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

Algoritmica grafurilor - Cursul 8

Graph Algorithms * C. Croitoru - Graph Algorithms *

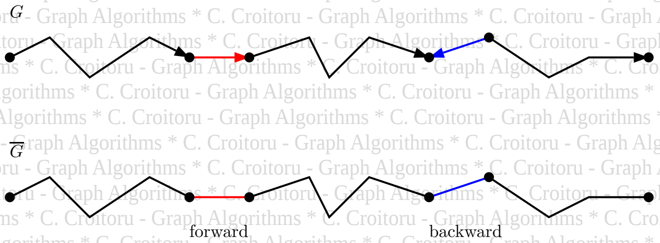
Decembrie 2018

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

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

Definiţie

Fie P un drum în $\widetilde{G}$ – graful suport al digrafului G – şi $e = v_iv_j$ o muchie a lui P, $e \in E(P)$. Dacă e corespunde arcului v_iv_j atunci e este un **arc înainte (forward)** *al lui P, altfel (e corespunde arcului v_jv_i) e este un* **arc înapoi (backward)** *al lui P.*



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

Definiţie

Fie $R = (G, s, t, c)$ *o reţea şi* x *un flux în* R. *Un* **A-drum** *(în* R *relativ la fluxul* x*) este un drum* P *în* $\widetilde{G}$ *astfel încât* $\forall ij \in E(P)$:

- *dacă* ij *este un arc forward, atunci* $x_{ij} < c_{ij}$,
- *dacă* ij *este a arc backward, atunci* $x_{ji} > 0$.

Dacă P *este un* A*-drum în* R *relativ la fluxul* x, *atunci* **capacitatea reziduală** *a arcului* $ij \in E(P)$ *este*

$$r(ij) = \begin{cases} c_{ij} - x_{ij}, & \text{dacă } ij \text{ este un arc forward al lui } P \\ x_{ji}, & \text{dacă } ij \text{ este un arc backward al lui } P \end{cases}$$

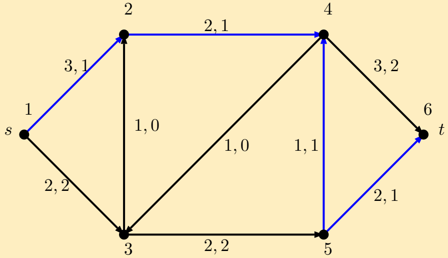
Capacitatea reziduală *a drumului* P *este* $r(P) = \min_{e \in E(P)} r(e)$.

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

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

Exemplu

Considerăm reţeaua de mai jos , unde, pe fiecare arc, prima etichetă este capacitatea iar a doua este fluxul.



$P: 1, 12, 2, 24, 4, 45, 5, 56, 6$ este un A-drum de la s la t cu arcele forward 12 ($x_{12} = 1 < c_{12} = 3$), 24 ($x_{24} = 1 < c_{24} = 2$), 56 ($x_{56} = 1 < c_{56} = 2$), şi arcul backward 45 ($x_{45} = 1 > 0$). Capacitatea rexiduală a lui P: $r(P) = \min\{2, 1, 1, 1\} = 1$.

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

Definiţie

Un **drum de creştere** *relativ la fluxul x în reţeaua $R = (G, s, t, c)$ este un A-drum de la s la t.*

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

Lema 1

Dacă P este un drum de creştere relativ la fluxul x în $R = (G, s, t, c)$, atunci $x^1 = x \otimes r(P)$ definit prin

$$x^1_{ij} = \begin{cases} x_{ij}, & \text{dacă } ij \notin E(P) \\ x_{ij} + r(P), & \text{dacă } ij \in E(P), ij \text{ este arc forward în } P \\ x_{ij} - r(P), & \text{dacă } ij \in E(P), ij \text{ este arc backward în } P \end{cases}$$

este un flux în R cu $v(x^1) = v(x) + r(P)$.

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

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

Demonstraţie. Din definiţia lui $r(P)$, constrângerile (i) sunt satisfăcute de x^1. Constrângerile (ii) - verificate de x - nu sunt afectate pentru x^1 în vreun nod $i \notin V(P)$.

Pentru $i \in V(P)$ există exact două arce în P incidente cu i, e. g. li şi ik. Avem următoarele două cazuri posibile:

a) li şi ik sunt arce forward:

$$\sum_{j} x^1_{ji} - \sum_{j} x^1_{ij} = \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x^1_{li} - x^1_{ik}$$

$$= \sum_{j \neq l} x_{ji} - \sum_{j \neq k} x_{ij} + x_{li} + r(P) - x_{ik} - r(P) = \sum_{j} x_{ji} - \sum_{j} x_{ij} = 0.$$

b) li este arc forward şi ik arc backward:

$$\sum_{j} x^1_{ji} - \sum_{j} x^1_{ij} = \sum_{j \neq l,k} x_{ji} - \sum_{j} x_{ij} + x^1_{li} + x^1_{ki}$$

$$= \sum_{j \neq l,k} x_{ji} - \sum_{j} x_{ij} + x_{li} + r(P) + x_{ki} - r(P) = \sum_{j} x_{ji} - \sum_{j} x_{ij} = 0.$$

G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms

c) li arc backward şi ik arc forward: similar cu b).

d) li şi ik arce backward: similar cu a).

$v(x^1)$ diferă de $v(x)$ datorită fluxului de pe arcul lt din P:

lt arc forward:

$$v(x^1) = \sum_j x^1_{jt} - \sum_j x^1_{tj} = \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x^1_{lt} =$$

$$= \sum_{j \neq l} x_{jt} - \sum_j x_{tj} + x_{lt} + r(P) = v(x) + r(P).$$

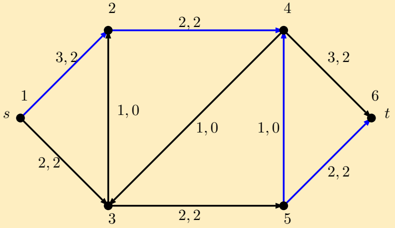
lt arc backward:

$$v(x^1) = \sum_j x^1_{jt} - \sum_j x^1_{tj} = \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - x^1_{tl} =$$

$$= \sum_j x_{jt} - \sum_{j \neq l} x_{tj} - (x_{tl} - r(P)) = v(x) + r(P). \square$$

G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms

Pentru exemplul de mai sus, fluxul $x^1 = x \otimes r(P)$ de valoare $v(x^1) = v(x) + r(P) = 3 + 1 = 4$ este:



Algorithms * G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms * G. Croitoru

Remarci

- Lema de mai sus explică şi numele drumurilor de creştere şi capacitatea reziduală.
- Din definiţie, dacă P este un drum de creştere, atunci $r(P) > 0$ şi $v(x \otimes r(P)) > v(x)$. Urmează că dacă există un drum de creştere relativ la fluxul x, atunci x nu este un flux de valoare maximă.

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

Definiţie

Fie $R = (G, s, t, c)$ o reţea. O **secţiune** *în R este o partiţie (S, T) a lui $V(G)$ cu $s \in S$ şi $t \in T$.* **Capacitatea secţiunii** *(S, T) este*

$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij}.$$

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

Lema 2

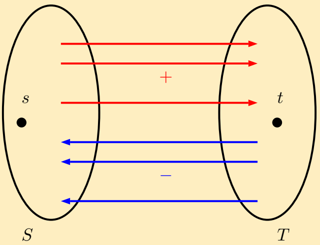
Dacă x este un flux în $R = (G, s, t, c)$ şi (S, T) este o secţiune în această reţea, atunci

$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji})$$

(valoarea fluxului este fluxul net care trece prin secţiune).

Problema fluxului maxim - Secţiuni

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



Demonstraţie.

$$v(x) = \left(\sum_j x_{sj} - \sum_j x_{js} \right) + 0 =$$

$$= \left(\sum_j x_{sj} - \sum_j x_{js} \right) + \sum_{i \in S, i \neq s} \left(\sum_j x_{ij} - \sum_j x_{ji} \right) =$$

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

Demonstraţie (continuare).

$$= \sum_{i \in S} \left(\sum_j x_{ij} - \sum_j x_{ji} \right) = \sum_{i \in S} \sum_j (x_{ij} - x_{ji}) =$$

$$= \sum_{i \in S} \sum_{j \in S} (x_{ij} - x_{ji}) + \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) =$$

$$= 0 + \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}). \; \square$$

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

Lema 3

Dacă x este un flux în $R = (G, s, t, c)$ şi (S, T) este o secţiune în această reţea, atunci

$$v(x) \leqslant c(S, T).$$

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

Demonstraţie. Din Lema 2

$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) \overset{x_{ij} \leqslant c_{ij}}{\leqslant} \sum_{i \in S} \sum_{j \in T} (c_{ij} - x_{ji}) \overset{x_{ji} \geqslant 0}{\leqslant}$$

$$\overset{x_{ji} \geqslant 0}{\leqslant} \sum_{i \in S} \sum_{j \in T} c_{ij} = c(S, T). \ \square$$

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

Remarcă

Dacă $\overline{x}$ este un flux în R şi $(\overline{S}, \overline{T})$ este o secţiune astfel încât $v(\overline{x}) = c(\overline{S}, \overline{T})$, atunci, $\forall x$ flux în R, avem $v(x) \leqslant c(\overline{S}, \overline{T}) = v(\overline{x})$, i. e., $\overline{x}$ este un flux de valoare maximă în R.

Similar, $\forall (S, T)$ secţiune în R, avem $c(S, T) \geqslant v(\overline{x}) = c(\overline{S}, \overline{T})$, adică, $(\overline{S}, \overline{T})$ este o secţiune de capacitate minimă în R.

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

Un flux x este un flux de valoare maximă dacă şi numai dacă nu există un drum de creştere relativ la x în R.

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

Demonstraţie. "$\Rightarrow$" Dacă P este un drum de creştere relativ la x, atunci $x \otimes r(P)$ este un flux în R de valoare strict mai mare.
"$\Leftarrow$" Fie x un flux în R cu proprietatea că nu există drum de creştere relativ la x în R. Fie

$$S = \{i \,:\, i \in V \text{ şi } \exists P \text{ un } A\text{-drum în } R \text{ de la } s \text{ la } i\}.$$

Evident $s \in S$ (există un A-drum de lungime 0 de la s la s) şi $t \notin S$ (nu există vreun A-drum de la s la t). Astfel, luând $T = V \setminus S$, (S, T) este o secţiune în R.

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

Demonstraţie (continuare). $\forall i \in S$ şi $\forall j \in T$ avem:

- dacă $ij \in E$, atunci $x_{ij} = c_{ij}$ şi
- dacă $ji \in E$, atunci $x_{ji} = 0$

(altfel A-drumul de la s la i poate fi extins la un A-drum de la s la j, astfel $j \in S$ - contradicţie). Urmează că $v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) = \sum_{i \in S} \sum_{j \in T} c_{ij} = c(S, T)$, i. e., x este un flux de valoare maximă (vezi remarca de mai sus). $\square$

* 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 - 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

Teorema 2

(Teorema fluxului întreg) Dacă toate capacităţile din R sunt întregi atunci există un flux întreg, x, de valoare maximă (toate $x_{ij} \in \mathbb{Z}_+$).

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

Demonstraţie. Considerăm următorul algoritm:

$x^0 \leftarrow 0$; $i \leftarrow 0$;
while ($\exists P_i$ un drum de creştere relativ la x^i) **do**
  $x^{i+1} \leftarrow x^i \otimes r(P_i)$; $i++$;
end while

Să observăm că "x^i are doar componente întregi" este un invariant al algoritmului (din definiţia lui $r(P_i)$, dacă toate capacităţile sunt întregi şi x^i este întreg, atunci $r(P_i)$ este întreg şi astfel x^{i+1} este întreg).

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

Demonstraţie (continuare). Mai mult, în fiecare iteraţie while, valoarea fluxului curent creşte (cu cel puţin 1), astfel algoritmul se opreşte în cel mult $c(\{s\}, V \setminus \{s\}) \in \mathbb{Z}_+$ iteraţii while.
Nu mai există drumuri de creştere relativ la fluxul final, astfel, din Teorema 1, fluxul este de valoare maximă. □

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

Remarcă

Algoritmul de mai sus se opreşte şi când toate capacităţile sunt numere raţionale.

* 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 - 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

Teorema 3

(Teorema flux maxim – secţiune minimă) Valoarea maximă a unui flux în reţeaua $R = (G, s, t, c)$ este egală cu capacitatea minimă a unei secţiuni în R.

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

Linia demonstraţiei. Dacă descriem un algoritm care, plecând cu un flux iniţial x^0 (e.g., $x^0 = 0$), construieşte într-un număr finit de paşi un flux x fără drumuri de creştere, atunci secţiunea considerată în demonstraţia Teoremei 1 satisface, împreună cu x, cerinţa teoremei. Pentru capacităţi raţionale, algoritmul considerat în demonstraţia Teoremei 2 satisface această condiţie. Pentru capacităţi arbitrare reale vom prezenta mai târziu un astfel de algoritm datorat lui **Edmonds şi Karp (1972)**.

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 -

Remarci

- O demonstraţie scurtă a teoremei de mai sus constă în a arăta că există un flux de valoare maximă şi de a aplica construcţia din demonstraţia Teoremei 1. Un flux de valoare maximă există întotdeauna observând că acesta este soluţia optimă a unei probleme de programare liniară (peste un politop nevid).
- Importanţa algoritmică a Teoremei 3 este dată de faptul că mulţimea tuturor secţiunilor dintr-o reţea este finită, pe când mulţimea tuturor fluxurilor este infinită.

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

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

Algoritmul întreţine o etichetare a nodurilor din reţea pentru a determina drumuri de creştere relativ la fluxul curent x. Când nu mai există drumuri de creştere, fluxul x este de valoare maximă.
Fie $R = (G = (V, E), s, t, c)$ o reţea şi x un flux în R.

Eticheta unui nod j, care are trei componente (e_1, e_2, e_3), semnifică: există un A-drum de la s la j, P, unde $e_1 = i$ este nodul dinaintea lui j pe acest drum, $e_2 \in \{forward, backward\}$ reprezintă direcţia arcului ij, iar $e_3 = r(P)$.

Iniţial s este etichetat $(\cdot, \cdot, \infty)$. Celelalte noduri primesc eventual etichete prin vizitarea (scanarea) nodurilor deja etichetate:

- 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 - Graph

procedure scan(i)
for ($j \in V$, neetichetat) **do**
  if ($ij \in E$ şi $x_{ij} < c_{ij}$) **then**
    etichetează j cu $e = (i, forward, \min\{e_3[i], c_{ij} - x_{ij}\})$;
  end if
  if ($ji \in E$ şi $x_{ji} > 0$) **then**
    etichetează j cu $e = (i, backward, \min\{e_3[i], x_{ji}\})$;
  end if
end for

Semnificaţia componentelor etichetelor este întreţinută de procedura scan.

Când, în procedura scan, nodul t este etichetat, se poate detecta un drum de creştere P, relativ la fluxul curent x, astfel:

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

$r(P)$ este componenta e_3 a etichetei lui t, nodurile lui P pot fi găsite în $\mathcal{O}(n)$ prin explorarea primei componente a etichetelor, şi modificarea $x \otimes r(P)$ se poate face în timpul acestei explorări folosind a doua componentă a etichetelor.

Pentru noul flux, se porneşte cu o nouă etichetare (de la s).

Dacă toate noduri etichetate au fost scanate şi nodul t nu a primit etichetă, urmează că fluxul curent nu admite drumuri de creştere, deci este de valoare maximă. Dacă S este mulţimea nodurilor etichetate şi $T = V \setminus S$, atunci (S, T) este o secţiune de capacitate minimă în R.

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

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

porneşte cu un flux iniţial $x = (x_{ij})$ (e. g., $x = 0$)
$e(s) \leftarrow (\cdot, \cdot, \infty)$;
while ($\exists$ noduri etichetate şi nescanate) **do**
    alege i un nod etichetat şi nescanat;
    scan(i);
    if (t a fost etichetat) **then**
        modifică fluxul pe drum dat de etichete;
        şterge toate etichetele; $e(s) \leftarrow (\cdot, \cdot, \infty)$;
    end if
end while
$S \leftarrow \{i \,:\, i \in V, i \text{ este etichetat}\}$;
$T \leftarrow V \setminus S$;

x este un flux de valoare maximă, (S, T) este o secţiune de capacitate minimă.

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

Complexitatea timp: Fiecare creştere a fluxului curent necesită cel mult $2m$ ($m = |E|$) inspecţii ale arcelor pentru etichetarea altor noduri. Dacă toate capacităţile sunt intregi, sunt necesare cel mult v creşteri (v fiind valoarea fluxului maxim). Astfel algoritmul are complexitatea timp $\mathcal{O}(mv)$.

Dacă U este un majorant al tuturor capacităţilor pe arce, atunci $v \leqslant (n-1)U$ (acesta este un majorant al secţiunii $(\{s\}, V \setminus \{s\})$), deci algoritmul are complexitatea timp $\mathcal{O}(nmU)$.

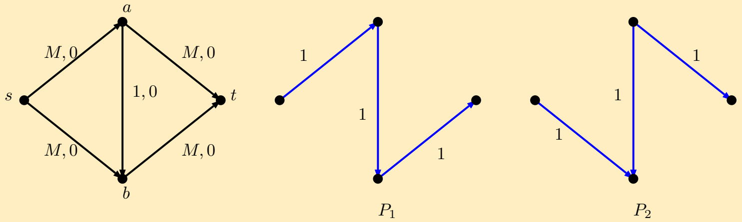
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Remarcă

E posibil ca algoritmul să nu se termine pentru capacităţi iraţionale. Această situaţie nu apare în implementările practice, dar neajunsul acestei descrieri a algoritmului este dat de faptul că numărul de creşteri ale fluxului curent depinde de capacităţi (şi nu de dimensiunile reţelei).

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

Exemplu



Dacă operaţia alege din algoritmul de mai sus determină drumurile de creştere $P_1, P_2, P_1, P_2, \ldots,$, unde $P_1 = (s, sa, a, ab, b, bt, t)$ şi $P_2 = (s, sb, b, ba, a, at, t)$, atunci fiecare creştere a fluxului se face cu 1, şi algoritmul necesită $2M$ creşteri, ceea ce este prea mult.

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

Modificarea lui Edmonds & Karp a algoritmului lui Ford & Fulkerson

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

Aceste deficienţe ale algoritmului pot fi evitate dacă alegerea nodurilor etichetate în vederea scanării nu se face aleatoriu (**Dinic (1970)**, **Edmonds & Karp (1972)**).

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

Definiţie

Un **cel mai scurt drum de creştere** *relativ la fluxul x în R este un drum de creştere de lungime minimă printre toate drumurile de creştere relativ la x.*

Fie x un flux în reţeaua R. Fie x^k $(k \geqslant 1)$ secvenţa de fluxuri:

$x^1 \leftarrow x$;

$x^{k+1} \leftarrow x^k \otimes r(P_k)$, P_k un cel mai scurt drum de creştere relativ la x^k;

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

Pentru a arăta că această secvenţă este finită, fie $\forall i \in V$ şi $\forall k \in \mathbb{N}^*$:

$$\sigma_i^k = \text{ lungimea minimă a unui } A\text{-drum de la } s \text{ la } i \text{ relativ la } x^k.$$

$$\tau_i^k = \text{ lungimea minimă a unui } A\text{-drum de la } i \text{ la } t \text{ relativ la } x^k.$$

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

Lema 4

$\forall i \in V$ şi $\forall k \in \mathbb{N}^*$ avem

$$\sigma_i^{k+1} \geqslant \sigma_i^k \text{ şi } \tau_i^{k+1} \geqslant \tau_i^k.$$

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

Demonstraţie. Omisă.

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

Teorema 4

(Edmonds & Karp, 1972) Dacă $x = x^1$ este un flux arbitrar în reţeaua R, atunci secvenţa de fluxuri $x^1, x^2, \ldots$, obţinută din x^1 prin creşteri succesive cu drumuri de creştere de lungime minimă, are cel mult $mn/2$ termeni (în cel mult $mn/2$ creşteri succesive se obţine un flux x cu proprietatea că nu există drum de creştere relativ la x).

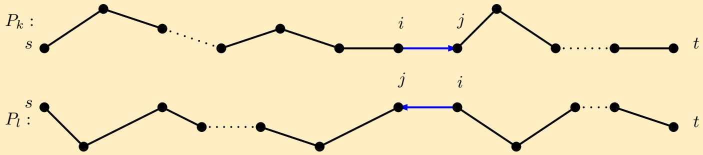
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Demonstraţie. Dacă P este un drum de creştere relativ la un flux x în R, cu capacitatea reziduală $r(P)$, un arc critic în P este un arc $e \in E(P)$ cu $r(e) = r(P)$. În $x \otimes r(P)$, fluxul de pe arce critice devine fie egal cu capacitatea (pe arcele forward), sau nul (pe arcele backward). Fie ij un arc critic de pe un cel mai scurt drum de creştere P_k relativ la x^k. Lungimea lui P_k este $\sigma_i^k + \tau_i^k = \sigma_j^k + \tau_j^k$

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

Deoarece ij este a critic în P_k , in x^{k+1} nu mai poate fi utilizat în aceeaşi direcţie ca în P_k. Fie P_l (cu $l > k$) primul cel mai scurt drum de creştere relativ la x^l în care fluxul de pe arcul ij va fi modificat, (când arcul va fi folosit în direcţie inversă decât în P_k). Avem două cazuri:

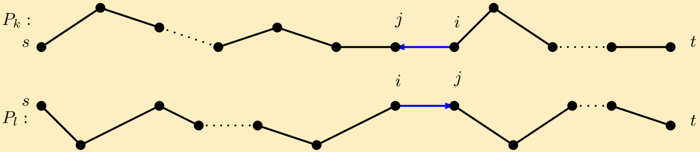
ij este un arc forward în P_k. Atunci $\sigma_j^k = \sigma_i^k + 1$; în P_l ij va fi arc backward, deci $\sigma_i^l = \sigma_j^l + 1$.



Urmează că $\sigma_i^l + \tau_i^l = \sigma_j^l + 1 + \tau_i^l \geqslant \sigma_j^k + 1 + \tau_i^k = \sigma_i^k + \tau_i^k + 2$ (din Lema 4). Am obţinut că $length(P_l) \geqslant length(P_k) + 2$.

G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms * G. Croitoru - Graph Algorithms

ij este un arc backward în P_k. Atunci $\sigma_i^k = \sigma_j^k + 1$; în P_l ij va fi arc forward, deci $\sigma_j^l = \sigma_i^l + 1$.



Urmează că $\sigma_j^l + \tau_j^l = \sigma_i^l + 1 + \tau_j^l \geqslant \sigma_i^k + 1 + \tau_j^k = \sigma_j^k + \tau_j^k + 2$. Obţinem că $length(P_l) \geqslant length(P_k) + 2$.

Astfel orice cel mai scurt drum de creştere pe care arcul ij este critic are lungimea cu cel puţin 2 mai mare decât lungimea drumului precedent pe care ij a fost critic. Deoarece lungimea unui drum în G este cel mult $n - 1$, urmează că un arc fixat nu poate fi critic de mai mult de $n/2$ ori (de-a lungul întregului proces de creştere).

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

Orice drum de creştere are cel puţin un arc critic. Astfel în secvenţa (P_k) avem cel mult $mn/2$ drumuri de creştere cele mai scurte.

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

Corolar

În orice reţea există un flux x cu proprietatea că nu există drumuri de creştere relativ la x.

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

Remarci

- Demonstraţia Teoremei 4 este acum încheiată.
- Singura modificare din **Algoritmul lui Ford & Fulkerson** este în alegerea nodului etichetat care va fi scanat: se foloseşte regula "primul etichetat-primul scanat" adică, se întreţine o coadă a nodurilor etichetate (iniţializată cu s, la fiecare început de creştere).

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

În concluzie, avem următoarea teoremă.

Teorema 5

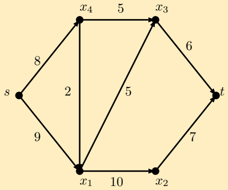
(Edmonds-Karp, 1972) Dacă în **Algoritmul lui Ford & Fulkerson** scanarea noduri etichetate se face într-o manieră **bfs**, atunci un flux de valoare maximă se obţine în $\mathcal{O}(m^2n)$ time.

Demonstraţie. Există $\mathcal{O}(mn)$ creşteri de flux (din Teorema 4), fiecare de complexitate $\mathcal{O}(m)$. $\square$

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 *

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

Exerciţiul 1. Considerăm reţeaua de transport de mai jos:



Determinaţi un flux maxim în reţeaua de mai jos utilizând algoritmul lui Edmonds-Karp pentru următoarele ordonări ale listelor de adiacenţă:

(a) $A(s) = (x_1, x_4)$, $A(x_1) = (s, x_3, x_2, x_4)$, $A(x_2) = (x_1, t)$, $A(x_3) = (x_1, x_4, t)$, $A(x_4) = (x_3, x_1)$.

(b) $A(s) = (x_4, x_1)$, $A(x_1) = (s, x_2, x_3, x_4)$, $A(x_2) = (x_1, t)$, $A(x_3) = (x_1, x_4, t)$, $A(x_4) = (x_3, x_1)$.

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

Exerciţiul 2. Considerăm o reţea $R = (G, s, t, ; c)$, unde $G = (V, E)$ are n noduri şi m arce, şi funcţia de capacitate are doar valori întregi ($c : E \to \mathbb{Z}_+$). Fie $C = \max_{e \in E} c(e)$.

(a) Arătaţi că valoarea maximă a unui flux în R este cel mult $m \cdot C$.

(b) Arătaţi că, pentru fiecare flux x din R şi pentru fiecare $K \in \mathbb{Z}_+$, putem găsi un drum de creştere P cu capacitatea reziduală $\delta(P)$ cel puţin K - dacă un asemenea drum există - în $\mathcal{O}(m)$ time complexity.

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

Exerciţiul 2 (continuare).

(c) Considerăm următorul algoritm:

SC-MAX-FLOW(R) {

$C \leftarrow \max\limits_{e \in E} c(e)$;

$x \leftarrow 0$;// x **fluxul curent;**

$K \leftarrow 2^{1+\lfloor \log C \rfloor}$;

while($K \geqslant 1$) {

while(x **are un drum de creştere** P **cu** $\delta(P) \geqslant K$)

$x \leftarrow x \otimes \delta(P)$;

$K \leftarrow K/2$;

}

return x;

}

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

Exerciţiul 2 (continuare).

1. Arătaţi că procedura **SC-MAX-FLOW**(R) returnează un flux de valoare maximă x în R.
2. Arătaţi că, după fiecare iteraţie while exterioară, valoarea maximă a unui flux în R este cel mult $v(x) + m \cdot K$.
3. Arătaţi că, $\forall\, K \in \mathbb{Z}_+$, există cel mult $2m$ iteraţii while interioare. În consecinţă procedura are complexitatea timp $\mathcal{O}(m^2 \log C)$.

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

Exerciţiul 3. Digraful $G = (V, E)$ reprezintă topologia unei reţele de procesoare.Fiecărui procesor, $v \in V$, îi cunoaştem încărcarea, $load(v) \in \mathbb{R}_+$. Folosind un flux maxim într-o anumită reţea determinaţi o strategie statică de echilibrare a încărcaării (static load balancing strategy) în G: indicaţi pentru fiecare procesor încărcarea ce trebuie trimisă şi cărui procesor aşa încât toate procesoarele să aibă aceeaşi încărcare.

Exerciţii pentru seminarul din săptămâna a 11-a

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

Exerciţiul 4. Fie $R = (G, s, t, c)$ o reţea şi $(S_i, T_i)(i = \overline{1,2})$ două secţiuni de capacitate minimă în R. Arătaţi că $(S_1 \cup S_2, T_1 \cap T_2)$ şi $(S_1 \cap S_2, T_1 \cup T_2)$ sunt de asemenenea secţiuni de capacitate minimă.

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

Exerciţiul 5. Fie $R = (G = (V, E), s, t, c)$ o reţea şi $c : E \to \mathbb{Z}_+$, $n = |V|$, $m = |E|$.

(a) Descrieţi un algoritm de complexitate timp $\mathcal{O}(n + m)$ pentru a determina o secţiune de capacitate minimă în R, având la îndemână un flux de valoare maximă x^*.

(b) Folosind un algoritm de flux maxim pentru o anumită funcţie de capacitate, arătaţi că se poate găsi o secţiune de capacitate minimă în R, (S_0, T_0), cu număr minim de arce.

- 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 - 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.

Exerciţiul 6. 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.

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

Exerciţiul 7. Fie $R = (G, s, t, c)$ o reţea care are toate capacităţile întregi pozitive şi pare. Arătaţi că există un flux maxim cu toate valorile întregi pozitive şi pare.

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 - 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 *