Nejkratší cesty v grafu

Zdeněk Hanzálek
hanzalek@fel.cvut.cz

ČVUT FEL Katedra řídicí techniky

29. března 2011

Formulace problému

a) Nejkratší cesta v grafu (Shortest Path)

- **Instance:** Orientovaný graf G, váhy $c : E(G) \rightarrow \mathbb{R}$ a dva vrcholy $s, t \in V(G)$.
- **Cíl:** Nalézt nejkratší $s - t$−cestu P, neboli tu s minimální délkou $c(E(P))$, nebo rozhodnout, že t není dosažitelný z s.

Další problémy hledající nejkratší orientovanou cestu:

b) z výchozího vrcholu s do každého vrcholu grafu (**Shortest Path Tree - SPT**)

c) z každého vrcholu grafu do cílového vrcholu t

d) mezi všemi uspořádanými dvojicemi uzlů (**All Pairs Shortest Path**)

Problém a) bývá řešen pomocí algoritmů pro b),c) nebo d), algoritmus s lepší časovou složitostí není znám (pro konkrétní instanci lze u speciálních grafů ukončit výpočet při dosažení vrcholu t).
Problém c) lze snadno převést na problém b) obrácením orientace hran.

Příbuzné úlohy

- Hledání **nejdelších cest** cest můžeme převést na hledání nejkratších cest obrácením znamének u délek všech hran. Tím jsme převedli hledání maxima na hledání minima.
- Někdy je třeba hledat nejkratší cesty v grafu, ve kterém jsou **ohodnoceny vrcholy**, nebo vrcholy i hrany a délka cesty je definována jako součet délek všech vrcholů i hran na cestě. Lze převést na hledání v grafu s ohodnocenými hranami:
  - každý vrchol v původního grafu **nahradíme dvojicí vrcholů** v_1 a v_2, spojíme je hranou o délce rovné původní hodnotě vrcholu v
  - hrany, které končily ve vrcholu v přesměrujeme do v_1
  - hrany, které vycházely z vrcholu v budou vycházet z vrcholu v_2

Záporné hrany a záporné cykly

Pro **orientované grafy**:

- připouštíme záporné délky hran
- pokud graf obsahuje cyklus se zápornou délkou, pak jde o **NP-obtížné** problémy
- omezíme se na instance, které nemají cyklus se zápornou délkou

Pro **neorientované grafy** provedeme převod na grafy orientované, ale omezíme se na instance s **nezápornými délkami hran**:

- každá neorientovaná hrana spojující vrcholy v a w je převedena na dvě orientované hrany (v,w) a (w,v)
- pro záporné neorientované hrany by podobným postupem vznikl cyklus záporné délky

Základní fakta - sled a cesta

Věta - existence nejkratší cesty

Jestliže v grafu existuje nějaká cesta z vrcholu s do vrcholu t, pak v něm existuje i nejkratší cesta z s do t.

Pozn: **pro sledy podobná věta neplatí** - obsahuje-li graf cyklus se zápornou délkou, pak ke každému sledu jdoucímu tímto cyklem lze nalézt sled kratší tak, že budeme dostatečně dlouho procházet tento cyklus.

Délka cesty je součet délek hran tvořících cestu.
Vzdálenost $l(s,t)$ **z vrcholu** s **do vrcholu** t definujeme jako délku nejkratší cesty z s do t.

Věta - nejkratší sled a cesta

Jestliže v grafu není **cyklus se zápornou nebo nulovou délkou**, pak každý nejkratší sled z s do t je nejkratší cestou z s do t.
Jestliže v grafu není **cyklus se zápornou délkou**, pak každý nejkratší sled z s do t obsahuje nejkratší cestu z s do t a délka této cesty je stejná.

Základní fakta - trojúhelníková nerovnost

Věta - trojúhelníková nerovnost

Jestliže graf neobsahuje cyklus se zápornou délkou, pak **pro všechny trojice vrcholů** i, j, k splňují **vzdálenosti** mezi nimi tuto nerovnost: $l(i,j) \leq l(i,k) + l(k,j)$.

Důsledky: nechť $c(i,j)$ **je délka hrany** z vrcholu i do vrcholu j, potom jestliže graf neobsahuje cyklus se zápornou délkou pak platí: $l(i,j) \leq c(i,j)$, $l(i,j) \leq l(i,k) + c(k,j)$ a $l(i,j) \leq c(i,k) + l(k,j)$

Věta - nejkratší cesta se skládá z nejkratších cest

Předpokládáme, že graf neobsahuje cyklus se zápornou délkou. Jestliže nejkratší cesta z i do j prochází přes vrchol k, pak úsek cesty z i do k je nejkratší cestou z i do k a podobně úsek cesty z k do j je nejkratší cestou z k do j a navíc platí $l(i,j) = l(i,k) + l(k,j)$.

Důsledek: (**Bellmanova rovnice**) jestliže graf neobsahuje cyklus se zápornou délkou platí: $l(i,j) \leq \min_{k \neq j} \{l(i,k) + c(k,j)\}$

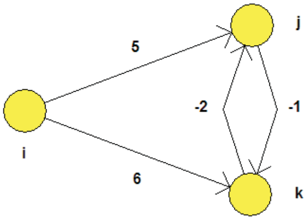
Základní fakta - cyklus se zápornou délkou

Algoritmy které uvedeme spoléhají na platnost výše uvedených vět (graf neobsahuje cyklus se zápornou délkou).

- (1) Jejich rychlost je založena na tom, že se **nestarají o opakování vrcholů v cestě** (nerozlišují nejkratší cestu a nejkratší sled) .
- (2) Pokud graf obsahuje cyklus se zápornou délkou, **nelze (1) použít**, neboť nejkratší sled vůbec nemusí existovat (pak je nalezení nejkratší cesty NP-obtížnou úlohou).

Příklad: graf v němž je záporný cyklus - v důsledku toho neplatí trojúhelníková nerovnost - složením dvou nejkratších cest vznikl sled obsahující záporný cyklus.

$$\begin{array}{rcllcl} l(i,j) & \leq & l(i,k) & + & l(k,j) \\ 6-2 & \leq & (5-1) & + & (-2) \\ 4 & \leq & 2 & \ldots & \text{spor} \end{array}$$



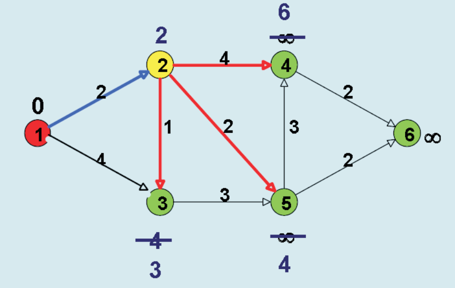
Dijkstrův algoritmus [1959] - nezáporné délky hran

Vstup: Orient. graf G, váhy $c : E(G) \rightarrow \mathbb{R}_0^+$ a vrchol $s \in V(G)$.
Výstup: Vektory l a p. Pro každý $v \in V(G)$ je $l(v)$ délkou nejkratší cesty z vrcholu s a $p(v)$ je předposlední vrchol na této cestě. Pokud v není dostupný z s, pak $l(v) = \infty$ a $p(v)$ není definován.

$l(s) := 0$; $l(v) := \infty$ pro $v \neq s$; $R := \emptyset$;
while $R \neq V(G)$ **do**
  Nalezni $v \in V(G) \setminus R$ takový, že $l(v) = \min_{w \in V(G) \setminus R} l(w)$;
  $R := R \cup \{v\}$;
  `// spočti hodnoty` $l(w)$ `pro vrcholy na rozhraní množiny` R
  for $w \in V(G) \setminus R$ pro které $(v, w) \in E(G)$ **do**
    if $l(w) > l(v) + c(v, w)$ **then**
      $l(w) := l(v) + c(v, w)$; $p(w) := v$;
    end
  end
end

Správnost Dijkstrova algoritmu

Intuitivní vysvětlení správnosti: postupně rozšiřujeme množinu R o **nejvhodnějšího kandidáta** v. V tomto okamžiku je hodnota $l(v)$ již **definitivní**. Hodnoty $l(v) \in R$ nelze zlepšit vedením cesty přes $w \in V(G) \setminus \{R \cup v\}$, jelikož $l(w) \geq l(v)$ a délky všech cest z w do v jsou nezáporné.



Dú: jak se liší Dijkstrův alg. pro SPT a Primův alg. pro výpočet minimální kostry (**Minimum Spanning Tree - MST**)?
Nalezněte co nejmenší příklad s odlišným SPT a MST.

Využití dodatečné informace

Pokud nás zajímá nejkratší cesta z vrcholu s **pouze do jednoho cílového vrcholu** t, pak lze Dijkstrův alg. ukončit jakmile odebereme tento vrchol z množiny R. Pro rychlejší hledání lze navíc použít **Dijkstrův alg. s heuristikou**:

- před výpočtem Dijkstrova algoritmu pro každý vrchol $v \in V(G)$ nastavíme $h(v)$ jako dolní odhad vzdálenosti z vrcholu v do cílového vrcholu t. Přitom pro libovolné dva vrcholy $v_1, v_2 \in V(G)$ musí platit $c(v_1, v_2) \geq h(v_1) - h(v_2)$ (například pro silniční síť lze použít vzdálenost od t vzdušnou čarou - pak tuto podmínku není potřeba ověřovat, jelikož vyplývá z geometrických vlastností prostoru)
- uvnitř Dijkstrova algoritmu volíme vrchol $v \in V(G) \setminus R$ takový, že $l(v) = \min_{w \in V(G) \setminus R} \{l(w) + h(w)\}$

Časová náročnost

- nejrychlejší známý algoritmus pro **SPT** s nezápornými hranami
- časová náročnost algoritmu je $O(n^2)$, respektive s využitím prioritní fronty $O(m + nlogn)$

Algoritmy s linární časovou složitostí řešící specifické problémy:

- pro planární grafy - Henzinger [1997]
- neorientované grafy s celočíselnými nezápornými váhami - Thorup [1999]

Bellman-Fordův algoritmus [1958] (Moore [1959])

Vstup: Orientovaný graf G bez záporných cyklů
váhy $c : E(G) \rightarrow \mathbb{R}$ a vrchol $s \in V(G)$.
Výstup: Vektory l a p. Pro každý $v \in V(G)$ je $l(v)$ délkou nejkratší cesty z vrcholu s a $p(v)$ je předposlední vrchol na této cestě. Pokud v není dostupný z s, pak $l(v) = \infty$ a $p(v)$ není definován.

$l(s) := 0$; $l(v) := \infty$ pro $v \neq s$;
for $i := 1$ **to** $n - 1$ **do**
    for pro každou hranu $(v, t) \in E(G)$ **do**
        if $l(t) > l(v) + c(v, t)$ **then**
            $l(t) := l(v) + c(v, t)$; $p(t) := v$;
        end
    end
end

Časová náročnost Bellman-Fordova algoritmu a záporné cykly

Toto je nejlepší známý algoritmus pro **SPT** bez záporných cyklů.
Časová náročnost algoritmu je $O(nm)$.
Všimněme si, že každá cesta má maximálně $n-1$ hran a že do každého dosažitelného vrcholu t vede hrana.
Bellman-Fordův algoritmus dokáže detekovat cykly záporné délky, ale existují efektivnější metody na detekci záporných cyklů [Cherkassky&Goldberg 1999].

Floydův algoritmus [1962] (Warshall [1962])

Vstup: Orient. graf G bez záporných cyklů a váhy $c : E(G) \rightarrow \mathbb{R}$.

Výstup: Čtvercové matice l a p. Prvek l_{ij} je délkou nejkratší cesty z vrcholu i do vrcholu j. Prvek p_{ij} je indexem předposledního vrcholu na této cestě (pokud existuje).

$l_{ij} := c((i,j))$ pro všechny $(i,j) \in E(G)$;
$l_{ij} := \infty$ pro všechny $(i,j) \notin E(G)$ kde $i \neq j$;
$l_{ii} := 0$ pro všechna i;
$p_{ij} := i$ pro všechny (i,j);
for $k := 1$ **to** n **do**   `// pro každý vrchol` k `ověř zda zlepšuje` l_{ij}
    for $i := 1$ **to** n **do**
        for $j := 1$ **to** n **do**
            if $l_{ij} > l_{ik} + l_{kj}$ **then**
                $l_{ij} := l_{ik} + l_{kj}$; $p_{ij} := p_{kj}$;
            end
        end
    end
end

Floydův algoritmus

- incializuje matici l^0 vzdálenostmi bez "prostředníků"
- počítá posloupnost matic $l^0, l^1, l^2, \cdots, l^k, \cdots, l^n$ kde platí:

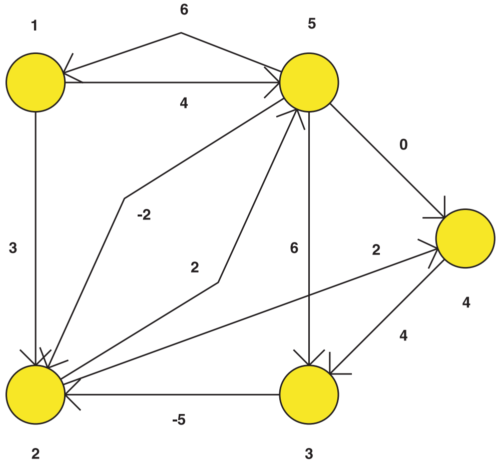
  l_{ij}^k je délka takové nejkratší cesty z i do j, že kromně vrcholů i, j cesta prochází pouze přes prostředníky z množiny $\{1, 2, \cdots k\}$

- známe-li matici l^{k-1}, můžeme snadno vypočítat matici l^k:

$$l_{ij}^k = min\{l_{ij}^{k-1}, l_{ik}^{k-1} + l_{kj}^{k-1}\}$$

Toto je nejlepší známý algoritmus pro **All Pairs Shortest Path** problém bez záporných cyklů.
Časová náročnost algoritmu je $O(n^3)$.
Graf obsahuje **cyklus záporné délky** právě když existuje i takové, že $l_{ii} < 0$.
Drobnou modifikací Floydova algoritmu ($l_{ii}^0 = \infty$) lze nalézt (nezáporný) **cyklus o minimální délce** - viz následující příklad.

Floydův algoritmus - příklad



$$l^0 = \begin{pmatrix} \infty & 3 & \infty & \infty & 4 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & -5 & \infty & \infty & \infty \\ \infty & \infty & 4 & \infty & \infty \\ 6 & -2 & 6 & 0 & \infty \end{pmatrix}$$

$$l = \begin{pmatrix} 10 & 2 & 8 & 4 & 4 \\ 8 & 0 & 6 & 2 & 2 \\ 3 & -5 & 1 & -3 & -3 \\ 7 & -1 & 4 & 1 & 1 \\ 6 & -2 & 4 & 0 & 0 \end{pmatrix}$$

$$p = \begin{pmatrix} 5 & 5 & 4 & 5 & 1 \\ 5 & 5 & 4 & 2 & 2 \\ 5 & 3 & 4 & 2 & 2 \\ 5 & 3 & 4 & 2 & 2 \\ 5 & 5 & 4 & 5 & 2 \end{pmatrix}$$

Příklad SPT1a: Odměřování vody (stavový diagram)

Jste na břehu jezera, máte k dispozici třílitrovou a pětilitrovou nádobu. Na počátku jsou obě nádoby prázdné a vaším úkolem je nabrat do větší nádoby přesně čtyři litry vody. Další pomocné nádoby nemáte, poměřování vody v obou nádobách provádět nesmíte.

a) Reprezentujte tuto úlohu grafem.

b) Zaveďte vhodné váhy v grafu a formulací úlohy nejkratších cest nalezněte řešení

- s nejmenším počtem přelití,
- s nejmenším množstvím manipulované vody,
- s nejmenším množstvím vylité vody.

c) Při některých operacích je potřeba zvýšená opatrnost - například při dolévání, kdy se jedna nádoba zcela naplní, ale druhá se nevyprázdní. Najděte způsob, který má nejmenší počet takových přelití.

d) Je možné odměřit 5 litrů vody pomocí 4-litrové a 6-litrové nádoby?

Příklad SPT1b: Vojáci na mostě (stavový diagram)

S využitím úlohy nejkratších cest zformulujte:
Čtyři vojáci mají jednu baterku a vracejí se přes most na jehož přechod potřebují 1,2,5, nebo 9 časových jednotek. Vždy jdou maximálně dva a pohybují se rychlostí toho pomalejšího ze dvojice. Jelikož potřebují baterku, tak jeden z vojáků ji musí přenést zpět. Rozhodněte, zda se všichni vojáci mohou přemístit přes most dříve než za 18 časových jednotek.

Příklad SPT2a: Umístění požární stanice (využití matice nejkratších cest)

S využitím úlohy nejkratších cest zformulujte:
Mějme silniční síť města.
a) Hledáme jedno nejvhodnější místo pro požární stanici tak, aby její vzdálenost od nejvzdálenějšího místa ve městě byla co nejmenší.
b) Jak se problém změní (ztíží), pokud bychom hledali umístění dvou požárních stanic?
c) Jak se problém změní (ztíží), pokud bychom hledali optimální počet a umístění požárních stanic tak, aby byla garantována dojezdová doba?

Příklad SPT2b: Umístění skladu (využití matice nejkratších cest)

S využitím úlohy nejkratších cest zformulujte:
Mějme silniční síť regionu. Hledáme jedno nejvhodnější místo pro sklad, ze kterého má být zásobeno n spotřebitelů, kteří týdně odebírají $q_1, \ldots, q_n$ hmotnostních jednotek produktu. U každého ze spotřebitelů se v zanedbatelné vzdálenosti nachází jedno vhodné místo pro sklad.
Předpokládáme, že ke každému spotřebiteli musí být vypraveno jedno auto zvlášť s náklady danými součinem vzdálenosti a přepravované hmotnosti.
Cílem je minimalizovat týdenní přepravní náklady.

Příklad SPT3: Nejspolehlivější spojení (ilustruje vlastnosti uzavřeného polookruhu)

Ve sdělovací síti je spolehlivost fungování přímé linky mezi místy i a j dána pravděpodobností $p(i,j)$. Předpokládejme, že poruchy na jednotlivých linkách jsou vzájemně nezávislé. Pak spolehlivost spojení mezi místy s a t je rovna součinu pravděpodobností $p(i,j)$ všech dílčích linek tvořících toto spojení. Nalezněte nejspolehlivější spojení mezi dvěma místy s a t.

Příklad SPT4: Přejezd nákladního tahače (ilustruje záporné hrany)

Mějme n měst s připravenými návěsy a tahač návěsů. Pro každou dvojici měst (i, j) známe náklady $c(i, j)$ na přejezd tahače z města i do města j. Dále o některých dvojicích (i, j) víme, že z města i do města j lze dopravit návěs a že za jeho převezení dostaneme zaplaceno $d(i, j)$. Naším úkolem je dostat tahač z města s do města t finančně co nejvýhodnějším způsobem bez ohledu na čas.

Nejkratší cesty v grafu - Shrnutí

- Jednoduchý optimalizační problém s řadou aplikací
  - OSPF (Open Shortest Path First) protokol pro směrování v sítích používá Dijkstrův alg.
  - hledání nejkratší-nejlevnější-nejrychlejší trasy v mapě
  - SPT1 - stavový diagram
  - SPT2 - využití Floydova algoritmu pro hledání centra grafu
  - SPT3 - uzavřené polookruhy
  - SPT4 - záporné hrany
  - ...
- Základ pro řadu dalších optimalizačních problémů
  - rozvrhování s relacemi následností
  - ...

Literatura

Jiří Demel.
Grafy a jejich aplikace.
Academia, 2002.

B. H. Korte and Jens Vygen.
Combinatorial Optimization: Theory and Algorithms.
Springer, fourth edition, 2008.