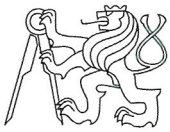


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

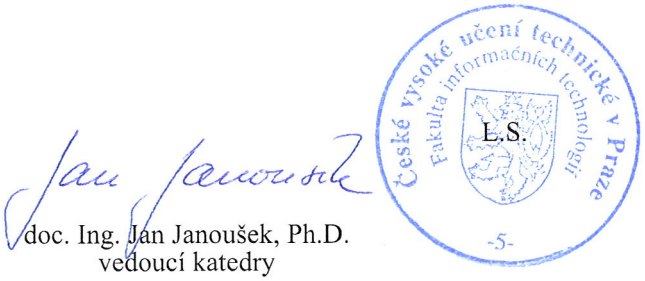
Název:	Přesné algoritmy pro geodetické číslo a metrickou dimenzi grafu
Student:	Marek Jílek
Vedoucí:	RNDr. Ondřej Suchý, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Nastudujte problém určování geodetického čísla a metrické dimenze grafu a jejich varianty.
Analyzujte některé známé algoritmy pro metrickou dimenzi grafu, zejména parametrizované maximálním počtem listů v kostře grafu.
Analyzujte známé algoritmy pro geodetické číslo grafu.
Pokuste se některé z výše uvedených algoritmů zobecnit pro vhodně parametrizovanou verzi problému určování geodetického čísla grafu, např. parametrizovanou velikostí nejmenšího vrcholového pokrytí nebo maximálním počtem listů v kostře grafu.
Vybraný z algoritmů nebo jejich zobecnění implementujte ve vhodném jazyce.
Výsledný program otestujte na vhodných vstupních datech, zhodnoťte jeho výsledky a v případě, že existují, porovnejte s jinými implementacemi algoritmů pro daný problém.

Seznam odborné literatury

David Eppstein: Metric Dimension Parameterized by Max Leaf Number. J. Graph Algorithms Appl. 19(1): 313-323 (2015)
Mitre Costa Dourado, Fábio Protti, Dieter Rautenbach, Jayme Luiz Szwarcfiter: Some remarks on the geodetic number of a graph. Discrete Mathematics 310(4): 832-837 (2010)



doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Přesné algoritmy pro geodetické číslo a metrickou dimenzi grafu

Marek Jílek

Vedoucí práce: RNDr. Ondřej Suchý, Ph.D.

10. května 2016

Poděkování

Chtěl bych poděkovat všem lidem, kteří mě během vytváření této práce podporovali. Rodině, mé slečně a především vedoucímu mé práce, který dokázal být velice chápavý a ochotný vždy, když toho bylo zapotřebí.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. května 2016 .....

Odkaz na tuto práci

Jílek, Marek. *Přesné algoritmy pro geodetické číslo a metrickou dimenzi grafu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá vytvořením parametrizovaného algoritmu hledajícího geodetické číslo na grafech s nízkým vrcholovým pokrytím. Geodetické číslo je velikost minimální množiny takové, že na nejkratších cestách mezi jednotlivými dvojicemi vrcholů množiny leží všechny vrcholy grafu. Na začátku jsou rozebrány řešení podobných problémů, například parametrizovaný algoritmus pro hledání metrické dimenze grafu. V práci je ukázáno, že lze využít určitých vlastností bipartitních grafů a grafů s nízkým vrcholových pokrytím pro zjednodušení složitosti algoritmu. Popsaný algoritmus je parametrizovaný velikostí množiny vrcholového pokrytí. Algoritmus je implementován v jazyce `C++`. Testováním byla potvrzena očekávaná redukce složitosti. Tento algoritmus lze tedy s úspěchem používat na zmíněných specifických grafech.

Klíčová slova geodetické číslo, parametrizovaný algoritmus, optimalizace složitosti algoritmu, metrická dimenze, NP-úplné problémy

Abstract

This bachelor thesis deals with developing a parameterized algorithm searching for the minimal geodetic number of graphs with low vertex cover. A geodetic number is a cardinality of the smallest set, such that on the shortest paths between pairs of vertices contained in the set lies every vertex of the graph. At the beginning of the thesis, a few solutions of similar problems are analysed, such as a parameterized algorithm for searching a metric dimension of a graph. This paper demonstrates the use of some specific properties of bipartite graphs and graphs with low vertex cover to simplify the complexity of the algorithm. The described algorithm is implemented in `C++`. Using tests, the expected complexity reduction was confirmed. Therefore, this thesis proves that we can use this algorithm on the specific graphs previously mentioned with success.

Keywords geodetic number, parameterized algorithm, algorithm's complexity optimization, metric dimension, NP-complete problems

Obsah

Seznam obrázků

Úvod

S grafovými algoritmy se setkáváme každý den: mobilní telefon hledá nejoptimálnější cestu do práce, při prohlížení emailů routery směrují pakety v síti. Příkladů bychom jistě našli mnoho. Často jde o výpočetně náročné úlohy, právě grafové algoritmy často spadají do třídy tzv. NP-úplných problémů a ani nejvýkonnější počítače často nedokáží vypočítat řešení v rozumném čase. Proto je vhodné zamýšlet se, zda nemůžeme využít znalostí struktury grafů pro značné zjednodušení hledání řešení problémů.

Jedním z NP-úplných problému je i hledání geodetického čísla grafu. Rozhodli jsme se řešit tento problém na grafech s nízkým vrcholovým pokrytím. Problém geodetického čísla je poměrně málo probádaný problém. Geodetické číslo grafu lze použít například v logistice podniků, lokační teorii a dalších oborech. Naši práci bude možné použít jako odrazový můstek pro další výzkum geodetického čísla.

V první části se věnujeme analýze současných znalostí o geodetickém čísle a rozboru parametrizovaného algoritmu řešící příbuzný problém hledání metrické dimenze grafu. V další části řešíme problém hledání minimální geodetické množiny na bipartitních grafech, kde jedna partita grafu je o mnoho větší, než druhá. Zároveň popisujeme parametrizovaný algoritmus, který na zmíněných grafech hledá geodetické číslo. V poslední části ukážeme souvislost mezi bipartitními grafy a grafy s nízkým vrcholovým pokrytím a změny, které je potřeba v algoritmu provést, aby fungoval i na grafech s nízkým vrcholovým pokrytím. Na úplný závěr provedeme měření konkrétní implementace algoritmu v jazyce `C++`.

Kapitola 1

Cíl práce

Cílem literární rešerše je získání obecného povědomí o problematice hledání geodetického čísla grafu. Rozebereme si článek F. Hararyho a kol., ve kterém bylo prvně definováno geodetické číslo a ukážeme si větu, díky které můžeme na split grafech hledat geodetické číslo v lineárním čase. Dalším cílem je analýza parametrizovaného algoritmu pro hledání metrické dimenze grafu.

Cílem praktické části bakalářské práce je vytvoření parametrizovaného algoritmu řešícího problém hledání geodetického čísla na grafech s nízkým vrcholovým pokrytím. Dále je cílem naimplementovat tento algoritmus v jazyce `C++` a danou implementaci otestovat.

Kapitola 2

Představení problému

V celé práci budeme jako G značit libovolný graf. Množinu vrcholů grafu G budeme značit jako V. Počet vrcholů grafu (tedy $|V|$) budeme značit jako n.

Prvním, kdo definoval geodetické číslo grafu, byl Frank Harary a kol. ve svém článku z roku 1993 [5]. Stejně jako v originálním článku si i my nejprve nadefinujme geodetický uzávěr. Uzávěr dvojice vrcholů $u, v \in V$ je definován jako množina všech vrcholů ležících na cestách nejkratší délky mezi vrcholy u a v. Uzávěr množiny $S \subseteq G$ je definován jako sjednocení geodetických uzávěrů všech dvojic vrcholů v množině. V práci budeme uzávěr množiny značit jako S^c. Množina $S \subseteq G$ se nazývá geodetická, pokud její geodetický uzávěr obsahuje všechny vrcholy grafu. Geodetické číslo grafu G je potom mohutnost nejmenší geodetické množiny.

Vrcholové pokrytí je taková množina $S \subseteq G$, že každá hrana grafu G má alespoň jeden krajní vrchol v množině S. Graf s nízkým vrcholovým pokrytím je tedy takový graf, který splňuje $|S| \ll |V|$. Příkladem takového grafu je hvězda.

Metoda větví a hranic byla poprvé představena v roce 1960 A. H. Landem a A. G. Doigem [7]. Jde o obecnou metodu omezení prohledávání stavového prostoru. Základní myšlenkou metody je to, že před expandováním nového stavu zkontroluji, zda je daný stav perspektivní, tedy zda je možné v jeho podstromu najít lepší řešení, než je současné nalezené. Pokud máme jistotu že ne, stav je zahozen (není expandován) a s ním i celý jeho podstrom. Pokud nalezneme nové minimum aktualizujeme předchozí. Tím dochází k postupnému zpřesňování a k častějšímu prořezávání neperspektivních větví. V důsledku toho je redukována velikost prohledávaného stavového prostoru.

Simpliciální vrchol $u \in V$ je takový vrchol, že jeho sousedé indukují kliku grafu G.

Marek Cygan a kol. ve své knize Parameterized Algorithms [2] definují parametrizované algoritmy pomocí tzv. FPT (fixed parameter tractable: parametrizovaně dostupný) třídy algoritmů. Problém nazveme FPT, pokud existuje algoritmus $\mathcal{A}$, funkce f a konstanta c taková, že je algoritmus schopen rozhod-

nout libovolný vstup v čase ohraničeném $f(k) \cdot |(x,k)|^c$, kde (x,k) reprezentuje libovolný vstup a k je parametrem. Jde nám tedy o to, najít určitý parametr, který bude dobře reprezentovat vnitřní strukturu problému. Pokud zajistíme, aby tento parametr byl malý, výsledná složitost algoritmu bude asymptoticky nižší než exponenciální.

Naším cílem je vytvořit algoritmus parametrizovaný velikostí množiny vrcholového pokrytí. Algoritmus bude hledat jednu minimální geodetickou množinu vstupního grafu, a tím pádem i geodetické číslo tohoto grafu (velikost nalezené množiny).

Kapitola 3

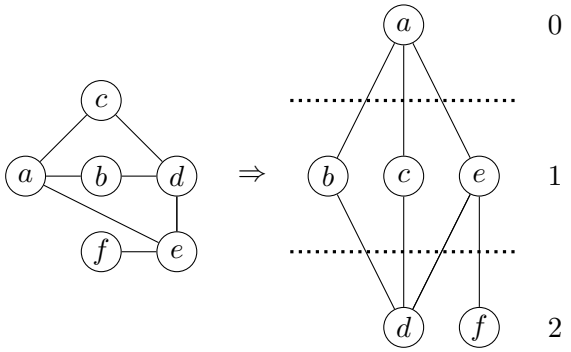
Literární rešerše

V literární rešerši se podíváme na existující algoritmy hledající geodetické číslo jak na obecném grafu, tak i na speciálním případu grafu. Dále si ukážeme parametrizovaný algoritmus řešící podobný problém: hledání metrické dimenze grafu. Představený algoritmus je parametrizovaný vlastností minimálního počtu vrcholů v kostře grafu G.

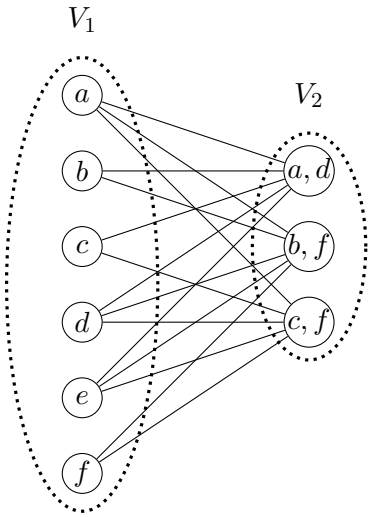
3.1 Hararyho algoritmus hledající geodetické číslo obecného grafu G

Frank Harary a kol. ve svém článku z roku 1993 [5] spolu s geodetickým číslem grafu představil i algoritmus hledající toto číslo na obecném grafu G. Jakým způsobem jeho algoritmus danou množinu hledá, si popíšeme v následující sekci.

Článek je poměrně starý a je v něm několik nesrovnalostí. Například není jasné, zda je MGC relací nebo množinou. My budeme tedy předpokládat, že jde o relaci. Dále je zde několik těžko pochopitelných částí v syntaxi pseu-



Obrázek 3.1: Transformace grafu G na vrstvený graf $G(a)$.



Obrázek 3.2: Vytvoření bipartitního grafu dle algoritmu F. Hararyho a kol.

dokódu, díky kterým je poměrně náročné zjistit detailní průběh algoritmu. Pokusíme se tedy představit alespoň základní myšlenky článku.

3.1.1 Maximální geodetický uzávěr

Autoři článku nadefinovali relaci maximální geodetický uzávěr vrcholů x a y grafu G označenou $MGC(x, y)$ následujícím způsobem: dvojice (x, y) taková, že $x, y \in V, x \neq y$, jsou v relaci $MGC(x, y)$ pokud geodetický uzávěr $\{x, y\}^c$ není podmnožinou geodetického uzávěru jiné dvojice vrcholů.

Dále si autoři vytvořil algoritmus pro hledání $mgc(x)$, ze kterého později pravděpodobně (z článku tento krok není patrný) získají vrcholy v relaci $MGC(x, y)$.

Algoritmus pro hledání $mgc(x)$ funguje následovně. Nejprve rozdělí graf G do vrstev a to tak, že pouze vrchol x je ve vrstvě 0 a pro $k > 0$ je vrchol u vzdálen od x na vzdálenost k je umístěn ve vrstvě k. Příklad této transformace můžeme vidět na obrázku 3.1. Pokud si vytvoříme takový graf, potom $mgc(x)$ je množina všech koncových vrcholů vzniklého grafu (tedy vrcholů, které nemají propojení na vrcholy v nižších vrstvách). Označme si tuto množinu těchto koncových vrcholů jako $mgc(x)$. Všimněme si, že platí $\{x, mgc(x)\}^c = V$.

Podle nás jsou vrcholy x a y v relaci $MGC(x, y)$, pokud platí $y \in mgc(x)$ a zároveň $x \in mgc(y)$. Důkaz správnosti tohoto kroku ale v originálním článku chybí.

3.1.2 Vytvoření bipartitního grafu z dvojic v relaci $MGC(x, y)$

Dále je vytvořen nový bipartitní graf G' z grafu G s partitami V_1 a V_2 takovými, že V_1 obsahuje všechny vrcholy grafu G a V_2 obsahuje množinu všech dvojic vrcholů, které jsou vzájemně relaci $MGC(x, y)$. Jak dostat všechny tyto dvojice, na základě předchozího algoritmu pro hledání $mgc(x)$, autoři neuvádí.

V případě grafu na obrázku 3.1 by partity obsahovali: $V_1 = \{a, b, c, d, e, f\}$, $V_2 = \{(a, d), (b, f), (c, f)\}$. Hrany mezi vrcholy potom povedeme tak, že mezi vrcholem $v \in V_1$ a dvojicí $(x, y) \in V_2$ vede hrana, pokud platí $v \in \{x, y\}^c$.

Bipartitní graf sestrojený na základě těchto pravidel z grafu na obrázku 3.1 můžeme vidět na obrázku 3.2. Například vrchol (a, d) je propojen s vrcholy a, b, c, d a e, protože platí $\{a, d\}^c = \{a, b, c, d, e\}$.

3.1.3 Popis vlastního algoritmu

Jak již bylo zmíněno na začátku sekce, článek obsahuje několik nesrovnalostí. Právě proto funkce algoritmu není zřejmá hned hned na několika místech. Z toho důvodu je potřeba si mnoho částí domyslet. Pokusíme se představit stručný popis toho, jak si myslíme, že autoři algoritmus zamýšleli.

Algoritmus hledající geodetické číslo využívá bipartitní graf vytvořený v předchozích sekcích. Pokouší se najít dominující množinu tohoto bipartitního grafu (ta potom bude i geodetickou množinou původního grafu). Stavový prostor prohledává jako strom. Postupně připojuje jednotlivé dvojice a ke každému vrcholu $v \in V_1$ počítá, v kolika uzávěrech již přidaných dvojic se nachází.

Pokud jsou již všechny vrcholy alespoň v jednom uzávěru připojených dvojic každá další dvojce nepřináší žádný efekt. Všechny nepoužité dvojice jsou označeny mínusem. Stejně tak jsou označeny mínusem všechny nepoužité dvojice, pokud je nalezená množina větší než současné minimum. Zda jde o nalezené minimum ověřují autoři tak, že zkontrolují, zda je počet dvojic aktuálního řešení menší než počet dvojic současného minima.

Algoritmus běží tak dlouho, dokud nejsou všechny dvojice označeny mínusem. Dále nemá cenu provádět expanzi, protože jsme buď vyzkoušeli přidat všechny dvojice, nebo by další prohledávání bylo zbytečné (jistě nalezené řešení bude horší než současné minimum).

Množina vrcholů v jednotlivých dvojicích minimální množiny nalezených dvojic je potom zřejmě sjednocena, a tím dostaneme hledanou minimální geodetickou množinu. Toto je však pouze naše domněnka. Autoři tento krok v originálním článku neuvádí. Nabízí se však otázka, zda se nemůže stát, že algoritmus určí jako minimum dvojici dvojic $\{(a, b), (c, d)\}$, která má po sjednocení 4 prvky, místo trojice dvojic $\{(a, b), (b, c), (a, c)\}$, jejichž sjednocení obsahuje 3 prvky. Tím by totiž nebyla nalezena minimální geodetická množina a algoritmus by chybně určil geodetické číslo grafu jako 4 místo 3. To z algoritmu není zřejmé.

3.1.4 Složitost hledání geodetického čísla

V článku autoři tvrdí, že problém hledání geodetického čísla je NP-úplný problém. Nepředkládají pro to ale žádný důkaz. Proto je třeba najít jiný zdroj, kde je NP-úplnost geodetického čísla dokázána.

Věta 1. *Nechť G je graf. Problém hledání geodetického čísla grafu G je NP-úplný problém.*

Důkaz. Důkaz lze dohledat v článku M. Aticiho z roku 2002 [1]. □

3.2 Geodetické číslo split grafu

V následující sekci si ukážeme vlastnosti minimálního geodetického čísla na split grafech. Split graf je takový graf, který lze rozdělit na podmnožiny V_1 a V_2, pro které platí $V_1 \cup V_2 = V$ takové, že V_1 je maximální nezávislá množina a V_2 je klika. Vyjdeme z článku Mitra Dourada a kol. z roku 2009 [3]. Zároveň si odvodíme i jednu důležitou vlastnost simpliciálních vrcholů při hledání geodetického čísla na obecném grafu G.

Vybereme z článku pro nás nejdůležitější větu.

Věta 2. *Mějme split graf G, mající množiny V_1 a V_2 takové, že V_1 je maximální nezávislá množina a V_2 je klika. Množina S obsahuje všechny simpliciální vrcholy grafu G. Označíme si množinu U, obsahující všechny vrcholy $u \in (V_2 \setminus S)$ mající právě jednoho souseda v množině V_1.*

Potom můžeme říci, že pokud je $|U| = 0$ potom platí $g(G) = |S|$. Pokud množina U je neprázdná a zároveň existuje vrchol $v \in (V_2 \setminus S)$ takový, že množina jeho sousedů v množině V_1 má prázdný průnik se sjednocením množin sousedů všech ostatních vrcholů v množině $V_2 \setminus S$, potom platí $g(G) = |S| + 1$. Pokud žádný vrchol v neexistuje, potom platí $g(G) = |S| + 2$.

Důkaz. Důkaz tvrzení lze dohledat v originální literatuře. □

V důkazu tvrzení se nachází pro nás významné tvrzení o simpliciálních vrcholech. V originálním článku tvrzení dokázáno není. Dokážeme si ho tedy sami.

Lemma 3. *Nechť G je graf a S je množinou všech simpliciálních vrcholů grafu G. Potom pro všechny geodetické množiny $\mathcal{G}$ grafu G platí $S \subseteq \mathcal{G}$.*

Důkaz. Důkaz provedeme sporem. Tvrdíme, že existuje nejkratší cesta mezi vrcholy $u, v \in V \setminus \{s\}$ taková, že vede přes simpliciální vrchol s. Jelikož má vrchol s sousedy pouze v klice $C \subseteq G$, existují sousedé vrcholu s v klice C (pojmenujme si je c_1 a c_2) takové, že přes ně vede nejkratší cesta mezi dvojicemi vrcholů (u, s) a (v, s). Jelikož ale víme, že c_1 a c_2 jsou v klice, jistě jsou propojeny hranou. Tím pádem existuje kratší cesta, než byla ta přes

vrchol s. Dostali jsme spor, tím pádem žádná nejkratší cesta přes simpliciální vrchol s neexistuje a všechny simpliciální vrcholy s musí být v každé geodetické množině grafu. □

3.3 Algoritmus hledající metrickou dimenzi parametrizovaný maximálním počtem listů

V poslední sekci literární rešerše se podíváme na parametrizovaný algoritmus řešící problém příbuzný s problémem hledání geodetického čísla: hledání metrické dimenze grafu. Představen bude algoritmus Davida Eppsteina z roku 2015 [4].

3.3.1 Úvod do problému

Nadefinujme si nejprve základní pojmy, které se v článku nacházejí.

Metrická dimenze byla poprvé definována nezávisle na sobě v roce 1975 Peterem J Slaterem [9] a v roce 1976 autory Frankem Hararym a Robertem Melterem [6]. Oba články nejprve definují rozlišovací množinu. Množina $S \subseteq G$ je rozlišovací, pokud pro libovolné dva vrcholy (označme si je u a v) grafu G existuje vrchol $s \in S$ takový, že vzdálenost mezi u a s je rozdílná od vzdálenosti v a s. Množina S si tedy lze představit jako množinu orientačních bodů grafu. Metrická dimenze grafu G je velikost její minimální rozlišovací množiny S [4].

Parametrem algoritmu byl zvolen maximální počet listů grafu G. Ten je definován jako maximum počtu listů přes všechny kostry grafu G. V dále tento počet budeme značit jako $m(G)$.

Dalším pojmem vyskytujícím se v práci je větev. Větev grafu G je maximální cesta nebo cyklus, jehož všechny vrcholy mají stupeň 2 v grafu G.

3.3.2 Teoretická příprava

Před představením samotného algoritmu uvádí autor několik základních poznatků jak o metrické dimenzi tak o maximálním počtu listů kostry grafu.

Prvním poznatkem je možnost převedení maximálního počtu listů na počet větví grafu. Graf G může obsahovat maximálně $\mathcal{O}(m(G)^2)$ větví. Naopak pokud graf obsahuje b větví, potom bude platit $m(G) < 2b$. Dále tedy autor nebude parametrizovat parametrem $m(G)$, ale místo toho parametrem b, tedy počtem větví grafu G. Na $m(G)$ bude možné algoritmus na konci převést. Algoritmus půjde použít na grafech s poměrně malým počtem větví, tedy takový graf mající převážně vrcholy stupně 2.

Je zaveden pojem nerozlišitelné množiny. Nerozlišitelná množina pro vrchol s a větvě A a B grafu G je definována jako množina všech dvojic (a, b), pro které platí $a \in A, b \in B$ a to, že vzdálenost mezi s a a je stejná jako mezi s a b.

Dále je nadefinován pojem kmen. Kmen je maximální souvislá podmnožina vrcholů větve C taková, že všechny nerozlišitelné množiny vrcholů s ve větvi C a všech dvojicích větví A a B mají stejnou kombinatorickou strukturu.

Pod pojmem „stejné kombinatorické struktury“ je myšleno to, že pokud máme vrchol s a větve A,B a posunujeme vrchol s po větvi C, nedochází ke zlomovým bodům změn nerozlišitelné množiny pro s, A a B. Tyto změny mohou nastat, pokud nejkratší cesta mezi koncovými vrcholy A nebo B a vrcholem s přestane procházet jedním koncem větve C a začne procházet druhým. Dalším případem, kdy dojde ke změně struktury, je situace, kdy koncové vrcholy větví A a B a vrcholy maximálně vzdálené od s si vymění svoje pozice, pokud jsou srovnané dle vzdálenosti od s. Takovýchto změn struktury může být pro s, A, B, a C maximálně $\mathcal{O}(1)$.

Další odvozeným vztahem je omezení počtu kmenů. Větev může mít maximálně $\mathcal{O}(b^3)$ kmenů.

Nakonec autor dokazuje, že metrická dimenze grafu G majícího b větví je velikosti $\mathcal{O}(b)$.

Důkazy všech výše zmíněných tvrzení lze dohledat v originálním článku.

3.3.3 Algoritmus

S přihlédnutím na předchozí vlastnosti metrické dimenze a grafů majících malý počet větví (a tím pádem malé číslo $m(G)$) je představen algoritmus hledající minimální rozlišovací množinu.

Algoritmus nejprve v lineárním čase zmenší všechny větve na hranu o váze délky zkrácené větve. Tato operace může být vykonána v čase $\mathcal{O}(n)$.

Potom algoritmus hledá rozlišovací množinu velikosti o maximální velikosti $\mathcal{O}(b)$ a to tak, že nejprve nalezne vhodnou velikost této množiny, a potom pro každý vrchol hledá kmen, který ho bude obsahovat. Celkem je takovýchto možností $2^{\mathcal{O}(b \log b)}$.

Při vygenerování určité možnosti je vždy třeba ověřit, zda jde o platnou rozlišovací množinu. Při ověřování je použit standardní algoritmus pro nízkodimenzionální celočíselné lineární programování. Ověření všech možností potom lze provést v čase $2^{\mathcal{O}(b \cdot \log b)} \cdot \log n$ a musíme to provést pro $\mathcal{O}(b^2)$ dvojic větví.

Složitost algoritmu je tedy $\mathcal{O}(n) + 2^{\mathcal{O}\left(b^3 \log b\right)} \cdot \log n$. Tato složitost může být převedena na $m(G)$ jako $\mathcal{O}(n) + 2^{\mathcal{O}\left(m(G)^6 \log(m(G))\right)} \cdot \log n$.

Autor tímto dokázal, že problém hledání metrické dimenze je FPT s parametrem maximálního počtu listů v kostře grafu. Přesto je algoritmus časově velice výpočetně náročný na to, aby byl použit v praxi.

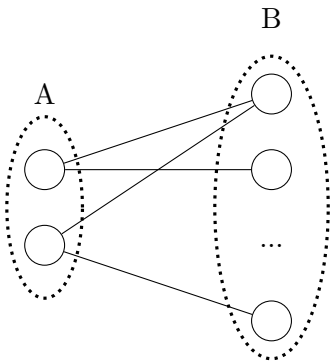
Kapitola 4

Vlastnosti geodetického čísla na bipartitních grafech

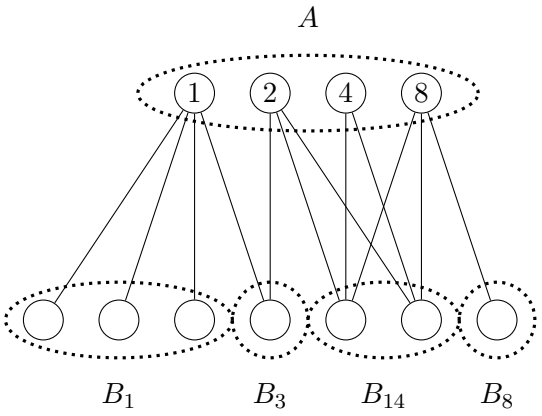
4.1 Počáteční podmínky

V této kapitole se budeme zabývat pouze souvislými bipartitními grafy. Pokud bychom na vstup dostali nesouvislý graf, lze ho rozdělit na jednotlivé bipartitní komponenty a ty poté řešit každou zvlášť. Jednotlivé partity grafu si označíme A a B. V rámci žádné partity nejsou dva vrcholy propojeny hranou (dle definice bipartitního grafu). Platí, že $A \cup B = V$.

Zaměříme se na takové bipartitní grafy, které splňují podmínky $|A| \ll |B|$. Příklad takového grafu a rozdělení na podmnožiny je vidět na obrázku 4.1. Dále tyto grafy budu nazývat „$|A| \ll |B|$ bipartitní“.



Obrázek 4.1: Příklad $|A| \ll |B|$ bipartitního grafu a rozdělení tohoto grafu na podmnožiny A a B.



Obrázek 4.2: Rozdělení partity B na homogenní skupiny a znázornění využití binární soustavy.

4.2 Rozdělení množiny B na homogenní skupiny

Pro budoucí potřeby si rozdělíme množinu B na homogenní skupiny podle sousednosti s vrcholy v množině A. Do určité skupiny budou patřit takové vrcholy, které mají stejné množiny sousedů. Tito sousedé jsou vždy v množině A.

Lemma 4. *Graf G má maximálně $2^{|A|} - 1$ homogenních skupin.*

Důkaz. Pro důkaz této vlastnosti použijeme binární soustavy. Označíme si vrcholy v množině A vzestupně mocninami dvojky $\{2^0, 2^1, \ldots, 2^{|A|-1}\}$ (například odshora dolů, nebo zleva doprava). Každou skupinu v množině B potom můžeme identifikovat součtem mocnin sousedů libovolného vrcholu ve skupině. Takovýchto indexů může být maximálně $2^{|A|}$. Jelikož ale víme, že musí jít o souvislý graf, je potřeba odečíst skupinu, kde vrcholy žádného souseda nemají. □

Příklad rozdělení do skupin a naznačení využití binární soustavy pro vytváření indexů skupin můžeme vidět na obrázku 4.2.

Dále v práci budeme používat značení B_m, kde pro m platí $m \in \{1, 2^{|A|} - 1\}$, pro označení konkrétní homogenní skupiny. Jako B_x potom budeme značit libovolnou neprázdnou homogenní skupinu z partity B.

4.2.1 Algoritmus pro rozdělení B na homogenní skupiny

Následující algoritmus rozděluje jednotlivé vrcholy množiny B do homogenních skupin. V algoritmu je využito dříve zmíněné vlastnosti identifikace skupiny, podle množiny sousedů libovolného vrcholu homogenní skupiny. Na konci

běhu algoritmu 1 dostaneme pole množin, ve kterém se nachází na pozici indexu množina vrcholů dané skupiny. Proměnná $a.index$ obsahuje index vrcholu $a \in A$, kde a nabývá postupně rostoucích hodnot $\{1, 2, 3, ...\}$. Funkce $N(u)$ vrací množinu sousedů vrcholu $u \in V$.

Algoritmus 1 Rozdělení vrcholů z B na homogenní skupiny.

```
1: sets[2^|A|] ← ∅                                   ▷ Inicializace pole množin
2: for all b ∈ B do
3:     index ← 0
4:     for all a ∈ N(b) do   ▷ Výpočet indexu skupiny, do které vrchol patří
5:         index ← index + 2^a.index
6:     end for
7:     sets[index] ← sets[index] ∪ b
8: end for
```

4.3 Geodetické číslo speciálních případů bipartitních grafů

V následující sekci se podíváme na geodetickou množinu speciálních případů bipartitních grafů a vyvodíme z nich určité pravidla pro obsah geodetické množiny. Ty později použijeme pro zjednodušení našeho algoritmu.

4.3.1 Hvězda

Triviálním případem bipartitního grafu je hvězda, tedy graf pro který platí $|A| = 1$. Pro důkaz lemmatu 6 se nám bude hodit následující pozorování.

Pozorování 5. *Nechť G je libovolný graf. Minimální geodetická množina obsahuje všechny vrcholy $v \in V$ stupně 1.*

Důkaz. Označme si libovolné dva vrcholy $x, y \in V$ různé od v. Jelikož víme, že v je stupně 1, jistě bude platit $v \notin \{x, y\}^c$. Tím pádem vrchol sám musí být v minimální geodetické množině, jinak by množina nebyla geodetickou. □

Lemma 6. *Pokud G je bipartitním grafem s partitami A, B takovými, že $|A| \leq |B|$ a $|A| = 1$ potom platí $g(G) = |B|$.*

Důkaz. V tomto speciálním případě grafu jsou všechny vrcholy v množině B stupně 1 v minimální geodetické množině, označme si ji $\mathcal{G}$. Jak víme z pozorování 5, pokud je v grafu vrchol stupně 1, pak jistě bude v geodetické množině. Menší tedy množina $\mathcal{G}$ být nemůže, jinak by nebyla geodetickou. Zároveň víme, že pro libovolné $b_1, b_2 \in B$ vzájemně různé a pro $a \in A$ platí $a \in \{b_1, b_2\}^c$. Tedy $V = B^c$. □

4.3.2 Úplný bipartitní graf

Dalším případem je úplným bipartitní graf. Pro ten platí následující tvrzení.

Lemma 7. *Pokud G je úplným bipartitním grafem s partitami A, B takovými, že $|A| \leq |B|$ a $|A| \geq 4$ potom platí $g(G) = 4$.*

Důkaz. Libovolné dva různé vrcholy a_1, a_2 v A a libovolné dva různé vrcholy b_1, b_2 v B tvoří minimální geodetickou množinu. Všechny vrcholy a v A leží na nejkratší cestě (délky 2) mezi b_1 a b_2, tím pádem $A \subseteq \{b_1, b_2\}^c$. Stejně tak platí, že $B \subseteq \{a_1, a_2\}^c$. Z toho vyplývá, že V je $\{a_1, a_2, b_1, b_2\}^c$. Množina je tedy geodetickou množinou.

Pokud $|A| = 4$ může navíc být minimální geodetickou množinou celá partita A. A je geodetickou množinou, protože pro libovolné dva vrcholy a_1, a_2 v A platí, že $B \in \{a_1, a_2\}^c$. Jelikož A je jistě v A^c (dle definice uzávěru), potom platí $V = A^c$. To, že celá partita A je geodetickou množinou zřejmě platí vždy, když můžeme vybrat dva různé vrcholy, tedy když platí $|A| > 1$.

Minimalitu dokážeme sporem. Předpokládejme pro spor, že existuje geodetická množina $\bar{\mathcal{G}}$, která má velikost menší než 4. BÚNO můžeme předpokládat, že platí $g(G) = 3$. Přidáním vrcholů do $\mathcal{G}$ jistě geodetičnost nenarušíme. Tyto tři vrcholy geodetické množiny si označme x, y a z. Mohou nastat 4 situace.

První z nich je $\{x, y, z\} \in A$. Jelikož ale víme, že $|A| \geq 4$ pak jistě $(A \setminus \{x, y, z\}) \notin \bar{\mathcal{G}}^c$ a množina $\bar{\mathcal{G}}$ tedy není geodetická.

Druhou situací je $\{x, y\} \in A, z \in B$. Znovu vidíme, že $(A \setminus \{x, y\}) \notin \bar{\mathcal{G}}^c$ a množina tedy není geodetická.

Obdobně bychom mohli ukázat nemožnost existence pro situace $\{x, y, z\} \in B$ a $\{x, y\} \in B, z \in A$. Odebráním vrcholů z množiny, která je geodetická jistě nevytvoříme geodetickou množinu, proto zřejmě výše zmíněné bude platit i pro menší množiny. Sporem jsme tedy dokázali, že minimální velikost geodetické množiny $\mathcal{G}$ je 4. □

Lemma 8. *Pokud G je úplným bipartitním grafem s partitami A, B takovými, že platí $|A| \leq |B|$ a $2 \leq |A| \leq 3$, potom platí $g(G) = |A|$.*

Důkaz. To, že partita A je geodetickou množinou (pro $|A| > 1$) jsme dokázali v lemmatu 7. Dokažme nyní minimalitu této geodetické množiny. Geodetické číslo $g(G)$ nemůže být rovno 1 (na grafech $|V| > 1$), protože vždy potřebujeme alespoň dva vrcholy, jejichž uzávěr bude obsahovat ostatní vrcholy grafu. Víme tedy, že geodetické číslo nebude rovno 1.

Zbývá dokázat že pokud platí $|A| = 3$, pak neexistuje geodetická množina velikosti 2. Důkaz znovu provedeme sporem. Tvrdíme, že existuje $\bar{\mathcal{G}}$, která je minimální geodetickou množinou a její velikost je 2. Její vrcholy označme x a y. První situací, která může nastat, je $x, y \in A$. Potom víme, že $(A \setminus \{x, y\}) \notin \bar{\mathcal{G}}^c$. Množina $\bar{\mathcal{G}}$ tedy není geodetickou množinou. Druhou situací je $x \in A, y \in B$. Všimněme si, že platí $\{x, y\}^c = \{x, y\}$, protože jde o sousední vrcholy. Tedy platí, že $(V \setminus \{x, y\}) \notin \bar{\mathcal{G}}^c$. Množina $\bar{\mathcal{G}}$ tedy znovu není geodetickou množinou.

Třetí situace, tedy situace kdy platí $x, y \in B$, je obdobná s první situací, protože víme, že platí $|B| \gg |A|$, a tím pádem platí $|B| \geq 3$. □

Partita B je v případě úplného grafu rozdělena pouze do jedné skupiny, protože všechny vrcholy partity B mají shodné množiny sousedů (všechny vrcholy z partity A). V případě kdy $|A| > 4$, jsou v minimální geodetické množině dva vrcholy homogenní skupiny nezávisle na velikosti skupiny. V případě kdy $|A| = 1$, jsou v geodetické množině všechny vrcholy skupiny, tedy všechny vrcholy partity B.

4.3.3 Využití jednoho vrcholu ze skupiny

V úplných bipartitních grafech jsme si všimli, že někdy je potřeba do minimální geodetické množiny použít všechny vrcholy určité homogenní skupiny, někdy dva vrcholy a někdy žádný. Existuje ale i takový graf, ve kterém v minimální geodetické množině bude použit právě jeden vrchol určité homogenní skupiny?

Lemma 9. *Existují grafy, jejichž minimální geodetická množina obsahuje právě 1 vrchol z homogenní skupiny B_x (pro kterou platí $|B_x| > 1$).*

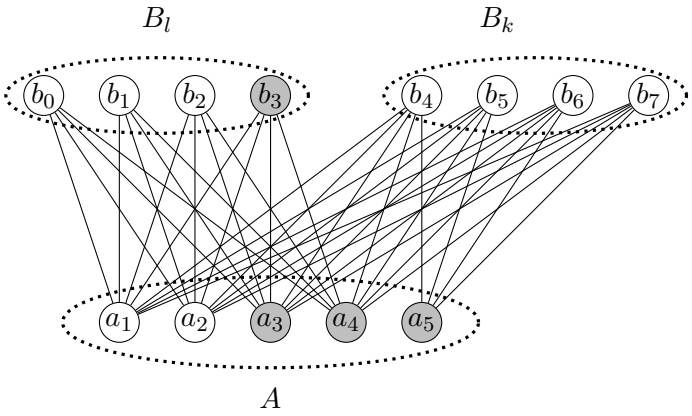
Důkaz. Jelikož potřebujeme dokázat existenci, stačí nám najít jeden konkrétní graf, pro který lemma platí. Příklad takového grafu můžeme najít na obrázku 4.3. Jedna z možných minimálních geodetických množin je zvýrazněna šedě. Skupiny B_k a B_l jsou opravdu odlišnými homogenními skupinami, liší se sousedstvím s vrcholem a_5. Vyznačená množina je opravdu geodetická, vidíme, že $B_l \subseteq \{a_3, a_4\}^c$, $B_k \subseteq \{a_4, a_5\}^c$ a $A \subseteq \{b_3, a_5\}^c$. Tedy platí, že $V = \{a_3, a_4, a_5, b_3\}^c$.

Pro spor s minimalitou předpokládejme, že existuje řešení velikosti 3. Nejprve zvolme takové prvky geodetické množiny, aby celá partita B byla v geodetickém uzávěru množiny. Tyto množiny (velikost menší nebo rovno třem) jsou tři. Rozeberme si každou zvlášť.

- Dvojice $x, y \in (A \setminus a_5)$ navzájem různých vrcholů. V následujícím kroku potřebujeme dostat do geodetického uzávěru zbývající vrcholy množiny A, aby byla množina geodetickou. Pokud ale přidáme libovolný vrchol z množiny B_l, nebo z množiny B_k, nebo vrchol a_5, nikdy nevytvoříme geodetickou množinu. Vrcholy $A \setminus \{a_5, x, y\}$ se nikdy nedostanou do geodetického uzávěru množiny.

- Trojice $x, y, z \in A$ navzájem různých vrcholů. V tomto případě již máme tři vrcholy, ale nejde o geodetickou množinu, protože $\{x, y, z\}^c = V \setminus (A \setminus \{x, y, z\})$. A jelikož $A \setminus \{x, y, z\}$ je neprázdná množina není množina $\{x, y, z\}$ geodetickou množinou.

- Trojice $\{a_5, u, v\}$, kde $u \in A \setminus \{a_5\}$ a $v \in B_l$. I v tomto případě již máme tři vrcholy, ale znovu nejde o geodetickou množinu, protože platí



Obrázek 4.3: Případ grafu, kdy je v minimální geodetické množině právě jeden vrchol z určité homogenní skupiny.

$\{x, y, z\}^c = V \setminus (B_l \setminus \{u\})$. A jelikož $A \setminus \{x, y, z\}$ je neprázdná množina, není $\{a_5, u, v\}$ geodetickou množinou.

Dostali jsme tedy spor. Minimální geodetická množina grafu vyznačeného na obrázku je velikosti čtyři, tedy geodetické číslo grafu G je rovno čtyřem. □

4.4 Výstupy pozorování

Na základě pozorování chování geodetického čísla na předchozích případech grafů můžeme odvodit několik základních poznatků o obsahu geodetické množiny bipartitních grafů.

V prvním lemmatu ukážeme, že pro libovolný bipartitní graf umíme najít poměrně malou množinu, která bude vždy geodetická.

Lemma 10. *Nechť graf G je bipartitním grafem s partitami A, B takovými, že platí $|A| \leq |B|$. Označme si množinu všech vrcholů stupně 1 z partity B jako B_{deg1}. Potom množina $A \cup B_{deg1}$ je geodetickou množinou. Nemusí však být minimální.*

Důkaz. Toto tvrzení si dokážeme sporem. Pro spor předpokládejme, že množina $A \cup B_{deg1}$ není geodetickou množinou. Tedy že existuje vrchol (označím si ho b) v množině B, který není stupně 1 a zároveň není na žádné nejkratší cestě mezi dvěma vrcholy z $A \cup B_{deg1}$. Takový vrchol vždy musí sousedit alespoň s dvěma vrcholy v množině A, označme si je a_1 a a_2. Jelikož jde o bipartitní graf, mezi a_1 a a_2 nevede hrana. Nejkratší cesty (může jich být více) mezi těmito dvěma vrcholy jsou tím pádem délky 2 a právě jedna z nich vede přes vrchol b. Tedy jistě alespoň na jedné nejkratší cestě vrchol b leží. Dostali jsme spor, množina $A \cup B_{deg1}$ je vždy geodetická. □

Předchozí lemma dále doplníme o upřesnění, že není vždy potřeba celé partity A a i tak dostaneme geodetickou množinu.

Lemma 11. *Nechť graf G je bipartitním grafem s partitami A, B takovými, že platí $|A| \leq |B|$. Označme si množinu všech vrcholů stupně 1 z partity B jako B_{deg1}. Dále si označme množinu $A_{neigh1} \subseteq A$, která obsahuje vrcholy, které sousedí alespoň s jedním vrcholem v partitě B stupně 1. Potom množina $(A \setminus A_{neigh1}) \cup B_{deg1}$ je geodetickou množinou grafu G. Stále však nemusí být minimální.*

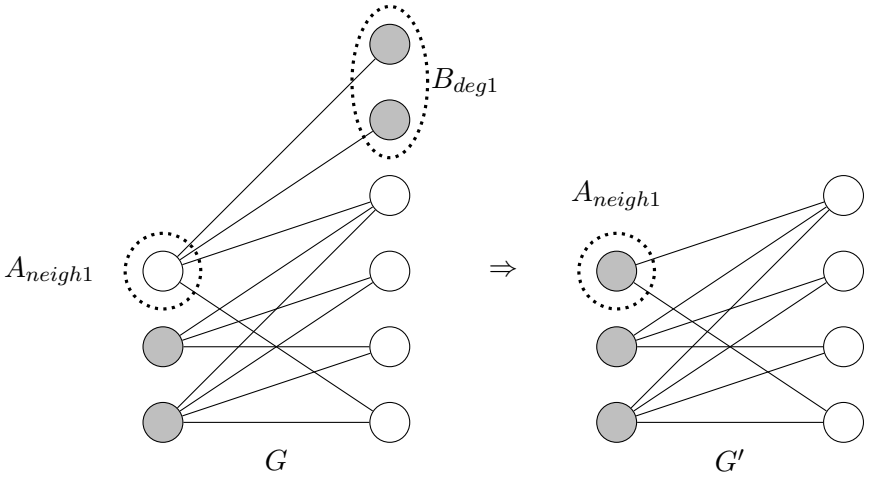
Důkaz. Důkaz provedeme sporem. Pro spor předpokládejme, že existuje taková množina $(A \setminus A_{neigh1}) \cup B_{deg1}$, která není geodetická. Tedy že existuje alespoň jeden vrchol (označme si ho x) takový, že nenáleží do uzávěru zmíněné množiny. Tento vrchol x musí být buď v množině $B \setminus B_{deg1}$, nebo v množině A_{neigh1}. Jak jsme dokázali v lemmatu 10, v $B \setminus B_{deg1}$ jistě ležet nemůže. Potom tedy musí ležet v množině A_{neigh1}. To znamená, že má souseda v množině B_{deg1}, který náleží do geodetické množiny. Jelikož víme, že geodetická množina bude vždy velikosti alespoň 2 a že je graf spojitý, pak zřejmě x musí ležet na nějaké nejkratší cestě mezi vrcholem, a tím pádem je i v uzávěru geodetické množiny. Dostali jsme spor, množina $(A \setminus A_{neigh1}) \cup B_{deg1}$ je vždy geodetická. □

Lemma 12. *Nechť graf G je bipartitním grafem s partitami A, B takovými, že platí $|A| \leq |B|$. Z každé homogenní skupiny partity B bude v minimální geodetické množině grafu G 0, 1, 2, nebo všechny vrcholy skupiny.*

Důkaz. Označme si minimální geodetickou množinu grafu G jako $\mathcal{G}$. To, že existuje graf nemající v geodetické množině žádný vrchol určité homogenní skupiny, jsme dokázali v lemmatu 7 v části, kdy $\mathcal{G} = A$. Existenci grafu, ve kterém je použit jediný vrchol skupiny jsme dokázali v lemmatu 9. V lemmatu 7 jsme dokázali existenci grafů, které mají v $\mathcal{G}$ použity 2 vrcholy určité homogenní skupiny. To, že existují grafy, ve kterých jsou použity všechny vrcholy skupiny, jsme dokázali v lemmatu 6.

Zbývá dokázat fakt, že těchto vrcholů nikdy nebude jiný počet. To si dokážeme sporem. Pro spor předpokládejme, že existuje graf, pro který existuje minimální geodetická množina (označená $\mathcal{G}$), která má z nějaké homogenní skupiny B_x použito m vrcholů, kde $2 < m < |B_x|$. BÚNO m bude rovno třem, označme si vrcholy $b_1, b_2, b_3 \in (\mathcal{G} \cap B_x)$.

Jako x si označme takový vrchol, pro který platí $x \in \mathcal{G}^c$ a zároveň platí $x \notin \{\mathcal{G} \setminus \{b_1, b_2, b_3\}\}^c$. Tato situace mohla nastat pouze ve dvou případech. Prvním případem, který mohl nastat: $x \in \{b_k, y\}^c$, kde platí $k = 1, 2, 3$ a $y \in \mathcal{G} \setminus \{b_1, b_2, b_3\}$. Potom stačí do geodetické množiny přidat pouze jeden vrchol ze skupiny B_x. Druhým případem by byla situace, kdy platí $x \in \{b_k, b_l\}^c$, kde $k, l \in \{1, 2, 3\}$ a $k \neq l$. Jiný případ nastat nemůže (protože víme, že velikost skupiny je $|B_x| > 3$). V druhém stačí, když do geodetické množiny



Obrázek 4.4: Ukázka transformace G na G'.

použijeme pouze dva vrcholy ze skupiny B_x. Dostali jsme tedy spor s minimalitou geodetické množiny, vždy bude použito pouze 0, 1, 2, nebo všechny vrcholy skupiny. □

V následujících lemmatech si ukážeme dvě transformace grafu odstraněním vrcholů stupně 1 a odvodíme, jak se bude měnit při těchto transformacích minimální geodetická množina grafů. Toho později využijeme k optimalizacím našeho algoritmu. Grafické znázornění první transformaci můžeme vidět na obrázku 4.4.

Lemma 13. *Nechť graf G je bipartitní graf mající partity A, B takové, platí že $|A| \leq |B|$ a mající geodetickou množinu $\mathcal{G}$. Vytvoříme nový graf G' takový, že z grafu G odstraníme všechny vrcholy stupně 1 z partity B (označme si množinu všech těchto vrcholů B_{deg1}). Dále označme množinu $A_{neigh1} \subseteq A$, množinu vrcholů z partity A sousedící s vrcholy stupně 1 v B. Potom je množina $\bar{\mathcal{G}} = (\mathcal{G} \setminus B_{deg1}) \cup A_{neigh1}$ geodetickou množinou grafu G'.*

Důkaz. Důkaz provedeme sporem. Pro důkaz předpokládejme, že $\bar{\mathcal{G}}$ není geodetickou množinou grafu G'. Tedy existuje takový vrchol $v \in (V \setminus \{A_{neight1} \cup B_{deg1} \cup \mathcal{G}\})$, pro který platí $v \in \mathcal{G}^c$ a zároveň $v \notin ((\mathcal{G} \setminus B_{deg1}) \cup A_{neigh1})^c$.

Pokud platí $v \in \{x, y\}^c$, kde $x, y \in (\mathcal{G} \setminus B_{deg1})$, pak se nic nezmění a vrchol v bude stále v uzávěru těchto dvou vrcholů. Pokud $v \in \{x, b_{deg1}\}^c$, kde $x \in (\mathcal{G} \setminus B_{deg1})$ a $b_{deg1} \in B_{deg1}$, potom víme, že pro každý vrchol b_{deg1} existuje jeho soused v partitě A. Tím pádem vždy můžeme najít takové $a_{neigh1} \in A_{neight1}$, že bude platit $v \in \{x, a_{neigh1}\}^c$.

V obou případech jsme dostali spor, množina $\bar{\mathcal{G}}$ je geodetickou množinou grafu G'. □

Následující lemma popisuje opačný směr předchozí transformace.

Lemma 14. *Nechť graf G je bipartitní graf mající partity A, B takové, že platí $|A| \leq |B|$. Vytvoříme nový graf G' takový, že z grafu G odstraníme všechny vrcholy stupně 1 z partity B (označme si množinu všech těchto vrcholů B_{deg1}). Dále si označíme množinu $A_{neigh1} \subseteq A$, množinu vrcholů z partity A, sousedící s vrcholy stupně 1 v B. Označme si $\bar{\mathcal{G}} = (\mathcal{G} \setminus B_{deg1}) \cup A_{neigh1}$ jako geodetickou množinou grafu G'. Pokud $\bar{G}$ je geodetická množina G', potom $G = (\bar{G} \setminus A_{neigh1}) \cup B_{deg1}$ je geodetická množina grafu G.*

Důkaz. Důkaz je obdobný jako v lemmatu 13. □

Vidíme tedy, že daná transformace je oboustranná (vratná). Tato transformace grafu nám pomůže snížit složitost vyhodnocení, zda je určitá množina geodetická.

Všech zmíněných vlastností využijeme pro zjednodušení hledání $g(G)$ na $|A| \leq |B|$ bipartitních grafech.

Kapitola 5

Algoritmus pro hledání $g(G)$ na $|A| \ll |B|$ bipartitních grafech

Ve větě 1 jsme si ukázali, že hledání geodetického čísla na obecných grafech je NP-úplný problém. Složitost naivního algoritmu řešící tento problém roste exponenciálně s počtem vrcholů. Proto je vhodné zamyslet se, zda lze využít vypozorovaných vlastností bipartitního grafu ke snížení celkové složitosti problému.

V následující kapitole si popíšeme algoritmus využívající specifických vlastností $|A| \ll |B|$ bipartitních grafů.

Pro zjednodušení popisu algoritmu jsme rozdělili celkový průběh do jednotlivých funkcí. Začneme s popisem nejzanořenějších funkcí a postupně budeme přecházet k těm méně zanořeným.

5.1 Funkce `closure()`

Tato funkce vrací pro vstupní vrcholy (označím si je u, v) jejich geodetický uzávěr, tedy množinu $\{u, v\}^c$. Musíme se tedy vypořádat se dvěma problémy:

1. nalezení délky nejkratší cesty mezi vrcholy u a v;
2. nalezení všech cest mezi u a v o minimální délce.

5.1.1 Úprava BFS algoritmu

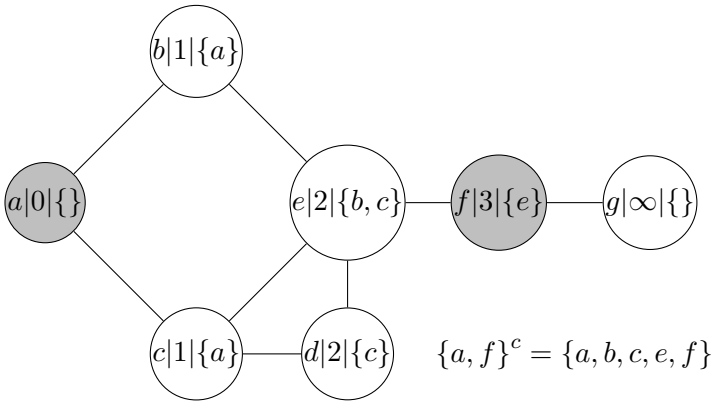
Pro vyhledání nejkratší cesty využiji BFS algoritmu. U něho máme jistotu, že nám najde nejkratší cestu mezi dvěma vrcholy. Mírně upravíme běh algoritmu tak, aby ke každému vrcholu zaznamenával vzdálenost od počátečního vrcholu a množinu vrcholů, přes které vede nejkratší cesta k počátečnímu vrcholu. Ty použijeme pro tzv. backtracking (zpětné vyhledávání).

Algoritmus 2 Funkce `closure()`

```
1: function CLOSURE(u, v)
2:     closure ← ∅                                   ▷ Inicializace množiny
3:     queue.push(u)                                 ▷ Inicializace fronty
4:     u.dist ← 0
5:     while !queue.empty() ∥ tmp = v do
6:         tmp ← queue.pop()
7:         tmp.state ← CLOSED
8:         for all k ∈ N(tmp) do
9:             if k.state = FRESH then
10:                k.dist ← tmp.dist + 1
11:                k.prev ← tmp
12:                stack.push(k)
13:            else
14:                if k.dist = tmp.dist + 1 then
15:                    k.rev ← k.rev ∪ tmp
16:                end if          ▷ Pokud tudy už vede jiná nejkratší cesta
17:            end if
18:        end for
19:    end while
20:    closure ← backtrack(v)
21:    return closure
22: end function
```

Na obrázku 5.1 můžeme vidět, jaké hodnoty by algoritmus v případě daného grafu jednotlivým vrcholům přidělil. Nejprve procházíme graf pomocí vyhledávání do šířky a ke každému vrcholu si zaznamenáme jeho vzdálenost od počátečního vrcholu a množinu předchozích vrcholů takových, že přes ně vede nejkratší cesta k počátečnímu vrcholu. Jak vidíme na obrázků 5.1, těchto vrcholů může být více (například u vrcholu e). Běh této části se zastaví ve chvíli, kdy jsme našli hledaný vrchol.

V algoritmu 2 můžeme vidět upravený algoritmus BFS zapisující informace nutné pro zpětné vyhledávání. Proměnná `dist` obsahuje vzdálenost od počátečního vrcholu a proměnná `prev` obsahuje množinu předchozích vrcholů všech nejkratších cest z aktuálního vrcholu do počátečního. Funkce `backtrack()` je svým způsobem shodná s funkcí `closure()`. Také používá BFS, jen jde „pozpátku“ a používá množinu `prev`. Nakonec vrací všechny vrcholy ležící na všech nejkratších cestách mezi počátečním a koncovým vrcholem. Funkce `N(v)` vrací množinu všech sousedů daného vrcholu.



Obrázek 5.1: Znázornění hodnot proměnných jednotlivých vrcholů po doběhnutí funkce `closure()` na konkrétním grafu pro hledání uzávěru $\{a, f\}^c$. Syntaxe zápisu je následující: jméno vrcholu | vzdálenost od počátečního vrcholu | množina předchozích vrcholů na všech nejkratších cestách do počátečního vrcholu.

5.1.2 Složitost

Složitost BFS algoritmu je $\mathcal{O}(|V| + |H(G)|)$ [8]. Jelikož jsme na bipartitních grafech, víme, že platí $|H(G)| \leq |A| \cdot |B|$. V tomto algoritmu je BFS použito dvakrát, jednou pro hledání nejkratších cest a podruhé pro zpětné vyhledávání. Výsledná složitost funkce `closure()` je tedy

$$\mathcal{O}(2 \cdot (|V| + (|A \mid \cdot |B|))).$$

Označme si celkový počet vrcholů jako n. Pokud zanecháme náš parametr $|A|$, můžeme složitost vyjádřit jako

$$\mathcal{O}(2 \cdot (n + |A| \cdot n)) = \mathcal{O}(|A| \cdot n).$$

Algoritmus 3 Funkce `isGeodetic()`

```
1: function isGeodetic(S, G)
2:     for all i ∈ (0, S.size − 1) do ▷ Iterování přes všechny unikátní dvojice
3:         for all j ∈ (i + 1, S.size) do
4:             set ← set ∪ closure(S[i], S[j])
5:         end for
6:     end for
7:     return set = {V(G)}          ▷ Porovnání sjednocení s vrcholy grafu
8: end function
```

5.2 Funkce `isGeodetic()`

Tato funkce ověřuje, zda je daná množina $S \subseteq V$ geodetická. Jak lze vidět na algoritmu 3 dělá to tak, že vygeneruje z S dvojice a pro ty zjistí jejich geodetický uzávěr. Všechny uzávěry sjednotí a porovná s množinou všech vrcholů grafu G. Pokud jsou tyto množiny shodné, jde o geodetickou množinu.

Generujeme pouze polovinu dvojic, protože je zřejmé, že pro libovolné vrcholy a, b platí $\{a, b\}^c = \{b, a\}^c$. Nemá také smysl, abychom vytvářeli dvojici vrcholu se sebou samým.

V naší implementaci provedeme jednu optimalizaci. Při každém volání funkce `closure()` v parametru předáváme odkaz na globální uzávěr. Funkce `closure()` potom v konstantním čase může provádět sjednocení. Proto se sjednocení jednotlivých uzávěr neprojeví v celkové složitosti funce `isGeodetic()`. Pro jednoduchost neuvádíme v pseudokódu.

5.2.1 Složitost

Maximální velikost ověřované množiny je $|A|$, viz optimalizace v podsekci **??**. Maximální celkový počet vygenerovaných dvojic bude:

$$\sum_{i=1}^{|A|} (|A| - i) = \frac{|A| \cdot (|A| - 1 + 0)}{2} = \frac{|A|^2 - |A|}{2}.$$

Tolikrát bude zavolána funkce `closure()`. Na konci je potřeba porovnat výsledný uzávěr a všechny vrcholy grafu. Celková složitost jednoho volání funkce `isGeodetic()` je tedy:

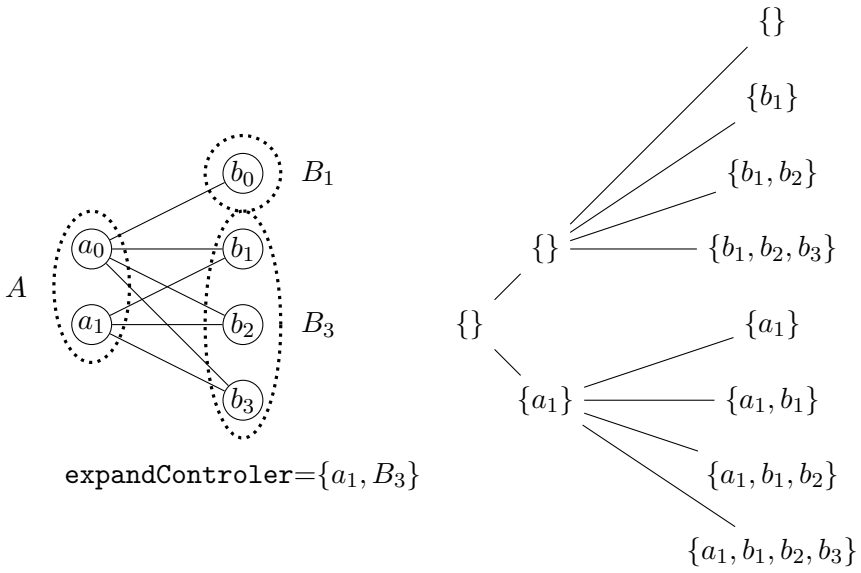
$$\mathcal{O}\left(\left(\frac{|A|^2 - |A|}{2}\right) \cdot (|A| \cdot n) + n\right) = \mathcal{O}\left(|A|^3 \cdot n\right).$$

Algoritmus 4 Funkce `testSet()`

```
1: function testSet(S)
2:     set ← S                          ▷ Inicializace množiny k testování
3:     graph ← G \ B_deg1               ▷ Inicializace grafu G'
4:     set ← set ∪ A_neigh1             ▷ Úprava testované množiny 𝒢 na 𝒢̄
5:     return isGeodetic(set, graph)
6: end function
```

5.3 Funkce `testSet()`

Samotná funkce `isGeodetic()` pouze ověřuje, zda je daná množina geodetická. My ale neexpandujeme všechny vrcholy, protože o některých víme, že



Obrázek 5.2: Stavový prostor a obsah proměnné `expandControler` pro jednoduchý graf, bez použití větví a hranic a dalších optimalizací.

jistě budou v minimální geodetické množině. Proto ve funkci `testSet()` předpřipravíme množinu a graf k testování.

Jak jsme si dokázali v lemmatech 13 a 14, můžeme si graf transformovat, abychom snížili celkový počet stavů, který je potřeba prozkoumat. K testovací množině, tedy připojíme množinu A_{neigh1}. Zároveň z grafu odstraníme všechny vrcholy v partitě B stupně 1. Na konci běhu programu je potřeba všechny vrcholy partity B stupně 1 připojit k nalezené geodetické množině.

5.4 Generování a prohledávání stavového prostoru

Celý algoritmus funguje na principu prohledávání redukovaného stavového prostoru. Stavový prostor budeme prohledávat jako strom, jak lze vidět na obrázku 5.2. Co vlastně znamená stav a jak probíhá jeho expandování na další stavy, si ukážeme v následující sekci.

5.4.1 Stav a jeho vlastnosti

Stav reprezentuje jednu kombinaci vrcholů, která může být hledanou minimální geodetickou množinou. Musíme ale vždy otestovat, zda je geodetická, protože ne všechny generované stavy reprezentují geodetické množiny. Dále si stav nese informaci o další expanzi: proměnnou `expandControler` a úroveň zanoření.

Kořen stromu stavů je stav, kdy v kontrolované množině není žádný vrchol. I takový stav může být geodetickou množinou: jak jsme zmínili v sekci 5.3 k testované množině vždy přidáváme i ty vrcholy, o kterých víme, že jistě budou patřit do geodetické množiny.

Jeden stav je potom expandován na další stavy v závislosti na hloubce zanoření a proměnné `expandControler`.

5.4.2 Proměnná `expandControler`

Tato proměnná svým obsahem určuje, jak bude probíhat expandování stavů. Jak již víme z předchozích lemmat, není potřeba zkoušet všechny vrcholy, ale pouze ty, o kterých s jistotou nevíme, zda jistě jsou (či nejsou) ve výsledné minimální geodetické množině.

Do `expandControler` tedy umístíme:

1. Všechny vrcholy z množiny A, které nemají souseda v B stupně 1.
2. Indexy všech skupin z množiny B, které neobsahují vrcholy stupně 1.

Obsah proměnné `expandControler` i s vygenerovaným kompletním stavovým prostorem k jednoduchému grafu lze vidět na obrázku 5.2.

Algoritmus 5 Rekurzivní funkce `geodRec()`

```
1:  function geodRec(setToTest, expandControler, depth, minSet)
2:      if setToTest.size ≥ minSet.size ∥ expandControler.size = depth then
3:          return                    ▷ Další expandování nelze nebo nemá smysl
4:      end if
5:      if testSet(setToTest) then                    ▷ Nalezeno nové minimum
6:          minSet ← setToTest
7:      end if
8:      next ← expandControler[depth]
9:      if isInA(next) then                ▷ Expandování vrcholu z partity A
10:         geodRec(setToTest, expandControler, depth + 1, minSet)
11:         geodRec(setToTest ∪ next, expandControler, depth + 1, minSet)
12:     else                          ▷ Expandování homogenní skupiny z partity B
13:         geodRec(setToTest, expandControler, depth + 1, minSet)
14:         b1 ← getBGroup(next)[0]
15:         geodRec(setToTest ∪ b1, expandControler, depth + 1, minSet)
16:         b2 ← getBGroup(next)[1]
17:         geodRec(setToTest ∪ b1 ∪ b2, expandControler, depth + 1, minSet)
18:         group ← getBGroup(next)
19:         geodRec(setToTest ∪ group, expandControler, depth + 1, minSet)
20:     end if
21: end function
```

5.4.3 Expandování stavu

Expandování znamená, že ke stávajícímu řešení přidáme (nebo nepřidáme) nový vrchol a zvýšíme hloubku zanoření. Toto expandování je popsáno v algoritmu 5. Pro zvýšení přehlednosti jsme z pseudokódu vynechali kontrolu, zda je daná skupina dostatečně veliká, tedy předpokládáme, že všechny skupiny v B mají větší mohutnost než 2. Funkce `isInA(u)` kontroluje, zda je vstupní vrchol u vrcholem v množině A. Funkce `getBGroup(id)` vrací množinu všech vrcholů nacházejících se v skupině s indexem id. Tento algoritmus již používá metodu větví a hranic.

Situace je poměrně jednoduchá, pokud se na pozici `expandControler[depth]` nachází vrchol náležící do množiny A, označme si ho a. Proměnná `depth` nese informaci o hloubce zanoření. Současný stav expandujeme tak, že vytvoříme dva nové stavy. V prvním přidáme do testovací množiny vrchol a a zvýšíme hloubku zanoření, v druhém pouze zvýšíme hloubku zanoření.

Pokud máme expandovat homogenní skupinu z množiny B, expandujeme ji všemi následujícími způsoby (viz lemma 12):

1. Nepřidáme žádný vrchol do `setToTest`, jen o jedničku zvýšíme `depth`.

2. Do `setToTest` přidáme jeden vrchol ze skupiny, o jedničku zvýšíme `depth`.

3. Do `setToTest` přidáme dva vrcholy ze skupiny, o jedničku zvýšíme `depth`. Toto expandování se použije pouze pokud $|B_x| \geq 2$.

4. Do `setToTest` přidáme všechny vrcholy ze skupiny, o jedničku zvýšíme `depth`. Toto expandování se použije pouze pokud $|B_x| \geq 3$.

Tímto způsobem bychom si nagenerovali celý stavový prostor. Tento prostor však není potřeba prohledávat celý, lze ho značně redukovat pomocí metody větví a hranic.

5.4.4 Metoda větví a hranic

Všimněme si, že každý nový expandovaný stav má v testovací množině stejně nebo více vrcholů než předchozí. Víme tedy s jistotou, že žádný stav v podstromu aktuálního stavu nebude mít méně vrcholů, než má aktuální.

Po celou dobu expandování si budeme držet globální minimum, které aktualizujeme vždy, když najdeme lepší řešení (menší geodetickou množinu).

Metodu větví a hranic v tomto případě aplikujeme tak, že se v každém stavu podíváme, zda jsem již našli menší (nebo stejně velikou) geodetickou množinu. Pokud ano, nebudeme stav dále expandovat. Na začátek stavu tedy přidám podmínku: pokud je velikost `setToTest` větší nebo rovna `minSet`, ukonči expandování.

Jako výchozí minimum si nastavíme množinu všech vrcholů z množiny A takovou, že do ní umístíme všechny vrcholy, které nemají souseda stupně 1. O této množině víme, že je geodetickou množinou, viz lemma 11.

5.4.5 Odvození počtu expandovaných vrcholů

Nejprve se podívejme na složitost bez použití metody větví a hranic. Máme maximálně $|A|$ vrcholů z partity A k expandování a k tomu máme maximálně $2^{|A|} - |A| - 1$ homogenních skupin z B. Odečítáme skupiny s vrcholy stupně 1 a skupinu, která nesousedí s žádným vrcholem z množiny A. Pokud bychom museli prozkoumat celý strom, celkem bychom generovali

$$2^{|A|+2\cdot\left(2^{|A|}-|A|-1\right)} = 2^{2\cdot 2^{|A|}-1}$$

stavů. V exponentu násobíme dvojkou proto, že pro homogenní skupiny je větvící faktor roven 4, tedy 2^2.

Metoda větví a hranic nám ovšem celkový počet expandovaných vrcholů značně redukuje, velikost počátečního minima je totiž $|A|$. Víme, že vždy když bude vygenerován stav s větším počtem vrcholů, celý podstrom bude prořízнut. Vlastně bychom měli počítat s $|A| - 1$, jelikož se ale v $\mathcal{O}$ tento detail neprojeví, budeme pro jednoduchost počítat s $|A|$. Celkem tedy víme, že listů stromu stavů bude maximálně

$$\binom{2^{|A|-1}+|A|}{|A|} \cdot |A| \cdot 3^{|A|}.$$

První část vyjadřuje počet možností, kolik bude vybráno vrcholů nebo skupin. Každou ze skupin můžeme navíc vybrat třemi způsoby (a jedním nevybrat), proto násobíme $3^{|A|}$. Navíc vrcholů může být i méně než $|A|$, pro započítání těchto možností je potřeba tímto číslem vynásobit.

Jelikož jde o strom, celkový počet vnitřních stavů bude maximálně tolik, kolik je počet listů. Tím pádem celkový počet vygenerovaných stavů je

$$2 \cdot \binom{2^{|A|-1}+|A|}{|A|} \cdot |A| \cdot 3^{|A|}.$$

Část vybírání $|A|$ vrcholů a skupin můžeme odhadnout

$$\binom{2^{|A|-1}+|A|}{|A|} \leq (2^{|A|-1}+|A|)^{|A|} \leq 2^{|A|^2}.$$

Tím pádem je celkový počet vygenerovaných stavů

$$\mathcal{O}\left(2^{|A|^2} \cdot 2 \cdot |A| \cdot 3^{|A|}\right) = \mathcal{O}\left(2^{|A|^2} \cdot |A| \cdot 3^{|A|}\right).$$

5.5 Celková složitost

Celkovou složitost vyjádříme ve dvou částech. První vyjadřuje složitost při zvyšování parametru a druhá při zvyšování počtu vrcholů grafu (tedy hlavně při zvětšování $|B|$). Výsledná složitost je potom rovna

$$\mathcal{O}\left(2^{|A|^2} \cdot |A| \cdot 3^{|A|}\right) \cdot \mathcal{O}\left(|A|^3 \cdot n\right) = \mathcal{O}\left(2^{|A|^2} \cdot |A|^4 \cdot 3^{|A|} \cdot n\right)$$

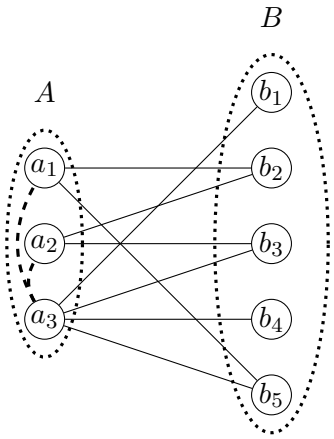
Jak tedy můžeme vidět, náš algoritmus lze použít, pokud je velikost parametru malá. Potom složitost roste lineárně k počtu vrcholů vstupního grafu. Vidíme, že ze nám podařilo parametrizovat problém hledání geodetického čísla na $|A| \ll |B|$ bipartitních grafech.

Kapitola 6

Hledání $g(G)$ na grafech s nízkým vrcholovým pokrytím

6.1 Počáteční podmínky

Stejně jako u bipartitních grafů nás budou zajímat pouze souvislé grafy. Nesouvislé grafy bychom znovu mohli rozdělit na jednotlivé komponenty a ty poté řešit každou zvlášť. Zajímat nás budou takové grafy G, které lze rozdělit na množiny A a B takové, že A je minimální množina vrcholového pokrytí a B je množina, která je pokryta množinou A. Navíc pouze takové, které mají poměrně malou $|A|$.



Obrázek 6.1: Změna grafu s nízkým vrcholovým pokrytím na $|A| \ll |B|$ bipartitní graf.

6.2 Souvislost grafu s nízkým vrcholovým pokrytím a $|A| \ll |B|$ bipartitního grafu

Jak si můžeme všimnout na obrázku 6.1 z grafu s nízkým vrcholovým pokrytím můžeme vytvořit $|A| \ll |B|$ bipartitní jednoduše tak, že odebereme všechny hrany mezi vrcholy množiny A, na obrázku vyznačeny čárkovaně.

Pozorování 15. *Mějme graf G mající množiny A a B takové, že A je minimální množina vrcholového pokrytí a pro B platí $B = V \setminus A$. Potom žádné dva vrcholy v množině B nejsou propojeny hranou.*

Důkaz. Pokud by existovala hrana mezi vrcholy $b_1, b_2 \in B$, potom by ani jeden vrchol hrany nebyl v množině vrcholového pokrytí, a tím pádem by A nebyla množinou vrcholového pokrytí. □

Potom velikost vrcholového pokrytí odpovídá partitě A bipartitního grafu. Jistě přidání těchto hran bude mít dopad na hledání $g(G)$, jak ale uvidíme dále, pouze malými úpravami již stávajícího algoritmu (hledajícího pouze na bipartitních grafech) docílíme požadované funkcionality.

6.3 Simpliciální vrcholy

Přidáním hran mezi vrcholy množiny A může vzniknout nová situace, kterou jsme zatím nemuseli řešit. V množině B můžou vzniknout skupiny simpliciálních vrcholů.

Pokud se podíváme na graf na obrázku 6.1 vidíme, že se zde nacházejí dva simpliciální vrcholy a to vrcholy b_3 a b_5. Navíc i vrchol b_1 je podle striktní definice simpliciální vrchol, ten ale máme již zahrnutý v množině B_{deg1}. Proto nás budou zajímat pouze simplicální vrcholy, sousedící s klikou o minimální velikosti 2.

Jak jsme si dokázali v lemmatu 3, o těchto vrcholech víme, že jistě budou patřit do každé geodetické množiny libovolného grafu G.

6.4 Upřesnění tvrzení

V této sekci si upřesníme dvě lemmata zmíněné dříve, přesněji lemma 13 a 14. Obdobná lemmata totiž platí i pro grafy s nízkým vrcholovým pokrytím.

Lemma 16. *Nechť G je graf mající množinu A, množinu vrcholového pokrytí a množinu $B = V \setminus A$ (tak, že $|A| \leq |B|$) a geodetickou množinu $\mathcal{G}$. Vytvoříme nový graf G' takový, že z grafu G odstraníme všechny vrcholy stupně 1 z partity B (označme si množinu všech těchto vrcholů B_{deg1}). Dále si označíme množinu $A_{neigh1} \subseteq A$, množinu vrcholů z partity A, sousedící s vrcholy stupně 1 v B. Potom platí $\bar{\mathcal{G}} = (\mathcal{G} \setminus B_{deg1}) \cup A_{neigh1}$ je geodetickou množinou grafu G'.*

Důkaz. Shodný s důkazem lemmatu 13. □

Lemma 17. *Nechť G je graf mající množinu A, množinu vrcholového pokrytí a množinu $B = V \setminus A$ (tak, že $|A| \leq |B|$) a geodetickou množinu $\mathcal{G}$. Vytvoříme nový graf G' takový, že z grafu G odstraníme všechny vrcholy stupně 1 z partity B (označme si množinu všech těchto vrcholů B_{deg1}). Dále si označíme množinu $A_{neigh1} \subseteq A$, množinu vrcholů z partity A, sousedící s vrcholy stupně 1 v B. Označme si $\bar{\mathcal{G}}$ jako geodetickou množinou grafu G'. Pokud $\bar{\mathcal{G}}$ je geodetická množina G', potom platí $G = (\bar{G} \setminus A_{neigh1}) \cup B_{deg1}$ je geodetická množina grafu G.*

Důkaz. Důkaz je obdobný jako v lemmatu 13. □

6.5 Změny v algoritmu

Předchozí algoritmus je navržen tak, že generuje možná řešení, kontroluje zda je dané řešení geodetická množina a hledá minimum této množiny. Proto jeho změna na algoritmus hledající $g(G)$ grafů s nízkým vrcholovým pokrytím nebude tak náročná.

Jedinou změnou, kterou je potřeba v algoritmu provést, je přidat novou množinu vrcholů ve funkci `testSet()`, tedy množiny vrcholů o kterých jistě víme, že budou v $\mathcal{G}$. Do této doby byli v této funkci přidávány pouze vrcholy A_{neigh1}. Nyní musíme přidat i množinu všech simpliciálních vrcholů.

Jelikož je algoritmus `closure()` navržen tak, aby fungovala na libovolných souvislých grafech, není potřeba ji žádným způsobem měnit. Stejně tak i další již zmíněné funkce.

Kapitola 7

Implementace algoritmu

V této kapitole krátce popíšeme, jak jsme náš algoritmus implementovali.

7.1 Základní informace o aplikaci

Pro implementaci byl zvolen jazyk `C++`. Byla vytvořena CLI aplikace, tedy konzolová aplikaci bez grafického prostředí. Vstupem programu je graf zadaný ve formátu:

- počet hran je na prvním řádku;
- ostatní řádky obsahují dvojice vrcholů, mezi kterými je hrana;
- poslední řádek obsahuje mezerou oddělené všechny vrcholy množiny vrcholového pokrytí.

Graf může být zadán jako parametr jménem souboru. Pokud daný parametr není zadán, je uživatel požádán o zadání grafu na standardní vstup na začátku běhu programu.

7.2 Implementace jednotlivých funkcí

Funkce `closure()`, `isGeodetic()`, `testSet()` a `geodRec()` jsou implementovány na základě pseudokódů v kapitole 5. Implementace se může mírně lišit od zmíněných pseudokódů, protože pseudokódy byly v některých případech mírně zjednodušeny. Základní myšlenka je však vždy zachována.

7.3 Inicializace a konečné úpravy výsledku

Nejprve je načten a zpracován graf na vstupu. Graf je zkontrolován, zda je souvislý. Ke kontrole souvislosti je použit základní BFS algoritmus.

Před spuštěním funkce `geodRec()` je třeba inicializovat proměnnou nesoucí informace o dalším expandování `expandControler`, viz podsekce 5.4.2. Dále je potřeba nastavit globální minimum (v implementaci pojmenováno `actualMin`) na hodnotu $A \setminus A_{neigh1}$, viz lemma 11.

Po nalezení minimální množiny je ještě potřeba přidat k nalezené množině všechny vrcholy, o kterých jistě víme, že jsou v každé geodetické množině, tedy vrcholy stupně 1 a simpliciální vrcholy, viz pozorování 5 a lemma 3. Výstupem programu je potom jedna nalezená minimální geodetická množina grafu. Mohutnost této množiny je potom geodetické číslo vstupního grafu.

7.4 Vnitřní reprezentace grafu

Pokud bychom chtěli čistě objektové řešení, základem grafu by byla třída např. `CVrchol`. Každý objekt této třídy by si nesl množinu všech svých sousedů, svoje jméno a příslušnost do partity. Tento způsob implementace by však byl velice paměťově náročný, především u velikých grafů. Proto zavedeme speciální struktury na uložení jmen, incidencí a příslušnosti do partit pro jednotlivé vrcholy.

7.4.1 Pole incidencí

Vytvoříme pole incidencí tak, že do pole vložíme za sebe sousedy všech vrcholů. Potom si pro každý vrchol stačí pamatovat, na které pozici v poli seznam jeho sousedů začíná a jak je dlouhý.

Touto reprezentací dojde ke značné paměťové úspoře, při porovnání s čistě objektovým řešením.

7.4.2 Vrchol

Vrchol by bylo možné reprezentovat jeho názvem v datovém typu `string`. Ovšem porovnávání, které je v programu velice časté, zabere poměrně dost času. Proto je lepší reprezentovat vrchol celočíselným indexem. Porovnání dvou indexů lze provést v jednom taktu procesoru.

Další informace, jako je jméno vrcholu v zadaném grafu a příslušnost do partity, je vhodné uložit do mapy, ve které budou mapované jednotlivé informace na indexy vrcholů. Vyhledání informace potom proběhne v logaritmickém čase k počtu vrcholů grafu.

Kapitola 8

Testování

V následující části si popíšeme, jak byla naše konkrétní implementace představeného algoritmu testována a jakých výsledků v testování dosáhla.

8.1 Generování testovacích grafů

Při generování testovacích grafů musíme dát pozor na to, zda jde opravdu o náhodný graf. Pro potřeby testování byly vytvořeny dva generátory náhodných grafů. Představíme si jejich algoritmy a ukážeme si, jaké náhodné grafy generují.

Algoritmus 6 Generátor náhodného grafu vypouštěním hran

```
1: function GENERATOR1(|A|, |B|, isBipartite, P_hrana)
2:     for all i ∈ ⟨0, |A|) do          ▷ Hrany mezi vrcholy v A a vrcholy v B
3:         for all j ∈ ⟨0, |B|) do
4:             if rand() < P_hrana then
5:                 createEdge(a_i, b_j)
6:             end if
7:         end for
8:     end for
9:     if !isBipartite then      ▷ Budeme vytvářet hrany mezi vrcholy v A?
10:        for all i ∈ (0, |A| − 1) do   ▷ Generování všechn unikátních dvojic
11:            for all j ∈ (i + 1, |A|) do
12:                if rand() < P_hrana then
13:                    createEdge(a_i, a_j)
14:                end if
15:            end for
16:        end for
17:    end if
18: end function
```

8.1.1 Generování vypouštěním hran z úplného bipartitního grafu

Pseudokód generátoru můžeme vidět na algoritmu 6. Funkce `createEdge(u, v)` vytvoří oba vrcholy, pokud neexistují a přidá mezi ně novou hranu. Funkce `rand()` vrací náhodné číslo v intervalu $(0, 1)$. Vstupní parametr P_{hrana} určuje pravděpodobnost výskytu hran v grafu.

V první části algoritmus vytvoří hranu s pravděpodobností P_{hrana}. Jde tedy vlastně o náhodné vypouštění hran z úplného bipartitního grafu.

V druhé části si algoritmus nageneruje všechny unikátní dvojice vrcholů množiny A a vytváří mezi nimi hrany s pravděpodobností P_{hrana}. Jde tedy vlastně o vypouštění hran s kompletního podgrafu G indukovaného množinou A.

Pro jednoduchost není v pseudokódu vyznačen mechanismus, který zajišťuje, aby nevznikl izolovaný vrchol. Ten funguje tak, že si pro každý vrchol zapamatujeme, zda již k němu byla vytvořena hrana a pokud ne na konci běhu algoritmu ho náhodně napojíme na vrchol v protější skupině (například vrchol b_1 na vrchol a_3).

Jednou z nevýhod tohoto algoritmu je to, že nedokážeme takto vygenerovat graf, který bude jistě souvislý. Může se totiž stát, že graf neobsahuje žádný izolovaný vrchol, a přesto není souvislý. Další nevýhodou je fakt, že grafy takto generované mají podobné sledované vlastnosti. Čím více bude v množině B vrcholů tím pravděpodobnější je, že se vyskytnou všechny možné homogenní skupiny. Tím pádem ale nevyzkoušíme například hodně veliké skupiny, protože pravděpodobnost jejich vygenerování předchozím algoritmem je mizivá.

Tento algoritmus budeme používat pro testy, kdy kdy bude potřeba fixní mohutnosti množiny B.

Z důvodů výše uvedených byl vytvořen další generátor beroucí v potaz homogenní skupiny množiny B.

8.1.2 Generování po homogenních skupinách

Pseudokód vylepšeného generátoru můžeme vidět na algoritmu 7. Parametr $|B_{max}|$ udává maximální velikost homogenní skupiny, $P_{skupina}$ určuje pravděpodobnost, s jakou se jednotlivé skupiny ve výsledném grafu objeví, funkce `addAToAEdges()` provádí stejnou operaci jako řádky 10 až 16 algoritmu 6 a funkce `createEdge()` se chová stejně jako v dříve zmíněné funkci.

Jak jsme si v sekci 4.2 ukázali, indexy homogenních skupin jsou úzce svázány s binární soustavou. Toho využijeme v tomto algoritmu.

Ve vnějším cyklu iterujeme přes všechny čísla v intervalu $\{1, 2^{|A|}\}$. Každé z těchto čísel reprezentuje jednu homogenní skupinu v množině B. Dle pravděpodobnosti $P_{skupina}$ se rozhodneme, kolik a které z těchto skupin vybereme.

Po vybrání určitého indexu si náhodně vybereme velikost skupiny z intervalu $\{1, |B_{max}|\}$.

Algoritmus 7 Generátor náhodného grafu dle homogenních skupin

```
1: function GENERATOR2(|A|, |B_max|, isBipartite, P_skupina)
2:     for all i ∈ ⟨0, 2^|A|) do
3:         if rand() < P_skupina then
4:             vertexes ← |B_max| · rand()
5:             for all j ∈ ⟨0, vertexes) do
6:                 tmp ← i
7:                 for k ← |A| downto 0 do
8:                     if tmp − 2^k ≥ 0 then
9:                         tmp ← tmp − 2^k
10:                        createEdge(a_k, b_{i,j})
11:                    end if
12:                end for
13:            end for
14:        end if
15:    end for
16:    if !isBipartite then
17:        addAToAEdges()
18:    end if
19: end function
```

Potom každý vrchol skupiny napojíme na příslušné vrcholy v partitě A dle jedniček ve dvojkovém zápisu čísla (detaily viz algoritmus 7).

Grafy generované tímto způsobem mají jiné vlastnosti. Neopakuje grafy s podobnými sledovanými vlastnostmi. Proto je použit tento generátor pro generování testovacích grafů, kde není potřeba fixní mohutnost množiny $|B|$. Tu totiž tímto postupem nemůžeme zafixovat.

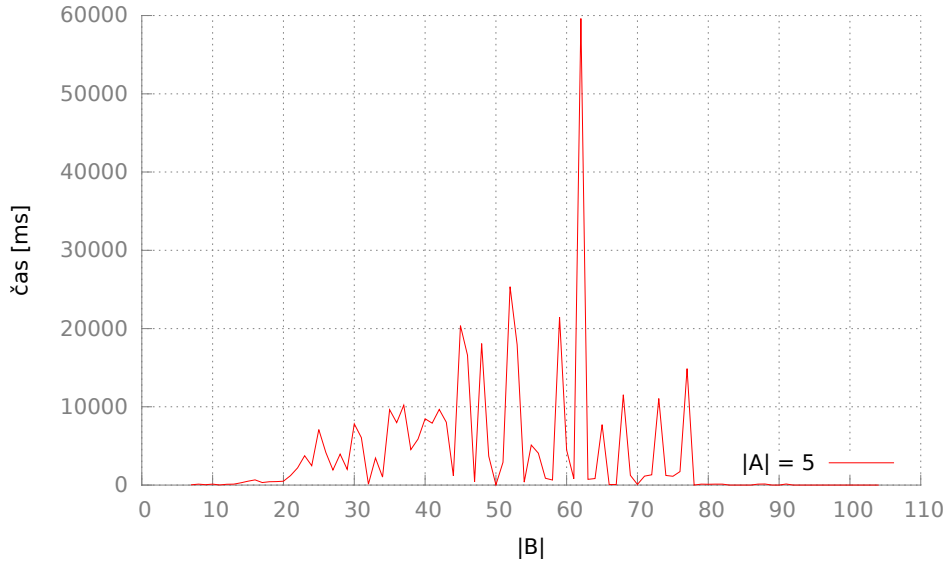
8.2 Průběh měření

Jak bylo zmíněno dříve algoritmus byl implementován v jazyce `C++`. Na tomto programu jsme potom měřili doby provádění daného běhu programu pomocí funkce vracející procesorový čas `omp_get_wtime()` dostupné v knihovně `OpenMP`.

Program byl kompilován v systému `Linux Mint 17` překladačem `gcc` ve verzi 4.8.4 s přepínačem `-O3`. Měření bylo prováděno na notebooku `Lenovo W500` mající dvoujádrový procesor `Intel(R) Core(TM)2 Duo T9600` pracující na frekvenci 2.80 GHz. Notebook má 4 GB RAM paměti.

Jelikož jde o náhodně generované grafy, doby provádění grafů se stejnými velikostmi $|A|$ i $|B|$ mohou vycházet různě. Proto každé měření opakujeme desetkrát a potom spočítáme průměr doby provádění jednotlivých pokusů.

I přes poměrně veliký počet měření, se občas v grafech vyskytnou závislosti, které neodpovídají očekávaným. Většinou je to způsobeno metodou větví a hranic. Pokud se totiž podaří brzy najít dobré řešení problému, mnoho pod-



Obrázek 8.1: Závislost doby provádění programu na $|B|$, $|A| = 5$.

stromů je prořízнuto, a proto celková doba provádění je výrazně nižší, než bychom čekali.

8.2.1 Problémy s měřením

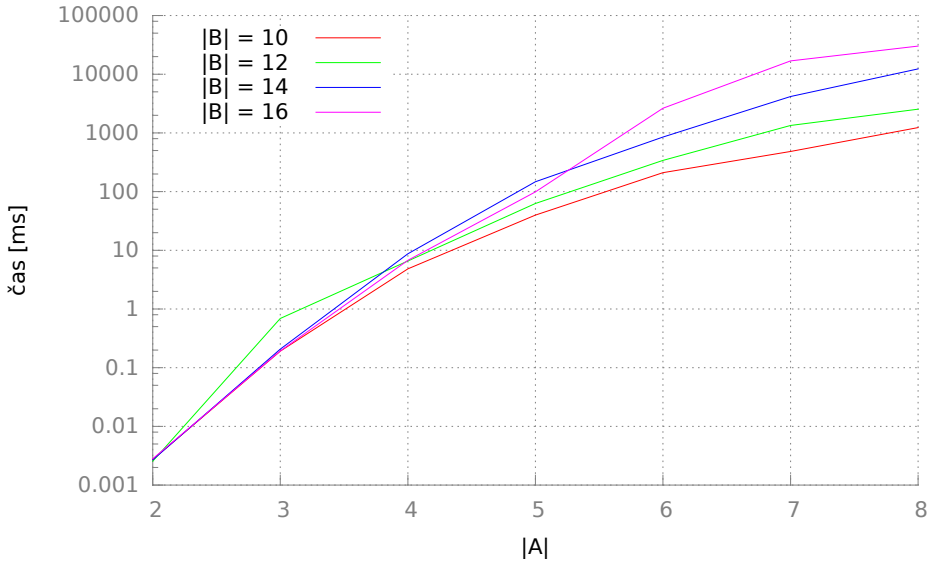
Při měření jsme dospěli k problému, že při měření závislosti času na mohutnosti množiny B (při fixní mohutnosti množiny A) většina grafů však vycházela podobně, jako je znázorněno na obrázku 8.1. Očekávali jsme ale se zvyšujícím se počtem vrcholů zvyšující se dobu provádění.

Při zkoumání tohoto problémů jsme přišli na to, že důvodem jsou vrcholy stupně 1. Ty způsobují značné zjednodušení (každý vrchol sníží exponent o 1, viz 4.4), a tím pádem významně mění dobu běhu algoritmu. Proto byla do generátoru přidána podmínka kontrolující vygenerované homogenní skupiny zda nejsou skupinou vrcholů stupně 1. Výsledné grafy jsou tedy horním odhadem reálných závislostí.

8.3 Závislost doby provádění programu na $|A|$

Parametrem našeho algoritmu je $|A|$. Podíváme se, jak bude vypadat závislost doby provádění programu, pokud pro různé fixní velikosti $|B|$ budeme zvyšovat $|A|$. Očekáván je exponenciální růst doby běhu programu (jak jsme si odvodili v sekci 5.5).

Jak můžeme vidět na grafu 8.2, námi implementované řešení opravdu očekávanou složitost má.



Obrázek 8.2: Závislost doby provádění programu na $|A|$ pro různé $|B|$.

8.4 Závislost doby provádění programu na $|B|$

Pro další testování zafixujeme velikost parametru $|A|$ a budeme postupně zvětšovat $|B|$. Graf závislost můžeme pozorovat na obrázku B.1 v příloze. Bohužel se nám nepodařilo proložit graf vhodnou lineární funkcí. To může být způsobeno metodou větví a hranic, která různým způsobem redukuje počet prozkoumaných stavů. Proto je v grafu zanesena kvadratická závislost, aby bylo zřejmé, že naše řešení je na daných datech řádově lepší, než kvadratické.

8.5 Vyhodnocení výsledků

Námi vytvořený algoritmus je parametrizovaný, je tedy použitelný pouze v případech, kdy je parametr poměrně malý. To jsme si ověřili v sekci 8.3, kde jsme mohli vidět, že s rostoucím parametrem algoritmu roste doba běhu exponenciálně.

Důležitější pro nás je ale graf B.1. Jak můžeme vidět pro malé velikosti parametru jsme schopni v rozumném čase najít geodetické číslo i pro grafy s poměrně velkým počtem vrcholů.

Závěr

Podařilo se nám nalézt parametrizovaný algoritmus hledající minimální geodetickou množinu na grafech s nízkým vrcholovým pokrytím. Algoritmus je parametrizován velikostí množiny vrcholového pokrytí. Z naměřených dat vyplývá, že při zafixování parametru dojde ke značné redukci časového složitosti řešení.

Algoritmus jsme nemohli porovnat s žádným existujícím, protože se nám nepodařilo vyhledat žádnou práci, která by se touto problematikou zabývala.

Grafy s nízkým vrcholovým pokrytím jsou pouze velice úzká množina grafů. Bylo by zajímavé rozšířit množinu vstupních grafů na nějakou obecnější množinu a pokusit je najít parametrizovaný algoritmus pro řešení tohoto problému.

Literatura

[1] M. Atici. Computational complexity of geodetic set. *International Journal of Computer Mathematics*, 79(5):587–591, 2002.

[2] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*, volume 4. Springer, 2015.

[3] M. C. Dourado, F. Protti, D. Rautenbach, and J. L. Szwarcfiter. Some remarks on the geodetic number of a graph. *Discrete Mathematics*, 310(4):832 – 837, 2010.

[4] D. Eppstein. Metric dimension parameterized by max leaf number. *Journal of Graph Algorithms and Applications*, 19(1):313–323, 2015.

[5] F. Harary, E. Loukakis, and C. Tsouros. The geodetic number of a graph. *Mathematical and Computer Modelling*, 17(11):89 – 95, 1993.

[6] R. A. Harary, Frank; Melter. On the metric dimension of a graph. *Ars Combinatoria*, 2:191–195, 1976.

[7] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):pp. 497–520, 1960.

[8] E. Moore. The shortest path through a maze. *Proceedings of an International Symposium on the Theory of Switching*, pages 285–292, 1959.

[9] P. J. Slater. Leaves of trees. *Congr. Numer*, 14(549-559):37, 1975.

Příloha **A**

Seznam použitých zkratek

BFS Vyhledávání do šířky (breadth-first search)

FPT Patamertizovaně dostupný (fixed parameter tractable)

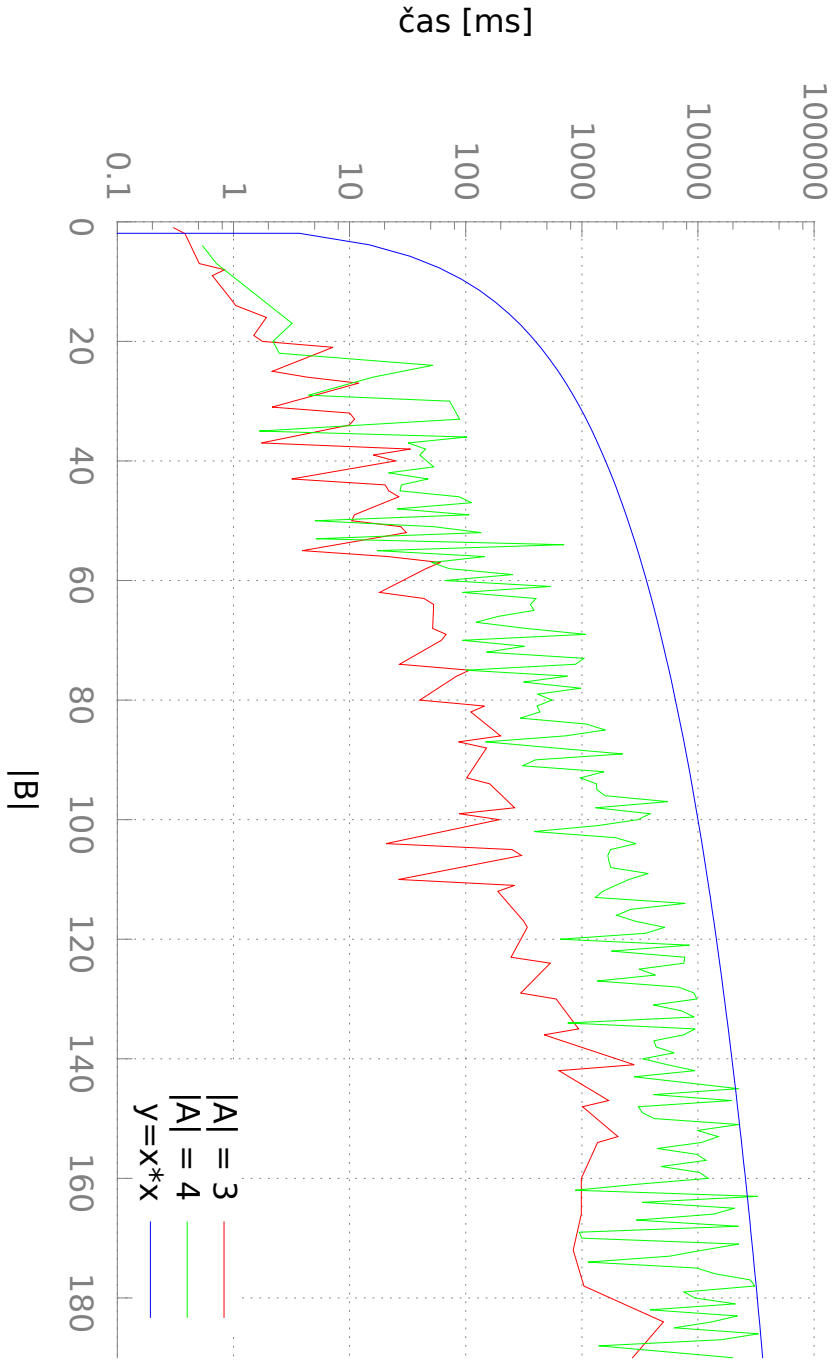
CLI Rozhraní příkazové řádky (command line interface)

RAM Operační paměť počítače (random access memory)

BÚNO Bez újmy na obecnost

Příloha B

Přiložené obrázky



Obrázek B.1: Závislost doby provádění programu na $|B|$ pro konstantní $|A|$.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src` ..... zdrojové kódy
  - `impl` ..... zdrojové kódy implementace a Makefile
  - `thesis.tex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF