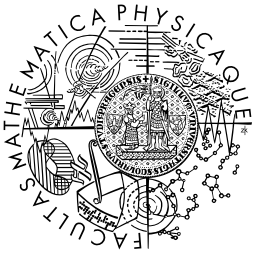


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Ondřej Houška

Vícekriteriální metody dělení grafů

Katedra numerické matematiky

Vedoucí bakalářské práce: prof. Ing. Miroslav Tůma, CSc.

Studijní program: Matematika

Studijní obor: Obecná matematika

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Děkuji svému vedoucímu bakalářské práce prof. Ing. Miroslavovi Tůmovi, CSc. za vstřícnost, podporu, za vysvětlení nejasností a za jeho čas.

Název práce: Vícekriteriální metody dělení grafů

Autor: Ondřej Houška

Katedra: Katedra numerické matematiky

Vedoucí bakalářské práce: prof. Ing. Miroslav Tůma, CSc., Katedra numerické matematiky

Abstrakt: Práce se zabývá dělením grafů a aplikací dělení grafů v paralelních algoritmech pro řešení velkých soustav lineárních rovnic s řídkou maticí. Problém dělení grafů je důkladně vyložen a jsou zde popsány standardní metody dělení grafů. Aplikační část se zaměřuje především na předpodmíněnou metodu sdružených gradientů. Jako předpodmínění se používá varianta neúplné Choleského faktorizace založená na odvrhovacím parametru. V práci je vysvětlena role dělení grafů v paralelní variantě této metody a zabývám se v ní vyvažováním zátěže na jednotlivých procesorech.

Klíčová slova: dělení grafů, paralelní výpočty, řešení soustav rovnic, metoda konjugovaných gradientů, řídké matice

Title: Multicriteria graph partitioning

Author: Ondřej Houška

Department: Department of Numerical Mathematics

Supervisor: prof. Ing. Miroslav Tůma, CSc., Department of Numerical Mathematics

Abstract: The thesis is about graph partitioning and applications of graph partitioning in parallel algorithms for solving big sparse linear equations. The problem of graph partitioning is thorougly described and standard graph partitioning algorithms are explained. The application part is focusing on the Conjugate Gradient method preconditioned by a variant of incomplete Cholesky factorization based on drop tolerance. The role of graph partitioning in the problem decomposition is described and a load balancing problem is studied.

Keywords: graph partitioning, parallel computations, solving linear systems, Conjugate Gradient method, sparse matrices

Obsah

Úvod

Pojem *graf* má v matematice dva významy. První a známější je graf ve smyslu grafického znázornění funkce. Druhý význam se objevuje ve 20. století v souvislosti s vytvářením schémat sítí. Graf v tomto smyslu je tvořen množinou vrcholů a množinou hran, které je propojují. Tato bakalářská práce se zabývá dělením grafů ve druhém významu. Zadaný graf se snažím rozdělit na několik přibližně stejných částí tak, aby byl minimalizován počet propojení jednotlivých částí a aby byla splněna případná další kritéria, jako například požadavek na souvislost částí nebo optimální vyvážení částí pro paralelní výpočty.

Dělení grafů se používá například při návrhu elektronických obvodů, kde je potřeba elektronické součástky, reprezentované vrcholy grafu, rozdělit na několik plošných spojů tak, aby se minimalizoval počet potřebných propojení, neboli počet hran vedoucích mezi různými částmi.

Další významnou aplikací dělení grafů je problém rozdělení rozsáhlých výpočtů mezi několik procesorů počítače nebo mezi počítače zapojené do výpočetní sítě tak, aby byl každý procesor/počítač přibližně stejně vytížen a aby se minimalizovalo množství komunikace mezi různými procesory/počítači.

Jiná aplikace se objevuje v řešení velkých soustav lineárních rovnic metodou Gaussovy eliminace. Významnou úlohu u těchto soustav hraje řídkost matice. Matici nazýváme řídkou, pokud je většina jejích prvků nulových. Řídkost umožňuje značné úspory výpočetních nákladů, ovšem při provádění Gaussovy eliminace se může postupně řídkost matice zhoršovat. Vhodným přeuspořádáním původní matice lze tento efekt zeslabit. Dělení grafů se používá jako jedna z metod hledajících nové uspořádání matice, které by omezilo počet dodatečně vzniklých nenulových prvků vznikajících v průběhu eliminace.

V první části své bakalářské práce formuluji problém dělení grafů podrobněji a popíši základní metody dělení grafů. Ve druhé části se budu zabývat využitím dělení grafů v numerické lineární algebře. Konkrétně se zaměřím na řešení velké soustavy lineárních rovnic pomocí předpodmíněné metody konjugovaných gradientů a bude mě zajímat, jak tuto úlohu nejlépe rozdělit.

Velké soustavy lineárních rovnic s řídkou maticí soustavy je potřeba v aplikacích řešit velmi často. Pro řešení takových soustav je potřeba využívat metody, které co nejvíce využívají řídkost matice soustavy. I při použití efektivních metod je někdy nutné výpočet paralelizovat. Na rozdělení úlohy se používá dělení grafů, protože dokáže zohlednit strukturu řídkosti matice soustavy.

1. Základní pojmy a značení

Transpozici matice A značím A^T, počet prvků množiny X značím $|X|$, dolní celou část čísla $r \in \mathbb{R}$ značím $\lfloor r \rfloor$, horní celou část čísla $r \in \mathbb{R}$ značím $\lceil r \rceil$, $\mathbb{N} = \{1, 2, 3, \ldots\}$.

Definice 1. *Nechť X je množina, $k \in \mathbb{N} \cup \{0\}$. Množinu jejích k-prvkových podmnožin budu značit*

$$\binom{X}{k} = \Big\{ P \subset X \;\Big|\; |P| = k \Big\}.$$

Definice 2 (neorientovaný graf)**.** *Nechť V je nějaká množina vrcholů a $E \subset \binom{V}{2}$ nějaká množina neorientovaných hran. Pak $G = (V, E)$ nazývám* neorientovaný graf.

Definice 3 (matice sousednosti)**.** *Nechť $G = (V, E)$ je neorientovaný graf, $n = |V|$, $V = \{v_1, \ldots, v_n\}$.* Matice sousednosti *grafu G je matice*

$$\mathrm{Soused}(G) = (s_{ij})_{i,j=1}^{n},$$

$$\textit{kde } s_{ij} = \begin{cases} 1, & \{v_i, v_j\} \in E \\ 0, & \{v_i, v_j\} \notin E. \end{cases}$$

Definice 4 (stupeň vrcholu)**.** *Nechť $G = (V, E)$ je neorientovaný graf, $v \in V$. Stupněm vrcholu v rozumím počet vrcholů, které jsou s vrcholem v spojeny hranou, tj.*

$$\deg(v) = \Big| \{\, u \in V \mid \exists v \in V : \{u, v\} \in E \,\} \Big|.$$

Definice 5. *Nechť $n \in \mathbb{N}$. Matice $A \in \mathbb{R}^{n \times n}$ se nazývá pozitivně definitní, pokud $\forall x \in \mathbb{R}^n \setminus \{0\}\colon\ x^T A\, x > 0$.*

2. Problém dělení grafů

Snažíme se rozdělit neorientovaný graf na k částí tak, aby

1. části byly přibližně stejně velké
2. množství propojení jednotlivých částí bylo co nejmenší.

Definice 6 (rovnoměrné dělení grafu)**.** *Nechť $G = (V, E)$ je neorientovaný graf, $k \in \mathbb{N}$. Disjunktní rozklad $V_1, \ldots, V_k$ množiny vrcholů V se nazývá dělení grafu G na k částí. Dělení se nazývá rovnoměrné, pokud splňuje $|V_1| = \ldots = |V_k| = \frac{|V|}{k}$.*

Poznámka. Rovnoměrné dělení grafu G na k částí existuje $\Leftrightarrow$ k dělí $|V(G)|$.

Poznámka. Dělení grafu na dvě části se nazývá *bisekce*.

Definice 7 (hranový separátor dělení)**.** *Nechť $G = (V, E)$ je neorientovaný graf, $P = (P_1, \ldots, P_k)$ dělení grafu G.* Hranový separátor dělení P *je množina hran*

$$\begin{aligned} H(P) &= \{\, e \in E \mid \textit{koncové body } e \textit{ jsou v různých oblastech} \,\} \\ &= \Big\{\, \{u, v\} \in E \;\Big|\; \exists\, i, j \in \{1, \ldots, k\}, i \neq j : u \in P_i \;\&\; v \in P_j \,\Big\}. \end{aligned}$$

Problém 1. *Najít pro zadaný graf G rovnoměrné dělení P tak, aby počet hran v $H(P)$ byl co nejmenší.*

Příklad. Ať $G = (V, E)$, $V = \{1, 2, 3, 4\}$, $E = \{\, \{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\} \,\}$. Pak existují dvě[1] optimální bisekce $(\{1, 2\}, \{3, 4\})$ a $(\{1, 4\}, \{2, 3\})$. Obě optimální bisekce mají velikost hranového separátoru 2. Kromě těchto bisekcí existuje už jen bisekce $(\{1, 3\}, \{2, 4\})$, která má hranový separátor celé E.

Optimálních dělení v problému 1 tedy může být více.

2.1 Počet rovnoměrných dělení grafu

Nechť máme graf G a $V_1, \ldots, V_k$ je rovnoměrné dělení grafu G na k částí. Pak každá část V_i je složena z $\frac{|V(G)|}{k}$ vrcholů. Označme $n = |V(G)|$. Počet možností, jak vybrat první část V_1 je $\binom{n}{\frac{n}{k}}$, protože máme k dispozici n vrcholů a vybíráme $\frac{n}{k}$, přičemž nezáleží na jejich pořadí. Počet možností, jak vybrat druhou část V_2 je $\binom{n-\frac{n}{k}}{\frac{n}{k}}$, protože máme o $\frac{n}{k}$ méně vrcholů na výběr. Obecně u výběru části V_i budeme mít $\binom{n-(i-1)\frac{n}{k}}{\frac{n}{k}}$, protože je už vybráno $i - 1$ částí a každá má $\frac{n}{k}$ vrcholů. Celkem je počet rovnoměrných dělení

$$\prod_{i=1}^{k} \binom{n - (i-1)\frac{n}{k}}{\frac{n}{k}} = \prod_{i=0}^{k-1} \binom{n - i \cdot \frac{n}{k}}{\frac{n}{k}}$$

Vzhledem k tomu, že mají dělení stejný počet neuspořádaných vrcholů, můžeme považovat dvě dělení $V_1, \ldots, V_k$ a $V_1^*, \ldots, V_k^*$ za ekvivalentní, pokud existuje permutace $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ splňující $V_{\pi(i)} = V_i^*$. Protože takových permutací je $k!$, počet rovnoměrných a vzájemně neekvivalentních dělení grafu G na k částí je

$$\frac{1}{k!} \prod_{i=0}^{k-1} \binom{n - i \cdot \frac{n}{k}}{\frac{n}{k}}.$$

[1] až na pořadí částí

Tvrzení 1 (dolní odhad binomického koeficientu)**.** *Pro každé* $n \in \mathbb{N}$, $s \in \{1, \ldots, n\}$ *platí*

$$\binom{n}{s} \geq \left(\frac{n}{s}\right)^s.$$

Důkaz.

$$\binom{n}{s} = \frac{n!}{s!(n-s)!} = \frac{n(n-1)\ldots(n-s+1)}{s!}\frac{(n-s)!}{(n-s)!} =$$
$$= \frac{n(n-1)\ldots(n-s+1)}{s(s-1)\ldots(s-s+1)} = \prod_{i=0}^{s-1}\frac{n-i}{s-i} \geq \prod_{i=0}^{s-1}\frac{n}{s} = \left(\frac{n}{s}\right)^s.$$

Zbývá dokázat použitou nerovnost. Zřejmě platí

$$n \geq s$$
$$\Rightarrow \forall i \in \{0, \ldots, s-1\}\colon\ in \geq is.$$

To je ekvivalentní s nerovností

$$-is \geq -in,$$

přičtením členu ns získáme $(n-i)s \geq n(s-i)$. Vydělením kladným členem $s-i$ a kladným členem s získáme

$$\frac{n-i}{s-i} \geq \frac{n}{s} \quad \text{pro každé } i \in \{0, \ldots, s-1\}.$$

Věta 2 (odhad počtu rovnoměrných dělení)**.** *Nechť* $k, n \in \mathbb{N}$, $k \leq n$, k *dělí* n. *Pak rovnoměrných dělení grafu s* n *vrcholy na* k *částí je alespoň* $(k!)^{\frac{n}{k}-1}$.

Důkaz. Z tvrzení 1 plyne odhad

$$\binom{n - i\cdot\frac{n}{k}}{\frac{n}{k}} \geq \left(\frac{n - i\cdot\frac{n}{k}}{\frac{n}{k}}\right)^{\frac{n}{k}} = (k-i)^{\frac{n}{k}}.$$

Z nezápornosti činitelů pak

$$\frac{1}{k!}\prod_{i=0}^{k-1}\binom{n - i\cdot\frac{n}{k}}{\frac{n}{k}} \geq \frac{1}{k!}\prod_{i=0}^{k-1}(k-i)^{\frac{n}{k}} = \frac{1}{k!}\left[\prod_{i=0}^{k-1}(k-i)\right]^{\frac{n}{k}} = \frac{1}{k!}\left[k!\right]^{\frac{n}{k}} = (k!)^{\frac{n}{k}-1}.$$

Poznámka. V knize [1] se dokazuje i opačný odhad

$$\binom{n}{s} \leq \left(\frac{e\cdot n}{s}\right)^s, \text{ kde } e \text{ je Eulerovo číslo.}$$

Obdobným postupem bychom získali horní odhad pro počet rovnoměrných dělení $(e\cdot k!)^{\frac{n}{k}-1}$.

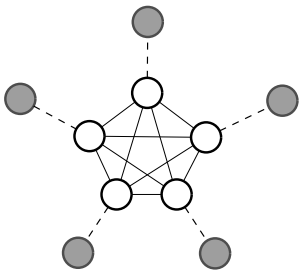
Nejjednodušší algoritmus na dělení grafů, tedy vyzkoušení všech možných rovnoměrných dělení, je tedy možné použít jen pro velmi malé grafy a velmi malé k. Konkrétně na mém osobním počítači tak lze v rozumném čase pro $k = 2$ dělit grafy o cca 22 vrcholech a pro $k = 3$ o cca 14 vrcholech.[2]

[2]implementace v programovacím jazyce Python, požadavek na délku běhu algoritmu do 2 s.

2.2 Dělení grafu a souvislost částí

Připomínám, že neorientovaný graf se nazývá souvislý, vede-li mezi každými dvěma vrcholy cesta (viz [1]). Grafy, které je třeba dělit jsou typicky souvislé,[3] přirozenou otázkou je, jestli vždy existuje optimální dělení na k částí, které má obě části *souvislé*.

Příklad. Následující graf o 10 vrcholech má (až na záměnu částí) jedinou rovnoměrnou bisekci minimalizující $|H(V_1, V_2)|$. Toto dělení obsahuje nesouvislou část.



2.3 Slabší kritéria rovnoměrnosti dělení

V aplikacích není potřeba mít dělení přesně rovnoměrné, stačí, když velikost separátoru bude malá. Pojem rovnoměrného dělení můžeme nepatrně zeslabit následujícím způsobem.

Definice 8 (skoro rovnoměrné dělení)**.** *Nechť G je graf, $k \in \mathbb{N}$. Označme $n = |V(G)|$. Dělení $P = (P_1, \ldots, P_k)$ nazýváme* skoro rovnoměrné*, pokud*

$$\forall i \in \{1, \ldots, k\} \colon |V_i| = \left\lfloor \frac{n}{k} \right\rfloor \text{ nebo } |V_i| = \left\lceil \frac{n}{k} \right\rceil.$$

Příklad. Skoro rovnoměrná dělení grafu o 10 vrcholech na tři části V_1, V_2, V_3 mjí velikosti částí $(|V_1|, |V_2|, |V_3|) \in \{\, (3,3,4),\ (3,4,3),\ (4,3,3) \,\}$.

V praxi se používá zejména následující pojem.

Definice 9. *Nechť $G = (V, E)$ je neorientovaný graf a $V_1, \ldots, V_k$ je dělení G. Označme $average = \frac{|V|}{k}$ a definujme*

$$\text{Imbalance}(V_1, \ldots, V_k) = \frac{1}{average} \max_{i=1,\ldots,k} (|V_i| - average) = \frac{\max_{i=1,\ldots,k} |V_i|}{\frac{|V|}{k}} - 1.$$

V některých aplikacích mají různé vrcholy grafu různou váhu, přičemž je třeba rozdělit graf na části tak, aby celkové váhy částí byly přibližně stejné. Pokud máme dány váhy vrcholů $w : V \to \mathbb{R}^+$, tak můžeme pro $V' \subset V$ označit $\tilde{w}(V') = \sum_{v \in V'} w(v)$. Minulá definice pak bude mít tvar

$$\text{Imbalance}(V_1, \ldots, V_k) = \frac{\max_{i=1,\ldots,k} \tilde{w}(V_i)}{\tilde{w}(V)/k} - 1.$$

[3] U nesouvislého grafu existuje bisekce (ne nutně rovnoměrná), která má prázdný hranový separátor.

Problém 2. *Pro zadaný graf G a danou toleranci $\varepsilon > 0$ najít dělení $V_1, \ldots, V_k$ tak, aby*

- $\text{Imbalance}(V_1, \ldots, V_k) \leq \varepsilon$,
- *v hranovém separátoru $H(V_1, \ldots, V_k)$ bylo co nejméně hran.*

2.4 Minimalizace komunikace mezi procesory

Zatím jsme se snažili minimalizovat velikost hranového separátoru. Pokud máme dané váhy hran $a : E \to \mathbb{R}^+$, tak můžeme obecněji velikostí separátoru rozumět

$$vol(H(V_1, \ldots, V_k)) = \sum_{e \in H(V_1, \ldots, V_k)} a(e)$$

a minimalizovat $vol(H(V_1, \ldots, V_k))$. Ovšem v problému minimalizace komunikace mezi procesory se ukazuje, že tento model je nedostatečný. $vol(H)$ totiž představuje *celkovou* komunikaci mezi procesory. Minimalizovat celkový objem komunikace má smysl například pokud se platí za množství přenesených dat, častěji nás zajímá co nejkratší čas komunikace. Čas strávený komunikací ale není přímo úměrný objemu komunikace, ale je to funkce tvaru

$$\frac{\text{objem komunikace}}{v} + c_0,$$

kde v je přenosová rychlost a c_0 představuje *latenci*. Je tedy potřeba zavést penalizační člen p_{ij}, který přičteme, pokud bude $H(V_i, V_j)$ neprázdná. Upravená velikost hranového separátoru pak bude

$$čas(H(V_i, V_j)) = p_{ij} h_{ij} + \sum_{e \in H(V_i, V_j)} \frac{a(e)}{v_{ij}},$$

kde $v_{ij} > 0$, $p_{ij} \geq 0$ a

$$h_{ij} = \begin{cases} 0, H(V_i, V_j) = \emptyset, \\ 1, \text{jinak.} \end{cases}$$

Pokud je $k > 2$, tak je situace složitější. Procesory totiž můžou být propojeny mnoha různými způsoby. Pokud spolu komunikují přes společnou sběrnici, tak

$$čas(H(V_1, \ldots, V_k)) = \sum_{i,j=1,\ldots,k, i<j} čas(H(V_i, V_j)).$$

Pokud naopak má každá dvojice procesorů vlastní komunikační kanál, tak

$$čas(H(V_1, \ldots, V_k)) = \max_{i,j=1,\ldots,k, i<j} čas(H(V_i, V_j)),$$

neboli celková doba komunikace je rovna době komunikace nejvytíženější linky. Typicky je situace kombinací minulých dvou.

Příklad. Několik osobních počítačů spojených přes internet.

2.5 Vrcholový separátor

Zatím jsme se snažili minimalizovat velikost hranového separátoru. Vrcholy v našem modelu značily výpočetní podúlohy, jejich váha $w(v)$ byla výpočetní náročnost a hrany reprezentovaly množství komunikace $a(\{v_1, v_2\})$.

Ovšem často se používá podobná, ale ne zcele ekvivalentní formulace problému rozdělení úlohy na více procesorů. V ní vrcholy představují data, se kterými se ve výpočtu pracuje, přítomnost hrany značí závislost mezi daty. Úkolem je rozdělit graf na přibližně stejně velké části tak, že množství vrcholů, které je nutno sdílet mezi více procesory, je co nejmenší. To odpovídá minimalizaci velikosti *vrcholového* separátoru.

Definice 10 (vrcholový separátor dělení)**.** *Ať G je graf, $P = (V_1, \ldots, V_k)$ jeho dělení. Pak $S \subset V(G)$ je* vrcholový separátor dělení P*, pokud*

$$\forall i, j \in \{1, \ldots, k\},\ i \neq j\colon \textit{každá cesta mezi } V_i \setminus S \textit{ a } V_j \setminus S \textit{ obsahuje vrchol z } S.$$

Vrcholový separátor S dělení P nazveme nezmenšitelný, pokud $\forall v \in S\colon\ S \setminus \{v\}$ není vrcholový separátor dělení P.

Lemma 3 (vztah vrcholového a hranového separátoru)**.** *Nechť $P = (V_1, \ldots, V_k)$ je dělení grafu $G = (V, E)$. Pak každý nezmenšitelný vrcholový separátor S dělení P splňuje*

$$|S| \ \leq\ |H(P)| \ \leq\ |S| \cdot \max_{v \in V} \deg(v).$$

Důkaz. Nechť S je vrcholový separátor dělení P. Pak pro každou hranu $\{v_1, v_2\} \in H(P)$ musí být $v_1 \in S$ nebo $v_2 \in S$, neboť jinak by cesta $(v_1, \{v_1, v_2\}, v_2)$ spojovala různé části grafu.

Označme $C = \{\, \{v_i, v_j\} \in E \mid v_i \in S \vee v_j \in S \,\}$ množinu hran, které mají alespoň jeden koncový vrchol ve vrcholovém separátoru S. Pak zřejmě platí $|C| \leq |S| \cdot \max_{v \in V} \deg(v)$, neboť z každého vrcholu v S může vycházet nejvýše $\max_{v \in V} \deg(v)$ hran. Zároveň dle výše dokázaného je pro každou hranu $e \in H(P)$ alespoň jeden koncový bod v S, a tedy $e \in C$. Tím jsem dokázal nerovnost

$$|H(P)| \leq |C| \leq |S| \cdot \max_{v \in V} \deg(v).$$

Označme S_0 množinu vrcholů z S, které jsou koncovými body $H(P)$. Pak S_0 je vrcholový separátor P. Z nezmenšitelnosti vrcholového separátoru S je potom $S = S_0$. Je-li S prázdná, pak $|S| \leq |H(P)|$ platí triviálně.

Nechť $v_1 \in S$. Vrcholu v_1 přiřadím libovolnou hranu e_1, která z bodu v_1 vychází. Taková hrana existuje, protože $S \subset S_0$. Nechť $s \in \mathbb{N}, s < n$, a nechť mám různým vrcholům $v_1, \ldots, v_s \in S$ přiřazené různé hrany $e_1, \ldots, e_s \in H(P)$. Vrcholu $v_{s+1} \in S \setminus \{v_1, \ldots, v_s\}$ přiřadím libovolnou hranu $e_{s+1} \in H(P) \setminus \{e_1, \ldots, e_s\}$, která vychází z vrcholu v_{s+1}. Taková hrana existuje, neboť kdyby byly všechny hrany vycházející z v_{s+1} již využity, byl by $S \setminus \{v_{s+1}\}$ vrcholový separátor, což je spor s nezmenšitelností S. Podle principu matematické indukce lze tedy každému vrcholu z S přiřadit jinou hranu z $H(P)$. Z toho plyne $|S| \leq |H(P)|$.

3. Metody dělení grafů

3.1 Setrvačná metoda

Setrvačná metoda je jednoduchý zástupce tzv. geometrických metod. Geometrické metody se snaží graf hrubě rozdělit na základě rozmístění vrcholů v prostoru/rovině bez informací o tom, mezi kterými body vede hrana. Potřebujeme tedy navíc znát souřadnice vrcholů $B : V \to \mathbb{R}^d$. Tyto souřadnice máme k dispozici např. při dělení grafů vzniklých z diskretizační sítě v metodě konečných prvků[1]. Tyto grafy mají zároveň velmi uniformní strukturu, která velmi těsně souvisí se souřadnicemi—zahození informace o hranách tedy není taková ztráta.

V setrvačné metodě spočteme těžiště vrcholů grafu a nadrovinu $\mathcal{H}$, podle které rozdělíme vrcholy do dvou skupin.

Popis metody

Nechť $n = |V|$, $V = \{v_1, \ldots, v_n\}$. Nejprve spočteme těžiště vrcholů

$$T = \frac{1}{n} \sum_{k=1}^{n} B(v_k).$$

Označme $\vec{x}(k) = B(v_k) - T$. Snažíme se zvolit osu otáčení tak, aby procházela těžištěm T a aby moment setrvačnosti pevně spojených bodů $v_1, \ldots, v_n$ kolem této osy byl co nejmenší.

K tomu použijeme tenzor setrvačnosti

$$\begin{pmatrix} I_{11}, & I_{12}, & I_{13} \\ I_{21}, & I_{22}, & I_{23} \\ I_{31}, & I_{32}, & I_{33} \end{pmatrix}$$

definovaný pomocí

$$I_{ij} = \begin{cases} \sum_{k=1}^{n} (x_1(k))^2 + (x_2(k))^2 + (x_3(k))^2 - (x_j(k))^2, & i = j, \\ \sum_{k=1}^{n} -x_i(k) x_j(k), & i \neq j. \end{cases}$$

Tenzor setrvačnosti je symetrická matice, podle věty z lineární algebry tedy existuje ortonormální báze $(\vec{o}_1, \ldots, \vec{o}_3)$, ve které má tenzor setrvačnosti tvar

$$\begin{pmatrix} \lambda_1, & 0, & 0 \\ 0, & \lambda_2, & 0 \\ 0, & 0, & \lambda_3 \end{pmatrix}.$$

Pro tenzor setrvačnosti platí, že moment setrvačnosti kolem osy určené jednotkovým vektorem $\vec{o}$ se spočte jako $I\vec{o}$. Z vyjádření v bázi $(\vec{o}_1, \ldots, \vec{o}_3)$ je jasné, že minimální hodnota momentu setrvačnosti se dosahuje pro vlastní vektor příslušný nejmenšímu vlastnímu číslu $\lambda_{\min}$.

[1] Finite Element Method (FEM) - známá metoda numerického řešení parciálních diferenciálních rovnic

Nyní rozdělme body $v_1, \ldots, v_n$ podle nadroviny $\mathcal{H}$, která prochází těžištěm T a je kolmá na spočtenou hlavní osu $\vec{o}_{\min}$. To, v jaké skupině se vrchol v nachází snadno zjistíme podle znaménka skalárního součinu $\vec{x}(k) \cdot \vec{o}_{\min}$. Může se stát, že nějaké vrcholy budou ležet přímo v nadrovině $\mathcal{H}$, pak skalární součin bude nulový a vrcholy rovnoměrně rozdělíme podle nějakého dodatečného kritéria.

Vzhledem k tomu, že nadrovina $\mathcal{H}$ prochází těžištěm, je výsledné dělení skoro rovnoměrné. Pokud jsme ovšem $\vec{o}_{\min}$ spočítali přibližně, je lepší rovnoměrnost dělení zkontrolovat.

Výpočetní náklady

- Těžiště n bodů $b(v_i)$ spočteme v čase $O(n)$.
- Každou složku tenzoru setrvačnosti spočteme v čase $O(n)$, složek je d^2, celý tenzor setrvačnosti tedy spočteme v čase $O(n)$.
- Výpočet hlavní osy setrvačnosti je nejsložitější část algoritmu, ale jsme to schopni udělat v konstantním čase.
- K přidělení vrcholu v do jeho stačí jen spočítat skalární součin, přidělení všech bodů jsme tedy schopni udělat v čase $O(n)$.

Setrvačný algoritmus můžeme tedy používat i pro velký počet vrcholů.

3.2 Metoda šíření

Nechť v_{start} je nějaký vrchol. Pak máme definovanou $d : V \to \mathbb{R}_0^+$, $d(v) = dist(v, v_{\text{start}})$ pro $v \in V$. Budeme nyní z vrcholu v_{start} provádět prohledávání grafu G do šířky Vrcholy, kterými budeme procházet, přidáme do první části dělení V_1. Až bude $|V_1| = n/k$, tak prohledávání zastavíme. V případě bisekce grafu pak položíme $V_2 = V - v_1$ a jsme hotovi V případě $k > 2$ opakujeme stejný algoritmus pro graf s množinou vrcholů $V' = V - V_1$ a s novým startovním vrcholem $v'_{\text{start}} \in V'$; dělit budeme tentokrát jen na $k - 1$ částí.

Volba prvních n/k vrcholů prohledáváním do šířky ze startovního vrcholu v_{start} odpovídá z hlediska teorie grafů volbě $m = n/k$ vrcholů $v_1, \ldots, v_m \in V$ tak, aby bylo co nejmenší $\max\{d(v_1), \ldots, d(v_m)\}$.

Kvalita výsledného dělení závisí hodně na volbě počátečního vrcholu v_{start}. Ideálně by se měl volit startovní vrchol s maximální výstředností.

Definice 11 (výstřednost vrcholu)**.** *Nechť G je graf, $v \in V(G)$. Pak* výstředností *(excentricitou) vrcholu v rozumíme číslo*

$$\epsilon(v) = \max_{u \in V} \text{dist}(v, u).$$

Největší výstřednost v grafu, tedy číslo

$$d = \max_{v \in V} \epsilon(v)$$

se nazývá průměr grafu G*. Vrchol nazveme* okrajový*, pokud je jeho výstřednost rovna průměru grafu.*

Proto, abychom našli okrajové vrcholy, bychom museli spočítat $dist(u, v)$ pro všechny $u, v \in V$. Protože je takový výpočet časově náročný, tak se většinou používají heuristické algoritmy, které pouze hledají vrchol s velmi vysokou výstředností (viz [2]).

Metoda šíření je jednoduchou metodou dělení grafů. Časová náročnost metody je nízká, konkrétně $O(|E|)$. V porovnání s ostatními metodami ale často dává dělení nižší kvality (viz [3]).

3.3 Kernighanův-Linův algoritmus

Popíšu zde základní variantu Kernighanova-Linova algoritmu pro $k = 2$ a $w(v) = 1 \, \forall v \in V$, tj. dělím na dvě části a všechny vrcholy mají stejnou váhu.

Kernighanův-Linův algoritmus předpokládá, že nějaké dělení již máme a snaží se toto dělení vylepšit. Ať tedy $A = V_1, B = V_2$ je rovnoměrné dělení grafu G. Budu uvažovat přemístění jednoho vrcholu z části A do části B a naopak přemístění jednoho vrcholu z části B do části A. Tím, že uvažuji tyto dvě změny společně, bude i nové dělení $(A \setminus \{v_a\}) \cup \{v_b\}, (B \setminus \{v_b\}) \cup \{v_a\}$ rovnoměrné a stačí se zabývat jen změnou velikosti hranového separátoru.

Nechť $v \in V_i$. Označme

$$out(v) = \{ \, u \in V \setminus V_i \mid u \text{ je soused } v \, \},$$

$$in(v) = \{ \, u \in V_i \mid u \text{ je soused } v \, \}.$$

O výhodnosti přesunutí vrcholu v rozhoduje

$$gain(v) = \sum_{u \in out(v)} a(\{v, u\}) - \sum_{u \in in(v)} a(\{v, u\}).$$

V zákl. případě, kdy $a(e) = 1 \, \forall e \in E$ platí

$$gain(v) = |\, out(v)| - |\, in(v)|.$$

Pokud je $gain(v)$ kladný, tak se přesunem v_a do části B zmenší velikost hranového separátoru.

Pokud se zabývám dvěma přesuny naráz, tak je potřeba definovat

$$g(v_a, v_b) = \begin{cases} gain(v_a) + gain(v_b), & \{v_a, v_b\} \notin E, \\ gain(v_a) + gain(v_b) - 2a(\{v_a, v_b\}), & \{v_a, v_b\} \in E, \end{cases}$$

pro $v_a \in A$, $v_b \in B$, protože případná hrana mezi v_a a v_b se před přesunem nachází v $out(v_a)$ i $out(v_b)$, ale i po přesunu bude stále v $H(A, B)$.

Můžeme spočítat $g(v_a, v_b)$ pro všechny $v_a \in A, v_b \in B$, a pak vybrat v_a, v_b maximalizující $g(v_a, v_b)$. Po přesunu v_a a v_b je třeba hodnoty g přepočítat. Problém takového přístupu je, že si musíme pamatovat řádově $O(|V|^2)$ hodnot g.

Jinou variantou algoritmu je si pamatovat pouze hodnoty *gain*. Těch je pouze $O(|V|)$. V prvním kroku algoritmu přesuneme vrchol s maximální hodnotou *gain* a přepočteme hodnoty *gain* u jeho sousedů. Ve druhém kroku přesuneme vrchol z opačné části s maximální hodnotou *gain* a přepočteme hodnoty *gain*. Výhoda této varianty spočívá v tom, že není těžké toto implementovat efektivně. Hodnoty

vrcholy totiž můžeme mít uložené ve dvou prioritních frontách (každá odpovídá jedné části dělení) tak, že jsou uspořádané podle hodnot *gain*. V implementaci prioritní fronty pomocí haldy jsou operace

- odebrání největšího prvku
- přidání nového prvku a jeho zařazení
- změna pořadí prvku v v důsledku změny hodnoty $cost(v)$

proveditelné v čase $O(\log(|V|))$. V jednom kroku metody je potřeba odebrat první vrchol z jedné z prioritních front, zařadit ho s opačnou hodnotou *cost* do druhé prioritní fronty a pak aktualizovat hodnoty *cost* u všech jeho sousedů. Pokud bude $d \in \mathbb{N}$ nějaká malá konstanta, tak ve třídě grafů, ve kterých jsou stupně vrcholů omezeny touto konstantou d je možné provést jeden krok této metody v čase $O((d+2)\log(|V|)) = O(\log(|V|))$.

Zbývá dodat, co bude algoritmus dělat v případě, že hodnoty g nebo *cost* budou všechny záporné. Jednou možností je v tomto případě algoritmus ukončit. V článku [3] se ale doporučuje ve výměnách pokračovat s tím, že se tím možná podaří dostat se z lokálního optima. Protože ale takový krok zhoršuje kvalitu dělení, je potřeba, aby si algoritmus před takovým spekulativním krokem nejlepší dosažené dělení zapamatoval.

Kernighanův-Linův algoritmus je základní heuristika, která se snaží vylepšit již získané dělení. Používá se tedy v kombinaci s jiným algoritmem na dělení grafů. Základní myšlenkou je výpočet zisku $gain(v)$ při přesunu vrcholu v do jiné části. Variant tohoto algoritmu je více, můžou mít různě efektivní implementace a různá ukončovací kritéria.

3.4 Víceúrovňová metoda

Víceúrovňová metoda je významnou technikou v algoritmech dělení grafů. Je založena na tom, že sousední vrcholy původního grafu G sloučíme do skupinek a tak vytvoříme menší/hrubší/zjednodušený graf G'. Graf G' má méně vrcholů, a tak je výpočet jeho dělení zpravidla mnohem rychlejší než u původního grafu G. Výsledné dělení pak promítneme zpět do grafu G.

Víceúrovňová metoda se typicky používá opakovaně, tedy sestrojuje se postupně posloupnost stále menších grafů $G = G_0, G_1, \ldots, G_m$. Nejmenší graf G_m se rozdělí vhodnou metodou. Získané dělení P_m pak promítneme na graf G_{m-1}, výsledné dělení P_{m-1} se promítne na G_{m-2}, atd. Nakonec získáme dělení P_0 grafu $G_0 = G$.

Významnou myšlenkou ve víceúrovňové metodě je po každém promítnutí získané dělení nechat chvíli vylepšovat Kernighanovým-Linovým algoritmem.

Na samotné rozdělení nejmenšího grafu G_m můžeme použít libovolnou metodu pro dělení grafů. Je ale nutné, aby byla schopna řešit problém dělení grafů se zadanými váhami hran a zadanými váhami vrcholů. Prakticky se používá např. spektrální metoda nebo opakovaná metoda šíření+KL.

Algoritmus 1: Víceúrovňová metoda

$G_0 \leftarrow G$
$i \leftarrow 0$
while $|V(G_i)| > n_{max}$ **do**
  $G_{i+1} \leftarrow coarsen(G_i)$
  $i \leftarrow i + 1$
end

$P_i \leftarrow findPartition(G_i)$

while $i \neq 0$ **do**
  $i \leftarrow i - 1$
  $B_i \leftarrow project(P_{i+1})$
  $P_i \leftarrow refinePartition(B_i)$
end
$P \leftarrow P_0$

Podrobnější popis slučování

Nyní podrobněji definuji, co rozumím pod pojmy hrubší graf a promítnutí dělení.

Definice 12 (hrubší graf)**.** *Je-li $G = (V, E)$ graf, $S_1, \ldots, S_r$ skupinky vrcholů splňující $S_i \subset V$, $S_1 \cup \ldots \cup S_r = V$, S_i neprázdné, $S_i \cap S_j = \emptyset$, pak hrubším grafem G rozumíme graf $G' = (V', E')$ s množinou vrcholů $V' = \{S_1, \ldots, S_r\}$ a s množinou hran $E' \subset \binom{\{S_1,\ldots,S_r\}}{2}$,*

$$e' = \{S_i, S_j\} \in E' \Leftrightarrow \exists u \in S_i \exists v \in S_j : \{u, v\} \in E.$$

Máme-li dány váhy vrcholů $w : V \to \mathbb{R}^+$, pak na hrubším grafu definujeme váhy vrcholů $w' : V' \to \mathbb{R}^+$ předpisem

$$w'(S_i) = \sum_{v \in S_i} w(v) \text{ pro každé } S_i \in V'.$$

Máme-li dány váhy hran $a : E \to \mathbb{R}^+$, pak na hrubším grafu definujeme váhy hran $a' : E' \to \mathbb{R}^+$ předpisem

$$a'(\{S_i, S_j\}) = \sum_{u \in S_i, v \in S_j} a(\{u, v\}) \text{ pro všechny } \{S_i, S_j\} \in E'.$$

Váhy hran a vrcholů jsou ve víceúrovňové metodě významné i v případě, že původní graf G má všechny vrcholy/hrany stejně významné.

Definice 13 (promítnutí dělení)**.** *Je-li G' hrubší graf G a máme-li dané dělení $V'_1, \ldots, V'_k$ hrubšího grafu G', pak promítnutí tohoto dělení na graf G je dělení $V_1, \ldots, V_k$ grafu G takové, že*

$$V_i = \bigcup_{S_j \in V'_i} S_j \text{ pro každé } k \in \{1, \ldots, k\}.$$

Vytváření skupin

Zbývá popsat, jakým způsobem se vybírá seskupení vrcholů do skupinek.

Nejjednodušší metodou je brát v náhodném pořadí hrany grafu a pokud jejich koncové vrcholy ještě nejsou v žádné skupince, tak vytvořit novou skupinku s těmito dvěma vrcholy. V případě, kdy jeden nebo oba koncové vrcholy už jsou v nějaké skupince, nedělat nic. Na konci nám většinou zbudou nějaké vrcholy, které nejsou v žádné skupince a všechny jejich sousedé už ve skupince jsou. U těchto vrcholů nezbývá než každému z nich vytvořit skupinku, ve které budou sami. Ideální by bylo mít těchto samostatných vrcholů co nejméně.

Metod vytváření skupinek je více, viz [4].

Shrnutí

Víceúrovňový přístup umožňuje efektivně nalézt kvalitní dělení pro mnohé grafy s velkým počtem vrcholů.

3.5 Spektrální algoritmus

Spektrální algoritmus je sofistikovaná metoda dělení grafů. Nejdříve přeformuluji problém dělení grafů.

3.5.1 Laplaceova matice grafu

Definice 14 (Laplaceova matice grafu)**.** *Nechť $G = (V, E)$ je neorientovaný graf, $n = |V|$, $V = \{v_1, \ldots, v_n\}$. Laplaceova matice grafu G je matice*

$$Q(G) = \begin{pmatrix} \deg(v_1) & & \\ & \ddots & \\ & & \deg(v_n) \end{pmatrix} - \text{Soused}(G),$$

kde $\deg(v_i)$ *je stupeň vrcholu* v_i *a* $\text{Soused}(G)$ *je matice sousednosti.*

Poznámka (souvislost Laplaceovy matice s Laplaceovým operátorem). Fourierův zákon pro vedení tepla má tvar $\vec{q} = -k \cdot$ (gradient teploty), $\vec{q}$ je tepelný tok a $k > 0$ je konstanta. Z něj odvozená rovnice vedení tepla má tvar

$$\frac{d(\text{teplota})}{dt} = -k\Delta(\text{teplota}).$$

Newtonův zákon ochlazování pro hranu $\{v_i, v_j\}$ má tvar (tepelný tok hranou) = $k_{ij} \cdot$ (rozdíl teplot v koncových bodech). Sečtením přes všechny hrany vycházející z bodu v_i odvodíme

$$\frac{d(teplota(v_i))}{dt} = \sum_{v_j \text{ je soused } v_i} k_{ij}(teplota(v_j) - teplota(v_i)) = {}^2$$

$$= \sum_{v_j \text{ je soused } v_i} -k(teplota(v_i) - teplota(v_j)) = -kQ \begin{pmatrix} teplota(v_1) \\ \ldots \\ teplota(v_n) \end{pmatrix}.$$

[2] zde pro jednoduchost $k = k_{ij}$ $\forall i, j$

Ukáži, že problém dělení grafu se dá přeformulovat jako problém minimalizace kvadratické formy s Laplaceovou maticí přes určitou množinu.

Věta 4 ([3]). *Nechť $G = (V, E)$ je neorientovaný graf, $V = \{v_1, \ldots, v_n\}$, $A \subset V$. Položme $B = V \setminus A$, $Q = Q(G)$. Označme*

$$x_i = \begin{cases} 1, & v_i \in A, \\ -1, & v_i \in B. \end{cases}$$

pro $i \in \{1, ..., n\}$. Pak

$$x^T Q\, x = 4\Big|H(A, B)\Big|.$$

Důkaz.

$$x^T Q\, x = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i q_{ij} x_j = \sum_{i=1}^{n} x_i x_i \deg(v_i) - \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j (Soused(G))_{ij} =$$

$$= \sum_{i=1}^{n} x_i^2 \deg(v_i) - \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j s_{ij} \quad (3.1)$$

Vrcholy v_i a v_j

1. jsou spojeny hranou, $v_i \in A, v_j \in A$, pak $x_i x_j s_{ij} = 1 \cdot 1 \cdot 1 = 1$
2. jsou spojeny hranou, $v_i \in B, v_j \in B$, pak $x_i x_j s_{ij} = -1 \cdot -1 \cdot 1 = 1$
3. jsou spojeny hranou, $v_i \in A, v_j \in B$, pak $x_i x_j s_{ij} = 1 \cdot -1 \cdot 1 = -1$
4. jsou spojeny hranou, $v_i \in B, v_j \in A$, pak $x_i x_j s_{ij} = -1 \cdot 1 \cdot 1 = -1$
5. nejsou spojeny hranou, pak $x_i x_j s_{ij} = x_i x_j \cdot 0 = 0$.

Tedy

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j s_{ij} = \sum_{e \in E \setminus H(A,B)} 2 - \sum_{e \in H(A,B)} 2 = 2\Big|E \setminus H(A, B)\Big| - 2\Big|H(A, B)\Big|.$$

Dále $x_i^2 = 1$,

$$\sum_{i=1}^{n} x_i^2 \deg(v_i) = \sum_{i=1}^{n} \deg(v_i) = 2\Big|E\Big|.$$

Celkem

$$\begin{aligned} x^T Q\, x &= 2\,|E| - \Big(\, 2\,|E \setminus H(A, B)| - 2\,|H(A, B)|\,\Big) = \\ &= 2\,|H(A, B)| + 2\,|E \setminus H(A, B)| - 2\,|E \setminus H(A, B)| + 2\,|H(A, B)| = \\ &= 4\Big|H(A, B)\Big|. \end{aligned}$$

3.5.2 Vlastnosti Laplaceovy matice

Tvrzení 5. *Nechť G je neorientovaný graf. Pak $Q(G)$ je symetrická a součet prvků v každém řádku je* 0*.*

Důkaz. Uvažujeme neorientovaný graf, matice souvislosti je tedy symetrická. Protože je $Q(G)$ součet diagonální a symetrické matice, je $Q(G)$ symetrická.

Součet prvků i–tého řádku matice $Q(G)$ je

$$\deg(v_i) - \sum_{j=1}^{n} s_{ij} \;=\; \deg(v_i) \;- \underbrace{\Big(\sum_{v_j \text{ je soused } v_i} 1\Big)}_{\deg(v_i)} \;- \Big(\sum_{v_j \text{ není soused } v_i} 0\Big) \;=\; 0.$$

Tvrzení 6. *Nechť $G = (V, E)$ je neorientovaný graf, $n = |V|$, $V = \{v_1, \ldots, v_n\}$, $m = |E|$, $E = \{e_1, \ldots, e_m\}$. Nechť u každé hrany e_k vybereme jeden z jejích koncových bodů a označíme ho jako první koncový bod e_k. Pak*

$$Q(G) = B^T B,$$

kde $B = (B_{ki})_{k,i=1}^{m,n}$ je orientovaná matice incidence,

$$B_{ki} = \begin{cases} 1, & v_i \text{ je první koncový bod } e_k, \\ -1, & v_i \text{ je druhý koncový bod } e_k, \\ 0, & \text{jinak.} \end{cases}$$

Důkaz. Pro $i = j$

$$(B^T B)_{ii} = \sum_{k=1}^{m} (B^T)_{ik} B_{ki} = \sum_{k=1}^{m} B_{ki} B_{ki} = \sum_{k=1}^{m} B_{ki}^2 \;=\; \sum_{v_i \text{ je koncový bod } e_k} 1 \;=\; \deg(v_i).$$

Pro $i \neq j$

$$(B^T B)_{ij} = \sum_{k=1}^{m} (B^T)_{ik} B_{kj} = \sum_{k=1}^{m} B_{ki} B_{kj} \;=\; \sum_{v_i, v_j \text{ jsou spojeny hranou}} (-1) \;=\; -s_{ij}.$$

Tvrzení 7. *Dimenze* $\mathrm{Ker}(Q(G))$ *je rovna počtu komponent souvislosti grafu G.*

Důkaz. Lze nalézt v [5] pod nadpisem „Proposition 2“.

Věta 8 (o vlastních číslech Laplaceovy matice)**.** *Laplaceova matice je unitárně diagonalizovatelná, všechna vlastní čísla jsou reálná a nezáporná. Dále je Laplaceova matice singulární, a tedy 0 je vlastní číslo.*

Důkaz. Z věty o unitární diagonalizaci pro symetrické reálné matice plyne, že existuje $U \in \mathbb{R}^{n \times n}$, $U^T U = I_n$ taková, že $U^T Q(G) U$ je diagonální. Nutně jsou potom všechna vl. čísla Laplaceovy matice reálná.

Nechť λ_i je vl. číslo Laplaceovy matice, $x \in \mathbb{R}^n$ je příslušný vlastní vektor, tedy $x \neq 0$, $Q(G)\,x = \lambda_i\,x$. Pak z tvrzení 6

$$x^T Q(G)\,x = x^T B^T B x = (Bx)^T (Bx) = |Bx|^2 \geq 0,$$

zároveň

$$x^T Q(G)\,x = x^T \lambda_i\,x = \lambda_i\,x^T x = \lambda_i\,|x|^2.$$

Máme tedy

$$\lambda_i\,|x|^2 \geq 0,\ x \neq 0 \ \Rightarrow\ \lambda_i \geq 0.$$

Součet prvků Laplaceovy matice v každém řádku je 0, takže vektor $x = (1, \ldots, 1)^T \in \mathbb{R}^n$ je vlastní vektor vl. čísla 0.

3.5.3 Odvození spektrálního algoritmu

Nechť G je *souvislý* neorientovaný graf s n vrcholy a aspoň jednou hranou, $n = |V|$, $\{v_1, \ldots, v_n\} = V(G)$, $Q = Q(G)$ je jeho Laplaceova matice. Nechť $d \in \mathbb{Z}$, $-n < d < n$. Pak diskrétní optimalizační problém

$$\begin{aligned} &\text{volba } A \subset V(G) \text{ splňující } |A| = |B| + d \text{ tak,} \\ &\qquad\qquad \text{aby } |H(A,B)| \text{ bylo minimální} \end{aligned} \tag{3.2}$$

je podle věty 4 ekvivalentní diskrétnímu optimalizačnímu problému

$$\begin{aligned} &\text{volba } x \in \mathbb{R}^n \text{ splňující } {}_{\forall i \in \{1,\ldots,n\}:}\ x_i = \pm 1 \text{ a } \sum_{i=1}^{n} x_i = d \text{ tak,} \\ &\qquad\qquad \text{aby } x^T Q\,x \text{ bylo minimální.} \end{aligned} \tag{3.3}$$

Z tohoto diskrétního optimalizačního problému nyní uděláme *spojitý* optimalizační problém

$$\begin{aligned} &\text{volba } x \in \mathbb{R}^n \text{ splňující } x_1^2 + \ldots + x_n^2 = n \text{ a } \sum_{i=1}^{n} x_i = d \text{ tak,} \\ &\qquad\qquad \text{aby } x^T Q\,x \text{ bylo minimální.} \end{aligned} \tag{3.4}$$

Zřejmě je množina povolených vektorů x u tohoto problémů nadmnožinou povolených vektorů u diskrétní varianty. Tento optimalizační problém teď budeme řešit.

Řešení problému 3.4. Q je unitárně diagonalizovatelná, označme její vlastní čísla $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Víme, že $\lambda_1 = 0$. Protože navíc předpokládáme souvislost grafu G, tak podle tvrzení 7 má prostor vlastních vektorů vl. č. 0 dimenzi 1, takže $\lambda_2 > 0$.

Můžeme zvolit $U \in \mathbb{R}^{n \times n}$, $U^T U = I_n$ takovou, že

$$\Lambda := U^T Q\,U = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}.$$

Označme j–tý sloupec matice U pomocí u_j. Pak je u_j jednotkový vlastní vektor příslušný vl. číslu λ_j. Dále označme $v = (1, \ldots, 1)^T \in \mathbb{R}^n$. Upravujme

$$\min_{\substack{x_1^2+\ldots+x_n^2=n \\ x_1+\ldots n=d}} x^T Q\, x = \min_{\substack{||x||^2=n \\ (x,v)=d}} x^T U \Lambda\, U^T x = \min_{\substack{||x||^2=n \\ (x,v)=d}} (U^T x)^T \Lambda\, (U^T x) =: M.$$

Nyní substitujme $U^T x =: y$. Pak $||x||^2 = ||Uy||^2 = ||y||^2$, neboť $U^T U = I_n \Rightarrow$ U zachovává normy.

Dále je potřeba dosadit do skalárního součinu $(x, v) = (Uy, v) = \sum_{i=1}^n (Uy)_i v_i$ $= \sum_{i=1}^n \sum_{k=1}^n U_{ik} y_k v_i = \sum_{k=1}^n \sum_{i=1}^n y_k U_{ik} v_i = \sum_{k=1}^n y_k (U^T v)_k = (y, U^T v)$. Víme, že $v = (1, \ldots, 1)^T \in \mathbb{R}^n$ je vlastní vektor příslušný vl. č. 0. Protože vl. č. 0 má násobnost 1 a $||v|| = \sqrt{n}$, tak platí $v = \pm\sqrt{n}\, u_1$,

$$U^T v = \begin{pmatrix} u_1^T \\ \vdots \\ u_n^T \end{pmatrix} v = \begin{pmatrix} u_1^T v \\ \vdots \\ u_n^T v \end{pmatrix} = \begin{pmatrix} \pm\sqrt{n}\, u_1^T u_1 \\ \vdots \\ \pm\sqrt{n}\, u_n^T u_1 \end{pmatrix} = \begin{pmatrix} \pm\sqrt{n} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

neboť U je ortogonální. Tedy $(x, v) = (y, U^T v) = (y, (\pm\sqrt{n}, 0, \ldots, 0)^T) = \pm\sqrt{n} y_1$, kde y_1 značí první složku vektoru y. Po substituci bude tedy

$$M = \min_{\substack{||y||^2=n \\ \pm\sqrt{n}\, y_1 = d}} y^T \Lambda\, y = \min_{\substack{y_1^2+y_2^2+\ldots+y_n^2=n \\ y_1=\pm d/\sqrt{n}}} \sum_{i=1}^n \lambda_i y_i^2 =$$

$$= \min_{\substack{y_1^2+y_2^2+\ldots+y_n^2=n \\ y_1=\pm d/\sqrt{n}}} \sum_{i=2}^n \lambda_i y_i^2 + \lambda_1 \cdot \Big(\pm \frac{d}{\sqrt{n}}\Big)^2 = \min_{(\pm d/\sqrt{n})^2+y_2^2+\ldots+y_n^2=n} \sum_{i=2}^n \lambda_i y_i^2 + 0 =$$

$$= \min_{y_2^2+\ldots+y_n^2=(n^2-d^2)/n} \sum_{i=2}^n \lambda_i y_i^2.$$

Protože $\lambda_2 \leq \ldots \leq \lambda_n$, můžeme odhadnout

$$\sum_{i=2}^n \lambda_i y_i^2 \geq \sum_{i=2}^n \lambda_2 y_i^2 = \lambda_2 \sum_{i=2}^n y_i^2 = \lambda_2 (n^2 - d^2)/n,$$

tedy $M \geq \lambda_2(n^2 - d^2)/n$ pro všechny volby $y = U^T x$. Zároveň volbou $y_2^2 = (n^2 - d^2)/n$, $y_i = 0$ pro $i = 3, \ldots, n$ můžeme takové hodnoty dosáhnout.

Spojitý optimalizační problém je tedy vyřešen pro $x = y_1 u_1 + y_2 u_2 + 0 = \frac{\pm d}{\sqrt{n}}(\frac{\pm 1}{\sqrt{n}} v) \pm \sqrt{\frac{n^2-d^2}{n}} \cdot u_2 = \frac{d}{n} \cdot (1, \ldots, 1)^T \pm \sqrt{\frac{n^2-d^2}{n}} \cdot u_2$.

Nakonec zbývá od spojitého problému přejít zpět k diskrétnímu. Potřebujeme hodnoty vektoru x zaokrouhlit tak, aby $x_i' = \pm 1$ a aby počet jedniček (tj. počet prvků A) byl $m_1 = \frac{n+d}{2}$ a počet minus jedniček (tj. počet prvků B) byl $m_2 = \frac{n-d}{2}$. To uděláme tak, že z vektoru x vybereme m_1 největších prvků x_i a těmto vybraným prvkům přiřadíme $x_i' = 1$, ostatním $n - m_1 = m_2$ prvkům přiřadíme $x_i' = -1$.

Tato „zaokrouhlovací“ procedura by skončila se stejným výsledkem, pokud bychom místo vektoru x začali s vektorem $x + (r, \ldots, r)^T$ pro nějaké $r \in \mathbb{R}$ nebo kdybychom začali s vektorem $c\, x$ pro nějaké $c \in \mathbb{R}^+$. Díky tomu stačí místo x použít $\pm c\, u_2$, kde $c \in \mathbb{R}^+$ libovolné, neboli libovolný vlastní vektor vl. čísla λ_2.

Definice 15 (Fiedlerův vektor)**.** *Nechť $G = (V, E)$ je neorientovaný graf, $n = |V|$, $V = \{v_1, \ldots, v_n\}$. Pak vektoru $x \in R^n \setminus \{0\}$ říkáme* Fiedlerův vektor*, pokud je to vlastní vektor nejmenšího nenulového vlastního čísla Laplaceovy matice $Q(G)$.*

Poznámka. Fiedlerův vektor existuje $\Leftrightarrow$ Laplaceova matice je nenulová. $\Leftrightarrow$ v grafu G existuje hrana. Pro použití Fiedlerova vektoru ve spektrálním algoritmu je ale potřeba souvislost grafu G.

Shrnutí

Spektrální algoritmus má tři části.

- Sestavení Laplaceovy matice $Q(G)$
- Výpočet (nějakého) Fiedlerova vektoru
- Vybrání m_1 největších prvků spočteného Fiedlerova vektoru, jejich pozice určí vrcholy přiřazené do části A

Laplaceova matice je obecně řídká, konkrétně má $|V| + 2|E|$ nenulových prvků. Fiedlerův vektor spočteme numericky, například Lanczosovým algoritmem.[3]

Podrobnější výsledky o spektrálním algoritmu jsou například v [6].

3.6 Rekurzivní dělení

Často se v dělení grafů pro velké k používá následující postup: Když máme rozdělit graf G na k částí, $k = s^p$, tak rozdělíme graf G nejprve na s částí $V_1, \ldots, V_s$. Každou z těchto částí V_i dál rozdělíme na části $V_{i,1}, \ldots, V_{i,s}$. Tento postup provedeme celkem p krát. Nakonec budeme mít graf G rozdělen na s^p částí $V_{i_1,\ldots,i_p}$, kde $i_r \in \{1, \ldots, s\}, r \in \{1, \ldots, p\}$.

Výhoda tohoto postupu je, že algoritmus pro $k' = s$ může být jednodušší na implementaci. Pro dělení grafů na 2^p částí stačí mít po užití tohoto postupu implementovanou pouze bisekci grafu. Zároveň tento postup nabízí určitou možnost, jak algoritmy dělení na mnoho částí paralelizovat.

Významnou otázkou je, jak kvalitní dělení se tímto postupem dají získat.

Poznámka. Chtěl bych upozornit na to, že v případě uvažování penalizačních členů z kapitoly 2.4 je vhodné rekurzivní dělení drobně upravit. Standardně po dělení grafu G na části $V_1, \ldots, V_s$ následuje dělení indukovaných grafů $G(V_1), \ldots, G(V_s)$. Ovšem při přítomnosti penalizačních členů je vhodné pro každou hranu $\{u, v\}$ hranového separátoru, kde $u \in V_i$, $v \in V_j$, $i \neq j$, přidat do indukovaného grafu $G(V_i)$ „vnější“ hranu $\{u, V_j\}$. Pomocí těchto „vnějších“ hran lze potom v algoritmu na dělení indukovaného grafu zohlednit penalizaci za propojení s částmi mimo indukovaný graf.

[3] Inverzní mocninnou metodu samozřejmě nelze použít, protože Q je singulární.

4. Využití dělení grafů v paralelních řešičích soustav lin. rovnic

Nechť $n \in \mathbb{N}$, $A \in \mathbb{R}^{n \times n}$ je regulární matice, $b \in \mathbb{R}^n$ je pravá strana. V této části bakalářské práce se budu zabývat numerickým řešením rovnice

$$Ax = b. \tag{4.1}$$

Konkrétně se budu zabývat situací, kdy je A *řídká* a kdy je potřeba řešit úlohu *paralelně*.

Definice 16. *Matice A se nazývá řídká, pokud je většina jejich prvků nulových. Počet nenulových prvků matice A budu značit* $\mathrm{nnz}(A)$. *Hustotou matice $A \in \mathbb{R}^{n \times n}$ budu nazývat hodnotu* $\frac{\mathrm{nnz}(A)}{n^2} \cdot 100\%$.

Budu se zaměřovat především na symetrické pozitivně definitní matice. Takové matice se v praxi vyskytují velmi často v numerických metodách pro eliptické parciální diferenciální rovnice. Na konkrétních metodách ukáži, jak se v paralelních řešičích využívá dělení grafů.

4.1 Přímé a iterační metody

Numerické metody na řešení soustav lineárních rovnic dělíme na přímé a iterační. V přímých metodách se po určitém počtu kroků spočte řešení. Pokud by se během výpočtu mezivýsledky nezaokrouhlovaly, tak by toto řešení bylo přesné. Přímé metody jsou typicky založeny na výpočtu Choleského, LU nebo QR rozkladu.

V iteračních metodách se postupně konstruuje posloupnost $x_0, x_1, x_2, \ldots$ přibližných řešení, která by v přesné aritmetice konvergovala k přesnému řešení. Základní skupiny iteračních metod jsou stacionární a nestacionární. Stacionární metody mají tvar $x_{k+1} = Bx_k + v$, kde B je matice a v je vektor. Z nestacionárních metod jsou nejvýznamnější Krylovovské metody. Ty přibližné řešení $Ax = b$ hledají v Krylovově prostoru

$$\mathcal{K}_n(A,\, b) = \mathrm{span}\{\, b, Ab, A^2b, \ldots, A^{n-1}b \,\}.$$

4.2 Struktura řídké matice

Definice 17 (graf struktury matice)**.** *Nechť $n \in \mathbb{N}, A \in \mathbb{R}^{n \times n}$, A je symetrická. Pak grafem struktury matice rozumíme graf s množinou vrcholů $V = \{1, \ldots, n\}$ a množinou hran*

$$E = \{\, \{i, j\} \mid A_{ij} \neq 0 \,\}.$$

Tento graf budeme značit $\mathrm{Struct}(A)$.

Poznámka. Výše uvedená definice s orientovanými hranami (i, j) by šla aplikovat i na nesymetrické čtvercové matice, ale výsledný graf by byl orientovaný. Často se u nesymetrických matic používá graf struktury symetrizace matice A, tedy graf struktury matice $\frac{1}{2}(A + A^T)$, což je graf o množině vrcholů $V = \{1, \ldots, n\}$ a množině hran

$$E = \{\, \{i, j\} \mid A_{ij} \neq 0 \vee A_{ji} \neq 0 \,\}.$$

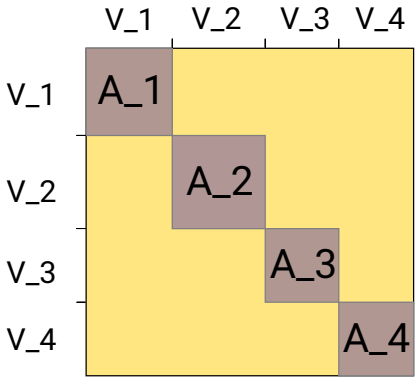
Poznámka. Pokud bychom chtěli zahrnout nejen informaci o struktuře matice A, ale i o numerických hodnotách jejích prvků, tak bychom mohli definovat

$$w : E \to \mathbb{R}^+$$

předpisem

$$w(\{i, j\}) = |A_{ij}|.$$

Je-li $V_1, \ldots, V_k$ dělení grafu Struct(A), tak můžeme matici A přeuspořádat permutací P tak, že ve výsledné matici $\tilde{A} = PAP^T$ jsou nejprve řazeny sloupce odpovídající vrcholům z části V_1, pak vrcholy z V_2 a tak podobně až po vrcholy z V_2 (viz obrázek 4.1). Poddiagonální nenulové prvky v podmaticích A_i odpovídají hranám mezi vrcholy z V_i. Poddiagonální prvky ve zbytku matice odpovídají hranám z hranového separátoru $H(V_1, \ldots, V_k)$.



Obrázek 4.1: Varianta A

4.3 Paralelizace přímých metod

Definice 18. *Řekneme, že čtvercová dolní trojúhelníková matice L je Choleského faktor matice A, jestliže*

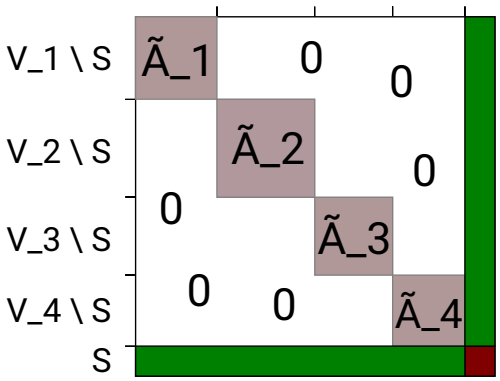
$$A = LL^T.$$

Pro symetrickou pozitivně definitní matici A existuje právě jeden Choleského faktor L. Tento faktor umíme najít pomocí varianty Gaussovy eliminace, přičemž výpočet je numericky stabilní pro libovolné uspořádání matice A.

Paralelizace výpočtu Choleského faktorizace

Pokud je matice A řídká, tak můžeme dělením grafu Struct(A) získat dělení $P = (V_1, \ldots, V_k)$ a nějaký vrcholový separátor S tohoto dělení P. Pak můžeme získat uspořádání matice A jako na obrázku 4.2). Tentokrát jsou mimo podmatice $\tilde{A}_i$ a pás příslušný separátoru S všechny poddiagonální prvky nulové, neboť mezi vrcholy z částí $V_i \setminus S$ a $V_j \setminus S$, $i \neq j$, nevede hrana.

Choleského faktorizace podmatic $\tilde{A}_1, \ldots, \tilde{A}_k$ lze počítat odděleně. Následně je třeba dodělat Gaussovu eliminaci pásu S, tato část výpočtu už paralelizovat nelze. Úspěšnost rozdělení úlohy tedy závisí na velikosti separátoru S.[1]



Obrázek 4.2: Varianta B

Příklad. Nechť matice A je hustá, konkrétně ať nemá žádný nulový prvek. Pak je Struct(A) úplný graf a vrcholový separátor každého dělení je tvořen všemi vrcholy.

Příklad. Nechť matice odpovídá rovnoměrné rovinné mřížce o rozměrech m_1 a m_2. Pak má optimální vrcholový separátor dělení na dvě části má velikost $|S| = \min\{m_1, m_2\}$.

Výhoda přeuspořádání matice na obrázku 4.2 získaného pomocí dělení grafu Struct(A) nespočívá pouze v možnosti paralelizace části výpočtu. Toto uspořádání zároveň zajišťuje, že v průběhu eliminace můžou vznikat nové nenulové prvky pouze uvnitř podmatic $\tilde{A}_1, \ldots, \tilde{A}_k$ a ve sloupcích a řádcích odpovídajícím separátoru S. Opakované dělení grafu Struct(A) (*nested dissection*) se používá jako jedna z metod snižujících narůstající zaplnění matice v průběhu eliminace.

Paralelizace výpočtu LU faktorizace

Kdyby A byla čtvercová nesymetrická, tak se dá dělit graf Struct($A + A^T$). Na rozdíl od minulého případu výpočet LU rozkladu už nemusí být numericky stabilní. Standardní technikou pro zvýšení numerické stability výpočtu je částečná pivotace. Dělení grafu určitým způsobem pivotaci omezuje, ale stále je možné provádět pivotaci v rámci eliminace podmatic $\tilde{A}_1, \ldots, \tilde{A}_k$.

[1] Připomínám, že algoritmy dělení se snaží rozdělit graf tak, aby byl vrcholový separátor co nejmenší.

4.4 Paralelizace iteračních metod

U Krylovovských metod je potřeba v jednom kroku provádět operace sčítání dvou vektorů, násobení vektoru skalárem, skalární součin dvou vektorů, násobení vektoru řídkou maticí A, případně A^T.

Tyto operace lze paralelizovat tak, že množinu indexů $\{1, \ldots, n\}$ rozdělíme na části $A_1, \ldots, A_k$ takové, že každý index $i \in \{1, \ldots, n\}$ se nachází v právě jedné z těchto částí. Na s–tém procesoru pak budeme uchovávat ty složky x_i vektoru $x \in \mathbb{R}^n$, pro které je $i \in A_s$.

Operace sčítání dvou vektorů nebo násobení vektoru skalárem se provádí po složkách, jednoduše se tedy provede na každém procesoru pro danou část. U skalárního součinu vektorů lze spočítat na každém procesoru částečný součet $\sum_{j \in A_s} x_i y_i$, pak je ale potřeba tyto částečné součty sečíst. Tím se nevyhneme přenášení dat mezi procesory a synchronizaci výpočtu.

Operace násobení vektoru řídkou maticí A, resp. A^T bude vyžadovat přenos dat mezi procesory. Tomuto přesunu dat se nelze vyhnout, ale můžeme se snažit množství přenášených dat minimalizovat. A právě zde se využije dělení grafů. Části nebudeme volit náhodně, ale tak, aby byly přibližně stejně velké a aby byl vrcholový separátor grafu Struct$(A + A^T)$ co nejmenší.

Paralelizovat Krylovovské metody by tedy šlo i bez dělení grafů, ale nebyla by minimalizována komunikace mezi procesory potřebná v každém kroku metody. Požadavek v dělení grafů na přibližně stejně velké části $V_1, \ldots, V_k$ navíc zajistí přibližně stejné vytížení procesorů při násobení řídkou maticí A, resp. A^T.

4.4.1 Metoda sdružených gradientů

Dále budu zabývat metodou sdružených gradientů. Tato metoda vyžaduje, aby A byla symetrická a pozitivně definitní. Odvození a analýza metody viz [7].

Algoritmus 2: Metoda sdružených gradientů (CG)

$x_0 \leftarrow 0$
$r_0 \leftarrow b$
$p_0 \leftarrow r_0$
$rscal_0 \leftarrow r_0^T r_0$
$k \leftarrow 0$
while $||r_k|| < tol$ **do**
  $w_k \leftarrow A p_k$ přenášení dat, násobení řídkou maticí
  $pscal_k \leftarrow p_k^T w_k$ synchronizace procesorů
  $\alpha_k \leftarrow \frac{rscal_k}{scal_k}$
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $r_{k+1} \leftarrow r_k - \alpha_k w_k$
  $rscal_{k+1} \leftarrow r_{k+1}^T r_{k+1}$ synchronizace procesorů
  $\beta_k \leftarrow \frac{rscal_{k+1}}{rscal_k}$
  $p_{k+1} \leftarrow r_{r+1} + \beta_k p_k$
  $k \leftarrow k + 1$
end

Vidíme tedy, že při paralelizaci této metody výše uvedeným způsobem budou potřeba v každém kroku dvě synchronizace procesorů kvůli výpočtu skalárního součinu a jeden rozsáhlejší přesun dat mezi procesory před výpočtem Ap_k.

Efektivnost výpočtu pomocí metody sdružených gradientů závisí na dvou faktorech:

1. rychlost konvergence metody
2. výpočetní náklady v každém kroku metody

Rychlost konvergence metody konjugovaných gradientů závisí především na rozložení vlastních čísel matice A. Pokud jsou vlastní čísla A velmi podobná, tak je konvergence CG velmi rychlá, pokud jsou naopak vlastní čísla A rovnoměrně rozprostřená, tak metoda konverguje velmi pomalu, viz [7]. Konvergence metody je také ovlivněna zaokrouhlovacími chybami.

Výpočetní náklady jednoho kroku metody jsou $O(n + \mathrm{nnz}(A))$.

4.4.2 Předpodmínění

V praxi se metoda konjugovaných gradientů používá zejména v kombinaci s tzv. *předpodmíněním*. Místo řešení rovnice $Ax = b$ se bude řešit rovnice $\tilde{A}\tilde{x} = \tilde{b}$, která má lepší numerické vlastnosti.

Nechť máme přibližný Choleského rozklad $A \approx LL^T$. Rovnici $Ax = b$ přenásobím zleva maticí L^{-1}, získám $L^{-1}Ax = L^{-1}\,b$, označím $\tilde{x} = L^T x$, $\tilde{b} = L^{-1}\,b$. Tím získáme upravenou rovnici soustavy

$$L^{-1}A(L^{-1})^T\,\tilde{x} = \tilde{b}. \tag{4.2}$$

Přitom matice $L^{-1}A(L^{-1})^T$ je symetrická pozitivně definitní a platí

$$L^{-1}A(L^{-1})^T \quad \approx \quad L^{-1}L\,L^T(L^{-1})^T = I_n I_n = I_n.$$

Algoritmus 3: Sdružené gradienty pro předpodmíněnou soustavu 4.2

$\tilde{x}_0 \leftarrow 0$
$\tilde{r}_0 \leftarrow L^{-1}b$
$\tilde{p}_0 \leftarrow \tilde{r}_0$
$\tilde{rscal}_0 \leftarrow \tilde{r}_0^T\tilde{r}_0$
$k \leftarrow 0$
while $||\tilde{r}_k|| < \tilde{tol}$ **do**
  $\tilde{w}_k \leftarrow L^{-1}A(L^{-1})^T p_k$
  $\tilde{pscal}_k \leftarrow \tilde{p}_k^T\tilde{w}_k$
  $\tilde{\alpha}_k \leftarrow \frac{\tilde{rscal}_k}{\tilde{pscal}_k}$
  $\tilde{x}_{k+1} \leftarrow \tilde{x}_k + \tilde{\alpha}_k\tilde{p}_k$
  $\tilde{r}_{k+1} \leftarrow \tilde{r}_k - \tilde{\alpha}_k\tilde{w}_k$
  $\tilde{rscal}_{k+1} \leftarrow \tilde{r}_{k+1}^T\tilde{r}_{k+1}$
  $\tilde{\beta}_k \leftarrow \frac{\tilde{rscal}_{k+1}}{\tilde{rscal}_k}$
  $\tilde{p}_{k+1} \leftarrow \tilde{r}_{r+1} + \tilde{\beta}_k\tilde{p}_k$
  $k \leftarrow k + 1$
end

Platí $x_k = (L^{-1})^T \tilde{x}_k$,

$$r_k = b - Ax_k = L(L^{-1}b + L^{-1}A\,x_k) = L(\tilde{b} - L^{-1}A(L^{-1})^T\,\tilde{x}_k) = L\,\tilde{r}_k.$$

Nyní označím $p_k = (L^{-1})^T \tilde{p}_k$, $w_k = Ap_k$. Pak

$$\begin{gathered}
x_0 = (L^{-1})^T \tilde{x}_0 = 0 \\
r_0 = L\,\tilde{r}_0 = LL^{-1}b = b \\
p_0 = (L^{-1})^T \tilde{p}_0 = (L^{-1})^T L^{-1} r_0 \\
\tilde{rscal}_k = \tilde{r}_k^T \tilde{r}_k = (L^{-1}r_k)^T (L^{-1}r_k) = r_k^T (L^{-1})^T L^{-1} r_k \\
\tilde{pscal}_k = \tilde{p}_k^T \tilde{w}_k = (L^T p_k)^T L^{-1} A (L^{-1})^T \tilde{p}_k = p_k^T L L^{-1} w_k = p_k^T w_k.
\end{gathered}$$

Zároveň

$$\begin{aligned}
&\tilde{x}_{k+1} = \tilde{x}_k + \tilde{\alpha}_k \tilde{p}_k \Rightarrow L^T x_{k+1} = L^T x_k + \tilde{\alpha}_k L^T p_k \Rightarrow && x_{k+1} = x_k + \tilde{\alpha}_k p_k \\
&\tilde{r}_{k+1} = \tilde{r}_k - \tilde{\alpha}_k \tilde{w}_k \Rightarrow L^{-1} r_{k+1} = L^{-1} r_k - \tilde{\alpha}_k L^{-1} w_k \Rightarrow && r_{k+1} = r_k - \tilde{\alpha}_k w_k \\
&\tilde{p}_{k+1} = \tilde{r}_{k+1} + \tilde{\beta}_k \tilde{p}_k \Rightarrow L^T p_{k+1} = L^{-1} r_{k+1} + \tilde{\beta}_k L^T p_k \Rightarrow && p_{k+1} = \\
& && (L^{-1})^T L^{-1} r_{k+1} + \tilde{\beta}_k p_k.
\end{aligned}$$

Dosazením získávám finální algoritmus:

Algoritmus 4: Sdružené gradienty s předpodmíněním (PCG)

$x_0 \leftarrow 0$
$r_0 \leftarrow b$
$z_0 \leftarrow (L^{-1})^T L^{-1} r_0$
$p_0 \leftarrow z_0$
$rscal_0 \leftarrow r_0^T z_0$
$k \leftarrow 0$
while $||r_k|| < tol$ **do**
  $w_k \leftarrow Ap_k$ násobení řídkou maticí
  $pscal_k \leftarrow p_k^T w_k$
  $\alpha_k \leftarrow \frac{rscal_k}{pscal_k}$
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $r_{k+1} \leftarrow r_k - \alpha_k w_k$
  $z_{k+1} \leftarrow (L^{-1})^T L^{-1} r_{k+1}$ aplikace předpodmínění
  $rscal_{k+1} \leftarrow r_{k+1}^T z_{k+1}$
  $\beta_k \leftarrow \frac{rscal_{k+1}}{rscal_k}$
  $p_{k+1} \leftarrow z_{r+1} + \beta_k p_k$
  $k \leftarrow k + 1$
end

Označím-li $M = LL^T$, tak $z_k = M^{-1} r_k$. Přitom M^{-1} se aplikuje tak, že se provede dopředná substituce $Ly_k = r_k$ a pak zpětná substituce $L^T z_k = y_k$.

Aby se mohla předpodmíněná metoda sdružených gradientů používat paralelně, tak je potřeba, aby se předpodmínění dalo používat paralelně. Napadají mě zde dvě možnosti:

A Počítat několik menších předpodmínění $A_i \approx L_i L_i^T$
(přeuspořádání matice jako na obrázku 4.1)

B Počítat jedno předpodmínění $\tilde{A} \approx \tilde{L}\tilde{L}^T$
(přeuspořádání matice jako na obrázku 4.2)

První možnost je plně paralelní — samostatně lze lokální předpodmínění L_i jak vypočítat, tak aplikovat. Zároveň má předpodmínění tvar

$$\begin{pmatrix} L_1 L_1^T & & \\ & L_2 L_2^T & \\ & & L_3 L_3^T \end{pmatrix} \approx \begin{pmatrix} A_1 & .. & . \\ .. & A_2 & . \\ . & . & A_3 \end{pmatrix},$$

tedy prvky odpovídající hranám v hranovém separátoru (znázorněné tečkami) nemůžou být zahrnuty do žádného z předpodmínění. To může snížit kvalitu získaného předpodmínění a následně pak zvýšit počet potřebných kroků PCG.

Druhá možnost tuto obtíž nemá, ale zároveň se výpočet a aplikace předpodmínění dá paralelizovat pouze částečně (podobně jako v kapitole o paralelizaci Choleského faktorizace). V této práci se dále zabývám první možností.

4.4.3 Neúplný Choleského rozklad

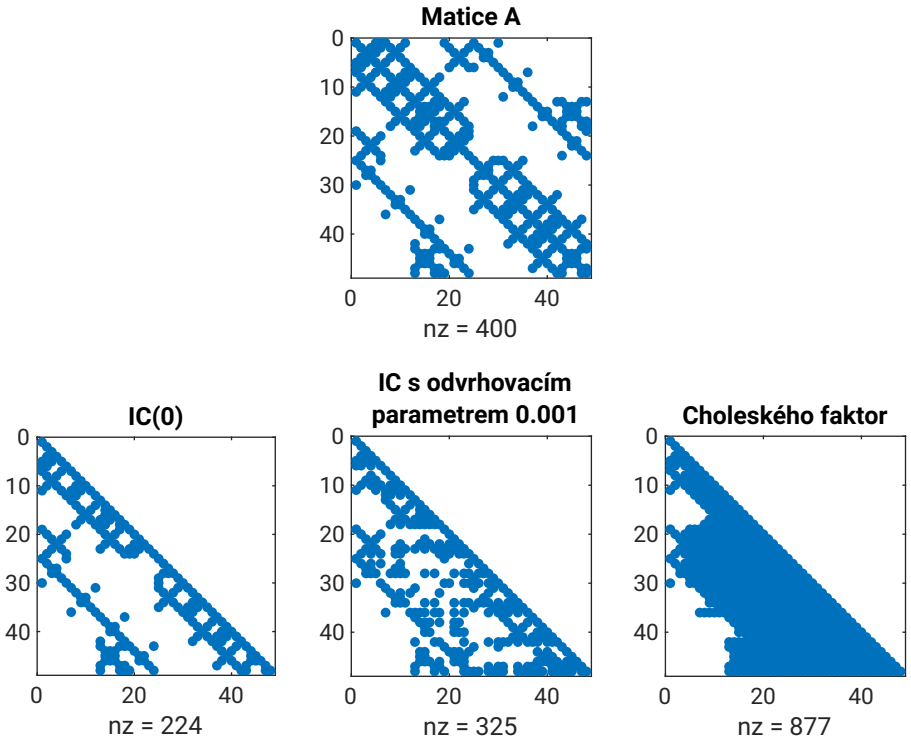
Potřebujeme nyní získat přibližný rozklad $A \approx LL^T$ takový, že $\mathrm{nnz}(L)$ je co nejmenší. Často používané předpodmínění je *neúplný Choleského rozklad*. Při výpočtu neúplného Choleského rozkladu se postupuje podobně jako v případě Choleského faktorizace, ovšem uvažují se pouze prvky na určitých pozicích.

Algoritmus 5: Neúplný Choleského rozklad (IC)

```
for k = 1 : n do
    L(k,k) ← √A(k,k)
    for i = (k+1) : n do
        if (i,k) ∈ S then
            L(i,k) ← A(i,k)/L(k,k)
            for j = (k+1) : i do
                if (j,k) ∈ S & (i,j) ∈ S then
                    A(i,j) ← A(i,j) − L(i,k) L(j,k)
            end
        end
    end
end
```

Různé varianty IC se liší podle výběru množiny $\mathcal{S} \subset \{\,(i,j) \mid i,j \in \mathbb{N}, i \leq n, j \leq n\,\}$. V základní verzi IC(0) se volí $\mathcal{S} =$ *pozice nenulových prvků v* A. Sofistikovanější verze IC určují množinu $\mathcal{S}$ v průběhu algoritmu podle numerických hodnot prvků nebo podle úrovně zaplnění prvku. Kromě IC se používá ještě modifikovaný neúplný Choleského rozklad MIC. Informace o neúplném Choleského rozkladu jsem čerpal z [8].

V této bakalářské práci budu využívat variantu ICT(τ) založenou na numerických hodnotách prvků matice A. Konkrétně budu používat funkci `ichol` v prostředí Matlab s nastavením `type='ict'` a nastavitelným odvrhovacím parametrem `droptol` $= \tau$. Používal jsem pevnou hodnotu $\tau =$ `1e-3`.



Obrázek 4.3: Ilustrace neúplných rozkladů IC(0), ICT($\tau =$ `1e-3`) a úplného Choleského rozkladu s množstvím nenulových prvků.

Existence neúplného Choleského rozkladu

I když je matice A symetrická pozitivně definitní, tak neúplný Choleského rozklad nemusí vždy existovat. V případě, že neexistuje, tak článek [9] doporučuje počítat neúplný Choleského rozklad matice $A + dI_n$, kde $d > 0$. Parametr d volím následujícím způsobem:

1. zkusím $d = 0$
2. pokud neexistuje neúplný Choleského rozklad, zkusím $d = d_{\text{start}}$
3. dokud neexistuje neúplný Choleského rozklad $A + dI_n$, tak zkusím $d = 2d$.

4.5 Vyvažování zátěže

U přepodmínění IC(0) je zajištěno, že $\text{nnz}(L_i) \approx \text{nnz}(L_j)$, $i \neq j$, protože dělení grafu zajišťuje $\text{nnz}(A_i) \approx \text{nnz}(A_j)$ a v IC(0) platí $\text{nnz}(L_i + L_i^T) = \text{nnz}(A_i)$. Ovšem u použití IC s odvrhovacím parametrem už vychází $\text{nnz}(L_i + L_i^T)$ úplně jinak než

$\mathrm{nnz}(A_i)$. Na různých částech tak může vycházet předpodmínění jinak zaplněné. To znamená, že na různých procesorech trvá aplikace těchto předpodmínění jinak dlouhou dobu. Ve zbytku bakalářské práce se snažím zjistit, jestli se s tímto problémem dá něco dělat.

Postup

Veškeré výpočty jsem prováděl v prostředí Matlab R2020a na počítači s procesorem AMD A6-6400K. Pro účely testování jsem vybral 21 symetrických pozitivně definitních matic různé velikosti ze sbírky řídkých matic [10]. U každé matice *mat* jsem rozdělil graf Struct(*mat*) pomocí programu Metis na dvě části. Následně jsem spočetl odpovídající přeuspořádání matice *mat* (viz obr. 4.1). Podmatice A_1 a A_2 jsem dále už nijak nepřeuspořádával, tedy je-li sloupec v podmatici A_1 vlevo od jiného sloupce v podmatici A_1, pak tomu tak bylo i v původní matici *mat*. [2]

Poté jsem na každé části spočetl $L_1 = \texttt{ichol}(A_1 + dI_n)$ a $L_2 = \texttt{ichol}(A_2 + dI_n)$ (detaily parametrů funkce `ichol` jsem popsal na minulé straně. Pokud byly L_1, L_2 jinak zaplněné, upravil jsem původní dělení získané pomocí programu Metis vlastním programem implementovaným v jazyce Python[3], který uměl přesunout s vrcholů z více zaplněné části do méně zaplněné části. Pak jsem opakoval část s přeuspořádáním matice *mat* a s výpočtem L_1', L_2'.

Tento postup jsem opakoval pro různá s, dokud nebyly oblasti vyrovnané. Pro volbu neznámého parametru s jsem použil metodu půlení intervalů kombinovanou s postupem už jednou použitým na minulé straně pro parametr d.

Poznámky

O vyrovnávání částí z hlediska zaplnění se již snažili v článku [11], ovšem tam se zároveň snažili mít na jednotlivých částech uspořádání, které by minimalizovalo počet vzniklých nenulových prvků v průběhu Choleského faktorizace. Taková uspořádání jsou velmi vhodná při použití přímé metody, ale při použití neúplného Choleského rozkladu se tím může snížit kvalita předpodmíněni. O vlivu uspořádání matice na rychlost metody ICCG pojednává článek [12].

Výsledky

Navržený postup se ukázal být funkční. U všech matic jsem dostal stejně zaplněné faktory L_1, L_2. Výjimka byla u matice `offshore` – matice byla velmi velká (4 242 673 nenulových prvků), výpočet trval dlouho a program ho ukončil předčasně.

U pěti z 21 zkoumaných matic byl rozdíl mezi velikosti faktorů zanedbatelný (to jsem definoval tak, že $|(\mathrm{nnz}(L_1)/\,\mathrm{nnz}(L_2)) - 1| \leq 0.01$). Výrazného zlepšení jsem dosáhl u matic `2cubes_sphere`, `685_bus`, `bcsstk13`, `bcsstk23` a `offshore`.

Bohužel se ukázalo, že tato úspěšná vyrovnání byla doprovázena zvětšením velikosti hranového separátoru. Nevím, jestli je to nutné, nebo je to pouze vlastnost mého algoritmu pro vyvažování. V následujících tabulkách značí h velikost hranového separátoru $|H(V_1, V_2)|$.

[2] Přeuspořádání jsem se původně zkoušel dělat, ale pak jsem to zavrhl pro přílišnou složitost interpretování výsledků.

[3] Program implementoval variantu Kernighanova-Linova algoritmu.

jméno matice	$\mathrm{nnz}(A_1)$	$\mathrm{nnz}(A_2)$	$\mathrm{nnz}(L_1)$	$\mathrm{nnz}(L_2)$	$\frac{\mathrm{nnz}(L_1)}{\mathrm{nnz}(L_2)}$	h
bcsstk03	320	320	177	177	1.000000	0
bcsstm07	3626	3410	1695	1692	1.001773	108
finan512	298352	298316	592634	592281	1.000596	162
G2_circuit	363197	362589	866371	871610	0.993989	444
nos4	275	293	333	332	1.003012	13

Tabulka 4.1: Matice, které neměly problém s vyvážením

jméno matice parametr s	$\mathrm{nnz}(A_1)$ $\mathrm{nnz}(A'_1)$	$\mathrm{nnz}(A_2)$ $\mathrm{nnz}(A'_2)$	$\mathrm{nnz}(L_1)$ $\mathrm{nnz}(L'_1)$	$\mathrm{nnz}(L_2)$ $\mathrm{nnz}(L'_2)$	$\frac{\mathrm{nnz}(L_1)}{\mathrm{nnz}(L_2)}$ $\frac{\mathrm{nnz}(L'_1)}{\mathrm{nnz}(L'_2)}$	h h'
2cubes_sphere	798027	832947	645886	851635	0.758407	8145
+8191	843180	654664	680699	654567	1.039923	74710
662_bus	1185	1245	1851	1898	0.975237	22
+2	1193	1237	1885	1883	1.001062	22
685_bus	1592	1641	2769	3425	0.808467	8
+31	1633	1510	2912	2880	1.011111	53
1138_bus	1943	2079	3326	3023	1.100232	16
-31	1850	2118	3068	3090	0.992880	43
bcsstk01	176	176	116	130	0.892308	24
+2	186	154	122	104	1.173077	30
bcsstk06	3720	3924	2244	2272	0.987676	108
+3	3723	3913	2247	2226	1.009434	112
bcsstk13	44855	33168	31684	19922	1.590403	2930
-319	34076	40523	23366	23777	0.982714	4642
bcsstk22	363	325	274	270	1.014815	4
-1	360	328	269	272	0.988971	4
bcsstk23	21007	22845	7781	5872	1.325102	663
-319	17778	24966	6541	6623	0.987619	1217
bcsstk24	79299	75915	6526	6240	1.045833	2348
-35	77790	77040	6341	6343	0.999685	2540
bcsstk38	173004	178020	117584	108244	1.086287	2218
-255	167735	180321	109234	109328	0.999140	3702
lund_a	1096	1127	676	688	0.982558	113
+1	1115	1104	692	678	1.020649	115
msc01440	21937	21373	16470	16860	0.976868	844
+11	22190	21074	16661	16582	1.004764	867
nasa2910	79523	86023	31587	33143	0.953052	4375
+35	81190	83954	32379	32282	1.003005	4576
nos3	7420	7936	8796	9417	0.934055	244
+16	7672	7678	9101	9088	1.001430	247
offshore	2104329	2119690	2337410	3084962	0.757679	9327
+32767	2174158	1757121	2401506	2684677	0.894523	155697

Tabulka 4.2: Vyvažování

Závěr

V první části bakalářské práce jsem důkladně rozebral problém dělení grafů. Odvodil jsem odhad pro počet rovnoměrných dělení, našel jsem příklad grafu, ve kterém každé optimální dělení obsahuje nesouvislou část. Formuloval a dokázal jsem lemma o vztahu velikostí hranového a vrcholového separátoru. V podkapitole 2.4 jsem rozebral nedostatky standardní formulace problému dělení grafů z hlediska aplikací v paralelním počítání.

Ve další kapitole jsem popsal standardní metody, které se pro dělení grafů používají, a u každé jsem shrnul její výhody a nevýhody. Přitom jsem vycházel zejména z textů [3] a [4]. Spektrálnímu algoritmu, který je ze všech popisovaných metod nejkomplikovanější, bylo věnováno více prostoru.

V poslední kapitole jsem se měl věnovat specifickým požadavkům kladeným na děliče grafů v problému rozdělení úlohy pro řešení rozsáhlých soustav lineárních rovnic. Po krátkém popsání problematiky u přímých metod jsem se věnoval iteračním metodám, konkrétně metodě ICCG (metoda sdružených gradientů předpodmíněná neúplným Choleského rozkladem). Známý algoritmus sdružených gradientů jsem připomněl a odvodil jsem jeho verzi s předpodmíněním. V podkapitole 4.4.3 jsem velmi stručně vyložil základy neúplné Choleského faktorizace. Na stranách 23 a 25 jsem vysvětlil roli dělení grafů v paralelizaci předpodmíněné metody sdružených gradientů.

V poslední části bakalářské práce jsem se zabýval problémem nevyrovnané zátěže jednotlivých procesorů způsobenou odlišným zaplněním matic L_1 a L_2 získaných variantou neúplné Choleského faktorizace ICT(τ). Tento typ neúplné Choleského faktorizace zohledňuje nejen strukturu řídkosti, ale i konkrétni numerické hodnoty matice. Navrhl jsem postup úpravy původního dělení a implementoval ho v programovacím jazyce Python a prostředí Matlab. Výsledky jsem shrnul v kapitole 4.5.

Seznam použité literatury

[1] J. Matoušek and J. Nešetřil, *Kapitoly z diskrétní matematiky.* čtvrté české vydání, Nakladatelství Karolinum, 2009.

[2] G. Paulino, I. Menezes, M. Gattass, and S. Mukherjee, "A new algorithm for finding a pseudoperipheral vertex or the endpoints of a pseudodiameter in a graph," *International Journal for Numerical Methods in Engineering*, vol. 10, no. 11, pp. 913–926, 1994.

[3] A. Pothen, "Graph Partitioning Algorithms With Applications To Scientific Computing," *Parallel Numerical Algorithms, Kluwer Academical Press*, pp. 323–368, 1997.

[4] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[5] U. von Luxburg, "A tutorial on spectral clustering.," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[6] A. Pothen, H. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, 1990.

[7] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain." available online at http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf, August 1994.

[8] T. F. Chan and H. A. Van der Vorst, "Approximate and Incomplete Factorizations," in *Parallel Numerical Algorithms* (D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds.), pp. 167–202, Dordrecht: Springer, 1997.

[9] T. Manteuffel, "An incomplete factorization technique for positive definite linear systems," *Math. Comp.*, vol. 34, pp. 473–497, 1980.

[10] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," *ACM Trans. Math. Softw.*, vol. 38, Dec. 2011. The collection is available online at https://sparse.tamu.edu/.

[11] I. Moulitsas and G. Karypis, "Partitioning Algorithms for Simultaneously Balancing Iterative and Direct Methods," Technical Report 04-014, Department of Computer Science and Engineering, University of Minnesota, 2004.

[12] I. S. Duff and G. A. Meurant, "The effect of ordering on preconditioned conjugate gradients," *BIT Numerical Mathematics*, vol. 29, pp. 635–657, 1989.