Minimální kostry

Štěpán Simsa

Abstrakt. Cílem příspěvku je seznámit s tématem minimálních koster, konkrétně s teoretickými základy, algoritmy a jejich analýzou.

Problém. (Minimální kostra) Je zadaný graf G s množinou vrcholů V a hranami E ohodnocený vahami $w : E \to \mathbb{R}$. Naším cílem je najít souvislý podgraf grafu G (na stejné množině vrcholů), který má ze všech takových grafů minimální součet vah vybraných hran.

Cvičení. Minimální kostra nemusí existovat jedna. Víme-li ale navíc, že jsou váhy hran po dvou různé, už jednoznačná je.

Teorie

Definice. Buď $T \subseteq G$ nějaká kostra grafu G. Pak:

- $T[x, y]$ bude značit cestu v T, která spojuje x a y. (Cestou opět míníme množinu hran.)
- $T[e] := T[x, y]$ pro hranu $e = xy$. Této cestě budeme říkat *cesta pokrytá hranou* e.
- Hrana $e' \in E \setminus T$ je *lehká vzhledem k* $T \equiv \exists e \in T[e'] : w(e') < w(e)$. Ostatním hranám neležícím v kostře budeme říkat *těžké.*

Věta. *Kostra T je minimální $\Leftrightarrow$ neexistuje hrana lehká vzhledem k T.*

Definice. Pro kostru T a hrany $e \in T, e' \notin T$ zaveďme $swap(T, e, e') := T - e + e'$.

Pozorování. Pokud $e' \notin T$ a $e \in T[e']$, je $swap(T, e, e')$ opět kostra. Stačí si uvědomit, že přidáním e' do T vznikne kružnice a vynecháním libovolné hrany z této kružnice získáme opět kostru.

Lemma. (O swapování) *Máme-li libovolné kostry T a T', pak lze z T dostat T' konečným počtem operací ($swap$).*

Lemma. (Monotónní o swapování) *Je-li T kostra, k níž neexistují žádné lehké hrany, a T' libovolná kostra, pak lze od T k T' přejít posloupností swapů, při které váha kostry neklesá.*

Všechny tradiční algoritmy na hledání minimální kostry lze popsat jako speciální případy následujícího meta-algoritmu. Formulujeme ho pro případ, kdy jsou všechny váhy hran navzájem různé.

Algoritmus. (Červenomodrý meta-algoritmus)

1. Na počátku jsou všechny hrany bezbarvé.
2. Dokud to lze, použijeme jedno z následujících pravidel:
   Modré pravidlo: Vyber řez (podmnožinu hran, jejichž odebráním graf přestane být souvislý) takový, že jeho nejlehčí hrana není modrá, a obarvi ji na modro.
   Červené pravidlo: Vyber cyklus takový, že jeho nejtěžší hrana není červená, a obarvi ji na červeno.

Lemma. (Modré lemma) *Je-li libovolná hrana e algoritmem kdykoliv obarvena na modro, pak $e \in T_{\min}$.*

Lemma. (Červené lemma) *Je-li libovolná hrana e algoritmem kdykoliv obarvena na červeno, pak $e \notin T_{\min}$.*

Lemma. (Bezbarvé lemma) *Pokud existuje nějaká neobarvená hrana, lze ještě použít některé z pravidel.*

Věta. *Pro Červenomodrý meta-algoritmus spuštěný na libovolném grafu s hranami lineárně uspořádanými podle vah platí:*

1. *Vždy se zastaví.*
2. *Po zastavení jsou všechny hrany obarvené.*
3. *Modře obarvené hrany tvoří minimální kostru.*

Klasické algoritmy

Algoritmus. (Kruskalův neboli Hladový)

1. Setřídíme hrany podle vah vzestupně.
2. Začneme s prázdnou kostrou (každý vrchol je v samostatné komponentě souvislosti).
3. Bereme hrany ve vzestupném pořadí. Pro každou hranu e se podíváme, zda spojuje dvě různé komponenty – pokud ano, přidáme ji do kostry (obarvíme ji na modro), jinak ji zahodíme (obarvíme ji na červeno).

Tvrzení. *Algoritmus má časovou složitost $O(m \log n)$. Máme-li hrany předem setřízené nebo můžeme-li je setřídit v lineárním čase, algoritmus lze implementovat v čase $O(m\alpha(m,n))$, kde $\alpha(m,n)$ je obdoba inverzní Ackermannovy funkce.*

Algoritmus. (Borůvkův) Opět si budeme pěstovat modrý les, avšak tentokrát jej budeme rozšiřovat ve fázích. V jedné fázi nalezneme ke každému stromečku nejlevnější incidentní hranu (hranu sdílející právě jeden vrchol s tímto stromečkem) a

všechny tyto nalezené hrany naráz přidáme (aplikujeme několik modrých pravidel najednou). Pokud jsou všechny váhy různé, cyklus tím nevznikne.

Tvrzení. *Algoritmus má časovou složitost* $O(m \log n)$.

Algoritmus. (Jarníkův) Jarníkův algoritmus je podobný Borůvkovi, ale s tím rozdílem, že nenecháme růst celý les, ale jen jeden modrý strom. V každém okamžiku nalezneme nejlevnější hranu vedoucí mezi stromem a zbytkem grafu a přidáme ji ke stromu (modré pravidlo); hrany vedoucí uvnitř stromu průběžně zahazujeme (červené pravidlo). Kroky opakujeme, dokud se strom nerozroste přes všechny vrcholy.

Tvrzení. *Algoritmus má časovou složitost* $O(m \log n)$. *S Fibonacciho haldou se dá naimplementovat v* $O(m + n \log n)$.

Důsledek. *Máme lineární algoritmus pro grafy s* $m \geq c \cdot n \log(n)$ *pro libovolnou konstantu* c.

Příklady

Příklad 1. Nechť úplný graf K_n na množině vrcholů $\{1, 2, \ldots, n\}$ má hranu $\{i, j\}$ ohodnocenou číslem

(i) $\max(i, j)$,
(ii) $i + j$.

Nalezněte minimální kostru a spočtěte její váhu.

Příklad 2. Nalezněte jednoduchý algoritmus pro výpočet minimální kostry v grafech ohodnocených vahami $\{1, \ldots k\}$ se složitostí $\mathrm{O}(mk)$ nebo dokonce $\mathrm{O}(m + nk)$.

Příklad 3. Uvažme v rovině n-bodovou množinu V. Definujme ohodnocení hran úplného grafu na V: ohodnocením hrany $\{x, y\}$ bude vzdálenost bodů x a y.

(i) Ukažte, že maximální stupeň vrcholu v libovolné minimální kostře je nejvýš 6.
(ii) Ukažte, že existuje minimální kostra, jejíž hrany se navzájem nekříží.

Příklad 4. Nechť V je množina n bodů ve čtverci o straně 1. Dokažte, že existuje kostra na V s celkovou délkou hran nejvýš $10\sqrt{n}$ (uvažujeme všechny kostry úplného grafu na V a váha hrany $\{x, y\}$ je euklidovská vzdálenost bodů x a y). Konstantu 10 lze podstatně zlepšit, ale nejlepší možná hodnota není známa (jaký nejlepší odhad se podaří najít vám?).

Příklad 5. Buď $G = (V, E)$ graf, w nezáporné ohodnocení jeho hran.

(i) Každá množina $E' \subseteq E$ navzájem disjunktních hran se nazývá *párování* v grafu G. Označme $\nu_w(G)$ maximální možnou hodnotu $w(E')$ pro párování $E' \subseteq E$. Hladový algoritmus pro maximální párování funguje podobně jako Kruskalův algoritmus pro maximální kostru, jenže zamítá hrany protínající některou dříve vybranou hranu. Ukažte, že takový algoritmus vždy najde párování váhy nejméně $\nu_w(G)/2$.

(ii) Ukažte, že odhad v (i) nelze zlepšit, tj. že pro libovolné $\alpha > \frac{1}{2}$ existuje zadání, pro nějž hladový algoritmus najde párování váhy menší než $\alpha\nu_w(G)$.

Rychlejší algoritmy

Definice. Kontrakce hrany spojí dva konce hrany do jednoho vrcholu, tuto hranu smaže, ale ostatní zachová (přičemž mohou vznikat paralelní hrany a smyčky).

Cvičení. Rozmyslete si, že Červenomodrý meta-algoritmus funguje i pro multigrafy.

Pozorování. Pokud $F \subseteq \mathrm{M}ST(G)$ (kde $\mathrm{M}ST(G)$ je minimální kostra grafu G), G' je graf vzniklý z G kontrakcí podél hran z F, pak kostra grafu G, která vznikne z $\mathrm{M}ST(G')$ zpětným expandováním kontrahovaných vrcholů, je $\mathrm{M}ST(G)$. Pokud kontrakcí vzniknou smyčky, můžeme je ihned odstraňovat; pokud paralelní hrany, ponecháme z nich vždy tu nejlehčí. To nás vede k následujícímu algoritmu:

Algoritmus. (Kontrahující Borůvkův algoritmus)

1. Ke každému vrcholu najdeme nejlevnější incidentní hranu – dostaneme množinu hran $F \subseteq E$.
2. Graf kontrahujeme podle F následovně:
3. Prohledáme do šířky graf (V, F) a přiřadíme každému vrcholu číslo komponenty, v níž se nachází.
4. Přečíslujeme hrany v G podle čísel komponent.
5. Odstraníme násobné hrany:
6. Setřídíme hrany lexikograficky přihrádkovým tříděním (násobné hrany jsou nyní pospolu).
7. Projdeme posloupnost hran a z každého úseku multihran odstraníme všechny až na nejlevnější hranu. Také odstraníme smyčky.
8. Pokud stále máme netriviální graf, opakujeme předchozí kroky.
9. Vrátíme jako minimální kostru všechny hrany, které se v průběhu algoritmu dostaly do F.

Tvrzení. *Algoritmus má pro rovinné grafy*[1] *časovou složitost* $O(n)$.

Algoritmus. (Kombinace Jarníkova a Borůvkova algoritmu)

1. Provedeme $\log\log n$ cyklů upraveného Borůvkova algoritmu s kontrahováním hran popsaného výše.
2. Pokračujeme Jarníkovým algoritmem (implementovaným Fibonacciho haldou).

Tvrzení. *Algoritmus má časovou složitost* $O(m \log\log n)$.

[1] Platí i silnější tvrzení, že algoritmus je lineární pro třídy grafů uzavřené na *minory*, kde *minor* vznikne z původního grafu jako podgraf, kterému kromě odebírání hran a vrcholů zároveň povolujeme kontrahovat hrany.

Algoritmus. (Jarníkův s omezenou velikostí haldy)

1. Opakujeme, dokud máme netriviální G (s alespoň jednou hranou):
2. $t \leftarrow |V(G)|$.
3. Zvolíme $k \leftarrow 2^{2m/t}$ (velikost haldy).
4. $T \leftarrow \emptyset$.
5. Opakujeme, dokud existují vrcholy mimo T:
6. Najdeme vrchol v_0 mimo T.
7. Spustíme Jarníkův alg. (s Fib. haldou) pro celý graf od v_0. Zastavíme ho, pokud:
8. $|H| \geq k$ (byla překročena velikost haldy) nebo
9. $H = \emptyset$ (došli sousedé) nebo
10. do T jsme přidali hranu oboustranně incidentní s hranami v T (připojili jsme novou podkostru k nějaké už nalezené).
11. Kontrahujeme G podle podkoster nalezených v T.

Tvrzení. *Časová složitost tohoto algoritmu je $O(m\beta(m,n))$, kde*

$$\beta(m,n) = \min\{i : \log^{(i)} n < m/n\}.$$

Všimněte si, že $\beta(m,n)$ je méně než $\log^* n$, kde $\log^* n$ (*iterovaný logaritmus*) je definován jako 0 pro $n \leq 1$ a jako $1 + \log^*(\log n)$ pro $n > 1$.

Důsledek. *Algoritmus je lineární pro grafy s $m \geq n \log^{(k)} n$ pro libovolnou konstantu k.*

Další výsledky

- $O(m\alpha(m,n))$, kde $\alpha(m,n)$ je obdoba inverzní Ackermannovy funkce definovaná podobně, jako je $\beta(m,n)$ obdobou $\log^*$.
- $O(\mathcal{T}(m,n))$, kde $\mathcal{T}(m,n)$ je hloubka optimálního rozhodovacího stromu pro nalezení minimální kostry v grafech s patřičným počtem hran a vrcholů. Jelikož každý deterministický algoritmus založený na porovnávání vah lze popsat rozhodovacím stromem, je tento algoritmus zaručeně optimální. Jen bohužel nevíme, jak optimální stromy vypadají, takže je stále otevřeno, zda lze minimální kostru nalézt v lineárním čase. Nicméně tento algoritmus pracuje i na Pointer Machine, pročež víme, že pokud je lineární složitosti možné dosáhnout, není k tomu potřeba výpočetní síla RAMu.[2]
- $O(m)$ pro grafy s celočíselnými vahami (na RAMu).
- $O(m)$, pokud už máme hrany setříděné podle vah: jelikož víme, že záleží jen na uspořádání, můžeme váhy přečíslovat na $1, \ldots, m$ a použít předchozí algoritmus.
- $O(m)$ průměrně: randomizovaný algoritmus, který pro libovolný vstupní graf doběhne v očekávaném lineárním čase.

[2] V tomto modelu dovolujeme přístupy na libovolné místo v paměti v konstantním čase.

- Na zjištění, zda je zadaná kostra minimální, stačí $O(m)$ porovnání a dokonce lze v lineárním čase zjistit, která to jsou. Z toho ostatně vychází předchozí randomizovaný algoritmus.

Návody

1.

(i) Sporem dokažte, že n-tý vrchol musí být spojený právě s jedním z předchozích vrcholů, pak indukce.

(ii) Analogicky, ale vrchol n musí být konkrétně spojený s vrcholem 1.

2. Dovolte si používat jen hrany s vahami $\leq k'$.

3.

(i) Pro vrchol v stupně ≥ 7 existují hrany $\{v, u_1\}, \{v, u_2\}$ svírající úhel $< 60°$. Jednu z nich lze nahradit hranou $\{u_1, u_2\}$.

(ii) Vezměte libovolné křížící se hrany $\{a, b\}$ a $\{c, d\}$. Odeberte je a uvědomte si, že po odebrání je právě jedna dvojice z vrcholů a, b, c, d ve stejné komponentě.

4. Jedno řešení: Rozdělte na síť $\sqrt{n} \times \sqrt{n}$.
Druhé řešení: Dokažte, že některé dva body jsou vzdálené $O(\sqrt{n})$, jeden vymažte a indukcí.

5.

(i) Buď E_M maximální párování, E_H párování nalezené hladovým algoritmem a každé hraně $e \in E_M$ přiřaďme první z hran E_H, která ji protíná.

(ii) Stačí graf na 4 vrcholech. Zařiďte, aby algoritmus kvůli vybrání největší hrany zahodil dvě jen o epsilon menší.

Literatura a zdroje

[1] Martin Mareš: *Krajinou grafových algoritmů*, *http://http://mj.ucw.cz/vyuka/ga/*

[2] Jiří Matoušek, Jaroslav Nešetřil: *Kapitoly z Diskrétní matematiky*, Nakladatelství Karolinum, 2009