12. Pravděpodobnostní algoritmus na řezy

Nahlédněme alespoň jednou kapitolou do světa pravděpodobnostních algoritmů. Není totiž výjimkou, že s pomocí generátoru náhodných čísel vyřešíme některé grafové problémy daleko snáze a často také efektivněji, než to dovedeme deterministicky. Pravděpodobnostní přístup si předvedeme na Kargerově-Steinově algoritmu [1] pro hledání minimálního řezu v neohodnoceném neorientovaném grafu. Připomeňme, že s deterministickými algoritmy jsme zatím dosáhli časové složitosti $\mathcal{O}(n^{5/3}m)$ pomocí toků nebo $\mathcal{O}(nm)$ Nagamochiho-Ibarakiho algoritmem.

Náhodné kontrakce

Uvažujme nejdříve následující algoritmus, který náhodně vybírá hrany a kontrahuje je, dokud počet vrcholů neklesne na ℓ. (Konkrétní hodnotu ℓ zvolíme později.)

Algoritmus Contract(G_0, ℓ):

1. $G \leftarrow G_0$.
2. Dokud $n > \ell$:
3. Vybereme hranu $e \in E$ rovnoměrně náhodně.
4. $G \leftarrow G/e$ (kontrahujeme hranu e, smyčky odstraňujeme, paralelní hrany ponecháme).
5. Vrátíme jako výsledek graf G.

Jaká je pravděpodobnost, že výsledný graf G má stejně velký minimální řez jako zadaný graf G_0? Všimněme si nejprve, že každý řez v grafu G/e je i řezem v grafu G (až na přeznačení hran při kontrakci, ale předpokládejme, že hrany mají nějaké identifikátory, které kontrakce zachovává). Podobně je-li v grafu G řez neobsahující hranu e, odpovídá mu stejně velký řez v G/e. Velikost minimálního řezu tedy kontrakcí nikdy neklesne – může pouze stoupnout, pokud skontrahujeme hranu ležící ve všech minimálních řezech,

Zvolíme nyní pevně jeden z minimálních řezů C v zadaném grafu G_0 a označíme k jeho velikost. Pokud algoritmus ani jednou nevybere hranu ležící v tomto řezu, velikost minimálního řezu v grafu G bude rovněž rovna k. Jaká je pravděpodobnost, že se tak stane?

Označme G_i stav grafu G před i-tým průchodem cyklem a n_i a m_i počet jeho vrcholů a hran. Zřejmě $n_i = n - i + 1$ (každou kontrakcí přijdeme o jeden vrchol). Navíc každý vrchol má stupeň alespoň k, jelikož jinak by triviální řez okolo tohoto vrcholu byl menší než minimální řez. Proto platí $m_i \geq kn_i/2$. Hranu ležící v řezu C tedy vybereme s pravděpodobností nejvýše $k/m_i \leq k/(kn_i/2) = 2/n_i = 2/(n-i+1)$. Všechny hrany z řezu C proto postoupí do výsledného grafu G s pravděpodobností

$$p \geq \prod_{i=1}^{n-\ell} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-\ell} \frac{n-i-1}{n-i+1} =$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \ldots \cdot \frac{\ell+1}{\ell+3} \cdot \frac{\ell}{\ell+2} \cdot \frac{\ell-1}{\ell+1} = \frac{\ell \cdot (\ell-1)}{n \cdot (n-1)}.$$

Ještě musíme ošetřit případ, kdy bychom hranu řezu smazali, protože se mezitím stala smyčkou. Ovšem smyčky vznikají pouze z hran paralelních s právě kontrahovanou hranou. Jelikož v libovolném svazku paralelních hran buďto všechny leží v C, nebo ani jedna neleží, museli jsme v takovém případě řez C rozbít už dříve. Odhad pravděpodobnosti to tedy neovlivní.

Můžeme tedy zvolit pevně ℓ, spustit na zadaný graf proceduru Contract a ve vzniklém konstantně velkém grafu pak nalézt minimální řez hrubou silou (to je obzvláště snadné pro $\ell = 2$ – tehdy stačí vzít všechny zbylé hrany). Takový algoritmus nalezne minimální řez s pravděpodobností alespoň c/n^2, kde c je konstanta závislá na ℓ.

Nabízí se otázka, k čemu je dobrý algoritmus, který vydá správný výsledek s pravděpodobností na řádu $1/n^2$. To opravdu není mnoho, ale stejně jako mnoho jiných randomizovaných algoritmů i tento můžeme iterovat: výpočet zopakujeme K-krát a použijeme nejmenší z nalezených řezů. Ten už bude minimální s pravděpodobností

$$P_K \geq 1 - (1 - c/n^2)^K \geq 1 - e^{-cK/n^2}.$$

(Druhá nerovnost platí díky tomu, že $e^{-x} \geq 1 - x$ pro všechna $x \geq 0$.) Pokud tedy nastavíme počet opakování K na $\Omega(n^2)$, můžeme tím pravděpodobnost chyby stlačit pod libovolnou konstantu, pro $K = \Omega(n^2 \log n)$ pod převrácenou hodnotu libovolného polynomu v n a pro $K = \Omega(n^3)$ už bude dokonce exponenciálně malá.

Implementace

Odbočme na chvíli k implementačním záležitostem. Jak reprezentovat graf, abychom stihli rychle provádět kontrakce? Bude nám stačit obyčejná matice sousednosti, v níž pro každou dvojici vrcholů budeme udržovat, kolik paralelních hran mezi těmito vrcholy vede. Pokud chceme kontrahovat hranu uv, stačí projít všechny hrany vedoucí (řekněme) z vrcholu u a zařadit je k vrcholu v. To zvládneme v čase $\mathcal{O}(n)$ na jednu kontrakci.

Pro náhodný výběr hrany budeme udržovat pole stupňů vrcholů, vybereme náhodně vrchol v s pravděpodobností úměrnou stupni a poté projitím příslušného řádku matice sousednosti druhý vrchol v s pravděpodobností úměrnou počtu hran mezi u a v. To opět trvá čas $\mathcal{O}(n)$.

Procedura Contract tedy pracuje v čase $\mathcal{O}((n-\ell)\cdot n)$ a celý K-krát ziterovaný algoritmus v $\mathcal{O}(Kn^2)$. Pokud se spokojíme s převráceně polynomiální pravděpodobností chyby, nalezneme minimální řez v čase $\mathcal{O}(n^4 \log n)$.

Kargerův-Steinův algoritmus

Předchozí prostinký algoritmus můžeme ještě podstatně vylepšit. Všimněme si, že během kontrahování hran pravděpodobnost toho, že vybereme „špatnou“ hranu ležící v minimálním řezu, postupně rostla z počátečních $2/n$ až po obrovské $2/3$ v poslední iteraci (pro $\ell = 2$). Pomůže tedy zastavit kontrahování dříve a přejít na spolehlivější způsob hledání řezu.

Pokud zvolíme $\ell = \lceil n/\sqrt{2}+1 \rceil$, pak řez C přežije kontrahování s pravděpodobností alespoň

$$\frac{\ell \cdot (\ell-1)}{n \cdot (n-1)} \geq \frac{(n/\sqrt{2}+1) \cdot n/\sqrt{2}}{n \cdot (n-1)} = \frac{n/\sqrt{2}+1}{\sqrt{2} \cdot (n-1)} = \frac{n+\sqrt{2}}{2 \cdot (n-1)} \geq \frac{1}{2}.$$

Jako onen spolehlivější způsob hledání řezu následně zavoláme stejný algoritmus rekurzivně, přičemž jak kontrakci, tak rekurzi provedeme dvakrát a z obou nalezených řezů vybereme ten menší, čímž pravděpodobnost chyby snížíme.

Hotový algoritmus bude vypadat následovně:

Algoritmus $\textsc{MinCut}(G)$:

1. Pokud $n < 7$, najdeme minimální řez hrubou silou.
2. $\ell \leftarrow \lceil n/\sqrt{2}+1 \rceil$.⟨1⟩
3. $C_1 \leftarrow \textsc{MinCut}(\textsc{Contract}(G, \ell))$.
4. $C_2 \leftarrow \textsc{MinCut}(\textsc{Contract}(G, \ell))$.
5. Vrátíme menší z řezů C_1, C_2.

Jakou bude mít tento algoritmus časovou složitost? Nejprve odhadněme hloubku rekurze: v každém kroku se velikost vstupu zmenší přibližně $\sqrt{2}$-krát, takže strom rekurze bude mít hloubku $\mathcal{O}(\log n)$. Na i-té hladině zpracováváme 2^i podproblémů velikosti $n/2^{i/2}$. Při výpočtu každého podproblému voláme dvakrát proceduru Contract, která spotřebuje čas $\mathcal{O}((n/2^{i/2})^2) = \mathcal{O}(n^2/2^i)$. Součet přes celou hladinu tedy činí $\mathcal{O}(n^2)$ a přes všechny hladiny $\mathcal{O}(n^2 \log n)$. Oproti původnímu kontrakčnímu algoritmu jsme si tedy moc nepohoršili.

Zbývá spočítat, s jakou pravděpodobností algoritmus skutečně nalezne minimální řez. Označme p_i pravděpodobnost, že algoritmus na i-té hladině stromu rekurze (počítáno od nejhlubší, nulté hladiny) vydá správný výsledek příslušného podproblému. Jistě je $p_0 = 1$ a platí rekurence $p_i \geq 1 - (1 - 1/2 \cdot p_{i-1})^2$. Uvažujme posloupnost g_i, pro kterou jsou tyto nerovnosti splněny jako rovnosti, a všimněme si, že $p_i \geq g_i$. Víme tedy, že $g_0 = 1$ a $g_i = 1 - (1 - 1/2 \cdot g_{i-1})^2 = g_{i-1} - g_{i-1}^2/4$.

Nyní zavedeme substituci $z_i = 4/g_i - 1$, čili $g_i = 4/(z_i + 1)$, a tak získáme novou rekurenci pro z_i:

$$\frac{4}{z_i + 1} = \frac{4}{z_{i-1}+1} - \frac{4}{(z_{i-1}+1)^2},$$

kterou už můžeme snadno upravovat:

$$\frac{1}{z_i+1} = \frac{z_{i-1}}{(z_{i-1}+1)^2},$$

$$z_i + 1 = \frac{z_{i-1}^2 + 2z_{i-1} + 1}{z_{i-1}},$$

$$z_i + 1 = z_{i-1} + 2 + \frac{1}{z_{i-1}}.$$

⟨1⟩ To je méně než n, kdykoliv $n \geq 7$.

Jelikož $z_0 = 3$, a tím pádem $z_i \geq 3$ pro všechna i, získáme z poslední rovnosti vztah $z_i \leq z_{i-1} + 2$, a tudíž $z_i \leq 2i + 3$. Zpětnou substitucí obdržíme $g_i \geq 4/(2i+4)$, tedy $p_i \geq g_i = \Omega(1/i)$.

Nyní si stačí vzpomenout, že hloubka rekurze činí $\mathcal{O}(\log n)$, a ihned získáme odhad pro pravděpodobnost správného výsledku $\Omega(1/\log n)$. Náš algoritmus tedy stačí ziterovat $\mathcal{O}(\log^2 n)$-krát, abychom pravděpodobnost chyby stlačili pod převrácenou hodnotu polynomu. Dokázali jsme následující větu:

Věta: Iterováním algoritmu MinCut nalezneme minimální řez v neohodnoceném neorientovaném grafu v čase $\mathcal{O}(n^2 \log^3 n)$ s pravděpodobností chyby $\mathcal{O}(1/n^c)$ pro libovolnou konstantu $c > 0$.

Cvičení

1. Zvolte lepší reprezentaci grafu, abyste prostorovou složitost snížili z $\Theta(n^2)$ na $\mathcal{O}(m)$. Může se tím zmenšit časová složitost?
2. Dokažte, že z našeho rozboru pravděpodobnosti chyby plyne, že v každém grafu je nejvýše $\mathcal{O}(n^2)$ minimálních řezů.
3. Ukažte, jak algoritmus upravit pro grafy s ohodnocenými hranami.

Literatura

[1] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996.