PB165 – Grafy a sítě

Toky v síti

26. 11. 2015

Obsah přednášky

Řez v grafu

Neformálně:

- ,,Rozříznutí" grafu napříč hranami (nikoliv skrz vrcholy) na dvě poloviny.
- Rozdělení vrcholů na dvě části.

Definice

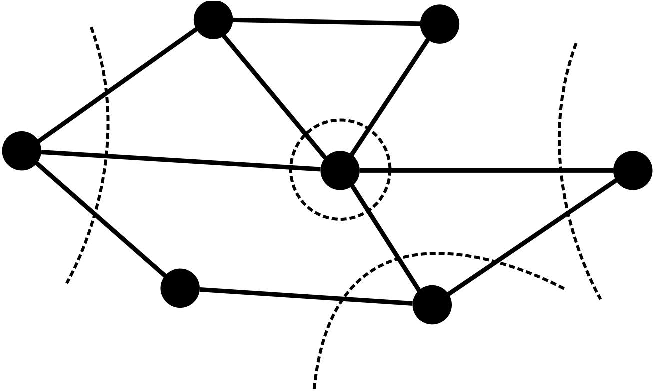
Řezem v grafu $G = (V, E)$ nazýváme rozklad množiny V na 2 neprázdné podmnožiny $P, \overline{P}$. $W_G(P)$ je množina všech hran, jejichž jeden vrchol je v P a druhý nikoliv.

Jelikož se jedná o rozklad, platí:

$$P \cap \overline{P} = \emptyset, P \cup \overline{P} = V$$

Řezy v grafu

V každém grafu existuje $2^{|V|-1} - 1$ řezů.

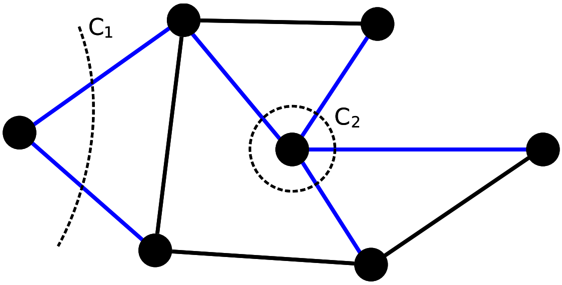


Obrázek : Příklady řezů v grafu jsou vyznačeny čárkovaně.

Hrany křižující řezy

Definice

Nechť řez C dělí vrcholy na množiny $P, \overline{P}$. O hranách (u, v), jejichž jeden vrchol leží v P a druhý nikoliv, říkáme že křižují řez C.



Obrázek : Hrany křižující řezy C_1, C_2 jsou vyznačeny modře.

Bipartitní grafy

Definice

Bipartitiní graf je takový graf G, jehož množina vrcholů je disjunktním sjednocením dvou množin S a T a platí $E(G) = W_G(S)$. Množiny S a T nazýváme stranami bipartitního grafu.

Každá hrana grafu G má jeden vrchol v S a druhý v T.

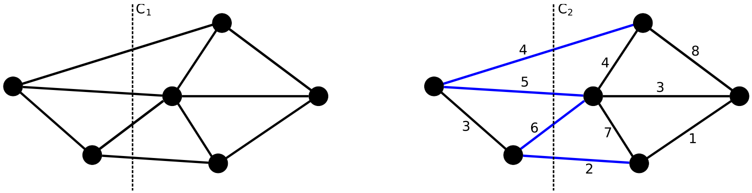
Definice

Úplný bipartitní graf je takový bipartitní graf, jehož každá dvojice vrcholů (s, t), $s \in S$ a $t \in T$ je spojena právě jednou hranou.

Váha řezu

Definice

Vahou řezu v hranově neohodnoceném grafu označujeme počet hran, které tento řez křižují.
V hranově ohodnoceném grafu se vahou rozumí součet ohodnocení všech hran křižujících tento řez.



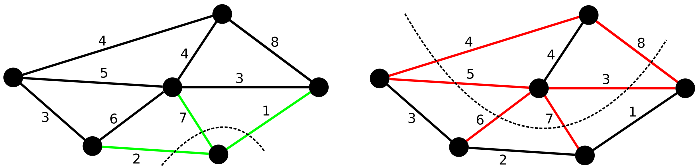
Obrázek : Váha řezu C_1 v neohodnoceném grafu je rovna 4. Váha řezu C_2 v ohodnoceném grafu je rovna 17.

Minimální a maximální řez

Definice

Minimálním rozumíme takový řez v grafu, jehož váha je minimální. Maximální řez je naopak ten s maximální vahou.

Minimální řez v grafu může být nalezen v čase polynomiálním vůči velikosti grafu. Naopak, problém maximálního řezu je NP-úplný.



Obrázek : Minimální řez ve vyobrazeném grafu je vyznačen zeleně, maximální červeně.

Síť a tok

Definice

Sítí nazýváme orientovaný, hranově ohodnocený graf $G = (V, E)$.

Definice

Tokem v síti nazýváme takové ohodnocení hran reálnými čísly $f : E(G) \to \mathbb{R}$, které pro každý vrchol v splňuje Kirchhoffův zákon

$$\sum_{e \in E^+(v)} f(e) = \sum_{e \in E^-(v)} f(e)$$

Takový graf si můžeme představit jako soustavu potrubí, pro níž platí zákon zachování hmoty, tj. kolik do vrcholu přitéká, tolik z něj zase vytéká.
Orientace hrany určuje směr proudění, záporný tok představuje proudění proti směru hrany.

Cirkulace a zdroj a spotřebič

Pokud Kirchhoffův zákon platí pro všechny vrcholy, mluvíme o *cirkulaci*.

Alternativou je tzv. *tok od zdroje ke spotřebiči*, kde dva vrcholy Kirchhoffův zákon nesplňují. Ve *zdroji* tok vzniká a ve *spotřebiči (stok, výlevka, sink)* zaniká.

Tok od zdroje ke spotřebiči můžeme vždy převést na cirkulaci přidáním hrany spojující zdroj a spotřebič. Takovou hranu nazýváme *návratovou hranou*.

Přípustný tok

Zpravidla omezujeme tok na hraně shora i zdola, tj. platí $f(e) \in \langle l(e), c(e) \rangle$. Číslo $c(e)$ nazýváme *kapacitou* hrany, případně *horním omezením toku v hraně*. Číslo $l(e)$ nazýváme *dolním omezením toku v hraně*. Tok, který splňuje $l(e) \leq f(e) \leq c(e)$ pro všechny hrany e nazýváme *přípustným tokem*.
V řadě praktických případů bývá dolní omezení toku zpravidla rovno 0, je-li však nenulové, není a priori jasné, zda existuje přípustný tok v síti.

Příklady sítí

Výše definované *sítě* jsou vhodnými reprezentacemi reálných sítí. Klasická teorie grafů se velmi často věnuje problémům toků na železničních, silničních a dalších dopravních sítích. Samostatnou oblastí jsou rozvodné sítě – vodovodní, plynové atd. Obecně mluvíme o *transportních sítích*. Většina úloh je věnována optimalizaci takovýchto sítí, případně nalezení úzkých míst, maximální kapacity (propustnosti) sítě, garance minimální propustnosti i při výpadku některých linek či vrcholů apod.
Pro nás jsou zajímavé toky v počítačových sítích.
Na řešení úloh s toky lze převést i řadu plánovacích úloh, např. tzv. *přiřazovací* úlohy. V těch máme za úkol přiřadit n úkolů mezi pracovníky tak, abychom minimalizovali náklad (provedení konkrétní úlohy konkrétním pracovníkem má svou cenu). Lze převést na bipartitní graf (pracovníci jsou zdroje a úlohy jsou spotřebiče, hrana představuje cenu práce).

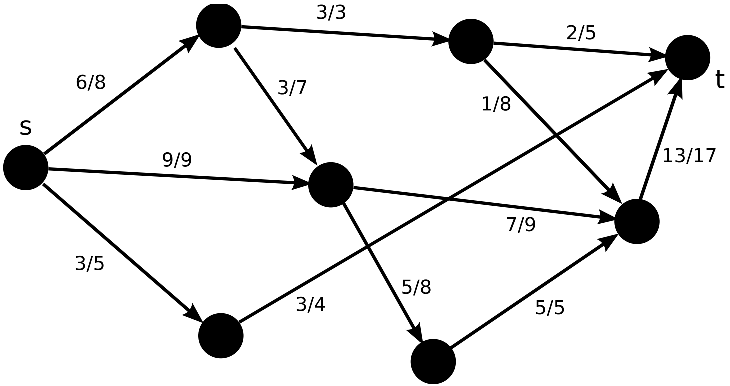
Reziduální tok

Definice

Reziduální kapacitou hrany e rozumíme číslo $c(e) - f(e)$, tj. rozdíl kapacity hrany a aktuálního toku.

Reziduální kapacity hran tvoří reziduální síť.
V mnoha případech potřebujeme zjistit, zda reziduální síť existuje.
Hrany s reziduální kapacitou nula nejsou v reziduální síti obsaženy (není možné přes ně vést nenulový tok).

Toky v síti – příklad



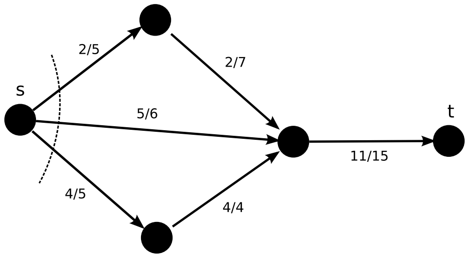
Obrázek : Příklad toku v síti. První číslo v hodnocení hrany je tok, druhé kapacita hrany

Velikost toku

Velikost toku značíme $F(f)$. Velikost toku od zdroje ke spotřebiči definujeme jako množství toku, které vzniká ve zdroji s.

$$F(f) = \sum_{e \in E^+(s)} f(e) - \sum_{e \in E^-(s)} f(e)$$

E^+, E^- označují součet toků vstupujících do vrcholu, resp. vystupujících z něj.



Obrázek : Velikost vyobrazeného toku (11) je dána množstvím toku opouštějícím zdroj.

Velikost toku přes řez

Nechť řez C dělí vrcholy grafu na množiny $P, \overline{P}$. Označíme jej C_P. Dále nechť zdroj toku náleží do množiny P a spotřebič do $\overline{P}$. Potom má smysl definovat velikost F_P toku přes řez C_P jako rozdíl mezi velikostí toku na hranách vedoucích z množiny P a velikostí toku na hranách vedoucích do této množiny. Říkáme, že řez C_P *odděluje* zdroj a spotřebič.

$$F_P(f) = \sum_{e \in W^+(P)} f(e) - \sum_{e \in W^-(P)} f(e)$$

W^+, W^- značí hrany vycházející z množiny W, resp. do ní vstupující.

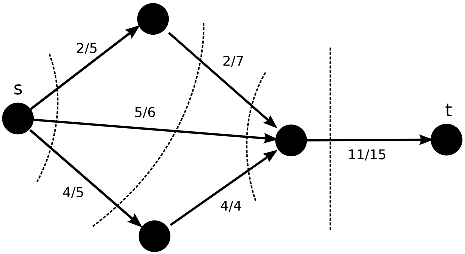
Shodnost toků přes řezy

Věta

Nechť C_P je libovolný řez, který odděluje zdroj a spotřebič. Potom pro velikost F_P toku přes C_P platí

$$F_P(f) = F(f)$$

Přes všechny řezy oddělující zdroj a spotřebič tedy protéká stejný tok.



Obrázek : Přes všechny vyznačené (i nevyznačené) řezy oddělující s a t protéka tok o velikosti 11.

Shodnost toků přes řezy – důkaz

Důkaz.

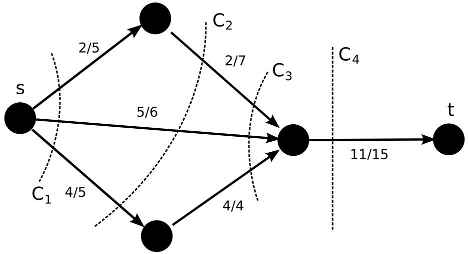
Důkaz povedeme indukcí:
Základ indukce: Položme $P = \{s\}$, kde s je zdroj toku. Tvrzení platí z definice.
Indukční krok: Do množiny P přidáme libovolný vrchol grafu, různý od spotřebiče. Jelikož pro tento vrchol musí platit Kirchhoffův zákon, nezmění se nikterak rozdíl mezi velikostmi toků z P vytékajících a do P vtékajících. Platnost tvrzení se tedy nezmění. Jelikož postupným přidáváním vrcholů lze získat libovolný řez, který odděluje zdroj od spotřebiče, platí tvrzení pro všechny řezy zdroj od spotřebiče oddělující. $\square$

Kapacita řezu

Kapacita řezu oddělujícího zdroj a spotřebič specifikuje, jaký maximální tok může tímto řezem protéct. Definována je jako součet kapacit všech hran, které tento řez protínají ve směru od zdroje ke spotřebiči zmenšená o součet minimálních kapacit hran opačně orienovaných.

$$C(C_P) = \sum_{e \in W^+(P)} c(e) - \sum_{e \in W^-(P)} l(e)$$

Obrázek : Kapacity řezů $C_1, \ldots, C_4$ jsou po řadě 16, 18, 17, 15.



Kapacita řezu má význam pro nalezení maximálního toku v grafu.

Maximální tok v grafu

Problém maximálního toku je hledání největšího toku v grafu od zdroje ke spotřebiči. Následující podmínky jsou ekvivalentní:

1. Tok f je maximální (maximalizujeme $F(f)$).
2. $|f|$ je kapacita některého řezu oddělujícího zdroj od spotřebiče.
3. V reziduální síti neexistuje cesta ze zdroje ke spotřebiči.

Algoritmy pro nalezení maximálního toku vycházejí z těchto ekvivalencí – hledají řez s minimální kapacitou nebo přidávají cesty mezi zdrojem a spotřebičem, dokud nějaké v reziduální síti existují.

Algoritmus – brutální síla

- Nejjednodušší algoritmus.
- Generuje postupně všechny podmnožiny vrcholů, pro každý provede následující kroky:
  - Najde mezi hranami všechny, které křižují řez definovaný touto množinou vrcholů.
  - Sečte kapacity hran křižujících tento řez, směřujících od zdroje ke spotřebiči.
- Výsledkem je řez s minimální vypočtenou kapacitou.

Jelikož všech řezů oddělujících zdroj od spotřebiče je $2^{|V|-2} - 1$, pro každý je potřeba zkontrolovat všech $|E|$ hran, celková časová složitost algoritmu hledání maximálního toku brutální silou je $\mathcal{O}(2^{|V|-2}|E|)$

Zlepšující cesta

Definice

Hranu nazveme hranou vpřed, je-li orientována ve směru průchodu cestou. Hrana vzad je pak orientována proti směru průchodu cestou.

Definice

Zlepšující cestou vzhledem k toku f nazveme takovou neorientovanou cestu ze zdroje ke spotřebiči, jejíž každá hrana splňuje $f(e) < c(e)$ pro hranu e vpřed a $f(e) > l(e)$ pro hranu vzad.

Definice říká, že aktuální tok lze zvýšit na hranách vpřed a snížit na hranách vzad o nějakou hodnotu $d > 0$.

Definice

Kapacitou zlepšující cesty pak rozumíme maximální hodnotu d, o kterou lze tok na zlepšující cestě změnit.

Ford-Fulkersonův algoritmus

- Využívá ekvivalence mezi maximalitou toku a neexistencí cesty ze zdroje ke spotřebiči v reziduální síti.
- Hledá *zlepšující* cesty mezi zdrojem a spotřebičem, dokud nějaká taková existuje.
- Pro hledání cest používá i zpětných hran.
- Z hran na cestě se vybere ta, jejíž reziduální kapacita je minimální.
  - V případě zpětné hrany (projité proti směru její orientace) se namísto hodnoty $c(e) - f(e)$ bere tok, který hranou protéká ve směru její orientace – tedy $f(e) - l(e)$. Toky se takto mohou vzájemně anulovat.
- O tuto minimální kapacitu se zvýší tok po všech dopředných hranách na nalezené cestě.
  - Naopak, na hranách zpětných se hodnota toku sníží o stejnou hodnotu.

Hledání zlepšující cesty

Můžeme použít *značkovací proceduru* ($P_V(e)$ je počáteční a $K_V(e)$ je koncový vrchol hrany e):

Inicializace Označkujeme vrchol zdroje, ostatní jsou bez značek.

Vpřed Existuje-li hrana e taková, že $P_V(e)$ má značku a $K_V(e)$ nemá a současně platí $f(e) < c(e)$, pak označkuj $K_V(e)$.

Vzad Existuje-li hrana e taková, že $K_V(e)$ má značku a $P_V(e)$ nemá a současně $l(e) < f(e)$, pak označkuj $P_V(e)$.

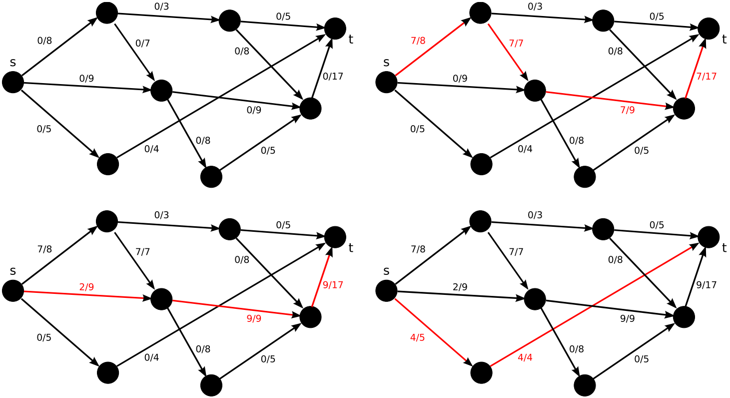
Ukončení Je-li označkován spotřebič, nalezli jsme zlepšující cestu. Nelze-li další vrchol označkovat, pak zlepšující cesta neexistuje.

Ford-Fulkersonův algoritmus

```
Pro všechny hrany (u,v)
| f(u,v) = 0
Dokud existuje zlepšující cesta p:
| Vyber minimální kapacitu d hrany na této cestě.
| Pro všechny hrany na cestě p:
| | f(u,v) = f(u,v) + d
| | f(v,u) = f(v,u) - d
```

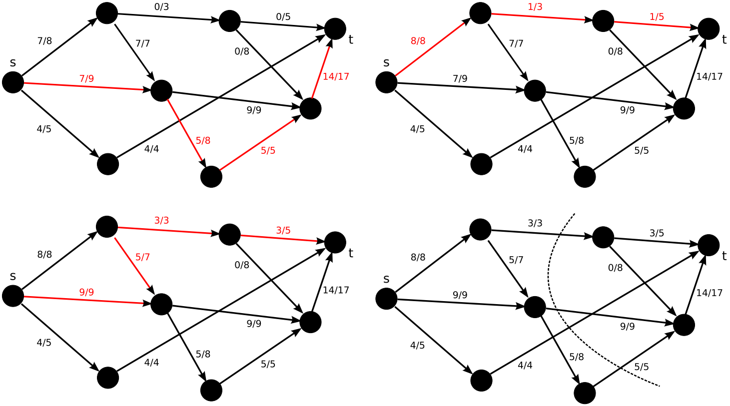
Algoritmus neříká, jakým způsobem se má cesta ze zdroje ke spotřebiči hledat. V praxi se používá obvykle průchod do hloubky, nebo průchod do šířky, čímž se z algoritmu stává Edmonds-Karpův (viz dále).

Ford-Fulkersonův algoritmus – příklad



Obrázek : Příklad běhu Ford-Fulkersonova algoritmu, 1. část.

Ford-Fulkersonův algoritmus – příklad



Obrázek : Příklad běhu Ford-Fulkersonova algoritmu, 2. část. Minimální řez je vyznačen na posledním obrázku. Kapacita je 21, což je i maximální tok v tomto grafu.

Ford-Fulkersonův algoritmus – složitost

- V obecném případě není možné dokázat, že běh algoritmu skončí.
- V některých případech nemusí hodnota nalezeného toku ani konvergovat k maximu.

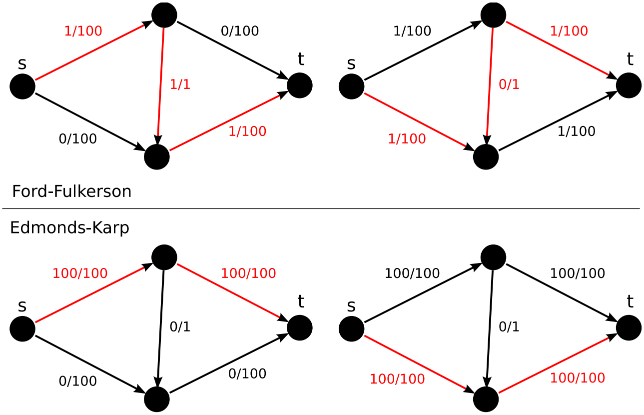
V reálných aplikacích jsou kapacity hran obvykle reprezentovány celými čísly. To zaručuje ukončení běhu algoritmu:

- Maximální tok má v takovém případě také celočíselnou hodnotu.
- Časová složitost nalezení zlepšující cesty je $\mathcal{O}(|E|)$
- S každou nalezenou zlepšující cestou se hodnota nalezeného toku zvýší minimálně o 1.
- Maximálně může tedy proběhnout nejvýše $F(f)$ iterací.

Edmonds-Karpův algoritmus

- Specializace Ford-Fulkersonova algoritmu.
- Pro hledání zlepšujicích cest je použit průchod do šířky.
- Pro potřeby průchodu do šířky jsou délky hran považovány za jednotkové.
- Průchod do šířky zajistí, že každá nalezená zlepšující cesta je nejméně tak dlouhá, jako předchozí nalezená.
- Maximální možná délka zlepšující cesty je $|V|$.
- Složitost algoritmu tak činí $\mathcal{O}(VE^2)$.

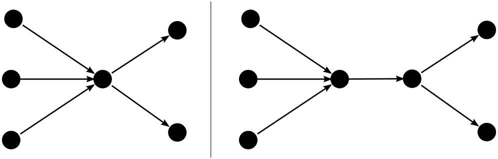
Edmonds-Karpův algoritmus – příklad



Obrázek : V horní části obrázku je znázorněn možný počátek běhu Ford-Fulkersonova algoritmu. Běh může pokračovat stejným způsobem i nadále, a tak potřebovat mnoho iterací. Edmonds-Karpův algoritmus nalezne odpověď během 2 iterací.

Omezení toku vrcholem

Reálné aplikace mohou klást i omezení na velikost toku procházejícího vrcholem – např. sběrné místo kanalizací, aktivní prvek v síti, rychlost zpracování dat na příjemci. Pokud jsou toky nezáporné, lze použít následující transformaci grafu:

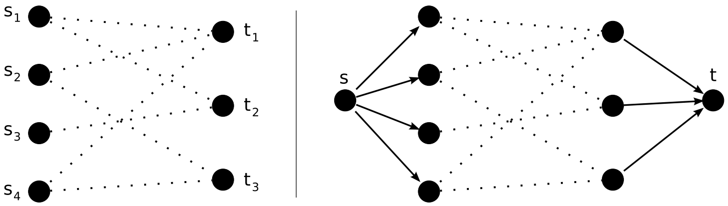


Obrázek : Vrchol je nahrazen dvěma vrcholy a hranou.

Vrchol v s omezenou kapacitou nahradíme vrcholy v_1, v_2, jejichž kapacita nebude omezena. Hrany směřující do v přesměrujeme do v_1, hrany z v vycházející budou vycházet z v_2. Vrcholy v_1, v_2 spojíme hranou, jejíž kapacita bude rovna původní kapacitě vrcholu v. Na takový graf je poté možno použít standardní algoritmy pro hledání maximálního toku.

Několik zdrojů a spotřebičů

Obdobně lze standardní algoritmy použít i v případě, kdy zadání obsahuje více než 1 zdroj nebo spotřebič:



K síti přidáme fiktivní zdroj a spotřebič. Z nově přidaného zdroje povedou hrany (s „neomezenou“ kapacitou) do všech zdrojů, obdobně přidáme hrany ze všech spotřebičů do nově přidaného.

Nejlevnější toky

Ke každé hraně je krom její kapacity definována i cena $a(e)$ jednotkového toku. Cena toku hranou e je potom rovna $a(e)f(e)$. Celková cena toku sítí je potom definována jako

$$\sum_{e \in E} a(e)f(e).$$

Úkolem je potom najít maximální tok sítí takový, že jeho cena bude zároveň minimální.

Přípustná cirkulace

Pokud se omezíme na *cirkulace*, je celá řada algoritmů (a odpovídající teorie) jednodušší. A platí, že úlohy týkající se přípustného toku od zdroje ke spotřebiči lze převést na hledání přípustné cirkulace přidáním návratové hrany.

Věta

V síti s omezeními toku l a c existuje přípustná cirkulace právě tehdy, když každý řez má nezápornou kapacitu

$$C(C_P) = \sum_{e \in W^+(P)} c(e) - \sum_{e \in W_-(P)} l(e) \geq 0$$

Algortimus pro přípustnou cirkulaci

Vstupem je síť G s omezeními toku l, c a libovolná (i nulová) cirkulace f. Výstupem je buď přípustná cirkulace f' nebo řez se zápornou kapacitou.

1. Najdeme hranu h s nepřípustným tokem. Pokud taková hrana neexistuje, výpočet končí a dosavadní tok f' je přípustný.
2. Je-li $f(h) < l(h)$, pak $z := K_V(h), s := P_V(h)$, v opačném případě $(f(h) > c(h))$ $z := P_V(h), s := K_V(h)$.
3. Nalezneme zlepšující cestu z vrcholu z do vrcholu s. Pokud cesta neexistuje, výpočet končí, přípustná cirkulace neexistuje a množina označkovaných vrcholů P určuje řez C_P, který má zápornou hodnotu.
4. Pokud cesta existuje, doplníme zlepšující cestu o hranu h, čímž vznikne zlepšující kružnice. Vypočteme její kapacitu, změníme toky na jejích hranách (viz předchozí algoritmy). Pak se vracíme zpět na krok 1.

Network Coding

26. 11. 2015

Motivace

Propustnost sítě je dána větou *MaxFlow-MinCut*. Ta říká, že maximální propustnost je rovna minimálnímu řezu takovému, že zdroj dat je v T a přijímající v T'.

Definice platí pro:

- unicast
- broadcast
- multicast

Unicast

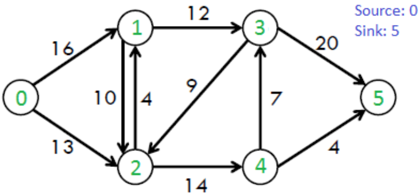
Po unicast umíme maximální tok najít pomocí algoritmu *Ford-Fulkerson*. Neformálně: Algoritmus využíva hledání *zlepšujících cest*. Začíná s nulovým tokem od zdroje k příjemci a postupně ho zvyšuje, dokud je to možné (nejsou překročeny kapacity linek).

Zlepšující cesta od zdroje k příjemci je taková cesta, na které můžeme zvýšit tok, aniž by byla překročena kapacita některé linky na cestě.

V každém kroku algoritmu nalezneme nějakou *zlepšující cestu* a zvýšíme tok.

Neexistuje-li zlepšující cesta, algoritmus končí.

$\Rightarrow$ umíme efektivně realizovat maximální tok v grafech s jedním příjemcem.



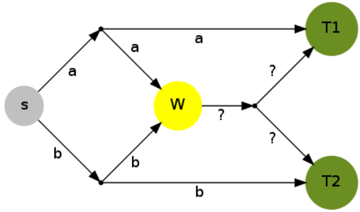
Multicast

Pro multicast pořád platí věta MaxFlow-MinCut.

Problémem ale je, že neexistuje efektivní algoritmus pro hledání maximální propustnosti.
$\Rightarrow$ neumíme prakticky realizovat maximální tok (kromě brute-force způsobu).

Multicast

Příklad:

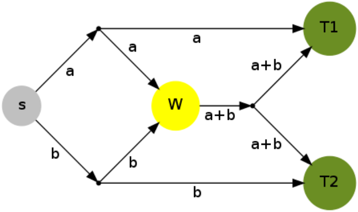


Hrany mají jednotkovou kapacitu a navěstí hran určují přenášenou informaci.

- Danou hranou umíme přenést jeden symbol za jednotku času.
- Uzel W může v daný okamžik přeposílat buď a nebo b.
- V obou případech bude přicházející tok v jednom z koncových uzlů pouze 1.
- Např. pošle-li uzel W symbol a, uzel T_1 obdrží dvakrát a (tok velikosti 1), a uzel T_2 obdrží a i b (tok 2).

Kódování

Situace se ale změní, povolíme-li uzlu W, aby *kódoval* přicházející informaci. Zvolme jednoduchou operaci kódování $+$ (konkrétně si za touto operací můžeme představit XOR).



- Protože velikost správy $a + b$ je 1, uzel W může tuto správu poslat v jednom kroku.
- Uzel T_1 obdrží a a taky $a + b$. Z toho jednoduše určí b jako $b = (a + b) - a$.
- Obdobně pro uzel T_2

Kódovací schema

Kódovací schema určuje pro každý uzel, jak má být vstupní informace kódována.

- Existují efektivní algoritmy pro návrh kódovacího schematu v obecnějších grafech.
- Jednotlivým uzlům přirazujeme **lokální kódovací funkci**
- **Globální kódovací funkce** vyjádřují, jak je informace transformována při přechodu sítí, t.j., jak má přijemce zrekonstruovat úvodní informaci.
- Bylo dokázáno, že pro dosahovaní maximální propustnosti postačují **lineární** kódovací funkce (dvě po sobě jdoucí lineární kombinace tvoří opět lineární kombinaci)
- Z praktického hlediska to znamená, že každý přijemce musí řešit systém lineárních rovnic, kde neznámými jsou původní informace.

Algoritmy

- Polynomiální algoritmy pro vytváření kódovacích schém (LIF, LIFE, LIFE-CYCLE, LIFE*)
- Procházejí graf od zdroje k příjemci
- Konstruují kódovací funkce (vektory) pro každý uzel tak, aby výslední systém obsahoval dostatečný počet lineárně nezávislých rovnic (jinak by neexistovalo jeho řešení a příjemce by nebyl schopen data zrekonstruovat)

Algoritmy II - praktické nevýhody

- kódovací schema musí být vytvořeno před přenosem
- pracují centralizovaně a na statických topologiích
- dojde-li ke změne topologie, musí se schema vypočíst znovu
- pracují se zjednodušeným modelem sítě
  - stejné kapacity linek
  - všechny datové toky jsou stejně velké
  - latence na všech linkách je stejná
  - operace v síti jsou synchronizované
  - ...

Dynamické sítě

Pro reálné sítě je vhodnější použít jiný přístup

$\Rightarrow$ **Náhodnostní kódování**

- využívá hluboké teoretické poznatky z oblasti *network coding*.
- ty ukazují, že znalost topologie není k dosahování maximální prospustnosti pomocí kódování potřebná.

Náhodnostní kódování – myšlenka

- uzly v síti kódují přicházející informace pomocí náhodně vygenerovaných koeficientů (ty se zasílají spolu s daty)
- při průchodu sítí se nám opět ,,nabalují" lineární kombinace, tentokrát ale náhodné
- aby přijímající mohl úspěšne dekódovat informace, potřebuje ,,nasbírat" dostatečný počet lineárně nezávislých kombinací
- kódujeme-li nad algebraickým polem vhodné velikosti, příjemce bude s vysokou pravděpodobností schopen dekódovat informace.
- neformálně: ve výsledku to funguje

Kontrolní otázka

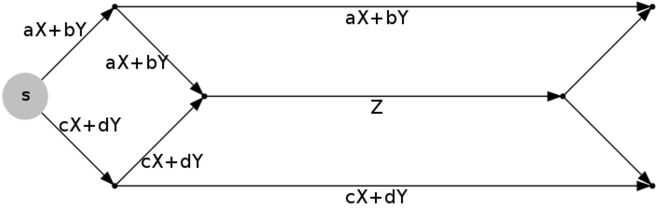
Jaký počet lineárních kombinací je **dostatečný**?

Kontrolní otázka

Jaký počet lineárních kombinací je **dostatečný**?

Každá lineární kombinace určuje jednu rovnici ve výsledném systému rovnic. Abychom mohli dekódovat, potřebujeme alespoň tolik rovnic, kolik jednotlivých bloků informace jsme obdrželi (obrázek na následujícím slajdu).

Random Linear Network Coding



$Z = e(aX + bY) + f(cX + dY) = (ea + fc)X + (eb + fd)Y$

Každý z příjemců řeší systém rovnic. Jedna z rovnic je v obou případech Z, druhou tvoří buď $aX + bY$ nebo $cX + dY$.

Aplikace I

Kromě dosahování maximální propustnosti se network coding využíva i v jiných oblastech, kde je třeba zvyšovat efektivitu šíření informace.

- Bezdrátové sítě
  - Systémy COPE, MORE
  - Streamování videa s prioritizací vrstev
  - Sítě pro spolupráci mobilních zařízení
  - Úprava TCP pro použití na ztrátových linkách
  - ...
- Peer-to-peer sítě
  - Poskytování obsahu velkému počtu uživatelů – systé Avalanche
  - Live Streaming

Aplikace II

- Distribuované úložiště
  - Navrhování robustních schémat
  - Snižování objemu dat přenášených v systému - Wuala
- Network Coding a GPU
  - Snaha o zrychlení kódovacích operací pomocí GPU, resp. spojeného CPU-GPU kódování
- ...