NI-LOM semestrální úloha

Michal Dvořák

2. února 2022

V této práci se věnujeme problému maximálního toku resp. minimálního řezu v síti. Cílem je porovnat různé formulace a strategie řešení tohoto problému pomocí lineárního programování.

1 Pojmy

Definice 1. *Síť* je uspořádaná pětice (V, E, s, t, c) přičemž (V, E) je orientovaný graf (se smyčkama) a $s, t \in V$ dva vyznačené vrcholy - *zdroj* a *stok*. $c\colon E \to \mathbb{R}^+$ je funkce přiřazující každé hraně e její kapacitu $c(e)$.

Definice 2. *Tok v síti* (V, E, s, t, c) je funkce $f\colon E \to \mathbb{R}^+$ splňující:

1. $\forall e \in E\colon 0 \le f(e) \le c(e)$
2. $\forall v \in V \setminus \{s, t\}\colon \sum_{(u,v)\in E} f((u, v)) = \sum_{(v,w)\in E} f((v, w))$

Velikost toku f je

$$w(f) = \sum_{(s,v)\in E} f((s, v)) - \sum_{(v,s)\in E} f((v, s))$$

Tok f je *maximální* pokud pro každý tok f' je $w(f) \ge w(f')$.

Dá se ukázat, že každá síť skutečně má maximální tok i když jsou kapacity libovolná reálná čísla. My se však omezíme na kapacity racionální (resp. celočíselné). Problém nalezení maximálního toku řeší několik standardních algoritmů. Pro srovnání s formulacemi a řešeními pomocí LP použijeme Dinitzův algoritmus. Pro detailní popis Dinitzova algoritmu odkazujeme čtenáře na [1]. Teoretická doba běhu Dinitzova algoritmu pro graf s n vrcholy a m hranami je $O(n^2m)$.

Poznamenejme, že problém maximálního toku je úzce spjat s problémem minimálního řezu. Problém nalezení minimálního řezu v síti je nalezení množiny $A \subseteq V$ takové, že $s \in A, t \notin A$ a $\sum_{u\in A, v\notin A} c((u, v))$ je minimální. Na problém minimálního řezu se dá pohlížet jako na duál maximálního toku. I bez teorie lineárního programování se dá ukázat, že velikost minimálního řezu je rovna velikosti maximálního toku v každé síti.

2 Formulace pomocí lineárního programování

Problém maximálního toku lze přirozeně formulovat pomocí lineárního programu

$$\begin{array}{lll}
\max & \sum_{(s,v)\in E} f((s, v)) - \sum_{(v,s)\in E} f((v, s)) & \\
\text{za podmínek} & \sum_{(u,v)\in E} f((u, v)) - \sum_{(v,w)\in E} f((v, w)) = 0 & \forall v \in V \setminus \{s, t\} \\
 & 0 \le f(e) \le c(e) & \forall e \in E
\end{array} \tag{1}$$

s proměnnými $f((u,v))$ pro každou hranu $(u,v) \in E$.

Duál programu 1 je

$$\begin{array}{lll}
\min & \sum_{e \in E} y_e c(e) & \\
\text{za podmínek} & y_e \geq 1 & e = (s,t) \in E \\
& y_e \geq -1 & e = (t,s) \in E \\
& y_e + y_v \geq 1 & \forall e = (s,v) \in E, v \notin \{s,t\} \\
& -y_u + y_e \geq -1 & \forall e = (u,s) \in E, u \notin \{s,t\} \\
& y_v + y_e \geq 0 & \forall e = (t,v) \in E, v \notin \{s,t\} \\
& -y_u + y_e \geq 0 & \forall e = (u,t) \in E, u \notin \{s,t\} \\
& y_v - y_u + y_e \geq 0 & \forall e = (u,v) \in E, u \neq s \wedge v \neq t \forall e \in E \\
& y_e \geq 0 & \forall e \in E
\end{array} \tag{2}$$

s proměnnými y_e za každou hranu a y_v za každý vrchol $v \in V \setminus \{s,t\}$.

Tento duál by měl v jistém smyslu odpovídat relaxaci minimálního řezu. Na první pohled není zřejmé, jak z proměnných y nějaký řez sestavit. Podívejme se na jinou formulaci problému maximálního toku (resp. minimálního řezu). Označme $\mathcal{P}$ množinu všech s-t cest v síti.

$$\begin{array}{lll}
\max & \sum_{p \in \mathcal{P}} x_p & \\
\text{za podmínek} & \sum_{p \in \mathcal{P}, e \in p} x_p \leq c(e) & \forall e \in E \\
& x_p \geq 0 & \forall p \in \mathcal{P}
\end{array} \tag{3}$$

s proměnnými x_p za každou s-t cestu. Problémem je, že s-t cest je obecně v grafu až exponencielně mnoho a tak program není možné v polynomiálním čase ani napsat. Podívejme se ale na duál.

$$\begin{array}{lll}
\min & \sum_{e \in E} y_e c_e & \\
\text{za podmínek} & \sum_{e \in p} y_e \geq 1 & \forall p \in \mathcal{P} \\
& y_e \geq 0 & \forall e \in E
\end{array} \tag{4}$$

Ten má proměnnou y_e za každou hranu $e \in E$ ale podmínku za každou cestu $p \in \mathcal{P}$. Myšlenka bude přidávat tyto podmínky do řešiče postupně. Lze nahlédnout, že polynomiálně mnoho podmínek bude stačit pro nalezení optima, protože i Dinitzův algoritmus uvažuje jen polynomiálně mnoho cest z s do t protože pracuje v polynomiálním čase. Ohodnocení hran y_e můžeme interpretovat jako váhové ohodnocení hran a podmínky za každou cestu jsou splněny všechny pokud vzdálenost[1] s od t je alespoň 1. Na hledání nejkratší cesty z s do t lze použít například Dijkstrův algoritmus. Pro rozbor Dijkstrova algoritma opět odkazujeme do [1]. Je použita implementace s binární haldou. Teoretická doba běhu této varianty Dijkstrova algoritmu je $O((n+m)\log n)$.[2]

3 Experimenty

Experimenty lze nalézt v jupyter-notebooku ve složce `visualization`

4 Závěr

Nic.

[1] vzdálenost $d(u,v)$ mezi dvěma vrcholy $u, v \in V$ v (orientovaném) grafu je délka nejkratší (orientované) cesty z u do v (případně $+\infty$ pokud žádná (orientovaná) cesta neexistuje)

[2] Lze zlepšit na teoretických $O(m + n\log n)$ s Fibonacciho haldou, ale v praxi je to spíš pomalejší.

Reference

[1] Mares, M. *Průvodce labyrintem algoritmů.* CZ. NIC, zspo, 2021.