Algoritmica Grafurilor - Cursul 9

6 decembrie 2019

Definiţie

Un **preflux** în $R = (G, s, t, c)$ este o funcţie $x : E(G) \to \mathbb{R}$ astfel încât

(i) $0 \leqslant x_{ij} \leqslant c_{ij}, \forall ij \in E$;

(ii) $\forall i \neq s,\ e_i = \sum_{ji \in E} x_{ji} - \sum_{ij \in E} x_{ij} \geqslant 0.$

e_i (pentru $i \in V \setminus \{s, t\}$) este numit **excesul** din nodul i. Dacă $i \in V \setminus \{s, t\}$ şi $e_i > 0$, atunci i este un **nod activ**. Dacă $ij \in E$, x_{ij} va fi numit **fluxul de pe arcul** ij.

Dacă în R nu există noduri active, atunci prefluxul x este un flux cu $v(x) = e_t$.

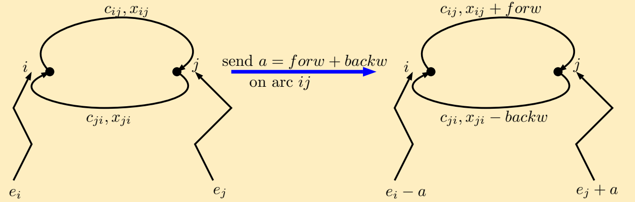
Ideea algoritmilor de tip preflux: un preflux iniţial în R este transformat prin modificarea fluxurilor de pe arce într-un flux cu proprietatea că **nu există drumuri de creştere în R relativ la el**.

G este reprezentat folosind liste de adiacenţă. Vom presupune că dacă $ij \in E$, atunci $ji \in E$ de asemeni (altfel, adăugăm arcul ji de capacitate 0). Astfel, G devine digraf simetric.

Dacă x este un preflux în R şi $ij \in E$, atunci **capacitatea reziduală** a lui ij este

$$r_{ij} = c_{ij} - x_{ij} + x_{ji},$$

(reprezentând fluxul suplimentar care poate fi trimis de la nodul i la nodul j folosind arcele ij şi ji).



În cele de urmează, **a trimite** flux de la i la j înseamnă creşterea fluxului pe arcul ij sau scăderea fluxului pe arcul ji.

Un **A-drum** în R relativ la prefluxul x, este un drum în G având toate arcele de capacitate reziduală strict pozitivă.

O **funcţie distanţă** în R relativ la prefluxul x este o funcţie $d : V \to \mathbb{Z}_+$ a. î.

(D_1) $d(t) = 0$,

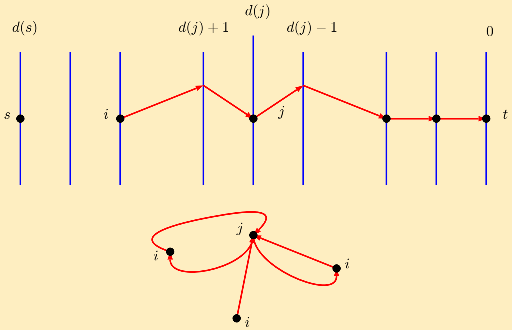
(D_2) $\forall ij \in E, r_{ij} > 0 \Rightarrow d(i) \leqslant d(j) + 1$.

Remarci

- Dacă P este un A-drum relativ la prefluxul x în R de la i la t, atunci $d(i) \leqslant \mathit{length}(P)$ (arcele lui P au capacităţi reziduale pozitive şi apoi se aplică (D_2) în mod repetat). Urmează că (τ_i este lungimea minimă a unui A-drum de la i la t): $d(i) \leqslant \tau_i$.

Remarci

- Ca de obicei, notăm cu $A(i)$ lista de adiacenţă a nodului i.



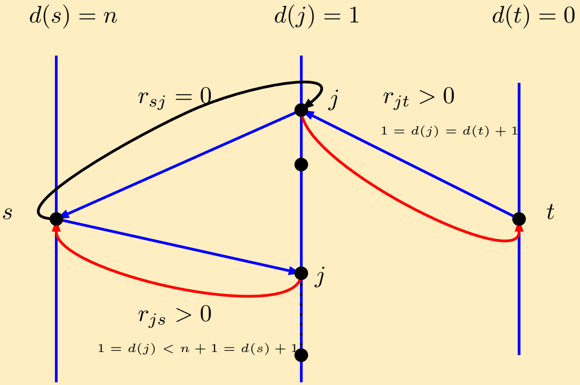
Definiţie

Fie x un preflux în R şi d o funcţie distanţă relativ la x. Un arc $ij \in E$ este numit **admisibil** *dacă* $r_{ij} > 0$ şi $d(i) = d(j) + 1$.

Dacă R este o reţea, considerăm următoarea procedură de iniţializare, care construieşte în $\mathcal{O}(m)$ un preflux x şi o funcţie distanţă d relativ la x.

procedure *initialization*();
for $(ij \in E)$ **do**
    if $(i = s)$ **then**
        $x_{sj} \leftarrow c_{sj}$
    else
        $x_{ij} \leftarrow 0$;
$d[s] \leftarrow n$; $d[t] \leftarrow 0$;
for $(i \in V - \backslash\{s, t\})$ **do**
    $d[i] \leftarrow 1$;

Se observă că după execuţia acestei proceduri, avem $r_{sj} = 0, \forall sj \in A(s)$.
Astfel condiţia (D_2) nu este not afectată de $d(s) = n$.
Pentru toate arcele ij, (D_2) este îndeplinită:



Alegerea $d(s) = n$ are semnificaţia: nu există A-drumuri de la s la t relativ la x (altfel, dacă P este un astfel de drum, lungimea lui trebuie să fie cel puţin $d(s) = n$, ceea ce este imposibil).

Dacă acesta va fi un invariant al algoritmilor de tip preflux, atunci când x va deveni un flux, va fi un flux maxim.

Considerăm următoarele două proceduri:

procedure $push(i)$; // i este un nod diferit de s, t
alege un arc admisibil $ij \in A(i)$;
"trimite" $\delta = \min\{e_i, r_{ij}\}$ (unităţi de flux) de la i la j;

Dacă $\delta = r_{ij}$ atunci avem o **push/pompare saturată**, altfel avem o **pompare nesaturată**.

procedure $relabel(i)$; // i este un nod diferit de s, t
$d[i] \leftarrow \min\{d[j] + 1 \ : \ ij \in A(i) \text{ şi } r_{ij} > 0\}$;

initialization;
while ($\exists$ noduri active în R) **do**
  alege un nod activ i;
  if ($\exists$ arce admisibile în $A(i)$) **then**
    $push(i)$;
  else
    $relabel(i)$;

Lema 1

"d este funcţie distanţă relativ la prefluxul x" este un invariant al algoritmului de mai sus. La fiecare apel $relabel(i)$, $d(i)$ creşte.

Demonstraţie. Am demonstrat deja că procedura de iniţializare construieşte un preflux x şi o funcţie distanţă d relativ la x. Arătăm că dacă perechea (d, x) satisface (D_1) şi (D_2) înaintea unei iteraţii while, atunci după această iteraţie cele două condiţii sunt de asemeni satisfăcute.

Avem două cazuri, în funcţie de care dintre procedurile push sau relabel este executată în iteraţia while curentă:

este apelată $push(i)$**:** singura pereche care poate viola (D_2) este $d(i)$ şi $d(j)$. Deoarece ij este admisibil, la apelul $push(i)$ avem $d(i) = d(j)+1$. După execuţia $push(i)$, arcul ji poate avea $r_{ji} > 0$ (fără a fi astfel înaintea apelului), dar condiţia $d(j) \leqslant d(i) + 1$ este evident satisfăcută.

este apelată $relabel(i)$**:** actualizarea lui $d(i)$ este astfel încât (D_2) este satisfăcută pentru fiecare arc ij cu $r_{ij} > 0$. Deoarece $relabel(i)$ este apelată când $d(i) < d(j) + 1$, $\forall ij$ cu $r_{ij} > 0$, urmează că, după apel, $d(i)$ creşte (cu cel puţin 1). □

Pentru a arăta că algoritmul se opreşte, este necesar să arătăm că (în timpul execuţiei) dacă un nod i este activ atunci în lista lui de adiacenţă, $A(i)$, există cel puţin un arc ij cu $r_{ij} > 0$. Aceasta demonstrează lema următoare.

Lema 2

Dacă i_0 este un x-nod activ în R, atunci există un $i_0 s$ A-drum relativ la x.

Demonstraţie. Dacă x este un preflux în R, atunci x poate fi descompus în $x = x^1 + x^2 + \cdots + x^p$, unde fiecare x^k are proprietatea că mulţimea $A^k = \{ij \,:\, ij \in E, x^k_{ij} \neq 0\}$ este

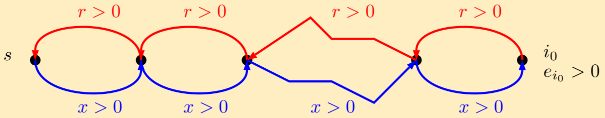
(a) mulţimea arcelor unui drum de la s la t, sau

(b) mulţimea de arce ale unui drum de la s la un nod activ, sau

(c) mulţimea arcelor unui circuit.

Mai mult, în cazurile (a) şi (c), x^k este un flux. (Demonstraţia este algoritmică: construim mai întâi mulţimile de tipul (a), apoi pe cele de tipul (c) şi (b); la fiecare pas, căutăm inversul unui drum de tip (a) sau (c) (sau (b)); prefluxul obţinut este scăzut din cel curent; deoarece excesele nodurilor sunt nenegative, construcţia poate fi realizată, ori de câte ori prefluxul curent este nenul; construcţia este finită, deoarece numărul de arce pe care fluxul curent este 0, creşte la fiecare pas.)

Deoarece i_0 este un nod activ în R relativ la x, cazul (b) va aparea pentru nodul i_0 (cazurile (a) şi (c) nu afectează excesul din nodul i_0). Arcele inverse de pe acest drum au capacitatea reziduală pozitivă (vezi imaginea de mai jos), astfel ele formează A-drumul cerut. $\square$



Corolarul 1

$\forall i \in V,\ d(i) < 2n.$

Demonstraţie. Într-adevăr, dacă i nu a fost reetichetat, atunci $d(i) = 1 < 2n$. Altfel, înaintea apelului *relabel*(i), i este un nod activ, astfel din Lema 2, există un *is*-drum P cu *length*$(P) \leqslant n - 1$. Din (D_2), urmează că, după reetichetare , $d(i) \leqslant d(s) + n - 1 = 2n - 1$ ($d(s) = n$ nu este modificat vreodată). $\square$

Corolarul 2

Numărul total de apeluri ale procedurii **relabel** nu este mai mare decât $2n^2$.

Demonstraţie. Într-adevăr, există $n-2$ noduri care pot fi reetichetate. Fiecare dintre ele poate fi reetichetat de cel mult $2n-1$ ori (din Lema 1, Corolarul de mai sus şi distanţa iniţială d). $\square$

Corolarul 3

Numărul total de pompări saturate nu este mai mare decât nm.

Demonstraţie. Într-adevăr, când un arc ij devine saturat, avem $d(i) = d(j)+1$. După aceea, algoritmul nu va mai trimite flux pe acest arc până când nu va trimite flux pe arcul ji, când vom avea $d'(j) = d'(i) + 1 \geqslant d(i) + 1 = d(j) + 2$.

Demonstraţie (continuare). Astfel o modificare a acestui flux pe arcul ij nu va apărea până când $d(j)$ nu va creşte cu 2. Urmează că un arc nu poate deveni saturat de mai mult de n ori şi (deoarece numărul total de arce este m), nu există mai mult de nm pompări saturate. □

Lema 3

(Goldberg şi Tarjan, 1986). Numărul total de pompări nesaturate nu este mai mare decât $2n^2m$.

Demonstraţie. Omisă.

Lema 4

Algoritmul de tip preflux returnează un flux de valoare maximă în R.

Demonstraţie. Din Lemele 1 şi 3 şi din Corolarul 3 al Lemei 2, algoritmul se opreşte în cel mult $2n^2m$ iteraţii while. Deoarece $d(s) = n$ nu se modifică niciodată, urmează că nu există vreun drum de creştere relativ la fluxul x obţinut, astfel x este de valoare maximă: dacă P ar fi un drum de creştere (în digraful suport al lui G), atunci înlocuind de-a lungul lui P fiecare arc invers cu simetricul său se obţine un A-drum de la s la t, deci $n = d(s) \leqslant d(t) + n - 1 = n - 1$. □

În locul demonstraţiei Lemei 3, vom prezenta **Algoritmul lui Ahuja & Orlin (1988)** care utilizează o **metodă de scalare - scaling method** pentru a reduce numărul de pompări nesaturate de la $\mathcal{O}(n^2m)$ **(Goldberg & Tarjan algoritm)** la $\mathcal{O}(n^2 \log U)$.

Să presupunem că $c_{ij} \in \mathbb{N}$ şi $U = \max\limits_{ij \in E}(1 + c_{ij})$. Fie $K = \lceil \log_2 U \rceil$.

Ideea algoritmului: Avem $K+1$ etape. În fiecare etapă p, cu p luând succesiv valorile $K, K-1, \ldots, 1, 0$, următoarele două condiţii sunt îndeplinite:

(a) la începutul etapei p, $e_i \leqslant 2^p$, $\forall i \in V \setminus \{s, t\}$.

(b) în timpul etapei p, procedurile **push-relabel** sunt folosite pentru a elimina nodurile active, i, cu $e_i \in \{2^{p-1} + 1, \ldots 2^p\}$.

Din definiţia lui K, în prima etapă ($p = K$), condiţia (a) este îndeplinită, şi dacă condiţia (b) va fi menţinută de-a lungul execuţiei algoritmului, după $K+1$ etape, dacă se menţine integralitatea exceselor de-a lungul algoritmului, atunci urmează că, după etapa $K+1$, excesul fiecărui nod $i \in V \setminus \{s, t\}$ este 0, deci avem un flux de valoare maximă.

Pentru a menţine (b) de-a lungul execuţiei algoritmului, schema generală a unui algoritm de tip preflux este adaptată după cum urmează:

- fiecare etapă p începe prin construirea listei $L(p)$ a tuturor nodurilor $i_1, i_2, \ldots, i_{l(p)}$ cu excese $e_i > 2^{p-1}$, ordonate crescător după d (aceasta poate fi făcută cu hash-sorting în timpul $O(n)$, deoarece $d(i) \in \{1, 2, \ldots, 2n-1\}$).
- nodul activ selectat pentru **push-relabel** în timpul etapei p va fi **primul nod din** $L(p)$. Urmează că, dacă se face o pompare (push) pe un arc admisibil ij, atunci $e_i > 2^{p-1}$ şi $e_j \leqslant 2^{p-1}$ (deoarece $d(j) = d(i) - 1$ şi i este primul nod din $L(p)$). Dacă δ, fluxul trimis de la i la j de către apelul $push(i)$, este limitat la $\delta = \min(e_i, r_{ij}, 2^p - e_j)$, atunci (deoarece $2^p - e_j \geqslant 2^{p-1}$) urmează că o **pompare nesaturată** trimite cel puţin 2^{p-1} unităţi de flux.

După apelul $push(i)$ excesul din nodul j (singurul pentru care excesul poate creşte) va fi $e_j + \min(e_i, r_{ij}, 2^p - e_j) \leqslant e_j + 2^p - e_j \leqslant 2^p$, deci condiţia (b) rămâne îndeplinită.

- etapa p este încheiată când lista $L(p)$ devine vidă.

Pentru a determina în mod eficient un arc admisibil pentru o pompare (**push**), sau pentru a inspecta toate arcele care părăsesc un nod i în vederea reetichetării (**relabel**), vom organiza listele de adiacenţă $A(i)$ după cum urmează:

- fiecare element din listă conţine: nodul j, x_{ij}, r_{ij}, un pointer către elementul corespunzător lui i din lista de adiacenţă $A(j)$ şi un pointer către următorul element din lista $A(i)$.
- lista are asociat un iterator pentru a-i înlesni parcurgerea.

Toate aceste liste sunt construite în $O(m)$, înaintea apelului procedurii **initialization**.

initialization; $K \leftarrow \lceil \log_2 U \rceil$; $\Delta \leftarrow 2^{K+1}$;
for $(p = \overline{K, 0})$ **do**
  construieşte $L(p)$; $\Delta \leftarrow \Delta/2$;
  while $(L(p) \neq \varnothing)$ **do**
    fie i primul nod din $L(p)$;
    caută în $A(i)$ primul arc admisibil sau până se ajunge la sfârşit;
    if (ij este arcul admisibil găsit) **then**
      $\delta \leftarrow \min(e_i, r_{ij}, \Delta - e_j)$;
      $e_i \leftarrow e_i - \delta$; $e_j \leftarrow e_j + \delta$;
      "trimite" δ unităţi de flux de la i la j;
      if $(e_i \leqslant \Delta/2)$ **then**
        şterge i din $L(p)$;
      if $(e_j > \Delta/2)$ **then**
        adaugă j la începutul listei $L(p)$;
    else
      calculează $d[i] = \min\{d[j] + 1 : ij \in A(i)$ şi $r_{ij} > 0\}$
      repoziţionează i în $L(p)$;
      setează poziţia curentă (a pointerului) la începutul listei $A(i)$;

Complexitatea timp a algoritmului este dominată de **pompările nesaturate** (ceea ce rămâne este de complexitate $\mathcal{O}(nm)$).

Lema 5

Numărul **pompărilor nesaturate** este cel mult $8n^2$ în fiecare etapă a scalării, astfel numărul total este $\mathcal{O}(n^2 \log U)$.

Demonstraţie. Fie

$$F(p) = \sum_{i \in V, i \neq s,t} \frac{e_i \cdot d(i)}{2^p}.$$

La începutul etapei p, $F(p) < \sum_{i \in V} \frac{2^p \cdot (2n)}{2^p} = 2n^2.$

Dacă, în etapa p, se execută *relabel*(i), atunci nu există arce admisibile ij, şi $d(i)$ creşte cu $h \geqslant 1$ unităţi. $F(p)$ va creşte cu cel mult h. Deoarece, $\forall i$, $d(i) < 2n$ urmează că $F(p)$ va creşte (până la finalul etapei p) cel mult până la $4n^2$.

Dacă, în etapa p, se execută *push*(i), atunci aceasta trimite $\delta \geqslant 2^{p-1}$ pe arcul admisibil ij cu $r_{ij} > 0$ şi $d(i) = d(j) + 1$. Astfel, după push, $F(p)$ va avea valoarea $F'(p) = F(p) - \dfrac{\delta \cdot d(i)}{2^p} + \dfrac{\delta \cdot d(j)}{2^p} = F(p) - \dfrac{\delta}{2^p} \leqslant F(p) - \dfrac{2^{p-1}}{2^p} = F(p) - 1/2$.

Această descreştere nu poate apărea de mai mult de $8n^2$ ori (deoarece $F(p)$ poate creşte cel mult până la $4n^2$ şi $F(p)$ este nenegativ). Evident, numărul pompărilor nesaturate este dominat de acest număr de descreşteri ale lui $F(p)$.

În concluzie:

Teoremă

(Ahuja-Orlin, 1988) Algoritm de tip preflux cu scalarea exceselor are complexitatea timp $\mathcal{O}(nm + n^2 \log U)$.

A. Determinarea unui cuplaj de cardinal maxim şi a unei mulţimi stabile de cardinal maxim într-un un graf bipartit.

Fie $G = (V_1, V_2; E)$ un graf bipartit cu n noduri şi m muchii.

Considerăm reţeaua $R = (G_1, s, t, c)$, unde

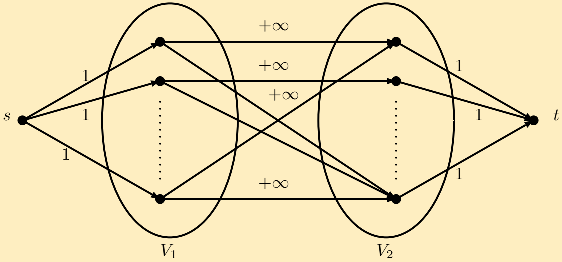
- $V(G_1) = \{s, t,\} \cup V_1 \cup V_2$;
- $E(G_1) = E_1 \cup E_2 \cup E_3$, cu

$$E_1 = \{sv_1 \ : \ v_1 \in V_1\}, E_2 = \{v_2 t \ : \ v_2 \in V_2\},$$

$$E_3 = \{v_1 v_2 \ : \ v_1 \in V_1, v_2 \in V_2\},$$

- $c : E(G_1) \to \mathbb{N}$ definită prin

$$c(e) = \begin{cases} 1, & \text{dacă } e \in E_1 \cup E_2 \\ \infty, & \text{dacă } e \in E_3 \end{cases}$$



Dacă $x = (x_{ij})$ este un flux întreg în R, atunci mulţimea $\{ij \ : \ i \in V_1, j \in V_2$ şi $x_{ij} = 1\}$ coresponde unui cuplaj M^x în graful bipartit G, cu $|M^x| = v(x)$.

Reciproc, orice cuplaj $M \in \mathcal{M}_G$ corespunde unei mulţimi de arce neadiacente din G_1; dacă pe fiecare astfel de arc ij ($i \in V_1$, $j \in V_2$) considerăm $x^M_{ij} = 1$ şi $x^M_{si} = x^M_{jt} = 1$, şi adăugăm $x^M(e) = 0$ pe restul arcelor, atunci fluxul întreg x^M satisface $v(x^M) = |M|$.

Astfel, dacă rezolvăm problema fluxului maxim în R (începând cu fluxul nul), atunci obţinem în $\mathcal{O}(nm + n^2 \log n)$ un cuplaj de cardinal maxim în G.

Fie (S, T) secţiunea de capacitate minimă (obţinută în $\mathcal{O}(m)$ dintr-un flux maxim determinat). Din Teorema de flux maxim- secţiune minimă, $c(S, T) = \nu(G)$.

Deoarece $\nu(G) < \infty$, luând $S_i = S \cap V_i$ şi $T_i = T \cap V_i$ $(i = \overline{1, 2})$, avem $|T_1| + |S_2| = \nu(G)$ şi $X = S_1 \cup T_2$ este o mulţime stabilă în G (pentru a avea $c(S, T) < \infty$). Mai mult, $|X| = |V_1 \setminus T_1| + |V_2 \setminus S_2| = n - \nu(G)$. Urmează că X este o mulţime stabilă de cardinal maxim, deoarece $n - \nu(G) = \alpha(G)$) (din teorema lui König).

B. Recunoaşterea secvenţelor digrafice

Considerăm următoarea problemă:

Date $(d_i^+)_{i=\overline{1,n}}$ şi $(d_i^-)_{i=\overline{1,n}}$, există un digraf $G = (\{1, \ldots, n\}, E)$ astfel încât $d_G^+(i) = d_i^+$ şi $d_G^-(i) = d_i^-$, $\forall i = \overline{1,n}$?

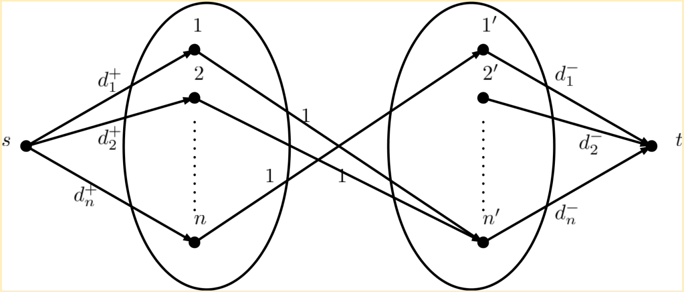
Condiţii evident necesare pentru un răspuns afirmativ sunt:

$$d_i^+ \in \mathbb{N}, 0 \leqslant d_i^+ \leqslant n-1 \text{ şi } d_i^- \in \mathbb{N}, 0 \leqslant d_i^- \leqslant n-1, \forall i = \overline{1,n};$$

$$\sum_{i=1}^{n} d_i^+ = \sum_{i=1}^{n} d_i^- = m \text{ (unde } m = |E|).$$

În aceste ipoteze considerăm reţeaua bipartită $R = (G_1, s, t, c)$.

G_1 este obţinut din graful complet bipartit $K_{n,n}$ cu bipartiţia $(\{1,2,\ldots,n\},\{1',2',\ldots,n'\})$, prin ştergerea mulţimii de muchii $\{11',22',\ldots,nn'\}$ şi prin orientarea fiecărei muchii ij' $(\forall i \neq j \in \{1,2,\ldots,n\})$ de la i la j', şi prin adăugarea a două noi noduri s, t, şi a tuturor arcelor si, $i \in \{1,2,\ldots,n\}$ şi $j't$, $j \in \{1,2,\ldots,n\}$.
Funcţia de capacitate: $c(si) = d_i^+$, $c(j't) = d_j^-$, $c(ij') = 1$, $\forall i,j = \overline{1,n}$.



Dacă în R există un flux întreg, x, de valoare maximă m, atunci din fiecare nod i vor pleca exact d^+_{ij} arce, ij', pe care $x_{ij'} = 1$, şi in fiecare nod j' vor intra exact d^-_i arce, ij', pe care $x_{ij'} = 1$.

Digraful dorit, G, este construit luând $V(G) = \{1, 2, \ldots, n\}$ şi punând $ij \in E(G)$ dacă şi numai dacă $x_{ij'} = 1$.

Reciproc, dacă G există, atunci inversând construcţia anterioară vom obţine un flux întreg în R de valoare m (deci de valoare maximă).

Urmează că, recunoaşterea secvenţelor digrafice (şi construirea digrafului, în cazul unui răspuns afirmativ) poate fi făcută în $\mathcal{O}(nm + n^2 \log n) = \mathcal{O}(n^3)$.

C. Determinarea numărului de conexiune pe muchii a unui graf

Fie $G = (V, E)$ un graf. Pentru $s, t \in V$, $s \neq t$, notăm

- $p_e(s, t) =$ numărul maxim de drumuri disjuncte pe muchii de la s la t în G,
- $c_e(s, t) =$ cardinalul minim al unei mulţimi de muchii astfel încât nu există drum de la s la t în graful obţinut prin ştergerea ei din G.

Teoremă

$p_e(s, t) = c_e(s, t)$.

Demonstraţie. Fie G_1 digraful obţinut din G prin înlocuirea fiecărei muchii printr-o pereche de arce simetrice. Fie $c : E(G_1) \to \mathbb{N}$ o funcţie de capacitate definită prin $c(e) = 1, \forall e \in E(G_1)$.

Demonstraţie (continuare). Fie x^0 un flux întreg de valoare maximă în $R = (G_1, s, t, c)$. Dacă există un circuit C în G_1 cu $x^0_{ij} = 1$ pe toate arcele lui C, atunci putem pune fluxul 0 pe toate arcele lui C fără a modifica valoarea fluxului x_0. Astfel, putem presupune că fluxul x^0 este aciclic şi atunci x^0 poate fi scris ca o sumă de $v(x^0)$ fluxuri întregi x^k fiecare cu $v(x^k) = 1$.

Fiecare flux x^k corespunde unui drum de la s la t în G_1 (considerând arcele pe care fluxul nu este 0), care este un drum de la s la t şi în graful G.

Urmează că $v(x^0) = p_e(s, t)$, deoarece orice mulţime de drumuri disjuncte pe muchii de la s la t în G generează un flux $0 - 1$ în R de valoare egală cu numărul acestor drumuri.

Proof (continuare). Fie (S, T) o secţiune de capacitate minimă în R; avem $c(S, T) = v(x^0)$, din Teorema flux maxim-secţiune minimă. Pe de altă parte, $c(S, T)$ este numărul de arce cu o extremitate în S şi cealaltă în T (deoarece toate capacităţile arcelor sunt 1). Această mulţime de arce generează în G o mulţime de muchii de acelaşi cardinal astfel încât nu există vreun drum de la s to t în graful obţinut prin ştergerea ei din G.

Astfel am obţinut o mulţime de $c(S, T) = v(x^0) = p_e(s, t)$ muchii în G care deconectează pe s de t prin ştergerea lor din G. Urmează că $c_e(s, t) \leqslant p_e(s, t)$. Deoarece inegalitatea $c_e(s, t) \geqslant p_e(s, t)$ este evidentă, teorema este demonstrată. $\square$

Dacă G este un graf conex, $\lambda(G)$, valoarea maximă alui $p \in \mathbb{N}$ pentru care G este p-muchie-conex, este

$$\min_{s,t \in V(G), s \neq t} c_e(s, t) \ (*)$$

Urmează că, pentru a calcula $\lambda(G)$, este necesar să rezolvăm $n(n-1)/2$ probleme de flux maxim descrise mai sus. Acest număr poate fi redus dacăe observăm că, pentru o pereche fixată (s, t), dacă (S, T) este o secţiune de capacitate minimă, atunci

$$\forall v \in S \text{ şi } \forall w \in T \ c_e(v, w) \leqslant c(S, T) \ (**)$$

În particular, dacă (s, t) este perechea pentru care minimul din $(*)$ se atinge, avem egalitate în $(**)$.

Dacă fixăm un nod $s_0 \in V$ şi rezolvăm cele $n-1$ probleme de flux maxim luând $t_0 \in V \setminus \{s_0\}$ vom obţine o pereche (s_0, t_0) cu $c(s_0, t_0) = \lambda(G)$ (t_0 nu va fi în aceeaşi clasă a bipartiţiei cu s_0 în (S, T)).

Concluzie: $\lambda(G)$ poate fi găsit în $\mathcal{O}(n \cdot (nm + n^2c)) = \mathcal{O}(n^2m)$.

D. Determinarea numărului de conexiune pe noduri a unui graf.

Fie $G = (V, E)$ un graf. C $s, t \in V$, $s \neq t$, dacă notăm

- $p(s, t)$ = numărul maxim de drumuri intern disjuncte (pe noduri) de la s la t în G,
- $c(s, t)$ = cardinalul minim al unei mulţimist-separatoare de noduri din G,

atunci, Din teorema lui Menger, avem

$$p(s, t) = c(s, t)(* * *)$$

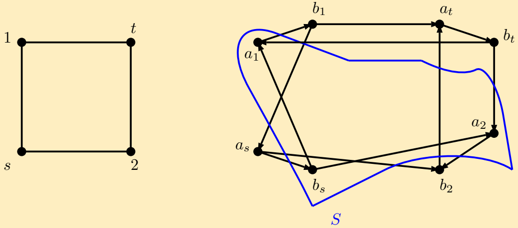
În plus, numărul de conexiune pe noduri, $k(G)$, al grafului G (valoarea maximă a lui $p \in \mathbb{N}$ pentru care G este p-conex) este

$$k(G) = \begin{cases} n - 1, & \text{dacă } G = K_n \\ \min\limits_{s,t \in V(G), s \neq t} c(s, t), & \text{dacă } G \neq K_n \end{cases} \quad (* * **)$$

Arătăm că egalitatea $(***)$ provine din Teorema flux maxim-secţiune minimă, aplicată unei reţele corespunzătoare.
Fie $G_1 = (V(G_1), E(G_1))$ digraful construit pornind de la G astfel:

- $\forall v \in V$, adăugăm $a_v, b_v \in V(G_1)$ şi $a_v b_v \in E(G_1)$;
- $\forall vw \in E$, adăugăm $b_v a_w, b_w a_v \in E(G_1)$.

Exemplu



Mai definim $c : E(G_1) \to \mathbb{N}$ prin

$$c(e) = \begin{cases} 1, & \text{dacă } e = a_v b_v \\ \infty, & \text{altfel} \end{cases}$$

Considerăm reţeaua $R = (G_1, b_s, a_t, c)$.

Fie x^0 un flux întreg în R de valoare maximă. În nodurile $b_v (v \in V)$ intră exact câte un arc de capacitate 1 şi din nodurile $a_v (v \in V)$ pleacă exact câte un arc de capacitate 1.

Urmează că (din legea de conservare a fluxului) că $x^0_{ij} \in \{0, 1\}$, $\forall ij \in E(G_1)$. Astfel x^0 poate fi descompus în $v(x^0)$ fluxuri x^k, fiecare de valoare 1, cu proprietatea că arcele pe care x_k este nenul corespond la $v(x^0)$ drumuri intern disjuncte din G.

Pe de altă parte, din orice mulţime de p st-drumuri intern disjuncte în G, putem construi p $b_s a_t$-drumuri intern disjuncte în G_1, pe care se poate transporta o singură unitate de flux. Urmează că $v(x^0) = p(s, t)$.

Fie (S, T) o secţiune de capacitate minimă în R astfel încât $v(x^0) = c(S, T)$. Deoarece $v(x^0) < \infty$, urmează că $\forall i \in S$, $\forall j \in T$ cu $ij \in E(G_1)$; avem $c(ij) < \infty$, deci $c(ij) = 1$, adică $\exists u \in V$ astfel încât $i = a_u$ şi $j = b_u$.

Astfel, secţiunea (S, T) corespunde unei mulţimi de noduri $A_0 \subseteq V$ astfel încât $c(S, T) = |A_0|$ şi A_0 este o mulţime *st*-separatoare.

Pe de altă parte, pentru orice mulţime *st*-separatoare, A, avem $|A| \geqslant p(s, t) = v(x^0)$. Deci

$$c(s, t) = |A_0| = c(S, T) = v(x^0) = p(s, t).$$

Demonstraţia de mai sus arată că, pentru a determina $k(G)$ este suficient să determinăm minimul în $(* * **)$ rezolvând $|E(\overline{G})|$ probleme de flux maxim, unde $\overline{G}$ este complementul lui G.

Avem astfel un algoritm cu complexitatea timp

$$\mathcal{O}\left(\left(\frac{n(n-1)}{2} - m\right)(nm + n^2 \log n)\right).$$

O observaţie simplă ne oferă un algoritm mai eficient. Evident,

$$k(G) \leqslant \min_{v \in V} d_G(v) = \frac{1}{n}\left(n \cdot \min_{v \in V} d_G(v)\right) \leqslant \frac{1}{n} \sum_{v \in V} d_G(v) = \frac{2m}{n}.$$

Dacă A_0 este a mulţime separatoare în G cu $|A_0| = k(G)$, atunci $G \setminus A_0$ nu este conex şi există o partiţie of $V \setminus A_0$ (V', V'') astfel încât $\forall v' \in V'$ şi $\forall v'' \in V''$ avem $p(v', v'') = k(G)$.

Urmează că rezolvând o problemă de flux maxim cu $s_0 \in V'$ şi $t_0 \in V''$ obţinem că $p(s_0, t_0) =$ valoarea fluxului maxim $= k(G)$.

Putem determina o astfel de pereche după cum urmează: fie $l = \left\lceil \frac{2m}{n} \right\rceil + 1$, alegem l noduri arbitrare din $V(G)$, şi pentru fiecare astfel de nod, v, rezolvăm toate problemele de flux maxim $p(v, w)$, cu $vw \notin E$. Numărul acestor probleme este $\mathcal{O}(nl) = \mathcal{O}\left(n\left((\frac{2m}{n} + 1\right)\right) = \mathcal{O}(m)$.

Astfel complexitatea timp a determinării lui $k(G)$ este $\mathcal{O}(m(nm + n^2 \log n))$.

Exerciţiul 1. Arătaţi că, utilizând un algoritm de flux maxim (într-o anumită reţea), se poate găsi, într-o matrice $0-1$, o mulţime de cardinal maxim de elemente egale cu 0, în care oricare două elemente nu se află pe aceeaşi linie sau pe aceeaşi coloană.

Exerciţiul 2. Fie S şi T mulţimi nevide, disjuncte şi finite. Se dă o funcţie $a : S \cup T \to \mathbb{N}$. Să se decidă dacă există un graf bipartit $G = (S, T; E)$ astfel încât $d_G(v) = a(v)$, pentru orice $v \in S \cup T$; dacă răspunsul este afirmativ trebuie returnate muchiile lui G (S şi T sunt clasele bipartiţiei lui G). Arătaţi că această problemă poate fi rezolvată în timp polinomial ca o problemă de flux maxim într-o anumită reţea.

Exerciţiul 3. Fiecare student dintr-o mulţime S de cardinal $n > 0$ subscrie la o submulţime de 4 cursuri opţionale dintr-o mulţime C de cardinal $k > 4$. Descrieţi un algoritm (de complexitate timp polinomială) care să determine (dacă există) o alocare a studenţilor către cursurile opţional din C astfel încât fiecare student să fie asignat la exact 3 cursuri (din cele 4 la care a subscris) şi fiecare curs să aibă asignaţi cel mult $\lceil n/k \rceil$ studenţi.

Exerciţiul 4.

(a) Adevărat sau fals? Într-o reţea $R = (G, s, t, c)$ cu capacităţi distincte există un singur flux maxim. De ce?

(b) Descrieţi şi demonstraţi corectitudinea unui algoritm de complexitate (timp) polinomială care să decidă dacă într-o reţea dată există un singur flux maxim.

Exerciţiul 5. O companie IT are n angajaţi $P_1, P_2, \ldots, P_n$ care trebuie să lucreze la m proiecte $L_1, L_2, \ldots, L_m$. Pentru orice angajat P_i avem o listă $\mathcal{L}_i$ a proiectelor la care poate lucra şi s_i numărul de projecte din $\mathcal{L}_i$ pe care le poate termina într-o săptămână ($s_i \leqslant |\mathcal{L}_i|$). Fiecare proiect va fi alocat unui singur angajat.
Cum se poate determina numărul minim de săptămâni necesare terminării tuturor proiectelor folosind fluxuri în reţele.

Exerciţiul 6. Planul de evacuare de urgenţă al unei clădiri este descris ca un grid $n \times n$; frontierele celulelor sunt rutele de eavcuare către exteriorul clădirii (grid-ului). O instanţă a problemei evacuării conţine dimensiunea n a grid-ului şi m puncte de plecare (colţurile celulelor).

Exerciţiul 6 (continuare). Instanţa primeşte un răspuns afirmativ dacă există m drumuri disjuncte către frontiera grid-ului plecând din cele m puncte de mai sus. Dacă drumurile acestea nu există, atunci instanţa primeşte un răspun negativ.
Determinaţi o reprezentare a acestei problem a evacuării ca o problemă de flux într-o reţea. Descrieţi un algoritm eficient pentru a recunoaşte instanţele pozitive ale problemei (care este complexitatea timp a algoritmului?).

Exerciţiul 7. Fie $G = (V, E)$ un graf cu n noduri $\{v_1, v_2, \ldots, v_n\}$ şi $c : E \to \mathbb{R}_+$ o funcţie de capacitate pe muchiile lui G. O tăietură în G este o bipartiţie (S, T) a lui V. Capacitatea unei tăieturi (S, T) este $c(S, T) = \sum\limits_{e \in E, |e \cap S| = 1} c(e)$.

Exerciţiul 7 (continuare). O *tăietură minimă* în G este o tăietură (S_0, T_0) astfel încât

$$c(S_0, T_0) = \min_{(S,T) \text{ tăietură în } G} c(S, T)$$

(a) Arătaţi că se poate determina o tăietură minimă în timp polinomial rezolvând un număr polinomial de probleme de flux maxim în anumite reţele.

(b) Pentru $G = C_n$ (circuit indus de ordin $n \geqslant 3$) cu toate capacităţile 1, arătaţi că există $\dfrac{n(n-1)}{2}$ tăieturi minime.

Exerciţiul 8. Fie $G = (V; E)$ un digraf şi $w : V \to \mathbb{R}$ astfel ca $w(V) \cap \mathbb{R}_+$ şi $w(V) \cap \mathbb{R}_- \neq \varnothing$. O submulţime $A \subseteq V$ se numeşte *izolată* în G dacă nu există nici un arc care iese din A. *Ponderea* unei submulţimi $A \subseteq V$ este $w(A) = \sum_{v \in A} w(v)$. Descrieţi un algoritm de compexitate polinomială bazat pe un algoritm de flux maxim într-o anumită reţea care să determine o submulţime izolată de pondere maximă în G.

Exerciţiul 9. Fie $G = (S, T; E)$ un graf bipartit. Demonstraţi teorema lui Hall (*există un cuplaj în G care saturează toate nodurile din S dacă şi numai dacă (H)* $\forall A \subseteq S, |N_G(A)| \geqslant |A|$) folosind *teorema flux maxim - secţiune minimă* pe o reţea particulară.