.,  $M = \emptyset$ );  
while ( $\exists P$  drum de creștere relativ la  $M$ ) do  
     $M \leftarrow M \Delta P$ ;  
end while
```

La fiecare iterație **while** cardinalul lui  $M$  crește cu 1, astfel, în cel mult  $n/2$  iterații obținem un cuplaj fără drumuri de creștere, adică de cardinal maxim. Neajunsul acestui algoritm este următorul: condiția din **while** trebuie implementată în timp polinomial. Acest lucru a fost făcut pentru prima oară de către **Edmonds (1965)**.

Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

## Lema 1

Fie  $M, N \in \mathcal{M}_G$ ,  $|M| = r$ ,  $|N| = s$  și  $s > r$ . Atunci în  $M \Delta N$  există cel puțin  $s - r$  drumuri de creștere relativ la  $M$  disjuncte pe noduri.

**Demonstrație.** Fie  $G' = (V, M \Delta N)$  și  $C_i$  ( $i = \overline{1, p}$ ) componentele conexe ale lui  $G'$ . Pentru fiecare  $1 \leq i \leq p$ , notăm cu  $\delta(C_i)$  diferența dintre numărul de muchii ale lui  $N$  din  $C_i$  și numărul de muchii ale lui  $M$  din  $C_i$ :

$$\delta(C_i) = |E(C_i) \cap N| - |E(C_i) \cap M|.$$

Se observă că, deoarece  $M$  și  $N$  sunt cuplaje,  $C_i$  sunt drumuri sau circuite. Astfel  $\delta(C_i) \in \{-1, 0, 1\}$ .  $\delta(C_i) = 1$  dacă și numai dacă  $C_i$  este un drum de creștere relativ la  $M$ .

**Demonstrație (continuare).** Deoarece

$$\sum_{i=1}^p \delta(C_i) = |N \setminus M| - |M \setminus N| = s - r,$$

urmează că există cel puțin  $s - r$  componente conexe ale lui  $G'$  cu  $\delta(C_i) = 1$ , adică, există cel puțin  $s - r$  drumuri de creștere disjuncte pe noduri conținute în  $M \Delta N$ .  $\square$

## Lema 2

Dacă  $\nu(G) = s$  și  $M \in \mathcal{M}_G$  cu  $|M| = r < s$ , atunci există în  $G$  un drum de creștere relativ la  $M$  de lungime cel mult  $2 \lceil r / (s - r) \rceil + 1$ .

**Demonstrație.** Fie  $N \in \mathcal{M}_G$  cu  $|N| = s = \nu(G)$ .

**Demonstrație (continuare).** Din Lema 1, există  $s - r$  drumuri de creștere disjuncte pe muchii (drumurile disjuncte pe noduri sunt și disjuncte pe muchii) conținute în  $M \Delta N$ . Urmează că cel puțin unul dintre ele are cel mult  $\lceil r/(s - r) \rceil$  muchii din  $M$ . Lungimea acestui drum de creștere este cel mult  $2\lceil r/(s - r) \rceil + 1$ .  $\square$

### Definiție

Fie  $M \in \mathcal{M}_G$ . Un **drum de creștere minim** relativ la  $M$  în  $G$  este un drum de creștere de lungime minimă printre toate drumurile de creștere relativ la  $M$  din  $G$ .

### Lema 3

Fie  $M \in \mathcal{M}_G$ ,  $P$  un drum de creștere minim relativ la  $M$ , și  $P'$  un drum de creștere relativ la  $M \Delta P$ . Atunci  $|P'| \geq |P| + 2|P \cap P'|$ .

**Demonstrație.** Fie  $N = (M \Delta P) \Delta P'$ . Atunci  $M \Delta N = P \Delta P'$  și  $|N| = |M| + 2$ . Din Lema 1, există două drumuri de creștere relativ la  $M$  disjuncte pe muchii,  $P_1$  și  $P_2$ , conținute în  $M \Delta N$ . Deoarece  $P$  este un drum de creștere minim relativ la  $M$ , avem  $|P \Delta P'| \geq |P_1| + |P_2| \geq 2|P|$  și, deci,  $|P| + |P'| - 2|P \cap P'| \geq 2|P|$ .  $\square$

Considerăm următorul algoritm:

$M_0 \leftarrow \emptyset; i = 0;$

while ( $\exists$  drumuri de creștere relativ la  $M_i$ ) do

  fie  $P_i$  un drum de creștere minim relativ la  $M_i$ ;

$M_{i+1} \leftarrow M_i \Delta P_i$ ;

$i++$ ;

end while

Fie  $P_0, P_1, \dots, P_{\nu(G)-1}$  secvența de drumuri de creștere minime construite de acest algoritm.

## Lema 4

- a)  $|P_i| \leq |P_{i+1}|$  și  $|P_i| = |P_{i+1}|$  dacă și numai dacă  $P_i$  și  $P_{i+1}$  sunt disjuncte pe noduri,  $\forall 1 \leq i \leq \nu(G) - 2$ .
- b)  $\forall i < j < \nu(G) - 1$ , dacă  $|P_i| = |P_j|$ , atunci  $P_i$  și  $P_j$  sunt disjuncte pe noduri.

- Demonstrație.** a) Luând  $P = P_i$  și  $P' = P_{i+1}$  în Lema 3, obținem  $|P_{i+1}| \geq |P_i| + 2|P_i \cap P_{i+1}| \geq |P_i|$ . Egalitatea are loc dacă și numai dacă  $P_i$  și  $P_{i+1}$  sunt disjuncte pe muchii, de unde rezultă că sunt disjuncte și pe noduri (fiind drumuri alternate).
- b) rezultă aplicând succesiv a)  $\square$

## Teorema 3

(Hopcroft, Karp, 1973) Fie  $G$  un graf cu  $\nu(G) = s$ . Numărul de întregi distincți din secvența  $|P_0|, |P_1|, \dots, |P_{s-1}|$  ( $P_i$  sunt drumuri de creștere minime construite de algoritmul de mai sus) nu este mai mare decât  $2\lfloor\sqrt{s}\rfloor + 2$ .

**Demonstrație.** Fie  $r = \lceil s - \sqrt{s} \rceil$ . Atunci  $|M_r| = r$  și

$$|P_r| \leq 2\lceil r/(s-r) \rceil + 1 = 2\lceil (s - \sqrt{s}) / (s - \lceil s - \sqrt{s} \rceil) \rceil + 1 < 2\lfloor\sqrt{s}\rfloor + 3.$$

Astfel, pentru orice  $i < r$ ,  $|P_i|$  este unul din cele  $\lfloor\sqrt{s}\rfloor + 1$  numere întregi impare nu mai mari decât  $2\lfloor\sqrt{s}\rfloor + 1$ .

În sub-secvența  $|P_r|, \dots, |P_{s-1}|$  există cel mult  $s - r \leq \lfloor\sqrt{s}\rfloor + 1$  întregi distincți. Urmează că în secvența  $|P_0|, |P_1|, \dots, |P_{s-1}|$  nu există mai mult de  $2\lfloor\sqrt{s}\rfloor + 2$  întregi distincți.  $\square$

$$2\lceil [s - \sqrt{s}] / (s - \lceil s - \sqrt{s} \rceil) \rceil + 1 < 2\lfloor \sqrt{s} \rfloor + 3 \Leftrightarrow$$

$$2\lceil [s - \sqrt{s}] / (s - \lceil s - \sqrt{s} \rceil) \rceil < 2\lfloor \sqrt{s} \rfloor + 2 \Leftrightarrow$$

$$\lceil [s - \sqrt{s}] / (s - \lceil s - \sqrt{s} \rceil) \rceil \leq \lfloor \sqrt{s} \rfloor + 1 \Leftrightarrow$$

$$(s - \lfloor \sqrt{s} \rfloor) / \lfloor \sqrt{s} \rfloor \leq \lfloor \sqrt{s} \rfloor + 1 \Leftrightarrow s \leq \lfloor \sqrt{s} \rfloor^2 + 2\lfloor \sqrt{s} \rfloor$$

If  $\lfloor \sqrt{s} \rfloor = k \in \mathbb{N}$ , then  $k \leq \sqrt{s} < k + 1$ , deci  $s = (\sqrt{s})^2 < k^2 + 2k + 1$  adică  $s = (\sqrt{s})^2 \leq k^2 + 2k$  - care este inegalitatea de mai sus.

Dacă algoritmul de mai sus este descompus în faze astfel încât în fiecare fază se determină o familie maximală de drumuri de creștere minime disjuncte pe noduri, atunci - din Lema 4 - lungimea drumurilor de creștere minime din faza următoare nu va descrește (altfel, mulțimea drumurilor de creștere minime construite în faza curentă nu este maximală). Din Teorema 3, obținem că numărul de faze nu este mai mare de  $2\lfloor \sqrt{\nu(G)} \rfloor + 2$ .

Astfel avem următorul algoritm pentru determinarea unui cuplaj de cardinal maxim într-un graf dat  $G$ :

$M \leftarrow \emptyset$ ;

repeat

determină  $\mathcal{P}$  o familie maximală de drumuri de creștere minime relativ la  $M$  disjuncte pe noduri;

for ( $P \in \mathcal{P}$ ) do

$M \leftarrow M \Delta P$ ;

end for

until ( $\mathcal{P} = \emptyset$ )

Complexitatea timp a algoritmului de mai sus este  $\mathcal{O}(\sqrt{n}A)$ , unde  $A$  este complexitatea timp a determinării familiei  $\mathcal{P}$ .

În cazul grafurilor bipartite, **Hopcroft** și **Karp** au arătat că aceasta poate fi obținută în  $\mathcal{O}(n+m)$  și, deci, întreg algoritmul are complexitatea timp  $\mathcal{O}(m\sqrt{n})$ .

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

Acest rezultat a fost extins la grafuri arbitrare de către **Micali** și **Vazirani (1980)** folosind o structură de date elaborată pentru a întreține etichetele asociate nodurilor pentru construcția drumurilor de creștere minime.

Să considerăm cazul grafurilor bipartite:  $G = (S, T; E)$  și  $M \in \mathcal{M}_G$ . Pornind cu una dintre clase, de exemplu  $S$ , considerăm mulțimea extremităților inițiale ale unor drumuri de creștere  $S \cap E(M)$ . Din fiecare astfel de nod pornim, în paralel, construcția unor drumuri alternate într-o manieră **bfs**. Primul drum de creștere obținut oprește construcția, oferind lungimea minimă a unui drum de creștere. Familia  $\mathcal{P}$  este obținută folosind etichetele și listele de adiacență în  $\mathcal{O}(n + m)$ .

Detaliile sunt omise; un exemplu este dat pe slide-ul următor.

Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -  
Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*





## Definiție

Un flux in  $R = (G, s, t, c)$  este o funcție  $x : V \times V \rightarrow \mathbb{R}$  a. î.

- (i)  $0 \leq x_{ij} \leq c_{ij}, \forall (i, j) \in V \times V,$
- (ii)  $\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0, \forall i \in V \setminus \{s, t\}.$

## Remarci

- Dacă  $ij \in E$  atunci  $x_{ij}$  este flux (transportat) de-a lungul arcului  $ij$ .
- Constrângerile (i) cer ca fluxul pe fiecare arc să fie ne-negativ și să nu depășească capacitatea.
- Constrângerile (ii), (legea de conservare), cer ca suma fluxurilor de pe arcele care intră într-un nod  $i$  să fie egală cu suma fluxurilor de pe arcele care ies din  $i$  (fluxul care intră este egal cu fluxul care iese).

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

- Să observăm că, din modul în care am extins funcția  $c$  la  $V \times V$ , constrângerile (i) implică  $x_{ij} = 0$  când  $ij \notin E$ . Astfel, un flux este de fapt o funcție definită pe  $E$ . Preferăm extensia sa pe  $V \times V$  pentru a simplifica notațiile.

Fie  $x$  un flux în  $R = (G, s, t, c)$ . Dacă adunăm toate constrângerile (ii) (pentru  $i \in V \setminus \{s, t\}$ ) obținem

$$\begin{aligned} 0 &= \sum_{i \neq s, t} \left( \sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} \right) = \sum_{i \neq s, t} \sum_{j \neq s, t} x_{ji} - \sum_{i \neq s, t} \sum_{j \neq s, t} x_{ij} + \\ &+ \sum_{i \neq s, t} x_{si} + \sum_{i \neq s, t} x_{ti} - \sum_{i \neq s, t} x_{is} - \sum_{i \neq s, t} x_{it} = \end{aligned}$$

Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

$$= \left( \sum_{i \in V} x_{si} - \sum_{i \in V} x_{is} \right) - \left( \sum_{i \in V} x_{it} - \sum_{i \in V} x_{ti} \right),$$

## Definiție

Valoarea fluxului  $x$  în  $R = (G, s, t, c)$  este

$$v(x) = \sum_{i \in V} x_{it} - \sum_{i \in V} x_{ti}.$$

În cuvinte,  $v(x)$  este **fluxul net care ajunge în destinația** rețelei sau, după cum am demonstrat mai sus, **fluxul net care părăsește sursa** rețelei.

Să observăm că în orice rețea  $R = (G, s, t, c)$  există un flux:  $x^0$ , **fluxul nul**, cu  $x_{ij}^0 = 0, \forall ij \in E$  și  $v(x^0) = 0$ .

Problema fluxului maxim: Dată  $R = (G, s, t, c)$  o rețea, să se determine un flux de valoare maximă.

Problema fluxului maxim poate fi văzută ca o problemă LP:

$$\begin{aligned} \max \quad & v \\ & \sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0, \forall i \neq s, t \\ & \sum_{j \in V} x_{js} - \sum_{j \in V} x_{sj} = -v \\ & \sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj} = v \\ & 0 \leq x_{ij} \leq c_{ij}, \forall ij \in E \end{aligned}$$

Cu toate acestea, vom considera a abordare combinatorială directă care este importantă când există constrângeri de integralitate a unor variabile (de exemplu fluxul de pe arce).

## Exerciții pentru seminarul 8

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

**Exercițiul 1.** Fie  $X$  o mulțime finită,  $X_1, \dots, X_n \subseteq X$ , și  $d_1, d_2, \dots, d_n \in \mathbb{N}$ . Arătați că există  $n$  submulțimi disjuncte  $Y_i \subseteq X_i$ ,  $|Y_i| = d_i$ ,  $\forall i = \overline{1, n}$  dacă și numai dacă

$$\left| \bigcup_{i \in I} X_i \right| \geq \sum_{i \in I} d_i,$$

pentru orice  $I \subseteq \{1, \dots, n\}$ .

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

**Exercițiul 2.** Orice graf  $p$ -regulat bipartit admite cuplaj perfect ( $p \geq 1$ ).

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

**Exercițiul 3.** Fie  $G = (S, T; E)$  un graf bipartit. Folosind teorema lui Hall demonstrați că, pentru fiecare  $0 \leq k \leq |S|$ ,  $G$  are un cuplaj de cardinal cel puțin  $|S| - k$  dacă și numai dacă  $|N_G(A)| \geq |A| - k$ ,  $\forall A \subseteq S$ .

## Exerciții pentru seminarul 8

**Exercițiul 4.** Utilizând teorema lui Tutte arătați că un graf 2-muchie conex și 3-regulat admite cuplaj perfect.

**Exercițiul 5.** Fie  $G = (V, E)$  un graf. O submulțime  $A \subseteq V$  este numită *m-independentă* dacă  $G$  are un cuplaj  $M$  care saturează toate nodurile din  $A$ . Arătați că, pentru orice două mulțimi *m-independente*  $A$  și  $B$  cu  $|A| < |B|$ , se poate determina un nod  $b \in B \setminus A$  astfel încât  $A \cup \{b\}$  este de asemenea *m-independentă* ( $\Rightarrow$  toate mulțimile maximal *m-independente* au același cardinal).

**Exercițiul 6.** Fie  $T = (V, E)$  un arbore cu rădăcină; îi notăm cu  $r$  rădăcina și cu  $parent(v)$  strămoșul direct al oricărui nod  $v \neq r$ . Un cuplaj  $M$  al lui  $T$  este numit *propriu* dacă orice nod nesaturat de  $M$ ,  $v \neq r$ , are un frate  $w$  astfel încât  $w \in M$ .

- Arătați că orice cuplaj *propriu* este un cuplaj de cardinal maxim.
- Găsiți în  $\mathcal{O}(n)$  un cuplaj *propriu* pentru un arbore de ordin  $n$ .

## Exerciții pentru seminarul 8

**Exercițiul 7.** Fie  $G = (V, E)$  un graf  $p$ -regulat bipartit ( $p \geq 1$ ). Considerăm următorul algoritm:

```
for ( $e \in E$ ) do
     $a(e) \leftarrow 1$ ;
end for
 $E^+ \leftarrow \{e \in E : a(e) > 0\}$ ;
while ( $G^+ = (V, E^+)$  conține un circuit  $C$ ) do
    fie  $C = M_1 \cup M_2$ , unde  $M_1$  și  $M_2$  sunt cuplaje a. î.  $a(M_1) \geq a(M_2)$ ; //
    pentru orice  $F \subseteq E$ ,  $a(F) = \sum_{e \in F} a(e)$ ;
    for ( $e \in E(C)$ ) do
        if ( $e \in M_1$ ) then
             $a(e) ++$ ;
        else
             $a(e) --$ ;
        end if
    end for
     $E^+ \leftarrow \{e \in E : a(e) > 0\}$ ;
end while
return  $E^+$ ;
```

**Exercițiul 7. (cont.)** Fie  $f(E^+) = \sum a^2(e)$ . Arătați că

- (a) după fiecare iterație **while**  $f(E^+)$  este un număr întreg care crește cu cel puțin  $|C|$  față de valoarea anterioară;
- (b) după fiecare iterație **while**  $\sum_{uv \in E^+} a(uv) = p, \forall u \in V$ ;
- (c) cât timp există muchii  $e$  cu  $0 < a(e) < p$ , algoritmul continuă; la final  $a(e) = p, \forall e \in E^+$  și în  $E^+$  se găsesc muchiile unui cuplaj perfect al lui  $G$ ;
- (d) numărul de iterații **while** este finit, la final  $f(E^+) = np^2/2 = pm$ , iar suma lungimilor tuturor circuitelor procesate este cel mult  $pm$ ;
- (e) toate circuitele pot fi găsite în complexitatea timp  $\mathcal{O}(\sum_{C \text{ cycle}} |C|)$  folosind parcurgeri **dfs**;
- (f) complexitatea timp a algoritmului în ansamblu este  $\mathcal{O}(pm)$ .

## Exerciții pentru seminarul 8

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

**Exercițiul 8.** Fie  $G = (S, T; E)$  un graf bipartit nenul. Arătați că următoarele afirmații sunt echivalente:

- (i)  $G - \{x, y\}$  are un cuplaj perfect,  $\forall x \in S, \forall y \in T$ .
- (ii)  $G$  este conex și orice muchie a lui  $G$  aparține unui cuplaj perfect.
- (iii)  $|S| = |T|$  și  $\emptyset \neq A \subsetneq S, |N_G(A)| > |A|$ .

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

**Exercițiul 9.** Fie  $G = (V, E)$  un graf conex care are un cuplaj perfect. Descrieți (și demonstrați corectitudinea) unui algoritm de complexitate timp  $\mathcal{O}(|V| + |E|)$  care să construiască un arbore parțial  $T$  al lui  $G$  astfel ca  $V(T)$  să admită o bipartiție în două mulțimi stabile de cardinal maxim ale lui  $T$ .

Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

