Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Cestová a stromová šírka kubických grafov

Bakalárska práca

2019
Filip Husár

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Cestová a stromová šírka kubických grafov

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	doc. RNDr. Robert Lukoťka, PhD.

Bratislava, 2019
Filip Husár

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Filip Husár
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Cestová a stromová šírka kubických grafov
Pathwidth and treewidth of cubic graphs

Anotácia: Cestová a stromová šírka sú parametre, ktoré sa bežne vyskytujú v parametrizovanej zložitosti. Za predpokladu ohraničenej cestovej/stromovej šírky možno efektívne riešiť mnohé problémy, ktoré sú vo všeobecnosti NP-ťažké. Fomin a Hoie [F. V. Formin, K. Hoie: Pathwidth of cubic graphs and exact algorithms, Information Processing Letters 97 (2006), 191-196] skonštruovali triedu algoritmov, pomocou ktorej je možné pre každé epsilon nájsť cestovú dekompozíciu kubického grafu so šírkou nanajvýš (1/6+epsilon).|V(G)|. Tento výsledok je možné použiť na nájdenie pomerne rýchlych algoritmov na riešenie NP-ťažkých problémov na kubických grafoch. Nie je však známa žiadna trieda kubických grafov, ktorej cestová šírka by sa približovala 1/6.|V(G)|. Cieľom práce je preto hľadať nové prístupy a heuristiky na nájdenie cestového/stromového rozkladu kubického grafu s malou šírkou.

Vedúci: doc. RNDr. Robert Lukoťka, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 25.09.2018

Dátum schválenia: 06.11.2018 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Týmto by som chcel poďakovať svojmu školiteľovi doc. RNDr. Robertovi Lukoťkovi, PhD. za cenné rady poskytnuté pri riešení problémov tejto práce a jej písaní.

Abstrakt

V tejto práci sa budeme zaoberať bisekciou grafu s ohraničenou šírkou, ktorá bude slúžiť ako vstupný údaj pre cestovú dekompozíciu s ohraničenou cestovou šírkou.

Obe časti implementujeme do efektívneho programu na hľadanie cestovej šírky.

Kľúčové slová: grafy, cestová dekompozícia, cestová šírka, bisekcia grafu

Abstract

In this paper we take a look on bisection with upper bounded cut-size, which we will need as an input to a path decomposition algorithm.

Both parts we implement to an efficient algorithm of path decomposition.

Keywords: graphs, path decomposition, pathwidth, bisection

Obsah

Úvod

Za posledné dekády rokov sa oblasť počítačových technológií posúva stále výraznejšími krokmi dopredu. Úlohy na spracovanie počítačmi sú čoraz komplikovanejšie, náročnejšie na výpočty a obvody už atakujú fyzikálne zákony určujúce ich teoretickú rýchlosť. Na to, aby mohol vývoj počítačov ísť ruka v ruke so stále narastajúcimi požiadavkami ľudstva, je potreba zameriavať svoju pozornosť na algoritmické problémy.

Teória grafov je veľká oblasť matematiky, ktorej sa vo veľkej miere venuje aj informatika. Grafové štruktúry sú súčasťou veľkého počtu programovacích jazykov. S rôznymi problematikami z oblasti teórie grafov sa môžeme stretnúť aj pri architektúre hardvérových komponentov počítaču, ako napr. VLSI[3](Very-Large-Scale Integration), ktorý pri navrhovaní intergovaných obvodov kombinuje milióny tranzistorov tak, aby výkonnosť bola čo najvyššia.

Podobne aj pri kompilovaní programov[4], kedy všetky medzivýpočty možno uchovávať buď v registroch alebo v hlavnej pamäti, rieši túto problematiku teória grafov.

Obidvom spomenutým problémom - VLSI aj kompilátorom, sa bližšie venuje cestová a stromová dekompozícia, resp. cestová a stromová šírka.

Cestová dekompozícia grafu je sekvencia množín vrcholov daného grafu taká, že každý vrchol grafu sa vyskytuje v súvislej sekvencii množín a každá hrana grafu sa vyskytuje v niektorej z množín (je reprezentovaná dvomi koncovými vrcholmi)

Cieľom cestovej dekompozície je, aby množiny neboli príliš veľké, nech algoritmy, v ktorých cestová dekompozícia zohráva dôležitú rolu, boli čo najefektívnejšie. Teda naším cieľom je, aby najväčšia z množín v sekvencii mala čo najmenej vrcholov.

Tento atribút nazývame aj cestová šírka[4].

Podobnou problematikou sa zaoberá aj stromová dekompozícia. Od cestovej dekompozície sa líši tým, že množiny nie sú nutne zoradené do sekvencie, ale môžu tvoriť strom.

V tejto práci sa budeme zaoberať tým, ako môžeme nadobudnúť na kubických grafoch s n vrcholmi cestovú šírku blížiacu sa k $(\frac{1}{6} + \epsilon)n$

Využijeme dva už známe články zaoberajúce sa horným ohraničením šírky bisekcie grafu[1][2], na čo bude nadväzovať článok o hornom ohraničení cestovej dekompozície.

Dôkazy týchto článkov sa následne pokúsime vhodne implementovať do programu, ktorý pre ľubovoľný kubický graf nájde cestovú dekompozíciu s cestovou šírkou, resp. šírkou blížiacou sa k nej.

Kapitola 1

Pojmy a definície

V tejto kapitole si vymenujeme základné prostriedky, ktoré potrebujeme na prácu v oblasti cestovej a stromovej šírky. Cestová a stromová šírka je jednou z novších oblastí teórie grafov, ktorá zohráva veľmi dôležitú rolu v algoritmických problémoch na grafoch. Hovorí nám o tom, ako dobrú cestovú alebo stromovú dekompozíciu vieme na danom grafe spraviť.

1.1 Základné definície teórie grafov

Pod pojmom graf budeme v tejto práci rozumieť množinu vrcholov a hrán ktoré ich spájajú. Takýmito grafmi možno zjednodušene znázorniť rôzne situácie z reálneho života. Vrcholy môžu znázorňovať ľudí a hrany priateľstvá medzi nimi. Vrcholy takisto môžu znázorňovať rohy ulíc a hrany medzi nimi cesty.

Definícia 1:

Neorientovaný graf [7] G je usporiadaná dvojica $G = (V, E)$, kde V je neprázdna konečná množina a E je systém dvojprvkových podmnožín množiny V.

Množinou V nazývame množinu vrcholov, množinou E nazývame množinu hrán. Ak je graf z kontextu daný, pre $|V| = n, |E| = m$ budeme vrcholy označovať ako $v_1, ..., v_n$ a hrany $e_1, ..., e_m$, pričom $e_i = \{v_j, v_k\}, 1 \leq i \leq m$ je hrana spájajúca vrcholy v_j a v_k

Pokiaľ si konkrétna situácia nebude vyžadovať inak, v grafe budeme povoľovať hrany $e_i = \{v_s, v_s\}$, nazývané aj slučky, a tiež aj viacnásobné hrany, teda n-ticu $\{e_{j+1} = \{v_t, v_u\}, e_{j+2} = \{v_t, v_u\}, ..., e_{j+n} = \{v_t, v_u\}\}$.

Cestová a stormová dekompozícia zoskupuje vrcholy grafu do množín, ktoré sú navzájom spojené hranami tak, že množiny tvoria líniu, resp. strom. Tieto množiny zvykneme nazývať aj *uzly*.

Súvislosť grafu hovorí o tom, či je graf zložený z jedného alebo viac komponentov. V jednom kompomentne sa vieme dostať po hranách z ľubovoľného vrcholu komponentu do ľubovoľného (nie nutne iného) vrcholu toho istého komponentu.

Pokiaľ má graf práve jeden komponent, je súvislý. Pokiaľ má graf viac komponentov, je nesúvislý. [7]

V grafe G sa kružnica nachádza práve vtedy, keď existuje dvojica vrcholov $v_1, v_2 \in V(G)$, že medzi vrcholmi v_1, v_2 existujú aspoň dve disjunktné cesty.[7]

Definícia 2:

Graf G nazývame *strom* [7] práve vtedy keď je súvislý a neobsahuje kružnicu.

Graf, ktorý je strom, má niektoré veľmi zaujímavé a praktické vlastnosti[8], ktoré nám môžu pomôcť k lepšej efektivite algoritmov na grafoch. Jedna zo základných vlastností stromu je tá, že medzi každými dvomi vrcholmi (v prípade cestovej dekompozície uzlami) existuje práve jedna cesta. Z nej vieme odvodiť ďalšie známe vlastnosti stromov ako napr. minimálnu spojitosť grafu či maximálnu acyklickosť grafu.

Na to, aby sme mohli používať pojmy ako stromová či cestová šírka, si potrebujeme najprv zadefinovať stromovú (resp. cestovú) dekompozíciu grafu.

1.2 Stromová dekompozícia

Označenia $V(G)$ a $E(G)$ v nasledujúcich definíciach odkazujú na vrcholy, resp. hrany grafu G.

Definícia 3: [8]

Majme graf $G = (V, E)$, strom T a rodinu množín vrcholov $\nu = (V_t)_{t \in T}$ takých, že $V_t \subseteq V(G)$ *Stromová dekompozícia*[**?**] je taká dvojica *(T,*ν*)*, ktorá spĺňa nasledujúce podmienky:

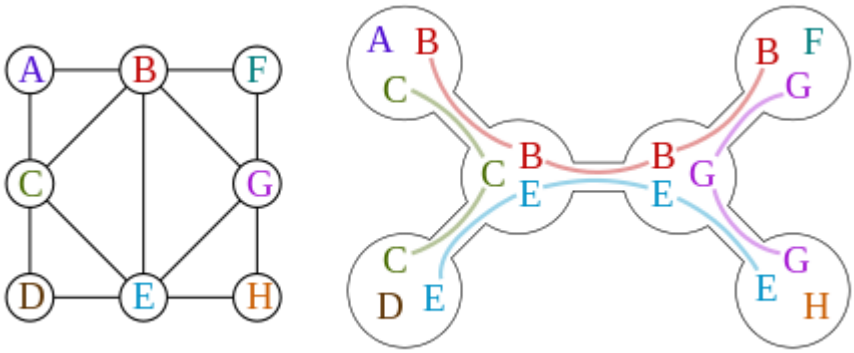
i. $V(G) = \bigcup_{t \in T} V_t$

ii. pre každú hranu $e \in E(G)$ existuje také $t \in T$, že oba koncové vrcholy hrany e ležia v V_t

iii. $V_{t_1} \bigcap V_{t_3} \subseteq V_{t_2}$ platí pre všetky $t_1, t_2, t_3 \in T$ pokiaľ t_2 leží na ceste spájajúcej t_1 s t_3

Podmienky i. a ii. hovoria o tom, že G je zjednotením podgrafov indukovaných vrcholmi množín $V_1, ..., V_n$.

Podmienka iii. hovorí o tom, že množiny vrcholov sú usporiadané do stromu.



Obr. 1.1: Príklad stromovej dekompozície

Na uvedenom príklade 1.1 môžeme spozorovať, že každá oblasť grafu je v stromovej dekompozícii reprezentovaná jednou množinou vrcholov.

1.3 Cestová dekompozícia

Cestová dekompozícia sa od stromovej dekompozície líši iba v malom detaile. Narozdiel od stromovej dekompozície nepriamo zakazuje uzol, ktorý má troch alebo viacerých susedov, teda každý uzol je stupňa najviac 2.

Jednoducho povedané, v stromovej dekompozícii sú uzly usporiadané do stromu, v cestovej dekompozícii do jednej línie - cesty.

Definícia 4: [?]

Majme graf G. Rodina množín vrcholov $(V_i)_{i \in I}$ indexovaných lineárne podľa I sa nazýva *lineárna dekompozícia* pokiaľ spĺňa nasledujúce podmienky:

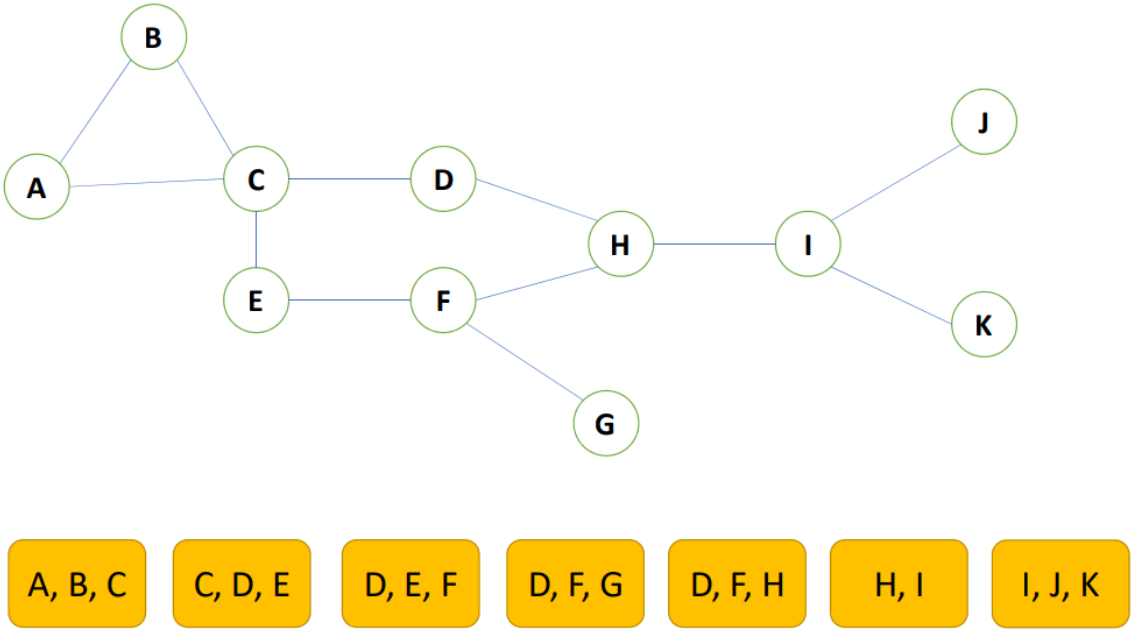
1. $V(G) = \bigcup_{i \in I} V_i$
2. pre každú hranu $e \in E(G)$ existuje také $i \in I$, že oba koncové vrcholy hrany e ležia v V_i
3. pre všetky $i, j, k \in I$ platí $i < j < k \implies V_i \bigcap V_k \subseteq V_j$

Lineárnu dekompozíciu konečného grafu G nazývame *cestová dekompozícia*.

V práci sa budeme zaoberať výlučne konečnými grafmi, preto je pre nás v tejto chvíli zaujímavá práve cestová a stromová dekompozícia.

1.4 Stromová a cestová šírka

Nie každá dekompozícia môže byť pre nás dostatočne uspokojivá. Podstatným atribútom, či už stromovej alebo cestovej dekompozície, je šírka rozkladu. Tá je odvodená od počtu vrcholov, ktoré sa nachádzajú v jej najväčšom uzle.



Obr. 1.2: Príklad cestovej dekompozície

Najprv si definujeme šírku rozkladu stromovej dekompozície.

Definícia 5:[4]

Šírka rozkladu (T, ν) je číslo

$$max|V_t| - 1 : t \in T$$

Pokiaľ sa jedná o cestovú dekompozíciu, šírka rozkladu sa určuje rovnako a má hodnotu

$$max|V_i| - 1 : i \in I$$

Definícia 6:[4]

Stromová šírka tw grafu G je najmenšia šírka rozkladu spomedzi všetkých stromových dekompozícií grafu G.

$$tw = max|V_t| - 1 : t \in T$$

Pozn.:Stromová šírka grafu na obrázku 1.1 je 3

Na základe predchádzajúcej definície o stromovej šírke poznamenávame, že cieľom stromovej dekompozície je dosiahnuť stromovú šírku, čo je ekvivalentné tvrdeniu, že stromová dekompozícia má za cieľ mať šírku rozkladu čo najmenšiu.

Cestová šírka je definovaná analogicky: množiny $V_t \in T$ nahradíme množinami $V_i \in I$ definovanými v cestovej dekompozícii

Definícia 7: [4]

Cestová šírka pw grafu G je najmenšia šírka rozkladu spomedzi všetkých cestových dekompozícií grafu G.

$$pw = max|V_i| - 1 : i \in I$$

Pozn.:Cestová šírka grafu na obrázku 1.2 je 3

Z definície stromovej šírky vidíme, že existuje trieda grafov s nulovou šírkou rozkladu. Pre túto triedu grafov platí, že neobasujú hrany. Ak by obsahovali hrany, dekompozícia by mala kvôli podmienke ii. v niektorom uzle dva vrcholy, teda stromová šírka by bola aspoň 1.

Ďalej môžeme nahliadnuť že pre triedu grafov so šírkou rozkladu $tw = 1$ platí, že pôvodné grafy sú lesy.[5] Vyplýva to z toho, že každá množina obsahuje najviac dva vrcholy, teda pôvodný graf nemôže obsahovať cyklus, lebo ak by obsahoval, neexistovala by taká dekompozícia, ktorá by tento cyklus rozbila na množiny s najviac dvomi vrcholmi.

Existuje mnoho ďalších takýchto tried grafov so zaujímavými vlastnosťami, avšak v našej práci sa budeme zaoberať hlavne cestovou dekompozíciou s cestovou šírkou, preto sa ďalej pozrime na to, prečo je cestová či stromová šírka taká podstatná v efektívnych algoritmoch.

Využitie v efektívnych algoritmoch

Na začiatku 70-tych rokov bolo spozorované, že veľká trieda optimalizačných problémov by mohla byť v prípade ohraničenosti dimenzie grafov efektívne vyriešená dynamickým programovaním.[6] Dimenzia grafu je pojem blízky stromovej šírke a síce hovorí o tom, že daný graf vieme vnoriť pri klasickej reprezentácii do euklidovského priestoru s určitou dimenziou.

Bolo ukázané, že pomocou dekompozície s cestovou, resp. stromovou šírkou vieme na grafoch pomocou dynmického programovania vyriešiť efektívnejšie množstvo problémov, ktoré sú $\mathcal{NP}$-ťažké.

Príklady:

- Jedným z takýchto problémov na grafoch je farbenie grafu. Vďaka dynamickému programovaniu poznáme algoritmy, ktoré pre graf s n vrcholmi a so stromovou šírkou k nájdu farbenie grafu v čase $O(k^{k+O(1)}n)$.[9]

- Z menej známych problémov spomenieme hľadanie celkového počtu nezávislých množín v grafe či hľadanie celkového počtu párení v grafe. Kolektív okolo Pengfeia Wana [10] vytvoril dva algoritmy, ktoré na základe stromovej šírky veľkosti k nájdu v n-vrcholovom grafe celkový počet nezávislých množín v čase $O(2^p \cdot pn^3)$ a počet párení v grafe v čase $O(2^{2p} \cdot pn^3)$.

1.5 Rez grafu

V kapitole 2 si uvedieme vetu o hornom ohraničení cestovej šírky spolu s dôkazom. Na jej dokázanie využijeme dôkaz vety o hornom ohraničení bisekcie 3-regulárnych grafov.

Na to aby sme vedeli čo to je bisekcia grafu, najprv si potrebujeme definovať *Rez Grafu*

Definícia 8:[8]

Rez $C = (S, T)$ grafu G je rozdelenie vrcholov grafu $V(G)$ do dvoch disjunktných množín S a T.

Rez, ktorý rozdelí vrcholy grafu $V(G)$ na množiny S, T ktorých veľkosť sa líši najvýš o jeden vrchol, sa nazýva *bisekcia*.

Definícia 9:[8]

Veľkosť rezu $C = (S, T)$ grafu G s neohodnotenými hranami je počet hrán $e = (v_j, v_k)$ takých, ktoré majú jeden koncový vrchol v množine S a druhý koncový vrchol v množine T.

$$\sum = \{(v_j, v_k) \in E(G) | v_j \in S, v_k \in T\}$$

Nasledovné 2 definície využijeme výlučne v dôkaze Lemy o 1-pomocných množinách.2.1

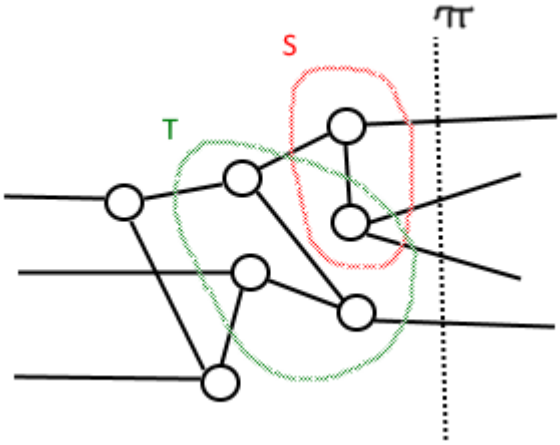
Definícia 10:[1]

Nech π je bisekcia grafu $G = (V, E)$.

Pre $S \subset V_p(\pi), p \in \{0, 1\}$ označme číslom

$$H(S) = |\{\{v, w\} \in E; v \in S, w \in V \setminus V_p(\pi)\}| - |\{\{v, w\} \in E; v \in S, w \in V_p(\pi) - S\}|$$

veľkosť zmenšenia (ak je $H(S) < 0$ tak zväčšenia) šírky rezu po presunutí množiny vrcholov S do druhej partície. Množinu S nazývame vzhľadom na rez π $H(S)$-pomocná.



Obr. 1.3: Ukážka rezu π a množín S a T

Príklady: (na obrázku 1.3)

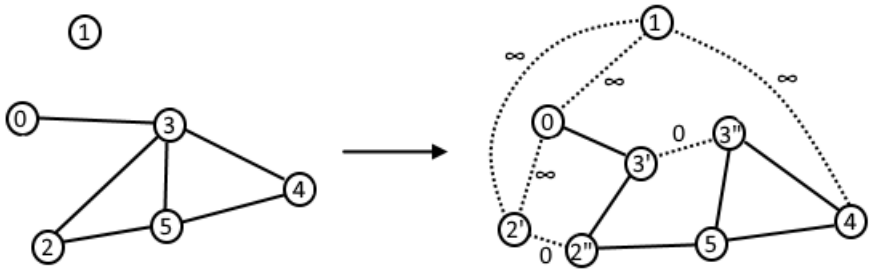
- Množina S je 2-pomocná. 3 hrany patria do rezu π, 1 hrana vychádzajúca z množiny S nepatrí do rezu π.
- Množina T je (-2)-pomocná. Takže po jej presunutí do druhej partície sa šírka rezu zväčší o 2 hrany.
- Vrcholy mimo množín S a T sú samé o sebe (-3)-pomocné

Definícia 11:[1]

Vrcholy množiny V_0 (V_1) klasifikujeme na základe toho ako ďaleko sa nachádzajú od rezu. V množine C sa budú nachádzať všetky vrcholy vo vzdialenosti 1 od rezu, t.j. vrcholy ktoré majú suseda v množine V_1 (V_0). V množine D sa budú nachádzať vrcholy vo vzdialenosti 2 od rezu, v množine E sa budú nachádzať vrcholy so vzdialenosťou aspoň 3 od rezu. Navyše, v množine $D_i, i \in \{1, 2, 3\}$ sa budú nachádzať všetky vrcholy z množiny D, ktoré susedia s i vrcholmi z množiny C. Teda $V_0 = C \uplus D_3 \uplus D_2 \uplus D_1 \uplus E$ Číslami $c(X), d_3(X), d_2(X), d_1(X), e(X)$ budeme označovať počet vrcholov daného typu v množine X$\subseteq V$.

1.6 Kubické grafy

V tejto bakalárskej práci sa budeme zaoberať stromovou a cestovou šírkou na kubických grafoch - teda takých, ktorých všetky vrcholy majú stupeň 3. Kubické grafy sú veľmi špecifickou triedou grafov. Mohlo by sa nám zdať, že stupeň vrchola 3 je malý, obmedzujúci a naše výsledky nebudeme môcť vedieť aplikovať na grafy vyšších stupňov. Avšak ako si ukážeme, každý graf vieme pretranformovať na kubický tak, že naň budeme môcť spustiť naše algoritmy a výsledky využiť v pôvodnom grafe.



Obr. 1.4: Príklad transformácie grafu na kubický graf. Pridané hrany sú Neohodnotené hrany majú základnú hodnotu 1.

Transformácia všeobecného grafu na kubický:

1. V prvom kroku upravíme každý vrchol $v \in V(G)$ s vyšším stupňom ako 3. Vrchol v vieme nahradiť vrcholmi v_1 a v_2. Vrcholu v_1 priradíme polovicu susedov vrchola v, vrcholu v_2 priradíme zvyšných susedov vrchola v. Medzi vrcholmi v_1 a v_2 bude hrana s váhou 0. Všetky hrany z pôvodného grafu budú mať váhu 1. Ohodnotením hrán vieme docieliť, že cesta medzi vrcholmi v_1 a v_2 má dĺžku 0, čo korešponduje so situáciou v pôvodnom grafe.

   Na transformácii uvedenej na obrázku 1.4 sme v prvom kroku rozdvojili vrchol 3 na vrcholy $3'$ a 3" a pridali medzi ne hranu s váhou 0.

2. V druhom kroku upravíme každý vrchol $v \in V(G)$ s menším stupňom ako 3. Nájdeme iný vrchol w_2 ktorý má menší stupeň ako 3. Pokiaľ taký vrchol nie je, tak aplikujeme predchádzajúci krok na vrchol w so stupňom 3, tým nám vzniknú dva nové vrcholy: w_3 so stupňom 3 a w_2 so stupňom 2. Vrcholy v a w_2 spojíme hranou s váhou ∞. Nová situácia v upravenom grafe korešponduje s pôvodným grafom. Hoci existuje hrana medzi vrcholmi, ktoré nie sú v pôvodnom grafe susedné, po pridanej hrane sa nedá prechádzať - algoritmy pri iterácii túto hranu preskočia.

   Na transformácii uvedenej na obrázku 1.4 sme v tomto kroku najprv pridali hrany $\{0,1\}, \{0,2\}, \{1,4\}$, nakoniec sme rozdvojili vrchol 2 na vrcholy $2'$ a 2" pridali hranu $\{1,2'\}$

Kapitola 2

Známe horné ohraničenia

V tejto kapitole si spravíme prehľad doteraz známych výsledkov v oblasti cestovej a stromovej šírky. Ukážeme si aký má cestová a stromová šírka vzťah k bisekcii grafu a ako vieme výsledky bisekcie grafu premostiť k zaujímavému výsledku na cestovej šírke.

Nakoľko sú všetky dôkazy vyslovených tvrdení v tejto kapitole konštrukčné, vynechávať budeme iba dôkazy o existencii jednotlivých entít v grafe. Väčšinu dôležitých konštrukčných krokov treba v tejto kapitole spomenúť, nakoľko ich budeme viac rozoberať v ďaĺších implementačných kapitolách 3 a 4

2.1 Šírka bisekcie

Poznáme širokú škálu problémov na grafoch, ktoré sa zaoberajú partíciami grafov. Úlohou je rozdeliť vrcholy grafu rovnomerne do určitého počtu partícií tak, aby počet hrán spájajúcich vrcholy z rôznych partícií bol čo najmenší. Pokiaľ počet partícií k nedelí počet vrcholov n grafu G, pripúšťame $n(mod)k$ množín takých, ktoré majú od všetkých ostatných množín počet vrcholov nanajvýš o 1 vyšší. Počet takýchto hrán zvykneme označovať ako *veľkosť rezu*.

Špeciálnym prípadom partície grafu je *bisekcia*, kedy vrcholy grafu máme rovnomerne rozdelené práve do dvoch množín. Veľkosť rezu bisekcie zvykneme označovať ako *šírka bisekcie*.

Začiatkom tohto tisícročia prišiel Burkhard Monien s Robertom Preisom k výsledkom [1] ktoré ukázali, že so zvyšujúcim sa počtom vrcholov kubických grafov sa horné ohraničenie šírky bisekcie týchto grafov blíži k $(\frac{1}{6})|V(G)|$ (zhora)

Vzťah bisekcie grafu a cestovej šírky

O pár rokov neskôr na základe predchádzajúcich výsledkov o bisekcii kubických grafov F.V.Fomin s K.Hoiem skonštruovali triedu algoritmov[2], ktoré pre ľubovoľné

$\epsilon > 0$ priradia $n_\epsilon \in \mathbb{N}$ a nájdu cestovú šírku grafu veľkú nanajvýš $(\frac{1}{6} + \epsilon)|V|$ pre všetky kubické grafy s počtom vrcholov $n > n_\epsilon$

Vzťahy medzi šírkou bisekcie a cestovej šírky však nie sú natoľko pevné. Ako vstupný parameter pre algoritmus pre cestovú dekompozíciu nepotrebujeme bisekciu, stačí nám dostatočne malý rez grafu, ktorý rozdeľuje vrcholy do množín ktoré sa veľkosťou nelíšia až príliš. takýto rez budeme nazývať *približná bisekcia.*() V kapitole 3 implementácia si *približnú bisekciu* definujeme presnejšie. Zatiaľ ju budeme v tejto kapitole používať ako abstraktný pojem.

Vo zvyšku kapitoly si predstavíme 2 lemy a vetu o šírke bisekcie na kubických grafoch spolu aj s ich konštrukčnými dôkazmi.

Lema o červených a čiernych hranách

Lema o červených a čiernych hranách sa so svojím konštrukčným dôkazom používa na nájdenie aspoň 1-pomocnej množiny spĺňajúcej lemu 2.1. Napriek tomu že jej použitie je v rámci dôkazu lemy 2.1 len jeden malý krok, zo všetkých dôkazov v tejto práci je tento najkomplikovanejší. Uvedieme si ho až na niektoré menej podstatné detaily v plnom znení.

Lema 1:

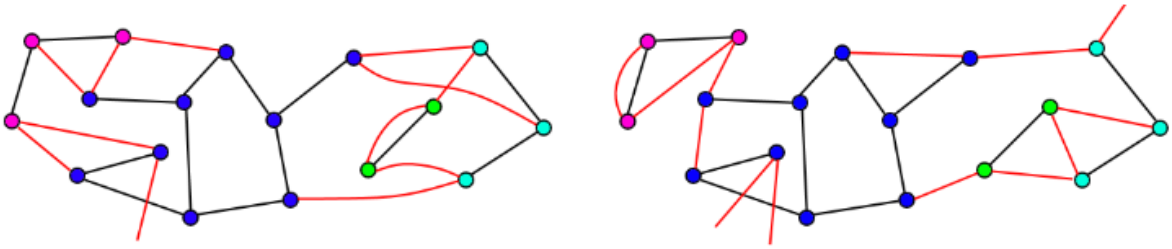
Nech $G = (V, E), E = B \uplus R$ je kubický graf s čiernymi hranami B a červenými hranami R. Nech je každý vrchol susedný s aspoň ednou čiernou hranou. Nech $|R| > (\frac{1}{2} + \epsilon)|V|$ pre $\epsilon > 0$. Potom existuje množina $S \subset V$ veľkosti $O(\frac{1}{\epsilon})$ taká, že počet červených hrán medzi vrcholmi množiny S je väčší ako počet čiernych hrán medzi vrcholmi množiny S a vrcholmi množiny $V \setminus S$

Dôkaz:

Uvažujme graf $G_{black} = (V, B)$ pozostávajúci z množiny vrcholov a čiernych hrán pôvodného grafu G. Nech F je rodina komponentov grafu G_{black}.

Komponent $I \in F$ považujeme v závislosti od počtu jeho vrcholov a zvoleného epsilon za malý, resp. veľký.

Komponent $I \in F$ považujeme za tenký pokiaľ je počet červených hrán spájajúcich vrcholy komponentu I s malými komponentami dostatočne malý v pomere k veľkosti zjednotenia tohto komponentu so všetkými malými komponentami cez dané červené hrany. V opačnom prípade považujeme komponent za hrubý.



Obr. 2.1: Príklad hrubého a príklad tenkého komponentu

Na obrázku 2.1 vľavo môžeme vidieť modrý komponent, ktorý je s malými komponentami spojený 8 červenými hranami a veľkosť jeho zjednotenia s týmito komponentami je 15. Modrý komponent považujeme za hrubý.

Vpravo na obrázku môžeme vidieť modrý komponent, ktorý je s malými komponentami spojený 3 červenými hranami a veľkosť jeho zjednotenia s týmito komponentami je 17. Modrý komponent považujeme za tenký.

Pokiaľ existuje malý komponent $I \in F$ taký, že v grafe G existuje červená hrana medzi vrcholmi tohto komponentu, daný komponent spĺňa lemu, keďže je dostatočne malý a má aspoň jednu internú (medzi vrcholmi komponentu) červenú hranu. Kvôli tomu že je to čierny komponent, neexistuje čierna hrana spájajúca I s $V(G) \setminus I$, teda červených interných hrán je viac ako čiernych externých.

Pokiaľ existujú dva malé komponenty $I, J \in F$ spojené červenou hranou, zjednotenie týchto komponentov spĺňa lemu, keďže je dostatočne malé a je tam minimálne jedna interná červená hrana a žiadna externá.

Pokiaľ sme doposiaľ zostali neúspešní, z F budeme vyhadzovať nasledovné komponenty:

1. Vyhoďme všetky malé komponenty $I \in F$ ktoré obsahujú čierne hrany tvoriace cyklus
2. Pokým existuje veľký a hrubý komponent $I \in F$, vyhoď I z F aj so všetkými malými komponentami, ktoré sú s ním spojené červenou hranou

pozn: Počas kroku ii. sa môžu tenké komponenty zmeniť na hrubé a naopak.

Po vykonaných úpravách všetky zvyšné komponenty v F sú

1. veľké a tenké, alebo
2. malé a čierne hrany tvoria strom

Pokiaľ sme doteraz nenašli množinu vrcholov spĺňajúcu lemu, v rodine komponentov F sa s istotou nachádza veľký a tenký komponent. (dôkaz o jeho existencii vynecháme)

Nech $I_{vt} \in F$ je veľký a tenký komponent. Nech $G_{I_{vt}}$ je graf indukovaný čiernymi hranami komponentu I_{vt}. Daný graf teraz v dvoch krokoch zredukujeme.

Z grafu $G_{I_{vt}}$ postupne zmažeme všetky vrcholy stupňa 1 až žiaden taký nezostane. Po ich zmazaní nahradíme v grafe $G_{I_{vt}}$ všetky súvislé úseky vrcholov stupňa 2 jednou hranou (spájajúcou vrcholy stupňa 3). Každá pridaná hrana e medzi vrcholy stupňa 3 nahrádza vymazané vrcholy, ktoré tvoria strom. Označme túto množinu ako $T(e)$.

Analogicky s tenkosťou, resp. hrubosťou množiny vrcholov budeme novopridané hrany označovať ako netučné, resp. tučné.

pozn: opäť kontrolujeme pomer medzi počtom červených hrán spájajúcich $T(e)$ s malými komponentmi a veľkosťou zjednotenia $T(e)$ s týmito množinami.

V grafe $G_{I_{vt}}$ existuje vrchol v, že počet červených hrán, ktoré spájajú $T(e)$ s malým komponentom, pričom e je netučná hrana susediaca s vrcholom v, je aspoň 4. (dôkaz o existencii vrchola vynechávame) Toto číslo označme ako $n(v)$.

- Pokiaľ pre vrchol v platí $n(v) = 4$, tak zjednotenie vrcholu v s vymazanými stromami na ktoré odkazujú susedné netučné hrany vrchola v, spolu s malými komponentami spojenými 4 červenými hranami, spĺňa lemu, keďže tam sú aspoň 4 interné červené hrany (spájajúce malé komponenty) a nanajvýš 3 externé čierne hrany (netučné hrany môžu byť 3, pričom so zvyškom grafu $G_{I_{vt}}$ sú spojené čiernou hranou)
- Pokiaľ $n(v) \geq 5$ a vrchol v je sused s dvomi netučnými hranami, kde každá z nich odkazuje na vymazaný strom ktorý spájajú dve červené hrany s malými komponentmi, potom zjednotenie vrcholu v s vymazanými stromami daných dvoch netučných hrán spolu s malými komponentmi spojenými 4 červenými hranami, spĺňa lemu, keďže tam sú aspoň 4 interné červené hrany (spájajúce malé komponenty) a 3 externé čierne hrany (netučné hrany sú 2, pričom so zvyškom grafu $G_{I_{vt}}$ sú spojené čiernou hranou, susedná hrana vrchola v ktorá nebola spomenutá je treťou externou čiernou hranou)

Pokiaľ sme stále nenašli množinu spĺňajúcu lemu, niektorá zo susedných hrán e vrchola v je netučná a $T(e)$ spájajú s malými komponentmi aspoň 3 červené hrany.

Ukážeme, ako vieme nájsť v $T(e)$ množinu vrcholov takú, že spolu s malými komponentami spojenými cez červenú hranu táto množina spĺňa lemu.

Vrcholy spojené s $T(e)$ čiernou hranou označme v_1, v_2. Majme množinu $K = T(e) \cup \{v_1, v_2\}$. K je strom. Zoberme jeden z vrcholov v_1, v_2 ako koreň a hrany nech sú orientované smerom od koreňa k listom. Označme v strome vrcholy ktoré spĺňajú nasledujúce podmienky ako kandidujúce:

- Vrchol v vo svojom podstrome neobsahuje žiaden z vrcholov $v_1.v_2$
- Podstrom vrchola je spojený s malými komponentmi 2 červenými hranami
- Podstrom vrchola neobsahuje žiaden iný vrchol ktorý spĺňa predošlé dve podmienky, t.j. vrchol je čo najďalej od koreňa

Pokiaľ pre niektorý z vrcholov čo spĺňa vyššie uvedené podmienky nie je zjednotenie jeho podstromu s malými komponentami spojenými 2 červenými hranami príliš veľké, toto zjednotenie spĺňa lemu (2 interné červené hrany spájajúce malé komponenty, 1 čierna externá hrana od vrcholu v smerom k rodičovi v strome K)

Pokiaľ je však pre všetky vrcholy spĺňajúce podmienky spomínané zjednotenie príliš veľké, vytvorme nový graf $\overline{G}$ pozostávajúci z vrcholov v_1, v_2, kandidujúcich vrcholov a všetkými cestami (vrcholmi stupňa 2) medzi nimi.

Spomínané cesty v grafe $\overline{G}$ korešpondujú s tučnými/netučnými hranami. Pre cestu P označme $T(P) = \bigcup_{v \in P} v \cup T(w)$, kde $w \in K \setminus \overline{G}$ je sused vrchola v. Ak taký nie je, $T(w)$ je prázdna množina. $T(P)$ teda obsahuje všetky všetky vrcholy na ceste P a všetky podstromy týchto vrcholov nepatriace grafu $\overline{G}$. Pre vrcholy v stupňa 3 v grafe $\overline{G}$ označíme $n(v)$ počet červených hrán spájajúcich $T(P)$ s malými komponentami červenou hranou pre všetky P ktoré susedia s vrcholom v. V grafe existuje vrchol v pre ktorý $n(v) \geq 4$. Dôkaz o jeho existencii vynecháme. Cesty P susediace s vrcholom v korešpondujú s netučnými hranami.

Budeme postupovať analogicky ako pri predchádzajúcom prípade. Pokiaľ $n(v) = 4$, vieme nájsť množinu vrcholov spĺňajúcu lemu. Takisto ak $n(v) \geq 5$ a pre dve z ciest P_1, P_2 platí, že $T(P_1)$ spájajú s malými komponentami 2 červené hrany, vieme nájsť množinu vrcholov spĺňajúcich lemu.

Pokiaľ sme ale stále nenašli množinu vrcholov spĺňajúcu lemu, pre niektorú cestu P susediacu s vrcholom v platí, že $T(P)$ spájajú s malými komponentami aspoň 3 červené hrany.

Vrchol $v \in P$ nazvime červený, ak má suseda $w \in K \setminus \overline{G}$ takého, že $T(w)$ je spojené s malým komponentom cez červenú hranu,

- Pokiaľ je $T(P)$ spojené s malými komponentami nanajvýš cez 4 červené hrany, zjednotenie cesty P a týchto malých komponentov spojených 3 červenými hranami je množina spĺňajúca lemu. Sú tam aspoň 3 interné červené hrany spájajúce malé komponenty, 2 externé čierne hrany spájajúce okraje cesty P so zvyškom grafu $\overline{G}$. Zjednotenie je pritom dostatočne malé.
- Pokiaľ je $T(P)$ spojené s malými komponentami aspoň cez 5 červených hrán, vieme nájsť takú trojicu $\{v_1, v_2, v_3\}$ červených vrcholov na ceste P, že zjednotenie $T(w_1), T(w_2), T(w_3)$, ciest medzi vrcholmi v_1, v_2, v_3 a malých komponentov aspoň cez 3 červené hrany je množina spĺňajúca lemu. Sú tam aspoň 3 interné červené hrany spájajúce malé komponenty, 2 externé čierne hrany spájajúce okraje cesty P so zvyškom grafu $\overline{G}$. Dôkaz, že je zjednotenie dostatočne malé, vynecháme.

Týmto sme dokázali pravdivosť lemy o červených a čiernych hranách.

Lema o 1-pomocných množinách

V tejto časti si ukážeme návod, ako nájsť množinu vrcholov takú, ktorej presunutím do druhej partície zmenčím veľkosť rezu.

Lema 2:

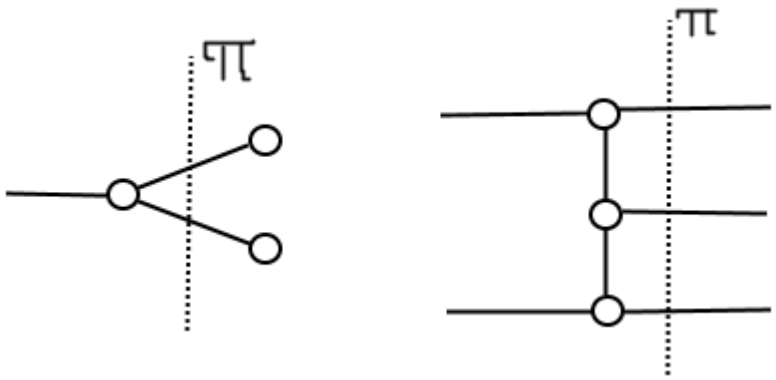
Nech π je *približná bisekcia*(2.1) grafu $G = (V, E), V = V_0 \uplus V_1$. Pokiaľ je šírka rezu väčšia ako $(\frac{1}{3} + 2\epsilon)|V_0|, \epsilon > 0$, vieme nájsť aspoň 1-pomocnú množinu $S \in V_0$ veľkosti $O(\frac{1}{\epsilon})$.

Teraz si v skratke predstavíme hlavné myšlienky dôkazu.

Dôkaz:

V množine V_0 sa môžu nachádzať situácie, ktoré priamo vedú k malej 1-pomocnej množine ktorá spĺňa lemu.

1. Pokiaľ má niektorý z C-vrcholov aspoň 2 susedov v V_1, tento vrchol sám o sebe tvorí aspoň 1-pomocnú množinu.



Obr. 2.2: Na obrázku môžeme vidieť jednoduché príklady 1-pomocných množín typu 1. a 2.

2. Množina 3 spojených C-vrcholov tvorí 1-pomocnú množinu.
3. Nech vrchol v má C-suseda, ktorý je susedom ďalšieho C alebo D_3 vrchola. Pokiaľ zvyšní dvaja susedia vrchola v sú z množiny $C \cup D_2 \cup D_3$, zjednotenie všetkých spomenutých vrcholov a ich C-susedov tvorí aspoň 1-pomocnú množinu veľkosti nanajvýš 11 vrcholov.

Ak sme doteraz nenašli žiadnu štruktúru spĺňajúcu lemu, na grafe G_{V_0}spravíme transformácie, po ktorých v grafe nezostanú žiadne D_3 vrcholy a taktiež ani hrany medzi C-vrcholmi.

Po vykonaní spomenutých transformácií použijeme na grafe $\overline{G_{V_0}}$ lemu o červených a čiernych hranách 2.1 na nájdenie 1-pomocnej množiny.

Nájdenú množinu na transformovanom grafe $\overline{G_{V_0}}$ upravíme tak, aby to bola 1-pomocná množina v pôvodnom grafe G_{V_0}

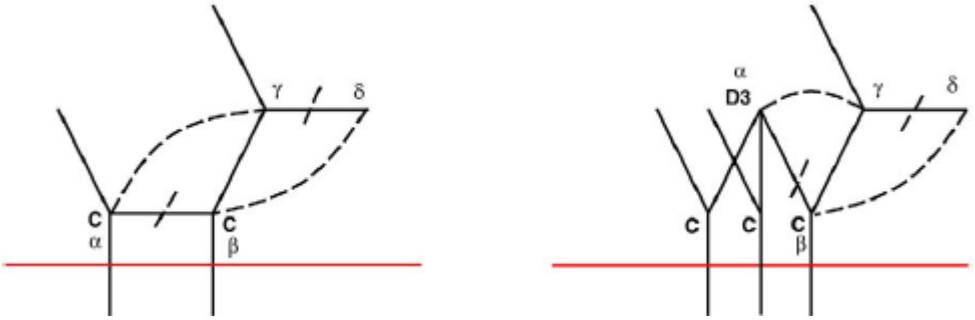
Transformácia 1:

Pôvodná situácia: $\alpha, \beta \in C, \gamma \in D_1 \cup D_2, \delta \in D_1 \cup E, \{\alpha, \beta\}, \{\beta, \gamma\}, \{\gamma, \delta\} \in E(G_{V_0})$. V transformovanom grafe $\overline{G_{V_0}}$ nahradíme hrany $\{\alpha, \beta\}, \{\gamma, \delta\}$ hranami $\{\alpha, \gamma\}, \{\beta, \delta\}$

Transformácia 2:

Pôvodná situácia: $\alpha \in D_3, \beta \in C, \gamma \in D_1 \cup D_2, \delta \in D_1 \cup E, \{\alpha, \beta\}, \{\beta, \gamma\}, \{\gamma, \delta\} \in E(G_{V_0})$. V transformovanom grafe $\overline{G_{V_0}}$ nahradíme hrany $\{\alpha, \beta\}, \{\gamma, \delta\}$ hranami $\{\alpha, \gamma\}, \{\beta, \delta\}$

Vytvorím nový graf K, ktorý bude pozostávať z $D \cup E$ vrcholov. Všetky hrany medzi vrcholmi $D \cup E$ v grafe $\overline{G_{V_0}}$ budú v grafe K čierne. Červené



Obr. 2.3: Transformácia 1 a transformácia 2

hrany budú medzi každými dvomi vrcholmi, ktoré sú v $\overline{G_{V_0}}$ spojené so spoločným C-vrcholom.

K spĺňa predpoklady pre lemu 1 pre $\overline{\epsilon} = 3\epsilon$. Použitím lemy 1 nájdeme 1-pomocnú množinu $\overline{S}$, ktorá spolu so susednými C-vrcholmi spĺňa lemu pre graf $\overline{G_{V_0}}$.

Zostáva nám ukázať, ako vieme 1-pomocnú množinu $\overline{S}$ v $\overline{G_{V_0}}$ pretransformovať na 1-pomocnú množinu S v G_{V_0}.

Všetky tranformácie ktoré sme vykonali na grafe G_{V_0} teraz budeme inverzne simulovať (teda aj vykonávať v opačnom poradí). V skutočnosti nebudeme pridávať ani odoberať hrany, ale iba upravovať 1-pomocnú množinu S v závislosti od konfigurácie vrcholov v grafe $\overline{S}$.

Pre vykonávanú inverznú tranformáciu si skontrolujeme či sa $\alpha, \beta, \gamma, \delta$ nachádzajú v 1-pomocnej množine.

Inverzná transformácia 2:

1. $\alpha, \gamma \in \overline{S}, \beta, \delta \notin \overline{S} \implies S = \overline{S} \cup \{\beta\}$
2. $\alpha, \gamma \notin \overline{S}, \beta, \delta \in \overline{S} \implies S = \overline{S} \cup \{\alpha, \gamma\} \cup \{v \in C; \{v, \alpha\} \in E\}$

Inverzná transformácia 1:

1. $\alpha, \gamma \in \overline{S}, \beta, \delta \notin \overline{S} \implies S = \overline{S} \cup \{\beta\}$
2. $\alpha, \gamma \notin \overline{S}, \beta, \delta \in \overline{S} \implies S = \overline{S} \cup \{\alpha, \gamma\}$

Obidve inverzné transformácie zväčšia 1-pomocnú množinu S prinajhoršom o konštantu, teda S spĺňa lemu.

Lema o 1-pomocnej množine nám hovorí o tom, že pokiaľ je šírka rezu emphpribližnej bisekcie(2.1) väčšia ako $(\frac{1}{3} + 2\epsilon)|V_0|, \epsilon > 0$ kde $|V_0| \geq |V_1|$, vieme vo V_0 nájsť dostatočne malú 1-pomocnú množinu.

Vždy po aplikovaní tejto lemy danú 1-pomocnú množinu presunieme do druhej partície.

Lemu budeme aplikovať až pokým sa nedostaneme do situácie, že šírka rezu je menšia ako $(\frac{1}{3} + 2\epsilon)|V_0|$, pričom keď $\frac{|V_0|}{|V_1} \approx 1$, horné ohraničenie šírky (približnej) bisekcie bude približne $(\frac{1}{6} + \epsilon)|V|$

2.2 Cestová šírka

V dôkaze o dekompozícii kubického grafu budeme potrebovať vetu o dekompozícii stromu, ktorú si teraz uvedieme bez dôkazu.

Veta a dekompozícii stromu[11]

Pre ľubovoľný strom T s počtom vrcholov $n \geq 3$ platí, že cestová šírka stromu T je nanajvýš $\log_3 n$

Lema o cestovej dekompozícii

V tejto časti si ukážeme možný návod, ako nájsť cestovú dekompozíciu s dostatočne malou šírkou rozkladu.

Lema 3:

Nech G je kubický graf. Pre každú množinu $X \subseteq V(G)$ existuje cestová dekompozícia $(X_1, X_2, ..., X_r)$ grafu G so šírkou nanajvýš

$$max\{|X|, \lfloor n/3 \rfloor\} + (2/3)\log_3 n + 1$$

pričom $X = X_r$

Dôkaz:

Lemu budeme dokazovať matematickou indukciou.

Na grafe s jedným vrcholom je tvrdenie lemy triviálne. Predpokladajme, že tvrdenie platí pre všetky grafy s počtom vrcholov menším ako n pre nejaké $n > 1$

Majme graf G s n vrcholmi a množinu $X \subseteq V(G)$.

Môžu nastať viaceré situácie:

Prípad 1: nájdeme vrchol $v \in X$ pre ktorý $N(v) \setminus X = \emptyset$, t.j. v nemá žiadneho suseda mimo množiny X.

Podľa indukčného predpokladu existuje cestová dekompozícia $(X_1, X_2, ..., X_r)$ grafu $G \setminus \{v\}$ so šírkou rozkladu nanajvýš $max\{|X| - 1, \lfloor (n-1)/3 \rfloor\} + (2/3)\log_3 n - 1 + 1$, pričom $X = X_r$. Pridaním vrchola v do množiny X_r nadobúda cestová dekompozícia šírku rozkladu nanajvýš $max\{|X|, \lfloor n/3 \rfloor\} + (2/3)\log_3 n + 1$

Prípad 2: nájdeme vrchol $v \in X$ pre ktorý $|N(v) \setminus X| = 1$, t.j. vrchol v má mimo množiny X práve jedného suseda. Označme tento susedný vrchol u.

Podľa indukčného predpokladu pre $G \setminus \{v\}$ a $X \setminus \{v\} \cup \{u\}$ existuje cestová dekompozícia $P' = (X_1, X_2, ..., X_r)$ grafu $G \setminus \{v\}$ so šírkou rozkladu nanajvýš $max\{|X|, \lfloor (n-1)/3 \rfloor\} + (2/3)\log_3 n - 1 + 1$, pričom $X \setminus \{v\} \cup \{u\}$

Vytvoríme novú cestovú dekompozíciu P z P' pridaním množín $X_{r+1} = X \cup \{u\}, X_{r+2} = X$, takže novovzniknutá cestová dekompozícia bude mať tvar $P = (X_1, X_2, ..., X_r, X_{r+1}, X_{r+2})$ a jej šírka rozkladu bude nanajvýš $max\{|X|, \lfloor n/3 \rfloor\} + (2/3)\log_3 n + 1$.

Prípad 3: pre každý vrchol $v \in X$ platí $|N(v) \setminus X| \geq 2$, t.j. vrchol má aspoň 2 susedov mimo množiny X. V takejto situácii rozlišujeme dva podprípady, podľa toho či je množina X veľká alebo malá. (detailnejší postup je ukázaný v pôvodnom článku)

Podprípad 3.A: pokiaľ je množina X vzhľadom na graf G veľká, vieme povedať, že v $G \setminus X$ sa nachádza komponent T, ktorý je strom. Podľa vety o dekompozícii stromov 2.2 existuje cestová dekompozícia $P^1 = (X_1, X_2, ..., X_r)$ grafu T so šírkou rozkladu nanajvýš $(2/3)\log_3 n$.

Podľa indukčného predpokladu existuje cestová dekompozícia $P^2 = (Y_1, Y_2, ..., Y_t = X)$ grafu $G \setminus V(T)$ so šírkou rozkladu nanajvýš $|X| + 1$. Kombináciou týchto dvoch cestových dekompozícií dostaneme dekompozíciu $P = (Y_1, Y_2, ..., Y_t = X, X_1 \cup X, X_2 \cup X, ..., X_r \cup X, X)$ so šírkou rozkladu nanajvýš $max\{|X|, \lfloor n/3 \rfloor\} + (2/3)\log_3 n + 1$

Podprípad 3.B: pokiaľ je množina X vzhľadom na graf G malá, pridáme do množiny X taký počet vrcholov z $V(G) \setminus X$ aby bola množina vzhľadom na graf G veľká. Po pridaní vrcholov do množiny X sa budeme nachádzať v niektorom z vyššie uvedených prípadov.

Matematickou indukciou sme dokázali tvrdenie lemy.

Veta o cestovej šírke

Veta 4:

Pre ľubovoľné $\epsilon > 0$ vieme nájsť celé kladné číslo n_ϵ, že pre každý 3-regulárny graf $G, |V(G)| > n_\epsilon$ existuje cestová dekompozícia so šírkou rozkladu nanajvýš

$$(\tfrac{1}{6} + \epsilon)|V(G)|$$

Dôkaz:

Majme graf G. Vďaka liem o červených hranách 2.1 a 1-pomocných množinách 2.1 vieme nadobudnúť *približnú bisekciu*(2.1) so šírkou rezu nanajvýš $(\frac{1}{6} + \epsilon)|V(G)|$.

Označme partície V_0, V_1. C-vrcholy v V_0 označme $\Delta(V_0)$, C-vrcholy v V_1 označme $\Delta(V_1)$.

Pomocou lemy o dekompozícii 2.2 vieme vytvoriť cestové dekompozície $P_0 = (X_1, X_2, ..., X_r = \Delta(V_0), P_1 = (Y_1, Y_2, ..., Y_t = \Delta(V_1)$podgrafov $G[V_0], G[V_1]$.

Tieto dekompozície majú veľkosť nanajvýš

$$max\{(\tfrac{1}{6} + \epsilon)n, \lfloor n/6 \rfloor + 1\} + (2/3)\log_3 n + 1$$

Teraz vytvoríme dekompozíciu $P_x = (Z_1, Z_2, ..., Z_s)$. Množina Z_1 bude obsahovať vrcholy $\Delta(V_0)$, Množina Z_s bude obsahovať vrcholy $\Delta(V_1)$. Pre Z_i, kde i je nepárne, vyberieme vrchol $v \in Z_i \setminus Z_s$ pričom ďalšie dve množiny konfigurujeme následovne: $Z_{i+1} = Z_i \cup N(v) \cap Z_s, Z_{i+2} = Z_{i+1} \setminus \{v\}$

Keďže v každom kroku vyhodím jeden vrchol z V_0 a nahradím ho jeho susedmi V_1, po konečnom počte krokov sa dostaneme k úprave množiny Z_s, ktorá obsahuje vrcholy z $\Delta(V_1)$.

Konečná dekompozícia grafu G sa skladá zo spomenutých 3 čiastkových dekompozícií a má tvar $P = (X_1, X_2, ..., X_r, Z_1, Z_2, ..., Z_s, Y_t, Y_{t-1}, ..., Y_1)$

Kapitola 3

Implementácia optimálneho rezu

V tejto kapitole si ukážeme, ako sme pri implementácii *približnej bisekcie*(2.1) riešili technické problémy rôzneho druhu.

Implementáciou dôkazov budeme prechádzať v chronologickom poradí ako v kapitole 2, preto sa budeme odvolávať na časti dôkazov iba keď to bude vyslovene potrebné.

V práci budeme používať knižnicu *ba_graph*, aj kvôli ktorej sme používali programovací jazyk *C++*. Na začiatok si teda predstavíme ktoré prostriedky tejto knižnice budeme v práci využívať.

Ako každá iná grafová knižnica, tak aj táto obsahuje základné triedy ako *Vertex* alebo *Edge*, teda poskytuje rozhranie pre vrcholy a hrany. My však budeme vo väčšine implementácie pristupovať k vrcholom cez triedu *Number*, ktorá má v sebe uložené číslo vrchola. To nám, ako uvidíme neskôr, pre problematiku cestovej šírky postačuje. Vrcholy grafu budeme mať pre rýchlosť prístupu uložené v dátovej štruktúre *set*.

K hranám grafu budem pristupovať cez štruktúru *multiset*, v ktorej budem mať uložené dvojice vrcholov, teda $\{Number, Number\}$. Dôvodom ukladania hrán do *multiset* je prípustnosť viacnásobných hrán, čo *multiset* povoľuje. Trieda *Number* je jedným z atribútov triedy *Vertex*, takže jednoduchosť zostane ako vstup programu graf obsahujúci vrcholy a hrany, z ktorého následne získam údaje o vrcholoch a hranách a Nepopierateľnú výhodu to má v tom, že spracovávanie programu bude pri triviálnejších triedach rýchlejšie.

Napriek ukázaniu možnej transformácie 1.4 všeobecného grafu na kubický budeme predpokladať, že vstupný graf je kubický.

Z knižnice sa budeme stretávať ešte s ďalším prvkom - triedou *Factory*, ktorá nadobúda význam hlavne vtedy keď pracujeme v jednom programe s viacerými grafmi paralelne. Nakoľko budeme vždy riešiť šírku rezu práve jedného grafu, touto triedou sa už viac nebudeme v tomto texte zaoberať.

3.1 Lema o červených a čiernych hranách

Cieľom lemy 1 je nájsť takú množinu vrcholov, v rámci ktorej je viac červených hrán medzi vrcholmi ako čiernych hrán vychádzajúcich z množiny vrcholov mimo množinu.

Aby sme mali všetky potrebné informácie o grafe, potrebujeme okrem počte vrcholov a zozname hrán vedieť aj ktoré hrany sú červené a ktoré sú čierne. Ako si neskôr ukážeme, červených hrán bude vo väčšine prípadov menej, teda ako druhý vstupný údaj budeme potrebovať zoznam červených hrán. Zvyšné hrany v grafe budú čierne. Nakoniec už len potrebujeme *Factory* a hodnotu ϵ, ktoré získame ako paramter z lemy 2, ktorú si ukážeme neskôr.

V prípade, že sa nachádza v grafe vrchol ktorý má červenú slučku, tento vrchol spĺňa lemu, keďže slučka je považovaná 2 hrany, teda vrchol má 2 interné červené hrany a jednu externú čiernu. (Tretia hrana musí byť čierna kvôli podmienke v leme)

Keď máme informáciu o tom ktoré hrany sú čierne a akú hodnotu má epsilon, môžeme si vytvoriť jednoduché funkcie, ktoré zistia, či je množina vrcholov veľká, malá, tenká, hrubá, či pozitívna (spĺňajúca lemu).

Na to aby sme zistili či je množina vrcholov I tenká alebo hrubá potrebujeme vedieť koľko červených hrán je takých, že spája danú množinu s malými komponentami. Označením SC budeme chápať zjednotenie všetkých malých čiernych komponentov.

Ukážeme si 3 rôzne prístupy:

- Prvá možnosť je prechádzať všetkými červenými hranami a zisťovať či jeden z jej koncových vrcholov je v I a druhý patrí do niektorého malého komponentu. Na to aby sme zistili, či je druhý v niektorej malej množine, musíme iteratívne prejsť všetky malé komponenty. Preto časová náročnosť tohto prístupu je $O(|R|\mu|SC|)$, čo pre nás nie je uspokojivé.

- Ďalší spôsob je namiesto červených hrán prejsť všetkými vrcholmi I, no opäť sa dostávame do podobného problému, kedy zistiť, či susedný vrchol cez červenú hranu patrí do malého komponentu je časovo náročné. Tento prípad je ale predsa o niečo lepší a trvá $O(|I|\mu|SC|)$.

- Keď vymeníme poradie iterácií a najprv prejdeme všetky vrcholy SC a overíme, či nie ich sused patrí do množiny I, celkový postup nás bude stáť $O(|SC|)$ času

Veľkosť zjednotenia množiny I so všetkými malými komponentami spojenými červenou hranou budeme riešiť analogicky. Čo však môžeme oproti predchádzajúcemu postupu vylepšiť je, že vždy keď pre daný malý komponent nájdeme červenú hranu spájajúcu komponent s I, zväčším zjednotenie o veľkosť tohto malého komponentu a preruším for cyklus pre vrcholy daného malého komponentu.

Keď už máme základné funkcie prichystané, uložíme si komponenty indukované čiernymi hranami do F vo formáte set(set<Number>). To spravíme jednoduchým prehľadávaním s využitím zaznačenia už navštívených vrcholov. Z daného vrcholu sa môžem pohybovať iba po čiernych hranách a do nenavštívených susedov, v ktorých budem postup rekurzívne opakovať. Po návrate z rekurzí funkcia uloží navštívené vrcholy. Keďže je graf konečný, táto funkcia tiež skončí.

Podľa dôkazu 2.1 potrebujeme overiť, či sa nenachádza v F taký malý komponent I, ktorý by bol pozitívny, resp. by bol spojený s iným malým komponentom červenou hranou.

Prvú možnosť vieme skontrolovať triviálne, pomocou už vytvorenej funkcie *isPositive*, pre druhý prípad použijeme funkciu *smallSetsViaRedEdge*, ktorá nájde všetky malé komponenty spojené s I.

Funkcia *smallSetsViaRedEdge* má parametricky dané, či chceme nájsť všetky malé komponenty alebo stačí jeden taký. V kroku spomenutom vyššie chceme práve jeden komponent, preto po jeho nájdení funkcia skončí. Táto funkcia bude v svojom plnom rozsahu využitá ešte neskôr.

V konštrukčnom dôkaze tejto lemy vrcholy aj hrany mažeme. Aby sme neprišli o dôležité údaje ktoré budeme neskôr potrebovať, nebudeme vrcholy ani hrany v skutočnosti mazať, ale iba ich budeme označovať za vymazané. Na druhú stranu si to však vyžaduje, aby sme vždy pri prechádzaní susedov vrcholov kontrolovali, či nie je daný sused vymazaný.

Pokiaľ sa nenachádza v F žiaden komponent spĺňajúci lemu, funkciami *step1, step2* upravíme množinu komponentov F podľa dôkazu.

- V kroku 1 využijeme fakt, že v grafe s vrcholmi stupňa ≤ 3 sa kružnica nachádza práve vtedy, keď je v ňom počet 1-stupňových vrcholov nanajvýš taký ako počet 3-stupňových. [8]

- Počas druhého kroku sa, narozdiel od prvého, môžu dodatočne vyskytnúť niektoré konfigurácie ktoré treba vymazať. Druhý krok ukončím až v momente, ako som spravil jednu celú iteráciu elementami z F bez toho aby som niektorý z nich vyhodil.

Redukciu niektorého veľkého tenkého komponentu riešim v nasledujúcich dvoch krokoch:

1. *reduceStep1*: Pre každý vrchol komponentu na redukovanie sa pozriem koľko má nevymazaných susedov spojených čiernou hranou. Pokiaľ má iba jedného takého suseda, označím vrchol za vymazaný a zavolám mazaciu funkciu *removeLine* na

susedný vrchol. Mazacia funkcia sa zastaví keď niektorý vrchol bude mať dvoch nevymazaných čiernych susedov. Keďže sa removeLine volá funkciou reduceStep1, ktorá prejde postupne všetkými vrcholmi, po redukcii nezostane žiaden vrchol stupňa 1.

2. *reduceStep2*: Vymazanie všetkých ciest tvorených vrcholmi stupňa 2 nie je problém. Treba si však zaznačovať vrcholy ktoré vymažeme a potom ich vymazať naraz. Postupné mazanie by zapríčinilo zníženie stupňa vrcholov a mohlo by sa stať, že by sme takto vymazali všetky vrcholy v komponente.

V reduceStep2 nepridávam hrany medzi vrcholy stupňa 3, spravím tak až vo funkcii *edgeRef*, ktorá ako parameter okrem iného vyžaduje zoznam vymazaných vrcholov. Pole vymazaných vrcholov znegujeme, teda vrcholy komponentu po redukcii budú označené ako vymazané. Potom zavoláme funkciu connectedComponents, ktorou získame zredukované stromy. Pre každý strom zistíme ktorý vrchol mimo stromu je spojený čiernou hranou. Takéto vrcholy sú dva a medzi nimi bude hrana odkazujúca na tento vymazaný strom.

Funkciu *isFat* vytvoríme analogicky ako funkciu *isThin*.

Vo funkcii *n(v)*, ktorá pre netučné hrany e incidentné s vrcholom v spočíta počet červených hrán spájajúcich $T(e)$ s malým komponentom, skombinujeme už vytvorené funkcie *isFat* a *alfa*, ktorá počíta červené hrany spájajúce množinu s malým komponentom.

V prípade ak pre niektorý vrchol v platí $n(v) = 4$, funkciou smallSetsViaRedEdge dostaneme požadovanú množinu spĺňajúcu lemu. Všimnime si, že v tomto prípade nechceme iba jeden komponent ako na začiatku.

Podobne postupujeme aj v prípade $n \geq 5$

V situácii, kedy vo vymazanom strome označujeme vrcholy ako kandidujúce, hlavne využívame funkcie *designateNodes* a *undesignateAbove*. Prvá z funkcií overí prvé dve podmienky pre kandidujúci vrchol.

Tretia podmienka hovorí o tom, že kandidujúci vrchol nemôže mať v podstrome žiaden iný kandidujúci vrchol. Inak povedané, medzi každou dvojicou *koreň-kandidujúci vrchol* nemôže byť iný kandidujúci vrchol, teda vždy po označení vrchola ako kandidujúceho funkcia undesignateAbove všetky vrcholy medzi koreňom a kandidujúcim vrcholom označí ako nekandidujúce.

Konštrukciu grafu $\overline{G}$ pozostávajúcu zo všetkých ciest medzi kandidujúcimi vrcholmi a koreňom vytvoríme pomerne jednoducho: pre každý kandidujúci vrchol pridám do množiny všetkých jeho predkov až po koreň. Pokiaľ som na ceste narazil na vrchol ktorý už je pridaný, pridávanie vrcholov do grafu ukončím.

Keďže vrcholy grafu máme uložené v dátovej štruktúre *set*, o duplicitu vrcholov sa nemusíme obávať.

Podobne získame všetky cesty medzi kandidujúcimi vrcholmi. Funkcie *getAllPaths* a *getPath* prechádzajú od každého kandidujúceho vrchola smerom hore.

- Ak daný vrchol nie je koreň, over či jeho rodič má iba jedno dieťa. Ak áno, pridaj ho do cesty, ak nie, je to vrchol stupňa 3, keďže má 2 deti a ako nekoreňový vrchol aj rodiča. V takom prípade ho do cesty nepridaj, cestu pridaj do množiny ciest a getPath spusti na 3-stupňovom rodičovi.

- Ak daný vrchol je koreň, doterajšiu cestu nahraďme novou, ktorú získame z funkcie *pathOfChildren*, ktorá vytvorí cestu z koreňa a vrcholov s jedným dieťaťom.

Nasledujúci úsek s funkciou $n(v)$ riešime analogicky ako v predošlom prípade.

V prípade, že $N(v) \geq 5$ a zároveň pre niektorú cestu P z vrchola v platí, že $3 \leq \alpha(T(P)) \leq 4$, požadovanú množinu nájdeme ľahko. Pokiaľ $\alpha(T(P)) \geq 5$, zistíme, ktoré vrcholy na ceste P sú červené, t.j. pre ktoré $v \in P$ platí $\alpha(T(v)) = 1$.

Aby sme mohli nájsť množinu spĺňajúcu lemu, potrebujeme si červené vrcholy dať do poradia *sequence* v akom nasledujú na ceste P. To spravíme tak, že z vrchola v zistíme, ktorý sused patrí do cesty P. Tento sused je na jednom jej kraji. Postupne budeme prechádzať na druhý koniec cesty a vždy pri prechádzaní červeným vrcholom tento vrchol pridáme do *sequence*. *sequence* musí byť vector, aby si prvky uchovával v lineárnom poradí.

Podobne v poradí budeme mať uložené aj cesty medzi červenými vrcholmi. Pre jednoduchosť algoritmu treba spomenúť, že prvý element v zozname týchto ciest nie je cesta medzi dvomi červenými vrcholmi.

Stačí nám už len prejsť všetky trojice červených vrcholov $v_i, v_{i+1}, vi + 2$ a dvoch ciest P_{i+1}, P_{i+2} medzi nimi. Každú takúto päticu skontrolujeme, či je zjednotenie $T(V_i) \cup T(v_{i+1}) \cup T(v_{i+2}) \cup T(P_{i+1}) \cup T(P_{i+2})$ ako doteraz najmenšie nájdené. Ak áno, označme túto päticu ako doteraz najmenšiu nájdenú.

Po $|P|-2$ krokoch nájdeme päticu, ktorá spolu s malými komponentami cez červené hrany spĺňa lemu.

3.2 Lema o 1-pomocných množinách

V tejto leme budú veľkú úlohu zohrávať transformácie - pridávanie a odstraňovanie hrán.

Podobne ako v predchádzajúcej leme, ani v tejto nebudeme pridávať či odstraňovať hrany na vstupnom grafe. Potrebujeme si ho zachovať pre ďalšie použitie. Transformované hrany si budeme uchovávať nasledovne.

V *neighbors* si budeme ukladať ktorý vrchol má akých susedov v pôvodnom grafe. V *transormedNeighbors* budeme mať uložených susedov daného vrchola po vykonaných transformáciách. Pre prípady viacnásobných hrán či slučiek máme susedov vrchola uložených v *multiset*.

Na začiatku overíme prítomnosť jednoduchých konfigurácií vrcholov, ktoré by priamo viedli k-pomocnej množine. Pokiaľ sme žiadne nenašli, vykonáme transformácie podľa dôkazu.

Vykonané transformácie si budeme ukladať do *vector*, keďže po nájdení 1-pomocnej množiny na transformovanom grafe budeme vykonávať inverzné transformácie v reverznom poradí, takže ich potrebujeme mať uložené v správnom poradí.

Pri transformácii 1 hľadáme 4 vrcholy priamo. Začneme od ľubovoľného C-vrcholu a zistím či nemá C-suseda. Ak áno, nájdem ďalšie vrcholy podľa dôkazu.

Je dôležité si zapamätať, kde sa nachádzal 3. a 4. vrchol. Aby sme vedeli, či γ treba vymazať z D_1 alebo z D_2 a pridať do D_2, resp. D_3. Podobne aj pri vrchole δ.

Aj pri transformácii 2 hľadáme požadované vrcholy priamo cez susedov. Tu si však nepotrebujeme pamätať do ktorej množiny patrí γ, keďže pridaná ani odobratá hrana z vrcholu γ ho nespájala s C-vrcholom, teda vrchol γ zostane v množine D_i v ktorej sa nachádza.

Vytvoríme graf s červeno-čiernymi hranami podľa dôkazu a nájdeme 1-pomocnú množinu.

Vykonáme transformácie typu 2 v rverznom poradí a následne aj transformácie typu 1 v reverznom poradí.

Upravená 1-pomocná množina spolu aj so susednými C-vrcholmi spĺňa lemu. Pokiaľ je však množina príliš veľká, že po jej presune do V_1 bude $|V_0| < \frac{|V(G)|}{3}$, vrátime prázdnu množinu, čo bude znamenať, aby sme ukončili hľadanie optimálneho rezu.

3.3 Optimálny rez

Ako vstupné parametre funkcie *cutSizeBisection* dostaneme graf a hodnotu ϵ. Vytvoríme si partície grafu V_0, V_1, ktoré sa budú líšiť veľkosťou najviac o jeden vrchol.

Aby následné hľadanie optimálneho rezu netrvalo dlho, nebudeme do V_0, resp. V_1 pridávať vrcholy náhodne. Na začiatok pridáme jeden vrchol do V_1 a všetky ostatné do V_0.

V premennej *cut* budem mať zoznam rezových hrán, teda z množiny V_0 do V_1.

Pokým $|V_1| < |V_0|$, v konštantnom čase zvolíme jednu rezovú hranu, jej koncový vrchol $v \in V_0$ vyhodíme z V_0 a pridáme do V_1. Po aktualizovaní partícií prejdeme hrany presunutého vrchola. Hrany ktoré sú vo V_1 vyhodíme z *cut*, hrany ktoré sú v V_0 pridáme

do *cut*. Treba si však uvedomiť, že to môžeme spraviť až dodatočne, keď sa skončí *for auto* cyklus, keďže pridaním alebo odstránením elementu z *cut* by iterátor stratil svoju pozíciu a program by padol. Nakoniec aktualizujeme *cut*.

Môžeme si všimnúť, že po takomto inicializovaní partícií neexistuje v V_1 vrchol, ktorý by bol vzhľadom na rez 3-pomocný, keďže každý vrchol má aspoň 1 suseda vnútri svojej partície, teda jeho presunutím rez zmenšíme nanajvýš o 1.

Aby sme implementovali čo najviac uspokojujúci optimálny rez, rozdelili sme sme si zmenšovanie šírky rezu do 2 fáz.

V prvej fáze hľadáme 1-pomocné množiny pokým je šírka rezu väčšia ako je požadované alebo pokým je jedna z partícií viac ako dvojnásobne väčšia oproti druhej.

Prvú fázu hľadania nám môže ukončiť aj situácia, kedy 1-pomocnú množinu nenájdeme.

V druhej fáze sa snažíme o ešte menšiu šírku rezu pokiaľ to ide. Samozrejme, kontrolujeme aby sme nemali niektorú z partícií príliš veľkú.

Po ukončení druhej fázy vrátime ako výsledok optimálneho rezu zoznam vrcholov v oboch partíciách.

Kapitola 4

Implementácia cestovej dekompozície

V tejto kapitole si ukážeme ako implementovať dôkaz o cestovej dekompozícii s horným ohraničením cestovej šírky.

4.1 Lema o cetovej dekompozícii

Lema 3 je dokázaná matematickou indukciou, teda na implementovaná funkcia bude rekurzívna.

Funkcia bude upravovať dekompozíciu, teda vektor množín vrcholov, ktorú si bude posúvať ako parameter do vnoreného volania.

Množinu X budeme nazývať *currentSet*. Keď v currentSet nie je žiaden vrchol, znamená to, že sme na konci dekompozície danej partície. Ak je množina currentSet neprázdna, môžeme sa dostať do 4 rôznych situácií.

Tri z nich vieme podľa dôkazu lemy pomerne priamo implementovať. Zostáva nám pozrieť sa na prípad *3.A*, kedy robíme dekompozíciu stromu.

Dekompozíciu stromu zrealizujeme tiež rekurzívne. Okrem aktuálneho vrcholu *vertex* budeme ako parameter posielať aj predošlý vrchol *from*. Na začiatku si niektorý vrchol stupňa ≤ 2 označíme ako from a dekompozíciu pustíme na jeho susedoch.

Idea je následovná:

1. Pokiaľ nemáš žiadneho suseda okrem from, do dekompozície pridaj množinu s vrcholom vertex

2. Pre každého suseda okrem from rekurzívne zavolaj túto funkciu, následne do každej pridanej množiny v dekompozícii pridaj aj vrchol vertex

Časová zložitosť síce nie je na úrovni ako prezentovali Ellis a spol. [11], avšak stále je logaritmická, čo je pre nás stále uspokojivé.

4.2 Veta o cestovej šírke

V tejto podkapitole si zhrnieme postup ako demonštrovať cestovú dekompozíciu s ohraničenou šírkou.

Ako vstup programu dostaneme ľubovoľný kubický graf. Na ňom nájdeme podľa 3 optimálny rez.

Na množinách V_0 a V_1 vykonáme podľa 4 cestové dekompozície P_0, P_1, ktorých krajné množiny zodpovedajú $cut \cap V_0$, resp. $cut \cap V_1$. Zostáva nám spraviť dekompozíciu spájajúcu P_0 a P_1. Implementácia z dôkazu tohto postupu je priama.

Nakoniec, aby výsledná dekompozícia bol prehľadnejšia, spravíme menšiu redukciu: každú množinu v dekompozícii, ktorá je podmnožinou susednej, vyhodíme.

Keďže nadmnožiny týchto množín sú susedné a zahrňujú všetky hrany čo tieto množiny, podmienky regulárnej cestovej dekompozície zostanú neporušené.

Záver

Cestová dekompozícia grafov je veľmi dôležitým faktorom efektívnych algoritmov na grafoch, ale aj mnohých iných oblastiach matematiky a informatiky.

Ako sme mohli vidieť, nadobudnúť dobrú šírku cestovej dekompozície je náročné a jej nadobudnutie si vyžaduje veľa študovania a skúmania.

Implementáciou cestovej dekompozície sme vytvorili vhodnú platformu na skúmanie súvislostí medzi grafmi s nižšou, ale aj vyššou cestovou šírkou, na ktorej vieme skúmať túto oblasť ešte do väčšej hĺbky.

Algoritmus v tejto práci spraví cestovú dekompozíciu grafu, ktorej šírka rozkladu je cestová šírka, resp. sa k nej blíži.

Stromová šírka je podobne ako cestová šírka dôležitý a využívaný pojem v algoritmoch na grafoch. Keďže však o nej nie sú známe natoľko dobré výsledky, zameriavali sme sa prevažne na cestovú šírku.

Literatúra

[1] Burkhard Monien - Robert Preis, 2006, Journal of Discrete Algorithms 4: Upper bounds on the bisection width of 3- and 4-regular graphs, 475s.-491s.

[2] Fedor V. Fomin - Kjartan Høie, 2005, Information Processing Letters 97: Pathwidth of cubic graphs and exact algorithms, 191s.-196s.

[3] Jianmin Li - John Lillis - Chung-Kuan Cheng, Dept. of Computer Sci. & Engr., University of California, San Diego: Linear Decomposition Algorithm for VLSI Design Applications

[4] Wikipedia, Pathwidth, použité 24.1.2019, [online] Dostupné na internete: <https://en.wikipedia.org/wiki/Pathwidth>

[5] Wikipedia, Treewidth, použité 24.1.2019, [online] Dostupné na internete: <https://en.wikipedia.org/wiki/Treewidth>

[6] Wikipedia, Tree decomposition, použité 24.1.2019 [online] Dostupné na internete: <https://en.wikipedia.org/wiki/Tree$_d$*ecomposition* >

[7] Keijo Ruohonen, Graph Theory, 2013

[8] Reinhard Diestel, Graph Theory, 1997

[9] Shuji Isobe - Xiao Zhou - Takao Nishizeki, A Polynomial-Time Algorithmfor Finding Total Colorings of Partialk-Trees, Graduate School of Information Sciences, Tohoku University

[10] Pengfei Wan - Jianhua Tu - Shenggui Zhang - Binlong Li, Computing the numbers of independent sets and matchings of all sizes for graphs with bounded treewidth, 2018, 42s.-47s.

[11] J.A. Ellis - I.H. Sudborough - J.S. Turner, The vertex separation and search number of a graph, Inform. and Comput. 113, 1994, 50s.–79s.