Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Geršak

Povečanje povezavne povezanosti grafa

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentorica: izr. prof. dr. Arjana Žitnik

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Povečanje povezavne povezanosti grafa

Tematika naloge:

Problem povečanja povezavne povezanosti v grafu zahteva, da za dani graf poiščemo najmanjšo množico povezav, ki jo je potrebno dodati grafu, da doseže neko predpisano povezavno povezanost.

V diplomskem delu obravnavajte problem povečanja povezavne povezanosti grafa. Opišite kaktusno reprezentacijo, ki dano množico najmanjših prerezov v grafu predstavi s kaktusom, ki ima enako množico najmanjših prerezov. Izpeljite algoritem, ki s pomočjo kaktusne reprezentacije poišče najmanjšo množico povezav, ki jo je treba dodati grafu, da mu povezavno povezanost povečamo za ena. Potem obravnavajte še splošnejši problem povečanja povezavne povezanosti grafa na dano vrednost in zanj predstavite učinkovit algoritem.

Iskreno se zahvaljujem mentorici, izr. prof. dr. Arjani Žitnik, za vso strokovno pomoč, potrpežljivost in vztrajnost. Zahvaljujem se tudi družini, ki me je spodbujala in podpirala tekom celotnega študija.

Kazalo

Povzetek

Naslov: Povečanje povezavne povezanosti grafa

Avtor: Jan Geršak

V diplomski nalogi obravnavamo problem povečanja povezavne povezanosti grafa. V prvem delu diplomske naloge predstavimo kaktusno reprezentacijo grafa in opišemo njeno konstrukcijo, za katero predstavimo tudi algoritem. V drugem delu diplomske naloge predstavimo povezavo med povezanostjo grafa in povezanostjo njegove kaktusne reprezentacije. S pomočjo te povezave določimo spodnjo mejo za število potrebnih povezav za povečanje povezavne povezanosti grafa za ena in dokažemo, da je vedno dosežena. Nato podamo algoritem, ki s pomočjo normalne kaktusne reprezentacije cikličnega tipa poveča povezavno povezanost grafa za ena. V tretjem delu diplomske naloge predstavimo splošno metodo razcepljanja povezav in jo uporabimo v Frankovem algoritmu za povečanje povezavne povezanosti grafa na dano vrednost. Tu tudi dokažemo Maderjev izrek, ki nam omogoča uporabo metode razcepljanja povezav za izvajanje Frankovega algoritma.

Ključne besede: graf, kaktusna reprezentacija, povečanje povezavne povezanosti, razcepljanje povezav, Frankov algoritem, Maderjev izrek.

Abstract

Title: Edge-connectivity augmentation of a graph

Author: Jan Geršak

In this thesis we consider the edge-connectivity augmentation problem. In the first part of the thesis we present a cactus representation of a graph and describe its construction for which we present an algorithm. In the second part of the thesis we consider the relation between edge-connectivity of a graph and its cactus representation. Using this relation we give a lower bound for the least number of edges to be added to increase the edge-connectivity of a graph by one. We also prove that the lower bound is always achievable. Then we give an algorithm for edge-connectivity augmentation by one by applying properties of the cycle-type normal cactus representation. In the third part of the thesis we present general edge splitting method which is used in Frank's algorithm for solving edge-connectivity augmentation problem. We also prove Mader's theorem which is needed to prove finiteness of edge splitting in Frank's algorithm.

Keywords: graph, cactus representation, edge connectivity augmentation, edge splitting, Frank's algorithm, Mader's theorem.

Poglavje 1

Uvod

Predstavljajmo si državo, ki načrtuje izgradnjo dodatnih cest v svojem obstoječem cestnem omrežju. Na žalost pa so v tej državi pogoste naravne nesreče in z njimi povezane nenačrtovane zapore cest. Predsednik države ima v načrtu, da izda naročilo za izgradnjo najmanjšega števila cest, da bo država prevozna kljub k naključnjim zaporam, kjer je k dovolj veliko naravno število. Tak problem imenujemo problem razširitve omrežja in ga rešujemo z algoritmi za povečanje povezavne povezanosti.

Rešitev problema povečanja povezavne povezanosti pa je uporabna tudi v navidez čisto drugačnih problemih, kot so problem togosti v mrežnih ogrodjih [1, 6], problem podatkovne varnosti [7, 9] in načrtovanje vezij [19].

Problem povečanja povezavne povezanosti na dano vrednost k so začeli raziskovati v letu 1976, ko so Eswaran in Tarjan v [2] kot tudi Plesnik v [17] pokazali, da ima rešitev problema povečanja povezavne povezanosti na dve polinomsko časovno zahtevnost. Nato sta leta 1987 Watanabe in Nakamura v [20] podala algoritem za rešitev problema za splošni k s časovno zahtevnostjo $O(k^2(kn+m)n^4)$, pri čemer je n število vozlišč in m število povezav grafa. Leta 1992 je Frank v [5] podal univerzalni pristop k različnim problemom povečanja povezavne povezanosti z uporabo Maderjevega izreka o razceplja-nju povezav s časovno zahtevnostjo $O(n^3 \log(k)(m + n \log n))$. Kasneje, leta 1997, sta Nagamochi in Ibaraki v [13] izboljšala Frankov pristop z uporabo

algoritma za najmanjši prerez na časovno zahtevnost $O((nm+n^2 \log n) \log n)$.

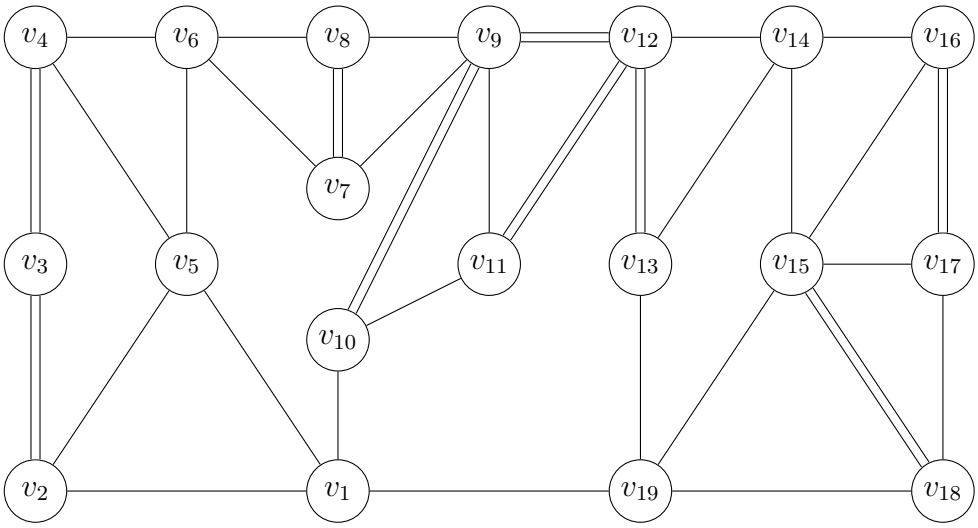
Mi se bomo najprej osredotočili na lažji problem povečanje povezavne povezanosti za ena. V ta namen bomo definirali kaktusno reprezentacijo za graf, ki predstavi najmanjše prereze grafa. Opisali bomo tudi algoritem za konstrukcijo kaktusne reprezentacije. Ta algoritem bi lahko uporabili tudi za povečanje povezavne povezanosti na dano vrednost tako, da bi ga večkrat poklicali, vendar bi s tem morda dodali več povezav, kot je nujno. S primerno izbiro vozlišča znotraj lista kaktusa pa lahko dosežemo, da dodamo najmanjše možno število povezav [15]. Predstavili bomo tudi Frankov pristop povečanja povezavne povezanosti na dano vrednost, ki je, ko je k velik, hitrejši, in predstavili ter dokazali Maderjev izrek.

Opišimo na kratko zgradbo diplomskega dela. Uvodnemu poglavju sledi poglavje o grafih, v katerem definiramo potrebne pojme. V tretjem poglavju podrobno obravnavamo kaktusno reprezentacijo. V četrtem poglavju obravnavamo problem povečanja povezavne povezanosti za ena. V zadnjem, petem poglavju pa obravnavamo problem povečanja povezavne povezanosti na dano vrednost s pristopom Franka.

Poglavje 2

Grafi

V tem poglavju bomo definirali osnovne pojme s področja teorije grafov in podali notacijo, ki jo bomo uporabljali v nadaljevanju. Navezovali se bomo na vire [14, 21, 18]. Začeli bomo z definicijo grafa in nekaterih osnovnih lastnosti.



Slika 2.1: Graf G.

Definicija 2.1. *Graf* G sestavlja neprazna množica elementov, ki jih imenujemo *vozlišča* grafa, in seznam (neurejenih) parov teh elementov, ki jih imenujemo *povezave* grafa. Množico vozlišč grafa označimo z $V(G)$, seznam

povezav pa z $E(G)$. Če sta v in w vozlišči grafa G, potem za povezavo $\{vw\}$ rečemo, da *povezuje* vozlišči v in w. Povezavo $\{vw\}$ bomo krajše pisali tudi kot vw ali wv.

Definicija 2.2. Dve povezavi ali več povezav, ki povezujejo isti par točk, poimenujemo *vzporedne povezave*. Povezava, ki povezuje neko točko s seboj, je *povratna povezava* ali *zanka*. Graf brez zank in večkratnih povezav poimenujemo *enostavni graf*.

Definicija 2.3. Naj bo G graf z množico vozlišč $V(G)$ in seznamom povezav $E(G)$ in G' graf z množico vozlišč V' in seznamom povezav E'. Če je V' podmnožica množice $V(G)$ in če je vsaka povezava iz seznama E' tudi v seznamu $E(G)$, potem je G' *podgraf* grafa G.

V nadajevanju bomo besedo graf uporabljali za graf brez zanke in z vzporednimi povezavami. Primer grafa, pri katerem sta upoštevani ti lastnosti, je na sliki 2.1.

Vsaki povezavi e med krajišči u in v ustreza par *usmerjenih povezav* (u, v) in (v, u). Vozlišče u je začetek, vozlišče v pa konec usmerjene povezave (u, v). Usmerjeno povezavo bomo v grafih označevali s puščico. Povezavo $e = (u, v)$, ki je usmerjena od u proti v, označimo krajše kar uv.

Definicija 2.4. Graf $G = (V, E)$ je *usmerjen*, če je E množica usmerjenih povezav.

Definicija 2.5. Zaporedje vozlišč V in povezav E v grafu G, $T = (v_1, e_1, v_2, e_2, ..., e_{k-1}, v_k)$ imenujemo *sprehod* med v_1 in v_k, če $v_1, v_2, ..., v_k \in V$, $e_1, e_2, ..., e_{k-1} \in E$ in $e_i = \{v_i, v_{i+1}\}$ za $i = 1, 2, ..., k-1$. Sprehod je *enostaven sprehod*, če so vse povezave sprehoda različne. Če so v enostavnem sprehodu vsa vozlišča različna, potem sprehod poimenujemo *pot*. Pot $P = (v_1, e_1, v_2, e_2, ..., e_{k-1}, v_k)$ lahko krajše zapišemo tudi kot zaporedje vozlišč $P = (v_1, v_2, ..., v_k)$ ali kot zaporedje povezav $P = (e_1, e_2, ..., e_{k-1})$. Za vozlišči $u, v \in V$ v grafu G pot med u in v imenujemo (u, v)-pot ali uv-pot.

Skrčitev povezave $e = vw$ je operacija na grafu G, pri kateri združimo vozlišči v in w v eno vozlišče ter ohranimo vse sosede vozlišč v, w. Odstranimo le zanke, ki nastanejo iz povezave e in morebitnih vzporednih povezav povezavi e. Graf, ki ga pridobimo s skrčitvijo te povezave, označimo z G/e.

Definicija 2.6. Za podmnožico povezav $F \subseteq E$ naj $G - F$ označuje graf, dobljen iz G z odstranitvijo povezav iz množice F. Graf G/F naj bo dobljen iz G s skrčitvijo vsake povezave $e \in F$ v eno vozlišče in odstranitvijo vseh zank. Podobno za poljubna podgrafa A, B grafa G označimo $A - B$ graf, pridobljen z odstranitvijo B iz A, kjer odstranimo vozlišča $v \in (B \cap A)$ in vse povezave, v katerih je v krajišče. Graf $A + B$ vsebuje vsa vozljišča in povezave A in B.

Definicija 2.7. Za vozlišče $v \in V$ v grafu $G = (V, E)$ naj $\Gamma_G(v)$ označuje množico sosedov v. Podobno naj za množico $X \subseteq V, \Gamma_G(X)$ oziroma $\Gamma(X)$ označuje množico $\{v \in V - X : uv \in E \text{ za nek } u \in X\}$ sosedov X v V. Z X^* označimo množico $X^* = V - (X + \Gamma(X))$ vozlišč, ki niso v X niti niso sosednja X.

Definicija 2.8. *Stopnja vozlišča* v v grafu G je število povezav s krajiščem v v in jo označimo z $deg(v)$. Najmanjšo stopnjo vozlišč v grafu G označimo z $\delta(G)$. Največjo stopnjo vozlišč v grafu G označimo z $\Delta(G)$.

Občasno bomo graf G z vzporednimi povezavami obravnavali kot enostaven *utežen* graf, kjer so uteži na vsaki povezavi $e = \{u, v\}$ predstavljene s številom vzporednih povezav med u in v. Utež na povezavi $e = \{u, v\}$ označimo z $c_G(e)$ ali $c_G(u, v)$ in je celoštevilska.

Definicija 2.9. Za podmnožici $X, Y \subseteq V$ (ne nujno disjunktni) naj $E(X, Y; G)$ označuje množico povezav e, ki povezujejo vozlišča iz X in Y (torej povezav oblike $e = \{u, v\}$, kjer $u \in X$ in $v \in Y$). Naj $d(X, Y; G)$ označuje $\sum_{e \in E(X,Y;G)} c_G(e)$. Množici $E(X, Y; G)$ in $d(X, Y; G)$ lahko zapišemo krajše kot $E(X; G)$ in $d(X; G)$, če velja $Y = V - X$. Zaradi praktičnosti privzamemo $d(\emptyset; G) = d(V; G) = 0$. Stopnja vozlišča je enaka $d(v; G)$. V primeru,

da je G jasen iz konteksta, lahko $E(X, Y; G)$ in $d(X, Y; G)$ zapišemo tudi kot $E(X, Y)$ in $d(X, Y)$.

Naslednji sklop definicij bo uvedel prereze v grafu in povezavno povezanost grafa.

Definicija 2.10. Naj bo $X \subseteq V$ neprazna množica in $X \neq V$. Potem particijo $\{X, V - X\}$ imenujemo *prerez*. Za prerez $\{X, V - X\}$ bomo v nadaljevanju uporabljali oznako $\{X, \overline{X}\}$ ali kar X.

Definicija 2.11. Množica povezav $E' \subseteq E$ je *prerezna množica* povezav za prerez $\{X, V - X\}$, če velja $E(X, V - X) \subseteq E'$. Povezava e je *prerezna povezava*, če je $\{e\}$ prerezna množica povezav za nek prerez. Če za prerezno množico povezav E' velja $E\{X, V - X\} = E'$, pravimo, da je prerez X *generiran* z E'.

Definicija 2.12. *Velikost prereza* X, generiranega s prerezno množico E', definiramo kot $d(X; G) = |E'|$. Za množici $S, T \subseteq V$ pravimo, da ju prerez X *ločuje*, če velja $S \subseteq X \subseteq V - T$ ali $T \subseteq X \subseteq V - S$. Če prerez X ločuje S in T, potem X imenujemo (S, T)-*prerez* oziroma če S in T vsebujeta samo po eno vozlišče s oziroma t, ga imenujemo (s, t)-prerez. Za vozlišči s in t v G imenujmo (s, t)-prerez X najmanjše velikosti *najmanjši* (s, t)-*prerez* in prerez velikosti $d(X; G)$ imenujemo *lokalna povezavna povezanost* med s in t in jo označimo z $\lambda(s, t; G)$. Po konvenciji je za $\lambda(v, v; G)$ določena vrednost $+\infty$.

Definicija 2.13. Prerez X v grafu G imenujemo *najmanjši prerez*, če velja $d(X; G) = \min_{u \in X, v \in \overline{X}} \lambda(u, v; G)$. V grafu G množico vseh najmanjših prerezov označimo z $C(G)$. Prerez Z v grafu G imenujemo *minimalni najmanjši prerez*, če je $Z \subset V$ v G najmanjši prerez in ne obstaja $X \subset Z$, da je X najmanjši prerez.

Definicija 2.14. *Povezavna povezanost* grafa $G = (V, E)$ je velikost najmanjšega prereza med vsemi (s, t) prerezi v grafu G. Označimo jo z $\lambda(G)$ in

je definirana kot $\lambda(G) = \min_{u,v \in V} \lambda(u, v; G)$. Pravimo, da je graf G *povezan*, če je $1 \leq \lambda(G)$, in k-povezavno povezan, če velja $k \leq \lambda(G)$ in $k \in \mathbb{N}$.

Definicija 2.15. Povezan graf G je *cikel*, če so vsa njegova vozlišča stopnje 2. Cikel, ki vsebuje k vozlišč, imenujemo *k-cikel*. Povezan graf G je *drevo*, če ne vsebuje nobenega cikla. Povezan graf G je *veriga*, če je sestavljen iz samih 2-ciklov in je vsako vozlišče največ stopnje 4. Povezan graf G je *zvezda*, če ima eno vozlišče stopnje $|V(G)| - 1$, vsa ostala vozlišča pa imajo stopnjo 1.

Definicija 2.16. Povezava $e = \{s, t\}$ v G je *kritična*, če velja $\lambda(s, t; G) = \lambda(G)$.

Če kritično povezavo odstranimao, zmanjšamo povezavno povezanost grafa G.

V nadaljevanju bomo obravnavali še različne sprehode v grafih. Sprehod, ki se začne in konča v istem vozlišču, imenujemo *obhod*, in če so vse povezave obhoda različne, ga imenujemo *enostavni obhod*. Povezan graf je *Eulerjev*, če obstaja enostavni obhod, ki vsebuje vse povezave grafa. Tak obhod imenujemo *Eulerjev obhod*.

Izrek 2.17. *Naj bo G povezan graf. Potem je G Eulerjev natanko tedaj, ko ima vsako vozlišče G sodo stopnjo.*

Gornji izrek je dokazan na primeru v ([21, stran 149]).

Iskanje Eulerjevega obhoda je lahek problem, zanj obstaja več algoritmov. Predstavili bomo Hierholzerjev algoritem iz [4] za iskanje Eulerjevega obhoda, ki je bil objavljen že leta 1873 in najde rešitev v času $O(|E(G)|)$, ne glede na graf.

Hierholzerjev algoritem (G)

1. Izberemo poljubno začetno vozlišče v in gradimo sprehod iz še neuporabljenih povezav, dokler se ne vrnemo v v. Na ta način dobljeni sprehod je obhod, za katerega pa ni nujno, da vsebuje vse povezave začetnega grafa.

2. Dokler na trenutnem obhodu obstaja vozlišče, ki ima sosednje vozlišče, ki ni del obhoda, začenjamo nov obhod iz u po še neuporabljenih povezavah, dokler se ne vrnemo v u. Nato nov obhod priključimo prejšnjemu.

Ker so stopnje vseh vozljišč sode, se pri gradnji sprehoda vedno enkrat vrnemo v začetno vozlišče, saj ko vstopimo v vozlišče, mora vedno obstajati povezava, po kateri lahko izstopimo.

Definicija 2.18. Naj bo G povezan graf in $s, t \in V(G)$. Pravimo, da sta dve st-poti po *povezavah disjunktni*, če nimata skupne povezave. Pravimo, da sta dve st-poti po *vozliščih disjunktni*, če nimata nobenega skupnega vozlišča (razen s in t).

Pomemben rezultat, ki povezuje najmanjše število povezav in število disjunktnih poti med dvema točkama grafa, je Mengerjev izrek. Poglejmo si množico povezav, ki ločijo s in t v poljubnem povezanem grafu. Ker odstranitev teh povezav uniči vse poti med s in t, mora vsaka st-pot vsebovati vsaj eno od teh povezav. Od tod sledi, da je največje število po povezavah disjunktnih st-poti manjše ali enako številu povezav v množici, ki loči s in t. To velja tudi za najmanjšo takšno množico. V tem primeru se izkaže, da sta števili enaki.

Izrek 2.19. ***Mengerjev izrek za grafe (oblika za povezave)***
Naj bo G povezan graf in $s, t \in V(G)$. Največje število po povezavah disjunktnih st-poti je enako najmanjšemu številu povezav, ki ločujejo s in t.

Za dokaz izreka glej ([21, stran 209]). S pomočjo Mengerjevega izreka lahko lokalno povezanost $\lambda(s, t; G)$ poiščemo tako, da poiščemo največje število disjunktnih poti med s in t. Le to pa je enostavno poiskati s pomočjo pretokov, na primer z algoritmom Haoa in Orlina, ki za neusmerjen graf lokalno povezanost najde v času $O(nm \log(n^2/m))$ [8].

Časovno zahtevnost za iskanje najmanjšega preseka in s tem povezavne povezanosti za G lahko najdemo v $O(mn \log(n^2/mk))$ v usmerjenem uteženem

grafu [8]. Najmanjši presek v neusmerjenem uteženem grafu lahko najdemo v enaki časovni zahtevnosti z uporabo algoritma Haoa in Orlina na usmerjenem grafu G', ki ga dobimo tako, da vsako neusmerjeno povezavo zamenjamo z dvema nasprotno usmerjenima povezavama z enako utežjo. S tem algoritmom je Fleischer v [3] dokazala konstrukcijo kaktusne reprezentacije z enako časovno zahtevnostjo.

Poglavje 3

Kaktusna reprezentacija

V tem poglavju bomo predstavili, kako s kaktusno reprezentacijo predstavimo množico vseh najmanjših prerezov grafa. Definirali bomo več različnih tipov kaktusnih reprezentacij in predstavili različne tipe particij. Nato bomo v razdelku *Osnovne operacije* predstavili ključne gradnike algoritma za konstrukcijo kaktusne reprezentacije grafa. Podali bomo tudi algoritem in njegovo časovno zahtevnost. Navezovali se bomo na vir [14].

3.1 Tipi reprezentacij

Povezan graf je *kaktus*, če vsaka povezava pripada največ enemu ciklu. Če ima graf samo eno vozlišče, ga imenujemo *trivialni kaktus*. Ostali preprosti primeri kaktusov so veriga, cikel, drevo in zvezda.

Definicija 3.1. Naj bo G graf in C množica prerezov grafa G. Par $(\mathcal{R}, \varphi)$, kjer je $\mathcal{R}$ grafa in $\varphi : V(G) \to V(\mathcal{R})$ preslikava, imenujemo *reprezentacija* za C, če zadošča naslednjima pogojema:

1. Za poljubni najmanjši prerez $\{S, V(\mathcal{R}) - S\} \in C(\mathcal{R})$ obstaja prerez $\{X, \overline{X}\}$ v G, definiran z $X = \{u \in V(G) |\ \varphi(u) \in S\}$ in $\overline{X} = \{u \in V(G) |\ \varphi(u) \in V(\mathcal{R}) - S\}$, ki pripada C.

2. Obratno, za vsak prerez $\{X, \overline{X}\} \in C$ obstaja tak najmanjši prerez $\{S, V(\mathcal{R}) - S\} \in C(\mathcal{R})$, da je $X = \{u \in V(G) |\ \varphi(u) \in S\}$ in $\overline{X} = \{u \in V(G) |\ \varphi(u) \in V(\mathcal{R}) - S\}$.

V reprezentaciji se lahko zgodi, da za neko vozlišče $x \in V(\mathcal{R})$ ne obstaja vozlišče $v \in V(G)$, da je $\varphi(v) = x$. Takšno vozlišče imenujemo *prazno vozlišče*. Po drugi strani pa lahko, kadar sta u in v povezani vozlišči in velja $\lambda(u, v; G) > \lambda(G)$, u in v preslikamo v isto vozlišče.

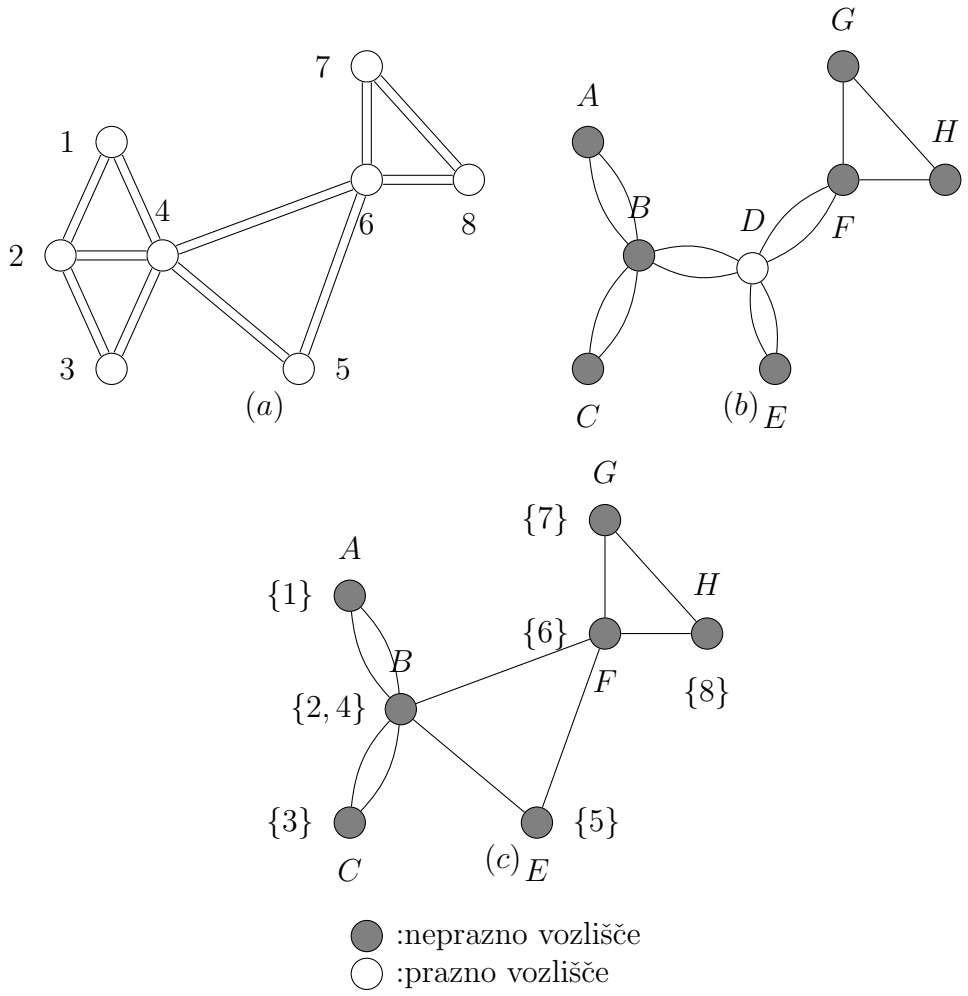
Reprezentacija $(\mathcal{R}, \varphi)$ je *kaktusna*, če je $\mathcal{R}$ kaktus. Kaktus vsebuje dve vrsti povezav: takšne, ki so vsebovane v kakšnem ciklu, in takšne, ki niso. Povezave, ki niso vsebovane v nobenem ciklu, imenujemo *drevesne povezave*. Vozlišče, ki se nahaja v k ciklih, imenujemo *k-stično* vozlišče.

Kaktusna reprezentacija je *normalna*, če nima drevesnih povezav in praznih 2-stičnih vozlišč, ki pripadajo 2-ciklom. Če za podmnožico $C \subseteq C(G)$ grafa G obstaja kaktusna reprezentacija, potem obstaja normalna kaktusna reprezentacija (glej [14, lema 5.5]).

Če normalna kaktusna reprezentacija ne vsebuje nobenega praznega 3-stičnega vozlišča, jo imenujemo *normalna kaktusna reprezentacija cikličnega tipa* (v nadaljevanju CNC). Takšno reprezentacijo lahko iz normalne kaktusne reprezentacije dobimo tako, da vsa prazna 3-stična vozlišča zamenjamo s po tremi praznimi vozlišči in jih povežemo v nov 3-cikel. V dobljenem grafu skrčimo 2-cikel C, ki vsebujejo prazno 2-stično vozlišče x v x', drugo vozlišče v C ter odstranimo s tem pridobljeno zanko.

Na sliki 3.1 sta prikazani različni kaktusni reprezentaciji (b) in (c) za vse najmanjše prereze grafa (a). Množice povejo, katera vozlišča iz grafa (a) se preslikajo v dano vozlišče pri kaktusu (c). Preverimo še, da je (b) reprezentacija za a. Označimo prereze v grafu (a) z $X_1 = \{1\}, X_2 = \{3\}, X_3 = \{1, 2, 3\}, X_4 = \{5\}, X_5 = \{6, 7, 8\}, X_6 = \{7, 8\}, X_7 = \{7\}, X_8 = \{8\}$. Le ti ustrezajo po vrsti prerezom $\{A\}, \{C\}, \{A, B, C\}, \{E\}, \{F, G, H\}, \{G, H\}, \{G\}$, $\{H\}$, v kaktusu (b) in kaktusu (c). Kaktusna reprezentacija (b) ni normnalna reprezentacija cikličnega tipa niti normalna, saj vsebuje drevesne povezave, (c) je normalna, ni pa cikličnega tipa, ker vsebuje 3-stično prazno vozlišče, in

(d) je CNC, ker ne vsebuje drevesnih povezav niti 3-stičnih praznih vozlišč. Na sliki 3.1 so z belo označena prazna vozlišča in s črno neprazna vozlišča. Ti oznaki bomo uporabili tudi v nadaljevanju diplomske naloge.



Slika 3.1: Kaktusne reprezentacije.

3.2 Particije in prerezi

Particije in prerezi imajo pomembno vlogo pri konstrukciji kaktusne reprezentacije grafa. Zato bomo v tem razdelku predstavili nekaj osnovnih tipov particij in z njimi povezanih kaktusnih reprezentacij.

Definicija 3.2. Naj bodo $V_i \subset V$ za $i = 1, ..., r$, kjer je $r \geq 2$. Potem imenujemo $(V_1, V_2, ..., V_r)$ *urejena particija*, če velja $\cup_{i=1}^{r} V_i = V$ in $V_i \cap V_j = \emptyset$ za $i \neq j$.

Definicija 3.3. Naj bo $(V_1, V_2, ..., V_r)$ urejena particija vozlišč in h, k takšni števili, da velja $1 \leq h \leq k \leq r$. Potem definiramo množico $V_{(h,k)} = V_h \cup V_{h+1} \cup ... \cup V_k$.

Za podmnožici $X, Y \subseteq V$ pravimo, da se *sekata*, če velja $X \cap Y \neq \emptyset$, $X - Y \neq \emptyset$ in $Y - X \neq \emptyset$, in da se *križata*, če se sekata in velja $V - (X \cup Y) \neq \emptyset$. Naslednjo lemo lahko dokažemo s štetjem povezav med ustreznimi množicami. Za natančen dokaz glej ([14, stran 145]).

Lema 3.4. *Naj bo $G = (V, E)$ utežen graf in naj bosta $\{X, \overline{X}\}, \{Y, \overline{Y}\} \in C(G)$ poljubna prereza, ki se križata. Označimo $V_1 = X \cap Y$, $V_2 = \overline{X} \cap Y$, $V_3 = \overline{X} \cap \overline{Y}$ in $V_4 = X \cap \overline{Y}$, potem veljata naslednji enakosti:*

1. $d(V_1, V_2) = d(V_2, V_3) = d(V_3, V_4) = d(V_4, V_1) = \lambda(G)/2$ *in*
2. $d(V_1, V_3) = d(V_2, V_4) = 0$.

Definicija 3.5. Za podmnožico $C' \subseteq C(G)$ urejeno particijo množice vozlišč $\pi_1 = (V_1, V_2, ..., V_r)$ imenujmo *urejena particija najmanjšega prereza* oziroma *MC particija*(angl. *minimum-cut o-partition*) nad C', če so $\{V_{(1,k)}, \overline{V_{(1,k)}}\}$, $1 \leq k \leq r - 1$, najmanjši prerezi v C' takšni, da velja:

$$C_1 = \{\{V_{(1,k)}, \overline{V_{(1,k)}}\} | 1 \leq k \leq r - 1\} \subseteq C'.$$

Naslednja lema nam bo podala preprosto strukturo za predstavitev množice prerezov iz $C(G)$, kjer vsi prerezi ločujejo določeno kritično povezavo. Ta rezultat pa nam bo pomagal pri konstrukciji (s, t)-kaktusne reprezentacije.

Lema 3.6. *([14, stran 146]) Za vsako kritično povezavo $e = \{s, t\}$ v uteženem grafu $G = (V, E)$ se nobena dva prereza v $C(G)$, v katerih sta s in t ločena, ne križata. Torej obstaja urejena particija $\pi_{s,t} = (V_1, V_2, ..., V_r)$ takšna, da $s \in V_1$ in $t \in V_r$ in množica $r - 1$ prerezoz*

$$\{V_{(1,i)}, V_{(i+1,r)}\}, \qquad 1 \leq i < r$$

je enaka množici vseh prerezov v $C(G)$, ki ločujejo s in t.

Dokaz. Najprej pokažimo, da se nobena dva najmanjša prereza $\{X, \overline{X}\}$ in $\{Y, \overline{Y}\}$ v $C(G)$, ki ločujeta s in t, ne križata. Brez škode za splošnost lahko predpostavimo, da je $s \in X \cap Y$. Če se ta prereza križata, je povezava $e = \{s, t\}$ vsebovana v $E(X \cap Y, \overline{X} \cap \overline{Y}; G)$, zato je $d(X \cap Y, \overline{X} \cap \overline{Y}; G) \geq c_G(e) > 0$. Ampak po lemi 3.4 je $d(X \cap Y, \overline{X} \cap \overline{Y}; G) = 0$, kar je protislovje. Zato se $\{X, \overline{X}\}$ in $\{Y, \overline{Y}\}$ v $C(G)$ ne križata.

Iz tega sledi, da vse najmanjše prereze $\{X_i, \overline{X_i}\}$ $(i = 1, ..., q)$ v $C(G)$, ki ločujejo s in t, in kjer brez škode za splošnost predpostavimo, da je $s \in X_i$, lahko uredimo na nasljedni način:

$$\{s\} \subseteq X_1 \subset X_2 \subset ... \subset X_q \subseteq V - \{t\}.$$

Tako je $(X_1, X_2 - X_1, ..., X_q - X_{q-1}, \overline{X_q})$ želena urejena particija, ki predstavlja vse najmanjše prereze $\{X_i, \overline{X_i}\}$ za $i = 1, ..., q$. □

Definicija 3.7. Urejeno particijo $\pi_{s,t}$ iz leme 3.6 imenujemo (s, t) *urejena particija najmanjšega prereza* ali krajše (s, t)*-MC particija*.

Lema 3.8. *[14, lema 4.10] Naj bo $\{s, t\}$ kritična povezava v uteženem grafu G. Potem lahko za poljuben največji pretok med s in t najdemo (s, t)-MC particijo v času in prostoru $O(m + n)$.*

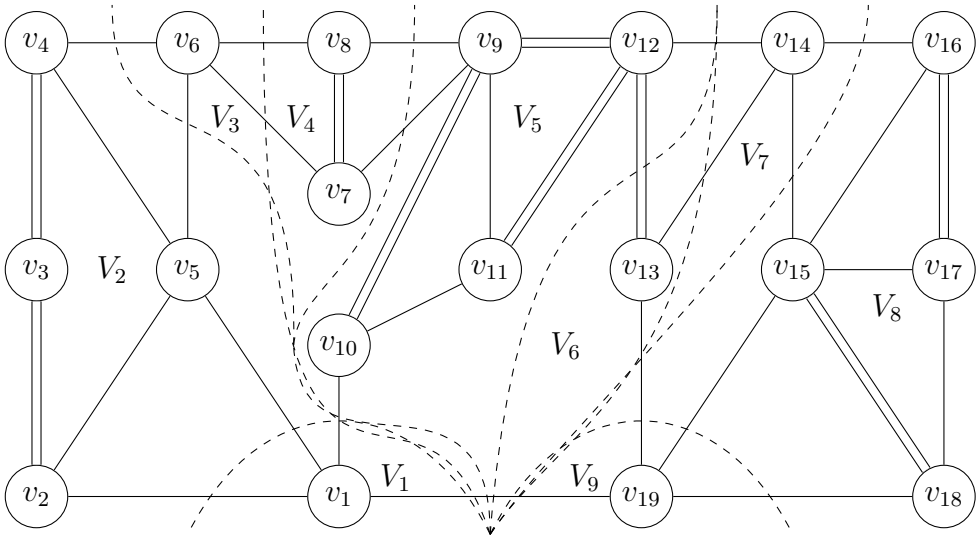
Dokaz leme 3.8 in algoritem za iskanje (s, t)-MC particije glej [10, 16].

Naj bo $\{s, t\}$ kritična povezava v grafu $G = (V, E)$. Naj bo X prerez in $\pi = (V_1, V_2, ..., V_r)$ particija v grafu $G = (V, E)$ potem je prerez X *kompatibilen* s particijo π, če obstaja $I \subset \{1, 2, ..., r\}$, da velja $X = \bigcup_{i \in I} V_i$. Naj

bo $\{s,t\}$ kritična povezava v grafu G in $\pi_{(s,t)}$-MC particija, ki ločuje s in t. Potem kaktusno reprezentacijo za množico vseh najmanjših prerezov, ki so kompatibilni s $\pi_{(s,t)}$ nad $C(G)$, imenujemo (s,t)-*kaktusna reprezentacija*.

Definicija 3.9. Za podmnožico $C' \subseteq C(G)$ urejeno particijo $\pi_2 = (V'_1, V'_2, ..., V'_{r'})$ imenujmo krožna MC particija oziroma CMC particija (angl. *circular minimum-cut o-partition*) nad C', če velja:

$$C_2 = \{\{V'_{(h,k)}, \overline{V'_{(h,k)}}\} | 1 \leq h \leq k \leq r' - 1\} \subseteq C'.$$

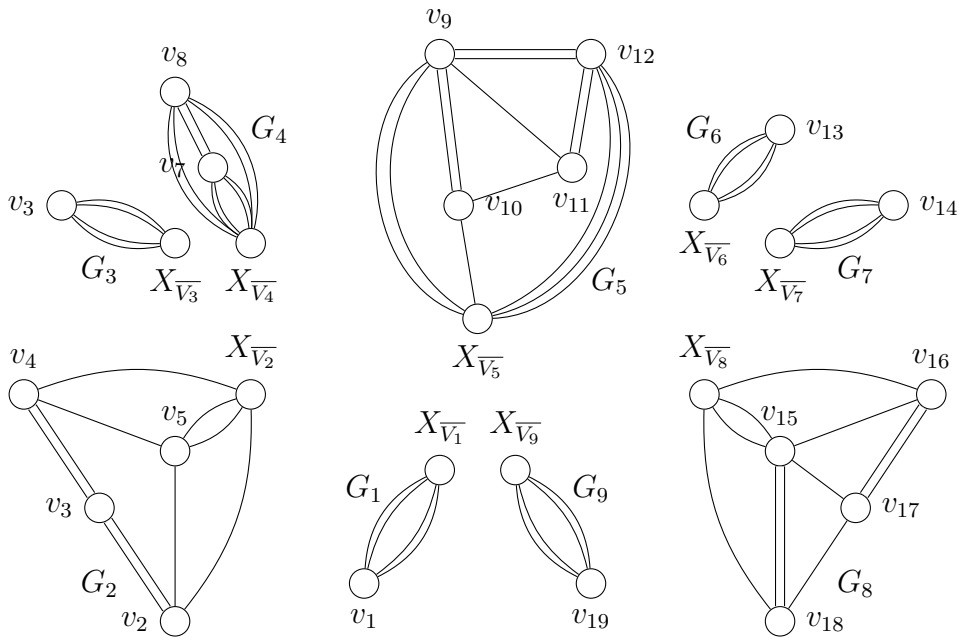


Slika 3.2: Prikaz (s,t)-MC particije.

Na sliki 3.2 je prikazana (v_1, v_{19})-MC particija grafa G, ki je 4-povezan po povezavah in je prikazan na sliki 2.1. Vsaka črtkana krivulja loči množici iz najmanjšega prereza, za kateri je povezava v_1, v_{19} v prerezni množici. Znotraj vsakega območja se nahaja oznaka V_i, s katero predstavimo množico vozlišč tega območja v particiji.

3.3 Osnovne operacije

V nadaljevanju bomo predstavili naivni algoritem za pridobitev CNC reprezentacije $\mathcal{R}$ grafa G. Kasneje bomo ta algoritem še dopolnili. V ta namen bomo definirali tri osnovne operacije.

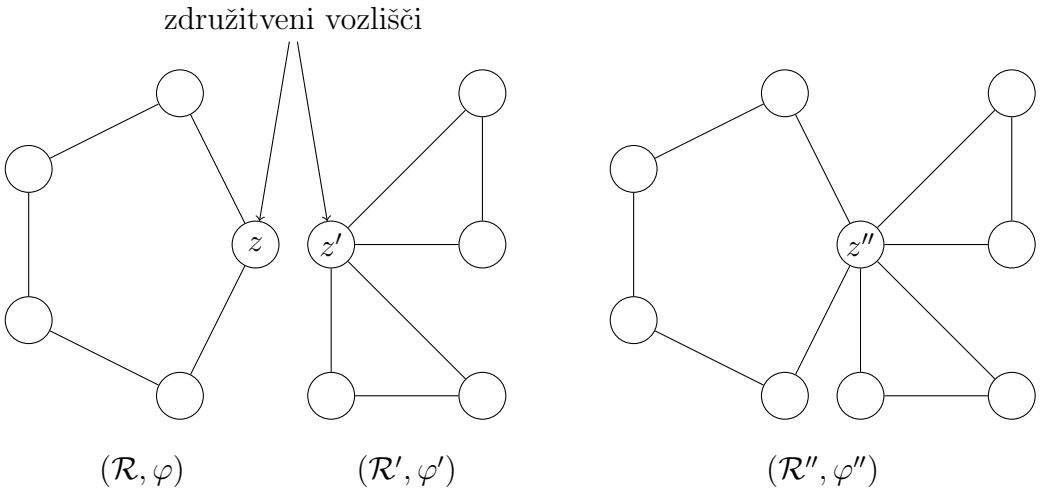


Slika 3.3: Prikaz (s,t)-dekompozicije.

(s,t)-dekompozicija Naj bo $\pi_{s,t} = (V_1, V_2, ..., V_r)$ $(s,t) - MC$ particija in $G_i = G/(V(G) - V_i)$. Zaporedje grafov $(G_1, G_2, ..., G_r)$, dobljeno iz grafa G na ta način, imenujemo (s,t)-dekompozicija. Slika 3.3 prikazuje (v_1, v_{19})-dekompozicijo za (s,t)-MC particijo, prikazano na sliki 3.2. Prikazani grafi $G_1, ..., G_9$, pri katerih je množica $\overline{V_i}$, ki smo jo skrčili, pa je označena z $X_{\overline{V_i}}$.

Združitvene funkcije $\oplus$ Potrebovali bomo še nekaj pojmov. Pravimo, da reprezentacija $(\mathcal{R}, \varphi)$ *povzroči* particijo $V = \{V_1, V_2, ...V_r\}$, če ima $\mathcal{R}$ natanko

r nepraznih vozlišč $x_1, x_2, ...x_r$ takšnih, da velja $\varphi^{-1}(x_i) = V_i$. Za particiji $\{V_1, V_2, ...V_r\}$ in $\{V_1', V_2', ...V_r'\}$ množice V, kjer je $r, r' \geq 2$, pravimo, da sta *komplementarni*, če obstajata podmnožici V_i, V_j' takšni, da velja $V_i \cup V_j' = V$.



Slika 3.4: Prikaz združitvene funkcije $\oplus$.

Naj bosta $\mathcal{R} = (W, F)$ in $\mathcal{R}' = (W', F')$ kaktusa, za katera velja $W \cap W' = \emptyset$, kjer je $(\mathcal{R}, \varphi)$ rprezentacija za $C \subseteq C(G)$ in $(\mathcal{R}', \varphi')$ za $C' \subseteq C(G)$. Za takšni reprezentaciji pravimo, da sta komplementarni, če sta particiji V, $\{V_1, V_2, ...V_r\}$ in $\{V_1', V_2', ...V_r'\}$, povzročeni z $(\mathcal{R}, \varphi)$ in $(\mathcal{R}', \varphi')$, komplementarni. Če sta $(\mathcal{R}, \varphi)$ in $(\mathcal{R}', \varphi')$ komplementarni, potem obstajata vozlišči $z \in W$ in $z' \in W'$ taki, da $\varphi^{-1}(z) \cup \varphi'^{-1}(z') = V$. Taki z in z' imenujemo *združitveni vozlišči*. Združitev dveh komplementarnih kaktusnih reprezentacij $(\mathcal{R}, \varphi)$ in $(\mathcal{R}', \varphi')$ zapišemo kot $(\mathcal{R}'', \varphi'') = (\mathcal{R}, \varphi) \oplus (\mathcal{R}', \varphi')$. Kaktusno reprezentacijo $(\mathcal{R}'', \varphi'')$, pridobljeno iz $\mathcal{R}$ in $\mathcal{R}'$, kjer vozlišči z in z' združimo v eno vozlišče z'', ki ima za sosede vse sosede z in z', in preslikavo $\varphi'' : V \to W \cup W' \cup \{z''\}$, definiramo kot:
$\mathcal{R}''(W'', F'') = (W \cup W' \cup \{z''\} - \{z, z'\}, F \cup F')$,
$\varphi''^{-1}(z'') = \varphi^{-1}(z) \cap \varphi'^{-1}(z')$,
$\varphi''^{-1}(x) = \varphi^{-1}(x)$ za $x \in W - z$,
$\varphi''^{-1}(x') = \varphi'^{-1}(x')$ za $x' \in W' - z'$.

Na sliki 3.4 je prikazana združitev dveh reprezentacij $(\mathcal{R}, \varphi)$ in $(\mathcal{R}', \varphi')$ v novo reprezentacijo $(\mathcal{R}'', \varphi'')$. Pokažimo, da z združitvijo res dobimo reprezentacijo.

Lema 3.10. *Naj bosta $(\mathcal{R}, \varphi)$ in $(\mathcal{R}', \varphi')$ reprezentaciji za $C \subseteq C(G)$ in $C' \subseteq C(G)$ komplementarni. Potem je $(\mathcal{R}'', \varphi'') = (\mathcal{R}, \varphi) \oplus (\mathcal{R}', \varphi')$ reprezentacija za $C \cup C'$.*

Dokaz. Naj bo $\{X, \overline{X}\} \in C \cup C'$. Recimo, da je $X \in C$ ($X \in C'$ lahko obravnavamo analogno) in naj bo $\{S, W - S\}$ ustrezen prerez v $C(\mathcal{R})$. Brez škode za splošnost lahko predpostavimo $X = \varphi^{-1}(S)$ in $Z \in W - S$. Po lastnostih funkcije φ''^{-1} vemo $\{S, W'' - S\} \in C(\mathcal{R}'')$, $\varphi''^{-1}(S) = X$ in $\varphi''^{-1}(W'' - S) = \overline{X}$, kar pomeni, da je X reprezentiran v $\mathcal{R}''$.

Poglejmo si še dokaz v drugo smer. Naj bo $\{T, W'' - T\}$ poljuben prerez v $C(\mathcal{R}'')$. Brez škode za splošnost lahko predpostasvimo $z'' \in W'' - T$. Potem velja, da je T podmnožica $W - z$ ali $W' - z'$. V nasprotnem primeru vozlišče z'', ki ločuje $T \cap W$ in $T \cap W'$, ni vsebovano v T, kar pomeni, da je $\{T \cap W, W - T\}$ ali $\{T \cap W', W' - T\}$ manjši prerez od najmanjšega prereza v $\mathcal{R}''$, kar je protislovje. Zato $\{T, W'' - T\}$ ustreza prerezu v $C \cup C'$. □

(s,t)-kaktusna reprezentacija Naj bo povezava $e = \{s, t\}$ kritična povezava v G. V nadaljevanju bomo s podmnožico $C \subset C(G)$, v kateri so vsi najmanjši prerezi, ki ločujejo vozlišči s in t, pokazali, da obstaja kaktusna reprezentacija za C. Ta reprezentacija ima ključno vlogo v algoritmu za konstrukcijo kaktusne reprezentacije za množico vseh najmanjših prerezov $C(G)$.

Naj bo $\pi_1 = (V_1, V_2, ..., V_r)$ MC particija in

$$C_1 = \{\{V_{(1,k)}, \overline{V_{(1,k)}}\} \mid 1 \leq k \leq r - 1\}.$$

Množica prerezov C_1 ima verižno reprezentacijo $(\ddot{\mathcal{R}}_{\pi_1}, \psi_{\pi_1})$, ki je definirana kot:

$$\ddot{\mathcal{R}}_{\pi_1} = (\{x_i \mid 1 \leq i \leq r\}, \{e_1, e_1', e_2, e_2', ..., e_{r-1}, e_{r-1}'\}),$$

$$\psi_{\pi_1}{}^{-1}(x_i) = V_i \ (1 \leq i \leq r),$$

kjer je $e_i = \{x_i, x_{i+1}\}$ in $e'_i = \{x_i, x_{i+1}\}$ za $1 \leq i \leq r-1$. Naj bo $\pi_2 = (V'_1, V'_2, ..., V'_{r'})$ CMC particija in

$$C_2 = \{\{V'_{(h,k)}, \overline{V'_{(h,k)}}\} \mid 1 \leq h \leq k \leq r'-1\}.$$

Množica prerezov C_2 ima ciklično reprezentacijo $(\mathring{\mathcal{R}}_{\pi_2}, \phi_{\pi_2})$, ki je definirana kot:

$$\mathring{\mathcal{R}}_{\pi_2} = (\{x_i \mid 1 \leq i \leq r'\}\{e_1, e_2, ..., e_{r'}\}),$$

$$\phi_{\pi_2}{}^{-1}(x_i) = V'_i \ (1 \leq i \leq r'),$$

kjer je $e_i = \{x_i, x_{i+1}\}$ za $1 \leq i \leq r'-1$ in $e_{r'} = \{x_{r'}, x_1\}$.

Naj bo $\pi_{(s,t)} = (V_1, V_2, .., V_r)$, (s,t)-MC particija nad $C(G)$ za kritično povezavo $e = \{s,t\}$ v grafu G. Najprej razdelimo $\{V_1, V_2, .., V_r\}$ v dve podmnožici:

$$A = \{V_i \mid d(V_i, \overline{V_i}; G) = \lambda(G), \ 1 < i < r\} \qquad \text{in}$$

$$B = \{V_1, V_r\} \cup \{V_i \mid d(V_i, \overline{V_i}; G) > \lambda(G), \ 1 < i < r\}.$$

Če odstranimo vse elemente A iz $\pi = (V_1, V_2, .., V_r)$, potem nam ostane zaporedje *segmentov*, to je podmnožic A, od katerih vsaka vsebuje eno ali več množic V_i z zaporednimi indeksi. Recimo da je $q+1$ takšnih segmentov $B_0, B_1, ..., B_q$, kjer je $V_1 \in B_0$ in $V_r \in B_q$. Potem je $\{B_j | j = 0, 1, ..., q\}$ particija B. Podobno razdelimo A na p segmentov $A_0, A_1, ..., A_p$, kjer je A_k največji segment, podan kot: $V_l \in A$ in $d(V_{(a_k,l)}, \overline{V_{(a_k,l)}}; G) = \lambda(G)$ za $l = a_k, a_k+1, ..., b_k$.

Natančneje indekse $a_1, b_1, a_2, b_2, ..., a_p, b_p$ dobimo s sledečim postopkom, ki je zapisan v algoritmu 2.

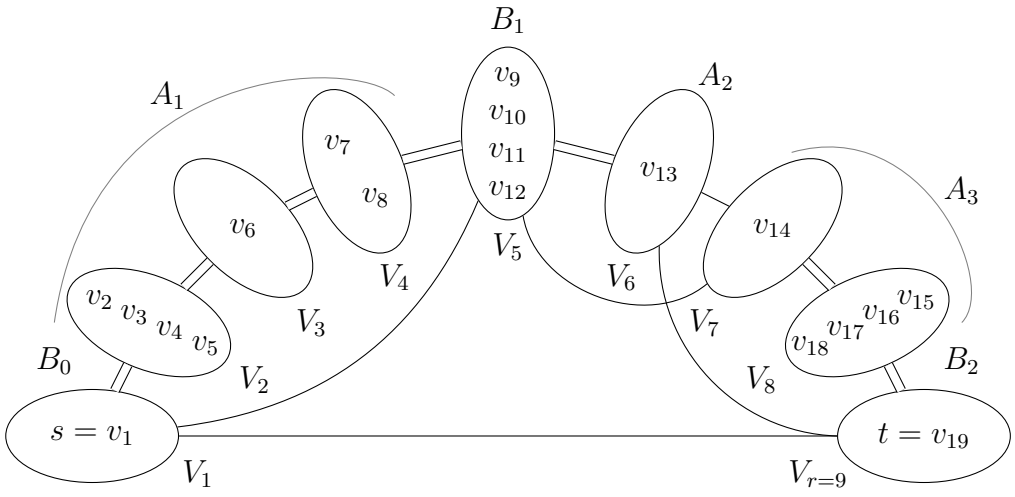
Algoritem 1 Razdelitev v največje segmente

Input: MC particija $(V_1, V_2, .., V_r)$ vozlišč grafa G in $\lambda(G)$.

Output: Segmenti $A_k, k = 1, 2, ...$

1: **procedure** ISKANJE SEGMENTOV
2: $a := l := 2$; $k := \Delta := 0$;
3: **while** $a < r$ **do**
4: **while** $d(V_{(a,l)}, \overline{V_{(a,l)}}; G) = \lambda(G)$ **do** $\Delta := 1$; $i := i + 1$;
5: **if** $\Delta = 1$ **then** $\Delta := 0$; $k := k + 1$; $A_k := V_{a,i-1}$; $l := l - 1$;
6: $a := l + 1$; $l := l + 1$;

Za vsako MC particijo π_{B_j}, kjer $j \in \{0, 1, ..., q\}$, označimo $(\ddot{\mathcal{R}}_{\pi_{B_j}}, \psi_{\pi_{B_j}})$ verižno reprezentacijo za najmanjše prereze, pripadajoče π_{B_j}, in za vsako CMC particijo π_{A_i}, kjer $i = 0, 1, ..., p$, naj bo $(\mathring{\mathcal{R}}_{\pi_{A_i}}, \phi_{\pi_{A_i}})$ ciklična reprezentacija za najmanjše prereze, pripadajoče π_{A_i}.



Slika 3.5: Prikaz segmentov.

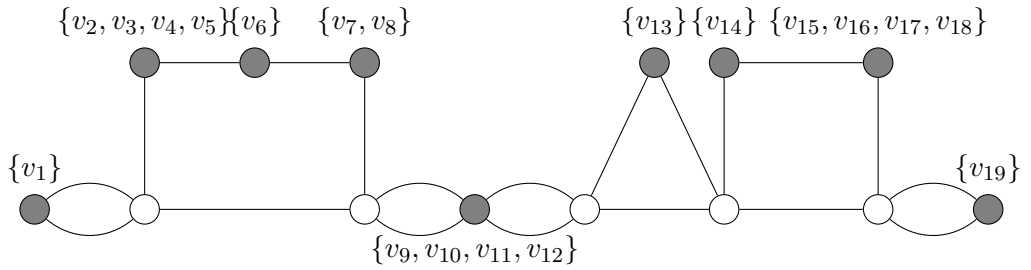
Na sliki 3.5 so segmenti grafa G iz slike 2.1, kot jih vrne algoritem 3.3. Iz segmentov bomo sestavili (s, t)-kaktusno reprezentacijo:

Algoritem 2 Združitev segmentov v (s,t)-katusno reprezentacijo

Input: Verižna reprezentacija $(\ddot{\mathcal{R}}_{\pi_{B_j}}, \psi_{\pi_{B_j}})$ za vsako MC particijo π_{B_j}, kjer je $j = 0, 1, ..., q$, in ciklična reprezentacija $(\mathring{\mathcal{R}}_{\pi_{A_i}}, \phi_{\pi_{A_i}})$ za vsako CMC particijo π_{A_i}, kjer je $i = 0, 1, ..., p$.

Output: (s,t)-katusna reprezentacija.

1: **procedure** (s,t)-KATUSNA REPREZENTACIJA
2: $(\mathcal{R}, \varphi) := (\ddot{\mathcal{R}}_{\pi_{B_0}}, \psi_{\pi_{B_0}})$;
3: **for** $k = 1, 2, ..., p$ **do**
4: $(\mathcal{R}, \varphi) := (\mathcal{R}, \varphi) \oplus (\mathring{\mathcal{R}}_{\pi_{A_k}}, \phi_{\pi_{A_k}})$;
5: **if** $b_k + 1 \neq a_{k+1}$ **then**$(\mathcal{R}, \varphi) := (\mathcal{R}, \varphi) \oplus (\ddot{\mathcal{R}}_{\pi_{B_j}}, \psi_{\pi_{B_j}})$;
6: kjer je B_j določen z $V_{b_k+1} \in B_j$;
7: $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)}) := (\mathcal{R}, \varphi)$;



Slika 3.6: Prikaz (s,t)-kaktusne reprezentacije.

Na sliki 3.6 je prikazana (v_1, v_{19})-kaktusna reprezentacija grafa G. Na njej je $r = 9$ nepraznih in $p + q = 5$ praznih vozlišč, ki nastanejo ob vsakem združevanju cikilične ali verižne kaktusne reprezentacije v (s,t)-kaktusno reprezentacijo. Ker vsaka ciklična ali verižna reprezentacija vsebuje vsaj eno neprazno vozlišče, velja $r \geq p + q$. To pomeni, da je v kaktusni reprezentaciji nepraznih vozlišč vedno več kot praznih, kar pomeni, da ima kaktus največ dvakrat toliko vozlišč kot graf. To opažanje bomo potrebovali pri časovni zahtevnosti povečanja povezavne povezanosti za 1.

3.4 Algoritem za kaktusno reprezentacijo

Sedaj lahko sestavimo algoritem za konstrukcijo celotne kaktusne reprezentacije za $C(G)$. Najprej bomo sestavili kaktusno reprezentacijo na naiven način s pomočjo operacij, predstavljenih v prejšnjem razdelku. Nato bomo to proceduro izboljšali v algoritem in zanj podali časovno zahtevnost.

KaktusNaivno(G)

1. Izberemo povezavo s, t v G, če le ta obstaja, če ne, vrnemo trivialni kaktus $(\mathcal{R}, \varphi)$.
2. Če $\lambda(s, t; G) > \lambda(G)$, potem združimo s in t v eno vozliščem. S tem dobimo graf G' in nadaljujemo s KaktusNaivno(G').
3. Sicer ($\lambda(s, t; G) = \lambda(G)$) in poiščemo (s, t)-MC particijo $\pi_{(s,t)} = (V_1, V_2, ..., V_r)$ ter pripadajočo (s, t)-kaktusno reprezentacijo $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$.
4. Poiščemo (s, t)-dekompozicijo $(G_1, G_2, ..., G_r)$ grafa G.
5. Izvedemo KaktusNaivno(G_i) na vsakem grafu G_i, da dobimo CNC reprezentacijo $(\mathcal{R}_{G_i}, \varphi_{G_i})$ za $C(G_i)$.
6. Združimo vse $\{(\mathcal{R}_{G_i}, \varphi_{G_i}), i = 1, ..., r\}$ in $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ v kaktusno reprezentacijo $(\mathcal{R}, \varphi)$ za $C(G)$ z uporabo združitvene funkcije $\oplus$.
7. Poenostavimo $(\mathcal{R}, \varphi)$ v CNC reprezentacijo in le to vrnemo.

Vendar ima ta algoritem nekaj težav. Če se KaktusNaivnoG' za graf G' sproži med izvajanjem KaktusNaivnoG'', potem imenujemo G' *otroka* oziroma *naslednika* G'' in G'' *starša* oziroma *prednika* G'. Če je $d(V_i, V(G) - V_i; G) = \lambda(G)$ za nek i, potem prerz $\{V_i, \overline{V_i}\}$ ostane najmanjši prerez v nasledniku G_i, čeprav smo ga odkrili že v staršu. To lahko privede do neskončno mnogo rekurzivnih klicev. Da preprečimo to možnost, hranimo množico starih (že najdenih) najmanjših prerezov. Najmanjši prerez označimo za novega pri G', če ga še nismo našli pri predniku G', sicer pa za starega. Opazimo,

da je najmanjši prerez star v G', če in samo če je v obliki $\{v, V(G') - v\}$ in je v skrčeno vozlišče $X_{\overline{V_i}}$ *(glej (s,t)-dekompozicija)* v predniku G'. Zato lahko hranimo množico starih najmanjših prerezov v grafu G' samo z označevanjem takšnih vozlišč kot stara. S tem lahko sedaj konstruiramo boljši algoritem.

Izrek 3.11. *([14]) Kaktusno reprezentacijo za vse najmanjše prereze v grafu G lahko konstruiramo v $O(mn + n^2 \log n)$ času in v $O(n + m)$ prostoru.*

Za podrobno izpeljavo časovne in prostorske zahtevnosti glej ([14, poglavje 5.3.4]).

Algoritem 3 Konstrukcija kaktusa

Input: Graf G, podmnožica $V^{stara} \subseteq V(G)$, celo število $\lambda(G) > 0$ in $\lambda = \lambda(G)$
Output: Kaktusna reprezentacija $(\mathcal{R}, \varphi)$ za množico najmanjših prerezov C', za katero je $C(G) - \{\{\overline{v}, V(G) - \{\overline{v}\}\} \mid \overline{v} \in V^{stara}\} \subseteq C' \subseteq C(G)$.

1: **procedure** KAKTUS(G, V^{stara}, λ)
2: **if** $|V(G)| = 1$ **then return** trivialni kaktus $(\mathcal{R}, \varphi)$;
3: **else**
4: Izberi povezavo $e = \{s,t\} \in E(G)$;
5: **if** $\lambda(s,t;G) > \lambda$ ali (s,t)-kaktusna reprezentacija $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ ne predstavlja najmanjšega prereza, drugačnega kot $\{\overline{v}, V(G) - \{\overline{v}\}\}$, $\overline{v} \in V^{stara}$ **then**
6: $G := G/\{s,t\}$;
7: $V^{stara} := V^{stara} - \{s,t\}$;
8: $(\mathcal{R}, \varphi)$:=Kaktus(G, V^{stara}, λ);
9: **return** $(\mathcal{R}, \varphi)$;
10: **else**
11: **for** vsak V_i v (s,t)-MC particiji $\pi_{(s,t)} = (V_1, V_2, ..., V_r)$ **do**
12: $G := G/(V(G) - V_i)$ z $x_{\overline{V_i}}$ označimo vozlišče, dobljeno s skrčitvijo $V(G) - V_i$;
13: **if** $d(V_i; G) = \lambda$ **then** $V_i^{stara} := (V^{stara} \cap V_i) \cup \{x_{\overline{V_i}}\}$;
14: $(\mathcal{R}_{G_i}, \varphi_{G_i})$:=Kaktus$(G_i, V_i^{stara}, \lambda)$;

15: $(\mathcal{R}, \varphi) := (\mathcal{R}_{(s,t)}, \varphi_{(s,t)}) \oplus (\mathcal{R}_{G_1}, \varphi_{G_1}) \oplus ... \oplus (\mathcal{R}_{G_r}, \varphi_{G_r})$;

16: Pretvori $(\mathcal{R}, \varphi)$ v CNC reprezentacijo in jo označi kot $(\mathcal{R}, \varphi)$;

17: **return** $(\mathcal{R}, \varphi)$;

Poglavje 4

Povečanje povezavne povezanosti za ena

Predstavili bomo, kako si lahko s CNC kaktusno reprezentacijo grafa pomagamo pri povečanju povezavne povezanosti grafa. V ta namen bomo pokazali nekaj povezav med povezanostjo grafa in njegovo CNC kaktusno reprezentacijo, podali in dokazali spodnjo mejo potrebnih dodatnih povezav za povečanje povezavne povezanosti ter izpeljali algoritem za povečanje povezavne povezanosti za ena. Podali bomo tudi primer izvajanja algoritma in njegovo časovno zahtevnost. Navezovali se bomo na vir [14].

4.1 Izpeljava algoritma

Naj bo $G = (V, E)$ multigraf, ki mu želimo povečati stopnjo povezavne povezanosti za ena. Poiskati moramo najmanjšo množico F, da velja $\lambda(G + F) = \lambda(G) + 1$. Takšna množica F bo uničila vse najmanjše prereze v G. Le te lahko predstavimo z normalno kaktusno reprezentacijo $(\mathcal{R}(\mathcal{V}, \mathcal{E}), \varphi)$ grafa G.

Naravna ideja je poiskati želeno množico F, tako da uniči vse najmanjše prereze v kaktusni reprezentacji. Za takšno množico povezav F nad vozlišči V naj bo $\varphi(F) = \{\{\varphi(u, v)\}|\{u, v\} \in F\}$. Za netrivialen graf kaktusna reprezentacija ne bo trivilen kaktus. Zato se lahko omejimo na netrivialne

kaktusne reprezentacije. V tem primeru za normalno kaktusno reprezentacijo $(\mathcal{R}, \varphi)$ velja, da je $\lambda(\mathcal{R}) = 2$, saj je vsaka povezava vsebovana v natanko enem ciklu. Ta lastnost nam pomaga pri naslednji trditvi.

Trditev 4.1. *([14, Lema 8.1]) Za množico novo dodanih povezav F grafu G velja $\lambda(G + F) \geq \lambda(G) + 1$ natanko tedaj, ko velja $\lambda(\mathcal{R} + \varphi(F)) \geq 3$.*

S tem lahko prevedemo problem iskanja najmanjše množice F na problem iskanja najmanjše množice $\mathcal{F} = \varphi(F)$, ki jo dodamo kaktusu $\mathcal{R}$ tako, da $\mathcal{F}$ uniči vse 2-prereze v $\mathcal{R}$. Pri tem mora povezava iz $\mathcal{F}$ povezovati zgolj neprazna vozlišča, sicer je ni možno dodati v F.

Naj spomnimo, da vozlišče stopnje 2 v kaktusu imenujemo list. Vsak list $x \in V(\mathcal{R})$ ustreza najmanjšemu minimalnemu prerezu $\{\varphi^{-1}(x), V - \varphi^{-1}(x)\} \in C(G)$. Sedaj z $\mathcal{M}(G)$ označimo množico vseh minimalnih najmanjših prerezov v grafu G. Iz tega opažanja in prejšnje trditve lahko izpeljemo trditev o minimalnem številu povezav, potrebnih za povečanje povezavne povezanosti grafa G in kaktusa $\mathcal{R}$.

Trditev 4.2.
1. *Če za kaktus $\mathcal{R}$ in množico novo dodanih povezav $\mathcal{F}$ velja $\lambda(\mathcal{R} + \mathcal{F}) \geq 3$, potem je $|\mathcal{F}| \geq \lceil |L(\mathcal{R})/2| \rceil$, pri čemer je $L(\mathcal{R})$ množica vseh listov.*
2. *Če za multigraf $G(V, E)$ in množico novo dodanih povezav F velja $\lambda(G + F) \geq \lambda(G) + 1$, potem je $F \geq \lceil |\mathcal{M}(G)/2| \rceil$.*

Dokaz. Dokažimo drugo trditev (za prvo trditev velja analogni dokaz). Naj bo F poljubna množica povezav, za katero velja $\lambda(G + F) \geq \lambda(G) + 1$. Za vsak prerez $X \in \mathcal{M}(G)$ mora obstajati taka povezava v F, da eno vozlišče pripada X, sicer bi veljalo $\lambda(G + F) \leq d(X; G + F) = d(G) = \lambda(G)$. Ker je presek minimalnih najmanjših prerezov prazen in so vsi v $\mathcal{M}(G)$, je število vozlišč na povezavah v F najmanj $|\mathcal{M}(G)|$. Od tod jasno sledi želena neenakost $F \geq \lceil |\mathcal{M}(G)/2| \rceil$. □

Po konstrukciji CNC reprezentacije v kaktusu ne obstaja prazno vozlišče, ki bi bilo list. Do sedaj smo pokazali, da je možno povečati povezavno poveza-

nost $\mathcal{R}$ (oziroma G) za 1 z dodatnimi $\lceil |L(\mathcal{R})/2| \rceil$ (oziroma $\lceil |\mathcal{M}(G)/2| \rceil$) povezavami. Če ima kaktus $\mathcal{R}$ sodo število listov, potem naj bo σ množica novih povezav, ki povezujejo vse liste in $|\sigma| = |L(\mathcal{R})/2|$. Sedaj moramo pokazati, da takšno združevanje listov iz množice σ realizira povezanost $\lambda(\mathcal{R}+\sigma) \geq 3$. Če ima $\mathcal{R}$ liho število listov, izberemo poljubno vozlišče $z \in V(\mathcal{R})$ in naredimo nov cikel $C = (z, z')$, pri čemer dodamo nov list z'. Zadostuje pokazati, da za tako dobljeni kaktus $\mathcal{R}'$ obstaja takšno združevanje listov σ, da je $\lambda(\mathcal{R}' + \sigma) \geq 3$. Zato je zadosti, da se omejimo na primer, ko ima $\mathcal{R}$ sodo število listov.

Trditev 4.3. *Naj bo $\mathcal{R}$ netrivialni kaktus s sodim številom listov. Potem veljata naslednji trditvi.*

1. *Vozlišča v $L(\mathcal{R})$ imajo ciklično ureditev $(z_1, z_2, ..., z_\ell)$ takšno, da za vsak 2-prerez $X \subset V(\mathcal{R})$ v $\mathcal{R}$ velja, da imajo vsa vozlišča $z_i \in X$ zaporedne indekse. Še več, takšno ciklično ureditev lahko najdemo v času $O(|E(\mathcal{R})|)$.*

2. *Za ciklično ureditev $(z_1, z_2, ..., z_\ell)$ iz prejšnje točke velja, da če $\mathcal{R}$ dodamo množico $\ell/2$ novih povezav $\mathcal{F} = \{\{z_i, z_{i+\ell/2}\} | i = 1, .., \ell/2\}$, se poveča povezavna povezanost na 3.*

Dokaz. 1. Vzemimo Eulerjev obhod grafa $\mathcal{R}$, ki ga lahko najdemo v linearnem času. Povezave v grafu $\mathcal{R}$ usmerimo tako, da vsak neusmerjen cikel postane usmerjen, in obiščimo vse cikle v tem vrstem redu, kot ga podaja metoda za iskanje v globino. Potem naj bo $(z_1, z_2, ..., z_\ell)$ ciklično zaporedje vseh vozlišč v $L(\mathcal{R})$ takšno, da se v Eulerjevem obhodu grafa $\mathcal{R}$ vozlišča $z_1, z_2, ..., z_\ell$ pojavijo v tem vrstnem redu. Za vsak 2-prerez $\{U, V(\mathcal{R}) - U\}$ v $\mathcal{R}$ po konstrukciji Eulerjevega obhoda velja, da imajo vsa vozlišča $z \in L(\mathcal{R}) \cap U$ zaporedne indekse.
2. Naj bo X poljuben 2-prerez v $\mathcal{R}$. Zadostuje pokazati, da množica $\mathcal{F} \cap E(X; \mathcal{R} + \mathcal{F})$ ni prazna. Brez škode za splošnost lahko privzamemo, da velja $\{z_1, z_2, ..., z_\ell\} \cap X = \{z_1, z_2, ..., z_p\}$, kjer je $p < \ell$. Oglejmo si povezavo $e =$

$\{z_{\lfloor p/2 \rfloor}, z_{\lfloor p/2 \rfloor + \ell/2}\} \in \mathcal{F}$: ker je $p < \lfloor p/2 \rfloor + \ell/2 < \ell$, velja $e \in \mathcal{F} \cap E(X; \mathcal{R} + \mathcal{F})$, kar je bilo treba dokazati. □

Sedaj lahko zapišemo algoritem za povečanje povezavne povezanosti za 1. V njem bomo uporabili Hierholzerjev algoritem za iskanje Eulerjevga obhoda iz drugega poglavja in algoritem za konstrukcijo CNC kaktusne reprezentacije iz tretjega poglavja.

Algoritem 4 Povečanje povezavne povezanosti za 1

Input: Graf $G = (V, E)$, stopnja povezavne povezanosti $\lambda(G)$.

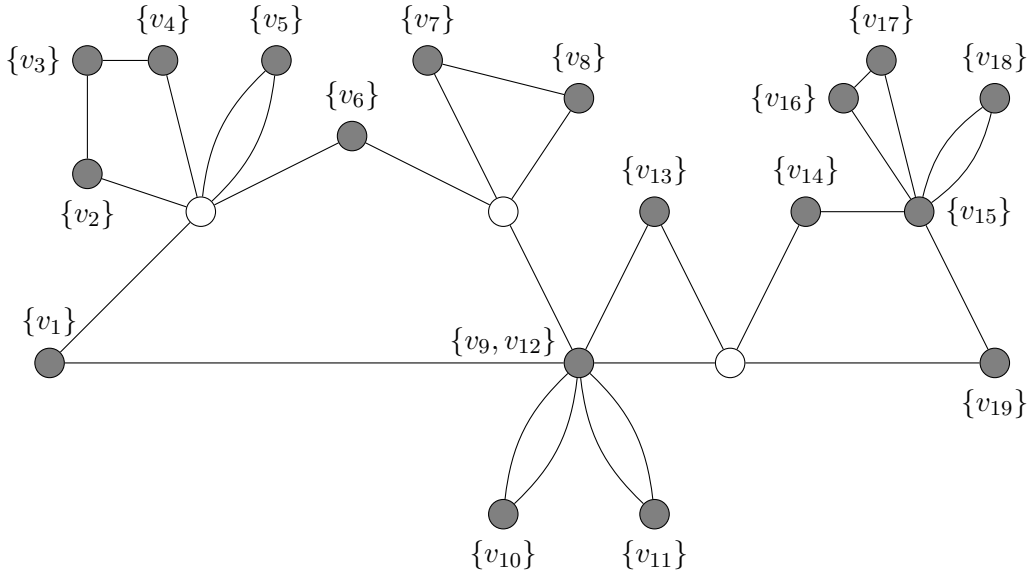
Output: Minimalna množica povezav F, da velja $\lambda(G) + 1 \leq \lambda(G + F)$.

```
1: procedure Povečanje za 1(G = (V, E), λ(G)
2:     Poiščemo kaktusno reprezentacijo
           CNC(R, φ) = KAKTUS(G, ∅, λ(G));
3:     E'=Hierholzerjev algoritem (R);
4:     i := 1, ℓ := 1;
5:     for e = uv v E' do
6:         if d(u) == 2 then
7:             u dodamo oznako z_ℓ; ℓ = ℓ + 1;
8:     F := ∅;
9:     while i ≤ ℓ/2 do
10:        izberemo x ∈ φ^{-1}(z_i) in y ∈ φ^{-1}(z_{i+ℓ/2});
11:        dodamo povezavo (x, y) v F;
12:        i = i + 1;
13:    return F;
```

Po trditvi 4.3 je graf $\mathcal{R} + \varphi(F)$ 3-povezan, zato po trditvi 4.1 velja $\lambda(G + F) \geq \lambda(G) + 1$ in po trditvi 4.2 velja, da je $|F|$ najmanjša, zato algoritem 4.1 vrne pravilen rezultat, njegovo časovno zahtevnost bomo obravnavali v razdelku 4.3.

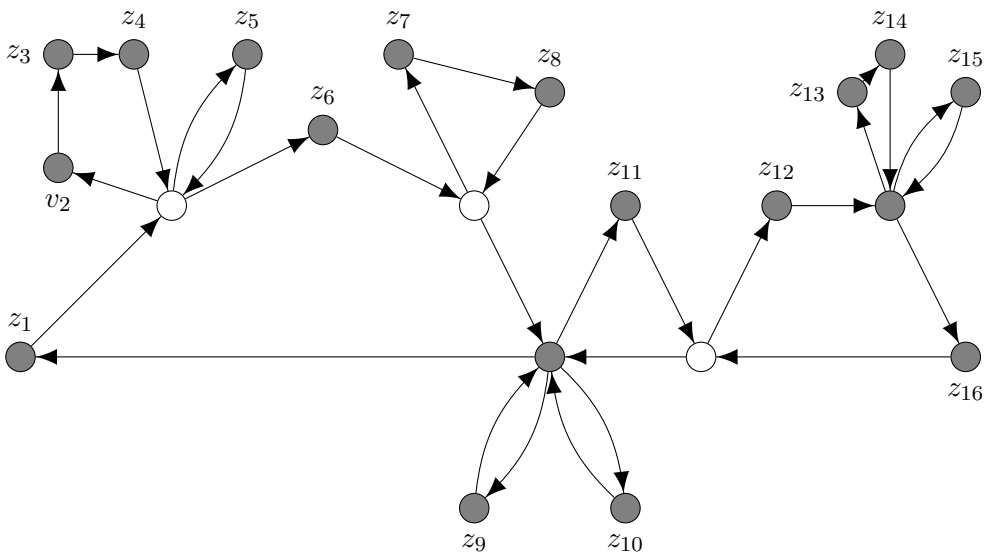
4.2 Primer izvajanja

V tem razdelku bomo predstavili vmesne korake prej opisanega algoritma 4.1. Za ena bomo povečali povezavno povezanost grafa G, prikazanega na sliki 2.1.

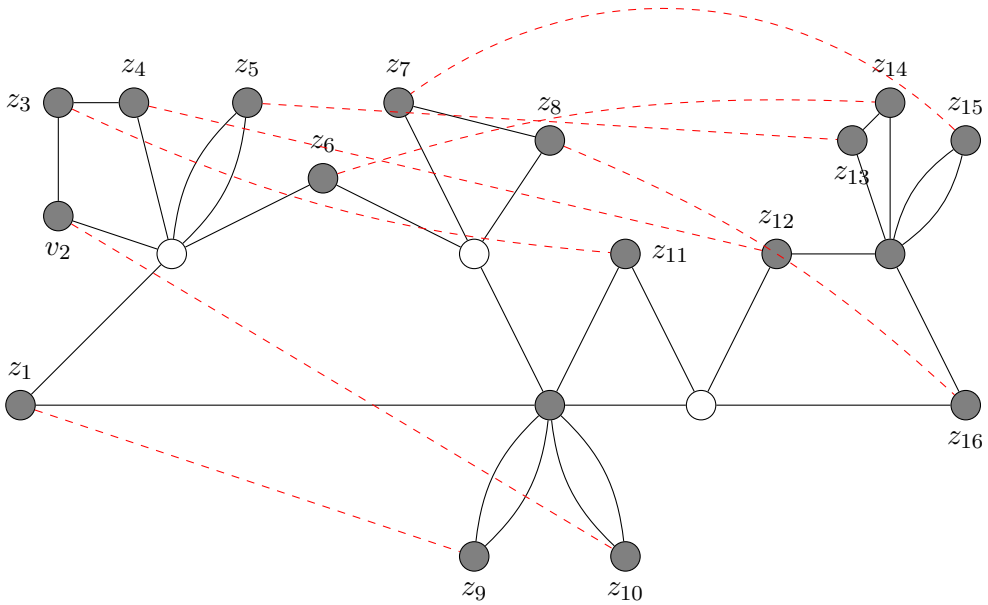


Slika 4.1: Kaktusna CNC reprezentacija za G 2.1.

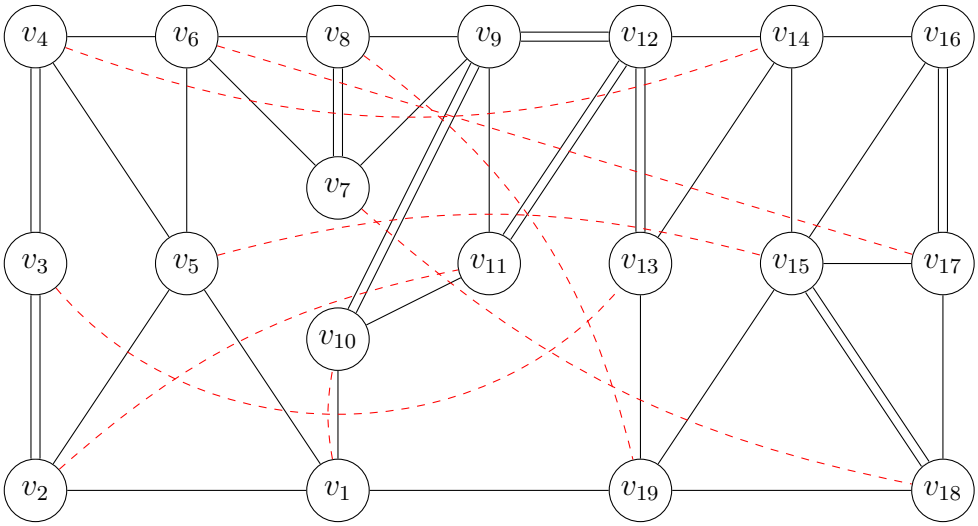
Na sliki 4.1 je prikazana CNC reperezentacija vseh najmanjših prerezov grafa G. Tej reprezentaciji smo dodali usmeritev in označili liste po postopku, opisanem v prejšnjem razdelku. Rezultat slednjega je na sliki 4.2. Označenim listom smo na sliki 4.3 dodali rdeče črtkane povezave po postopku 2. To množico povezav smo s preslikavo φ^{-1} preslikali na začetni graf G. Končni rezultat tega postopka je na sliki 4.4.



Slika 4.2: Usmerjanje in označitev listov.



Slika 4.3: Povečanje povezanosti na kaktusu.



Slika 4.4: Povečanje povezanosti na grafu.

Na prejšnjih štirih slikah je prikazan primer povečanja povezavne povezanosti na grafu G, ki je 4-povezan. Najmanjše prereze v G smo predstavili s kaktusno reprezentacijo $(\mathcal{R}, \varphi)$, ki vsebuje množico listov $L(\mathcal{R}) = \{z_1, z_2, ..., z_{16}\}$. Po trditvi 4.3 velja, da množica povezav $\mathcal{F} = \{(z_1, z_9), (z_2, z_{10}), ..., (z_8, z_{16})\}$ zadošča pogoju $\lambda(\mathcal{R} + \mathcal{F}) \geq 3$. Za množico povezav $\mathcal{F}$ najdemo takšno množico novih povezav F, da velja $\varphi(F) = \mathcal{F}$. Na zgledu je $F = \{(v_1, v_{10}), (v_2, v_{11}), (v_3, v_{13}), (v_4, v_{14}), (v_5, v_{15}), (v_6, v_{17}), (v_7, v_{18}), (v_8, v_{19})\}$. Za F velja $|F| = |\mathcal{F}| = \lceil |L(\mathcal{R})|/2 \rceil = \lceil |\mathcal{M}(G)|\ /2 \rceil = 8$. Po trditv 4.1 velja $\lambda(G + F) \geq \lambda(G) + 1$ in po trditvi 4.2 velja, da je $|F|$ najmanjša, zato je F optimalno povečanje za povezanost grafa G.

4.3 Časovna zahtevnost

V tem razdelku bomo podali časovno zahtevnost za izvedbo predhodno opisanega postopka za povečanje povezavne povezanosti na multigrafu $G = (V, E)$. Pri tem se bomo sklicevali na rezultate v [14]. Naj bo $n = |V(G)|$, $m = |E(G)|$ in $n^* = |V(\mathcal{R})|$. Pri (s, t)-kaktnusni reprezentaciji smo opazili, da velja $n^* < 2n$ (glej konec razdelka 3.3). Da lahko na multigrafu G

začnemo izvjati postopek, potrebujemo kaktusno reprezentacijo R, ki ima časovno zahtevnost $O(nm + n^2 \log n)$ (glej izrek 3.11).

Potrebujemo še ciklično ureditev listov v $\mathcal{R}$, ki jo lahko najdemo s Hierholzerjevim algoritmom v času $O(|E(\mathcal{R})|)$. Z njo lahko dodamo potrebne povezave v času $O(|\mathcal{L}(\mathcal{R})|)$. Če upoštevamo, da so lahko listi zgolj neprazna vozlišča v $\mathcal{R}$, dobimo $|\mathcal{L}(\mathcal{R})| < n$. Pokažimo še, da velja $E(\mathcal{R}) < 2V(\mathcal{R})$. Opazimo, da se neenakosti približamo, ko je kaktus unija 2-ciklov, v tem primeru velja $E(\mathcal{R}) = 2V(\mathcal{R}) - 2$. Denimo, da je kaktus z največ povezavami na danem številu vozlišč in ni sestavljen iz samih 2-ciklov. Potem obstaja k-cikel, kjer je $k \geq 3$, ki je podgraf kaktusa. Če v k-ciklu odstranimo eno povezavo, vsem ostalim pa dodamo vzporedno povezavo, je dobljeni graf še vedno kaktus, število povezav pa se je povečalo. Od tod sledi, da kaktus, ki je unija verig, vsebuje maksimalno število povezav, torej prejšnja neenakost velja. Sedaj velja $O(|E(\mathcal{R})|) < O(4n) = O(n)$.

Na koncu seštejemo vse dobljene časovne zahtevnosti in dobimo časovno zahtevnost $O(nm + nm \log n)$.

Poglavje 5

Povečanje povezavne povezanosti na dano vrednost

V tem poglavju bomo opisali, kako povečati povezavno povezanost na dano vrednost z uporabo Frankovega algoritma iz [5]. Ta temelji na Maderjevem izreku o razcepljanju, podanem v [12], za katerega bomo podali krajši dokaz.

5.1 Splošna Frankova metoda

V tem razdelku bomo predstavili splošno shemo Frankove metode za probleme povečanja povezanosti. To bomo uprabili za rešitev problema povečanje povezavne povezanosti na dano vrednost, lahko pa bi jo tudi za sorodne probleme, kot je problem povečanje vozliščne povezanosti na dano vrednost, problem povečanje povezavne povezanosti na dano vrednost z omejitvami in problem povečanje povezavne povezanosti na usmerjenih grafih.

V ta namen definirajmo nekaj pojmov in zaradi priročnosti predpostavimo nekaj lastnosti.

Definicija 5.1. Dan je graf $G = (V, E)$ in funkcija $r : V \times V \to \mathbb{Z}_+$, kjer je $\mathbb{Z}_+$ množica nenegativnih celih števil. Če se povezava (u, v) v G pojavi večkrat, le to predstavimo s celoštevilsko utežjo na tej povezavi. Za rešitev *problema povečanja lokalne povezanosti* je potrebno poiskati najmanjšo (ozi-

roma najcenejšo) množico novih povezav F, da v $G + F$ velja:

$$\lambda(u, v) \geq r(u, v) \qquad \text{za vsak } u, v \in V. \tag{5.1}$$

Funkcijo r imenujemo *funkcija zahtev*. Za problem povečanja povezavne povezanosti na dano vrednost k je fukcija $r(u, v) = k$ za vsaka $u, v \in V$.

Za r bomo predpostavili, da usteza pogojem:

$$r(u, v) \geq \lambda(u, v) \qquad \text{za vsak } u, v \in V, \tag{5.2}$$

$$r(u, v) \geq \min\{r(u, w), r(v, w)\}. \tag{5.3}$$

Definicija 5.2. Za podmnožici $X, Y \subseteq V$ v grafu G naj $d_G(X, Y)$ oziroma $d(X, Y)$ označuje število povezav med $X - Y$ in $Y - X$ v G in $\overline{d}_G(X, Y)$ oziroma $\overline{d}(X, Y) = d(X \cap Y, V - (X \cup Y))$. Za krajši zapis bomo v primiru $d(X, V - X)$ uporabljali kar $d(X)$.

Definicija 5.3. Za dano funkcijo zahtev r definiramo *funkcijo množice zahtev* $R(.)$ kot:

$$R(X) = \max\{r(u, v) | u \in X, v \in V - X\}.$$

Iz tega sledi $R(V) = R(\emptyset) = 0$.

Definicija 5.4. Količino $q(X) := R(X) - d(X)$ imenujemo *pomanjkanje* X.

Predstavimo dve enakosti o številu povezav med poljubnima podmnožicama v V in direktni posledici.

Trditev 5.5. *Za vsak graf $G = (V, E)$ in vsaki podmnožici $X, Y \subseteq V$ velja*

$$d(X) + d(Y) = d(X \cap Y) + d(X \cup Y) + 2d(X, Y), \tag{5.4}$$

$$d(X) + d(Y) = d(X - Y) + d(Y - X) + 2\overline{d}(X, Y). \tag{5.5}$$

Dokaz. Opazimo, da je $d(X) = d(V - X)$. S tem opažanjem bomo dokazali 5.4:
$d(X) + d(Y) = d(V - X) + d(Y) = d((V - X) \cap Y) + d((V - X) \cup Y) +$
$2d((V - X) - Y, Y - (V - X)) = d(X \cap Y) + d(X \cup Y) + 2d(X - Y, Y - X) =$

$d(X \cap Y) + d(X \cup Y) + 2d(X, Y)$
in 5.5:
$d(X)+d(Y) = d(V-X)+d(Y) = d((V-X)\cap Y)+d((V-X)\cup Y)+2d((V-X)-Y, Y-(V-X)) = d(X-Y)+d(Y-X)+2d(X\cap Y, V-(Y\cup X)) = d(X-Y)+d(Y-X)+2\overline{d}(X,Y)$. □

Trditev 5.6. *Za vsaki podmnožici $X, Y \subseteq V$ velja vsaj ena od naslednjih lastnosti:*

$$R(X) + R(Y) \leq R(X \cap Y) + R(X \cup Y), \tag{5.6}$$

$$R(X) + R(Y) \leq R(X - Y) + R(Y - X). \tag{5.7}$$

To lahko dokažemo z uporabo enačb (5.4),(5.5) in nekaj osnovne analize primerov. Dokaz pa tudi najdemo v [5, v trditev 5.4].

5.1.1 Metoda razcepljanja

Opisali bomo splošno shemo, ki jo je razvil A. Frank [5] za reševanje problema povečanja povezanosti.

Najprej vpeljimo nekaj pojmov. Pravimo, da povezavi $e = su$ in $f = sv$ *razcepimo*, če ju zamenjamo z novo povezavo uv. Torej G zamenjamo z $G^{ef} = G - \{e, f\} + uv$, kjer je uv nova povezava. Za dano lastnost $\mathcal{P}$ (npr. k-povezavno povezanost) grafa G pravimo, da je razcepljanje povezave e, f *$\mathcal{P}$-legalno* ali da je graf G *e, f razcepljiv*, če $\mathcal{P}$ velja tudi za graf G^{ef}. Razcepljanje je ena izmed pogosto uporabljenih osnovnih tehnik v algoritmih za povečanje povezanosti. Na sledečem generičnem problemu bomo predstavili splošno shemo, ki temelji na razcepljanju povezav.

Splošna shema za reševanje problema povezanosti
Input: Graf $G = (V, E)$ in zahtevana lastnost povezanosti $\mathcal{P}$.
Output: Množica novih povezav F najmanjše moči oziroma z najmanjšo ceno, takšna, da za $G + F$ velja $\mathcal{P}$.

1. **Razširitev:** Razširimo G, tako da dodamo novo vozlišče s in dodajamo nove povezave med s in V, dokler zahtevana lastnost povezanosti $\mathcal{P}$ ne drži za nov graf (predvidevamo, da je takšna konstrukcija izvedljiva).

2. **Odstranitev:** Odstranjujemo nove povezave in pri tem ohranjamo zahtevano lastnost povezanosti $\mathcal{P}$. To počnemo, dokler ni vsaka nova povezava kritična. Dobljeni graf H imenujemo *kritična razširitev* grafa G.

3. **Razcepljanje:** Ponavljamo $\mathcal{P}$-legalno razcepljanje na grafu H, dokler je to možno.

4. **Zaključek:** Uporabimo sosede s (če le ti obstajajo), med katerimi najdemo takšno množico povezav F', da $H+F'-s$ zadostuje zahtevi $\mathcal{P}$. Ta korak je odvisen od problema, pri povečanju povezavne povezanosti ga ne bomo potrebovali.

5.1.2 Določanje spodnje meje in robnih komponent

Problem povečanja povezavne povezanosti podamo s parom (G, r), kjer je G graf in $r : V \times V \to \mathbb{Z}_+$ funkcija zahtev. Naj bo $F \subseteq V \times V$. Če $G' = G + F$ zadostuje zahtevi povezanosti (5.1), definirani z r, po Mengerjevem izreku, velja:

$$d_{G'}(X) \geq R(X) \qquad \text{za vsako množico } X \subseteq V. \tag{5.8}$$

Označimo z $d_F(X)$ število povezav iz F, ki imajo obe krajišči v množici X. Ker je F množica novih povezav, iz neenakost (5.8) sledi:

$$d_F(X) = d_{G'}(X) - d(X) \geq R(X) - d(X) = q(X) \qquad \text{za vsak } X \subseteq V. \tag{5.9}$$

Razcepljanje povezav e, f je v našem primeru legalno, če $G'^{e,f}$ ohranja zahtevano povezavno povezanost. Torej za vsak $u, v \in V$ mora veljati $\lambda_{G'^{e,f}}(u, v) \geq r(u, v)$.

Izpeljimo sedaj spodnjo mejo za F. Množica povezav F je (G, r)- *povečanje*, če graf $G + F$ zadošča zahtevam povezanosti, podanih z r. Množica F je *optimalna* za problem povečanja povezanosti, če ima najmanjšo moč. Naj *opt(G,r)* označuje najmanjšo velikost množice povezav za povečanje.

Problem povečanja lokalne povezanosti lahko formuliramo kot problem funkcije pokritja povezav. Naj bo $p : 2^V \to \mathbb{Z}_+$ funkcija, ki slika iz množice v cela pozitivna števila, podana na urejeni množici V. Množica povezav F na V je *povezavno pokritje* p oziroma *p-pokritje*, če velja $d_F(X) \geq p(X)$ za vsak $X \subseteq V$. Iz (5.9) sledi, da je za problem povečanja povezavne povezanosti primerna izbira za p funkcija pomanjkanja q.

Definicija 5.7. Družina $\mathcal{F}$ paroma disjunktnih neprazniih podmnožic V se imenuje *podparticija*. Definirajmo število potrebnih novih povezav kot $\nu(G, r) = \max\{\sum_{X \in \mathcal{F}} q(X) : \mathcal{F}$ je podparticija $V\}$.

Opazimo, da je vsaka množica povezav povečanja F tudi pokritje povezav q. Ker imajo množice v $\mathcal{F}$ prazen presek, lahko katera koli povezava iz F seka največ dve množici. S tem dobimo $opt(G, r) \geq \lceil \nu(G, r)/2 \rceil$. Vendar ta meja ni vedno dosegljiva. V nadaljevanju bomo definirali robne komponente in dokazali, da kadar jih graf ne vsebuje, je spodnja meja dosežena, v nasprotnem primeru pa bomo obravnavali podgraf brez robnih komponent in robne komponente ločeno.

Definicija 5.8. Komponenta C v grafu G je *robna* glede na funkcijo zahteve r, če veljata naslednji trditvi:

$$r(u, v) \leq \lambda(u, v) + 1 \qquad \text{za vsaka } u \in C, v \in V - C, \tag{5.10}$$

$$r(u, v) \leq \lambda(u, v) \qquad \text{za vsaka } u, v \in C. \tag{5.11}$$

Z r' bomo označili funkcijo zahteve za G'.

Lema 5.9. *Naj bo C robna komponenta glede na funkcijo zahteve r v grafu G, $G' = G - C$ in r' omejitev za r v $V - C$ (predvidevamo, da držita* (5.2) *in* (5.3)*). Če je F' optimalna rešitev za (G', r'), potem:*

1. *Če $q(C) = 0$, potem je F' optimalna rešitev za (G, r).*
2. *Če $q(C) = 1$, potem je $F' + vc$ optimalna rešitev za (G, r) za poljuben $c \in C$ in $v \in V - C$ z $r(v, c) = 1$.*

Dokaz. Primer $q(C) = 0$ je očiten. Če je $q(C) = 1$, dokažemo trditev s protislovjem. Po dodanju F' in povezave vc v graf G obstaja par vozlišč v', r' z $r(v', c') = 0$. Ker sta c, c' povezana, iz (5.2) dobimo $r(c, c') \geq 1$ in iz (5.3) dobimo $r(v', c) \geq 1$ ter $r(v, v') \geq 1$. Ker je F' rešitev za (G', r'), obstaja v, v'-pot v G' in z vc ter cc'-potjo v C dobimo $v'c'$-pot v $G + F' + vc$. To je protislovje, ker je $opt(G, r) \leq opt(G', r') + q(C)$. Poglejmo si še dokaz v drugo smer. Naj bo $G_C = G/C$ in r_C zmanjšana zahteva r za graf G_C. Opazimo, da lahko povezanost povečamo samo na skrčitvi grafa, kjer se komponenta C skrči v v_C, saj velja $opt(G_C, r_C) \leq opt(G, r)$. Naj bo F_C optimalno povečanje (G_C, r_C) z minimalnim $t := |\{x : xv_C \in F_C\}|$. Če je $t = 0$, je to enako, kot da je $q(C) = 0$, tako je $t \geq 1$. Naj bo $t = 1$ in $f \in F_C$ povezava s krajiščem v v_C. Potem je $F - f$ rešitev za (G', r') in velja $opt(G', r') \leq |F_C| - 1 \leq opt(G_C, r_C) - q(C) \leq opt(G, r) - q(C)$, kot zahtevamo. Naj bo $t \geq 2$ in naj bo $u_1v_C, .., u_tv_C$ t povezav s krajiščem v v_C, ki pripadajo F_C. V tem primeru opazimo, da lahko vsako povezavo $u_iv_C, 2 \leq i \leq t$ zamenjamo z u_iu_1 in dobimo optimalno rešitev za (G_c, r_c) z manj povezavami s krajiščem v_C, kar je protislovje. □

Z uporabo leme 5.9 lahko prevedemo problem povečanja povezavne povezanosti na primer brez marginalnih komponent na sledeči način: naj bodo $C_1, .., C_t$ komponente G takšne, da je C_i marginalna komponenta za $G - (C_1 \cup ... \cup C_{i-1})$ in $G - (C_1 \cup ... \cup C_{t-1})$ nima marginalnih komponent. Optimalna rešitev za celoten graf G je optimalna rešitev za $G - (C_1 \cup ... \cup C_t)$ z dodajanjem komponent C_i, $i = t, ..., 1$ in povezav $q(C_i)$ za vsako komponento C_i kot v lemi 5.9.

5.1.3 Uporaba metode razcepljanja

Na osnovi zgoraj predstavljenih ugotovitev bomo uporabili metodo razcepljanja na grafu G brez robnih komponent. Graf G bo od slej e, f razcepljiv, če ima graf $G^{e,f}$ enako povezavno povezanost kot graf G. Na tem mestu tudi spomnimo, da smo s s označili dodatno vozlišče.

Definicija 5.10. Množica X je *kritična*, če velja $d_{G'}(X) = R(X)$.

Lema 5.11. *Naj bo G' kritična razširitev G za lastnost $\mathcal{P}$, podano s* (5.1). *Potem je $d_{G'}(s) = \nu(G, r)$.*

To lemo bomo pokazali s pomočjo dveh trditev. Dokaz trditve 5.12 najdemo [5, trditev 3.6].

Trditev 5.12. *Za poljubni dve kritični množici X, Y, ki nista disjunktni, velja vsaj ena od naslednjih lastnosti:*

1. *$X \cup Y$ in $X \cap Y$ sta kritični.*

2. *$X - Y$ in $Y - X$ sta kritični in $\overline{d}(X, Y) = 0$.*

Naj bo S množica sosedov s v G'. Povezava *su* preživi proces odstranjevanja, če in samo če obstaja kritična množica X, ki vsebuje u. Naj bo $\mathcal{F} = \{X_1, X_2, ..., X_t\}$ družina kritičnih množic pokritja S, takšnega, da je t najmanjši in zanj je $\sum_{i=1}^{t} |X_i|$ najmanjša.

Trditev 5.13. *Naj bosta množici $X, Y \subseteq \mathcal{F}$. Potem je $X \cap Y$ prazen.*

Dokaz. Množica $X \cup Y$ ne more biti kritična, ker je $\mathcal{F}$ najmanjša, iz česar sledi, da sta $X - Y$, $Y - X$ kritični in velja $\overline{d}(X, Y) = 0$. Od tod dobimo $S \cap (X \cap Y) = \emptyset$ in ker je $\sum |X_i|$ najmanjša, dobimo $|X| = |X - Y|$ in $|Y| = |Y - X|$, iz česar sledi $X \cap Y = \emptyset$, kar smo želeli dokazati. □

Dokaz. Dokazali bomo lemo 5.11. Iz trditve 5.13 sledi, da je $\mathcal{F}$ podparticija V in $d_{G'}(s) = \sum_{X \in \mathcal{F}} (d_{G'}(X) - d(X)) = \sum_{X \in \mathcal{F}} q(X) \leq \nu(G, r)$. □

Če je število $d_{G'}(s)$ liho, dodamo še eno povezavo (vzporedno z obstoječo). Velja, da nobena prerezna povezava nima krajišča v s. Če bi $e = vs$ bila prerezna povezava, potem bi $G' - e$ vsebovala robno komponento v. Da izvedemo korak 3 v splošni shemi, potrebujemo naslednjo različico Maderjevega izreka, ki ga bomo dokazali kasneje.

Izrek 5.14. *(Maderjev izrek)*
Naj bo graf $G = (V \cup \{s\}, E)$, kjer je $d(s)$ sodo število in nobena prerezna povezava nima krajišča v s. Potem lahko na množici povezav, ki imajo krajišče v s, naredimo particijo na $d(s)/2$ razcepljivih parov povezav.

5.2 Algoritem

Splošno shemo bomo zapisali v obliki algoritma za povečanje povezavne povezanosti na dano vrednost. Multigraf oziroma graf bomo predstavili kot utežen graf, kjer so teže $c(e)$ celoštevilske. Če povezave e v grafu ni, naj velja $c(e) = 0$. Naj bo $r_{\max} = \max\{r(u, v) \mid u, v \in V\}$ največja vrednost za r.

1. **Razširitev:** Razširimo G, tako da dodamo novo vozlišče s in za vsako vozlišče $v \in V$ dodamo povezavo sv z utežjo $r_{\max}$. S tem je pogoj (5.1) izpolnjen.

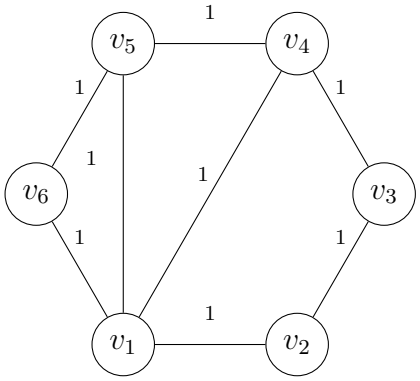
2. **Odstranitev:** Za vsako novo povezavo e zmanjšujemo utež $c(e)$, dokler pri tem ohranjamo zahtevano lastnost (5.1).

3. **Razcepljanje:** Ponavljamo (uteženo) legalno razcepljanje, dokler je to možno.

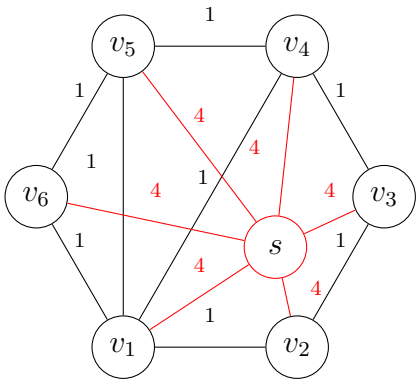
Oglejmo si še pravilnost algoritma. Kot smo razložili že za splošno shemo, algoritem najde možno rešitev. Po izreku 5.14 dobimo po koraku 2 graf, v katerem vozlišče s nima sosedov (zato korak 4 ni potreben). Po lemi 5.11 je stopnja vozlišča s pred korakom 3 enaka $\nu(G)$. S tem je število dodanih povezav enako prej dokazani spodnji meji $\lceil \nu(G, r)/2 \rceil$, kar pomeni, da je rešitev optimalna.

5.2.1 Primer izvajanja

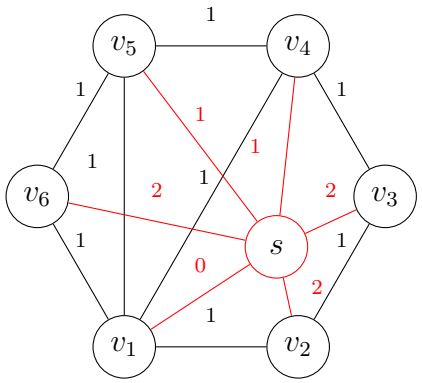
Na spodnjih štirih slikah je prikazan primer izvajanaj prej opisanega algoritma. Na sliki 5.1 smo predstavili naključen graf G, ki je 2-povezan. Želimo mu povečati povezavno povezanost na 4. Zato smo na sliki 5.2 izvedli prvi korak algoritma tako, da smo dodali vozlišče s in ga povezali z vsemi vozlišči s povezavami, ki imajo utež 4. V drugem koraku smo uteži zmanjšali, kar je prikazano na sliki 5.3. Na koncu smo na novo pridobljene povezave razcepili. Ker je stopnja vozlišča s soda, nam dodatne povezave ni bilo potrebno dodati. Končen rezultat je prikazan na sliki 5.4.



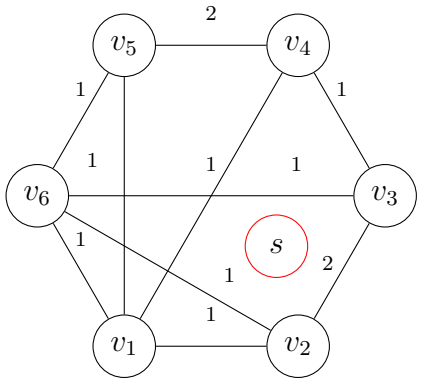
Slika 5.1: Graf G.



Slika 5.2: Dodanje vozlišča s v graf G.



Slika 5.3: Kritična razširitev grafa G.



Slika 5.4: Razcepljanje.

5.2.2 Časovna zahtevnost

Pokazali bomo, da so vsi koraki izvedljivi v polinomskem času in podali časovno zahtevnost v O-notaciji glede na vmesne rezultate, podane v [14, 5]. Naj bo $n = |V|$ in $m = |E|$. Potrebovali bomo še naslednjo definicijo in rezultat.

Definicija 5.15. Za vsako množico vozlišč $X \subseteq V$ velja $d(X) \geq R(X)$. Definirajmo *presžek* za X kot: $s(X) = d(X) - r(X)$. Potem za množico X

pravimo, da je *tesna*, če je $s(X) = 0$, in *nevarna*, če je $s(X) \leq 1$.

Lema 5.16. *Če par su, sv povezav v grafu G ni razcepljiv, potem ne bo postal razcepljiv po razcepitvi drugega para povezav sx, sy.*

Dokaz. Denimo, da par su, sv ni razcepljiv, potem obstaja nevarna množica X, ki vsebuje u, v. Opazimo, da po razcepljanju poljubnega para sx, sy, ki ni enak prvotnemu, kar pomeni $x \neq v$,$y \neq u$ ali $x \neq u$,$y \neq v$, se $d(X)$ ne bo spremenila (v primeru, da povezava prispeva k $d(X)$, bo tudi nova povezava prispevala k $d(X)$). Tako bo X ostala nevarna in par su, sv nerazcepljiv. □

Korak 1. je izvedljiv v času $O(n)$, kar je linearno.

Korak 2. lahko izvedemo tako, da za vsako povezavo najdemo najmanjšo utež, ki ustreza (5.1). Najdemo jo z binarnim iskanjem v domeni $[\,0, r_{\max}]$. Preverjanje, ali ustreza (5.1) pa izvedemo z algoritmom za največji pretok, ki ima časovno zahtevnostjo $O(nm \log(n^2/m))$ (glej konec poglavja o grafih). Vsaka povezava, ki ima krajišče v s, se pregleda le enkrat. Iz tega sledi časovna zahtevnost $O(n^2 \log(r_{\max})(m + n \log n))$.

Korak 3. lahko izvedemo s poskušanjem izvedbe *uteženega razcepljanja*, kjer za vsak par vozlišč $u, v \in V$ poiščemo največji α, tak, da po zmanjšanju $c(sv)$, $c(su)$ in povečanju $c(uv)$ za α, dobljeni graf ustreza (5.1). To je možno implementirati z binarnim iskanjem za α v domeni $[\,0, \min(c(su),\ c(sv))]$. Po lemi 5.16 lahko vsak par obravnavamo zgolj enkrat. Iz tega sledi časovna zahtevnost $O(n^3 \log(r_{\max})\ (m + n \log n))$, kar je po lastnostih O-notacije tudi časovna zahtevnost celotnega algoritma.

Problem bi lahko rešili tudi tako, da bi večkrat klicali algoritem za povečanje povezavne povezanosti za ena. Naor, Gusfield in Martel [15] so pokazali, da s pravilno izbiro vozlišč v listih kaktusa vedno najdemo optimalno

reširev. Vendar je algoritem večkratnega povečevanja bolj občutljiv na velikost povečanja k, saj je njegova časovna zahtevnost ob uporabi algoritma Haoa in Orlina za pretok enaka $O(k^2 + mn^2 \log(n^2/m))$. Frankov algoritem je zato dosti boljši za velike k, medtem ko je večkratno poklican algoritem povečanja povezavne povezanosti za ena hitrejši, kadar je k majhen, n pa velik.

5.3 Dokaz Maderjevega izreka

Maderjev izrek obravnava razcepljanje, pri katerem se ohranja lokalna povezavna povezanost. Tako e, f razcepljanje je legalno, če $G^{e,f}$ ohranja lokalno povezavno povezanost, torej za vsak $x, y \in V$ velja $\lambda_{G^{e,f}}(x, y) \geq \lambda_G(x, y)$ in funkcija zahteve je enaka lokalni povezavni povezanosti $r(X) = d(X)$. Za celotno podpoglavje velja, da $G = (V \cup \{s\}, E)$ nima prereznih povezav s krajiščem v s. Množico sosedov vozlišča s bomo krajše pisali S in za razcepljanje $\{su, sv\}$, kjer sta $\{u, v\} \subset V$ soseda vozlišča s, bomo krajše pisali $\{u, v\}$ razcepljanje ali u, v razcepljanje.

Pri dokazu izreka bomo potrebovali naslednje trditve in izreka. Večina dokazov je enostavnih, najdemo jih v [11].

Trditev 5.17. *Za poljubni podmnožici vozlišč $X, Y \subseteq V$ velja vsaj ena od naslednjih neenakosti:*

$$s(X) + s(Y) \geq s(X \cap Y) + s(X \cup Y) + 2d(X, Y), \tag{5.12}$$

$$s(X) + s(Y) \geq s(X - Y) + s(Y - X) + 2\overline{d}(X, Y). \tag{5.13}$$

Trditev 5.18. *$x, y \in S$ sta razcepljivi, če in samo če ne obstaja nevarna množica X taka, da $x, y \in X$.*

Trditev 5.19. *Naj bo T tesna množica. Par vozlišč $\{u, v\}$ je razcepljiv v G, če je pripadajoči par $\{u', v'\}$ razcepljiv v $G' = G/T$.*

Trditev 5.20. *Če je vsaka tesna množica sestavljena iz enega elementa, potem $\lambda(x, y) = \min\{d(x), d(y)\}$ velja za vsak x, y.*

Izrek 5.21. *Naj bo $G = (V \cup \{s\}, E)$ povezan graf z $d(s) \neq 3$. Potem obstaja razcepljiv par povezav e, f.*

Izrek 5.22. *Naj bo $G = (V \cup \{s\}, E)$ graf in $d(s)$ sodo število. Potem obstaja particija množice povezav s krajiščem v s na $d(s)/2$ disjunktnih razcepljivih parov.*

Trditev 5.23. *Če sta e, f razcepljivi v G, za katerega velja, da nima prereznih povezav s krajiščem v s, potem ta lastnost velja tudi za G^{ef}.*

Trditev 5.24. *Izreka 5.21 in 5.22 sta ekvivalentna.*

Dokaz. Maderjevega izreka o razcpljanju.
Naj bo $G(V \cup \{s\}, E)$ protiprimer z najmanjšim številom vozlišč, za katerega je $d(s)$ sodo. Po trditvi 5.23 je zadosti, da pokažemo, da ne obstaja razcepni par povezav. Ker je G minimalen, po trditvi 5.19 velja, da je vsaka tesna množica sestavljena iz enega elementa, kot v trditvi 5.20. Naj bo $t \in S$. Potem velja $d(t) = \min\{d(v) | v \in S\}$. Ker je dokaz zapleten, ga bomo sedaj razdelili v več krajših trditev.

Trditev 5.25. *$R(X - t) \geq R(X)$ za vsako nenevarno množico X, ki vsebuje t.*

Dokaz. Naj bo $u \in ((X \cap S) - t)$ in naj bo par vozlišč (v, z) takšen, da velja $v \in X$, $z \in V - X$, $\lambda(v, z) = R(X)$. Če $v \neq t$, potem $R(X - t) \geq \lambda(v, z) = R(X)$. Drugače veljajo pogoji za trditev 5.20, od koder dobimo $R(X) = \lambda(t, z) = \min\{d(t), d(z)\} \leq \min\{d(u), d(z)\} = \lambda(u, z) \leq R(X - t)$, kot zahtevano. □

Trditev 5.26. *Za nevarno množico X velja $d(s, X) \leq d(s, V - X)$.*

Dokaz. $R(V-X) = R(X) \geq d(X)-1 = d(V-X)-d(s, V-X)+d(s, X)-1 \geq R(V-X)-d(s, V-X)+d(s, X)-1$ od tod $d(s, V-X) \geq d(s, X)-1$, saj velja $d(s, V-X) \neq d(s, X)$, drugače bi bil $d(s)$ lih. □

Noben par u, t za $u \in S$ ni razcepljiv, tako je vsako vozlišče iz S v nevarni množici, ki vsebuje t. Naj $\mathcal{L}$ označuje minimalno družino nevarnih množic, ki vsebujejo t. Tedaj $S \subseteq \bigcup_{X \in \mathcal{L}} X$.

Trditev 5.27. *Minimalna družina nevarnih množic $\mathcal{L}$ vsebuje vsaj 3 nevarne množice.*

Dokaz. Recimo, da je $|\mathcal{L}| = 2$, torej $\mathcal{L} = \{X, Y\}$. Dvakrat bomo uporabili dejstvo, da je t v X, Y in ni v komplementu. Po trditvi 5.26 velja $d(s, X) \leq d(s, V - X) < d(s, Y) \leq d(s, V - Y) < d(s, X)$, kar je protislovje. □

Fiksirajmo $\{X_1, X_2, X_3\} = \mathcal{F} \subseteq \mathcal{L}$. Vsak X_i vsebuje vozlišče x_i, ki ni vsebovano v nobeni drugi množici $\mathcal{L}$ (in $\mathcal{F}$) zaradi minimalnosti $\mathcal{L}$.

Trditev 5.28. *Za vsaka dva $X, Y \in \mathcal{F}$ velja* (5.3).

Dokaz. Recimo, da (5.3) ne velja. Potem velja (5.2). Množica $\mathcal{L}$ je minimalna, zato $X \cup Y$ ni nevarna. Potem velja $s(X \cup Y) \geq 2$. Zato $1+1 \geq s(X)+s(Y) \geq s(X \cap Y)+s(Y \cup X) \geq 0+2$ in tako je množica $X \cap Y$ tesna, kar pomeni, da vsebuje zgolj en element, po definiciji $X \cap Y = \{t\}$. Potem $X - Y = X - t$, $Y - X = Y - t$ in s trditvijo 5.25 dobimo $s(X) + s(Y) \geq s(X - Y) + s(Y - X) + 2\overline{d}(X, Y)$. Torej (5.3) drži, kar je protislovje. □

Trditev 5.29. *Za vsaka dva $X, Y \in \mathcal{F}$ je $|X - Y| = |Y - X| = 1$ in $\overline{d}(X, Y) = 1$.*

Dokaz. Iz trditve 5.28 izhaja $1 + 1 \geq s(X) + s(Y) \geq s(X - Y) + s(Y - X) + 2\overline{d}(X, Y) \geq 0 + 0 + 2$. Sledi $\overline{d}(X, Y) = 1$ in množici $X - Y$, $Y - X$ sta tesni, torej vsebujeta en element. □

Sedaj lahko zaključimo dokaz Maderjevega izreka. Naj bo $M = X_1 \cap X_2 \cap X_3$. Po zadnji trditvi velja $X_i = M + x_i$ in tak M je presek vsakih dveh množic v $\mathcal{F}$. Iz $\overline{d}(X_i, X_j) = 1$ sledi $d(M) = 1$, torej edina povezava izven komponente M je st, ki je v tem primeru prerezna povezava s krajiščem v s, kar je protislovje. □

Literatura

[1] Jorgen Bang-Jensen, Harold N Gabow, Tibor Jordán, and Zoltán Szigeti. Edge-connectivity augmentation with partition constraints. *SIAM Journal on Discrete Mathematics*, 12(2):160–207, 1999.

[2] Kapali P. Eswaran and R. Endre Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976.

[3] L Fleischery. Building chain and cactus representations of all minimum cuts from hao-orlin in the same asymptotic run time. 1999.

[4] Herbert Fleischner. *Eulerian graphs and related topics*, volume 1. Elsevier, 1990.

[5] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992.

[6] Harold N. Gabow and Tibor Jordán. How to make a square grid framework with cables rigid. *SIAM Journal on Computing*, 30(2):649–680, 2000.

[7] Dan Gusfield. Optimal mixed graph augmentation. *SIAM Journal on Computing*, 16(4):599–612, 1987.

[8] Jianxiu Hao and James B. Orlin. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 165–174. Society for Industrial and Applied Mathematics, 1992.

[9] Ming-Yang Kao. Data security equals graph connectivity. *SIAM Journal on Discrete Mathematics*, 9(1):87–100, 1996.

[10] Alexander V Karzanov and Eugeniy A Timofeev. Efficient algorithm for finding all minimal edge cuts of a nonoriented graph. *Cybernetics*, 22(2):156–162, 1986.

[11] Gilad Liberman. *Connectivity Augmentation Problems*. PhD thesis, The Open University, 2005.

[12] Wolfgang Mader. A reduction method for edge-connectivity in graphs. In *Annals of Discrete Mathematics*, volume 3, pages 145–164. Elsevier, 1978.

[13] Hiroshi Nagamochi and Toshihide Ibaraki. Deterministic $\widetilde{O}(nm)$ time edge-splitting in undirected graphs. *Journal of Combinatorial Optimization*, 1(1):5–46, 1997.

[14] Hiroshi Nagamochi and Toshihide Ibaraki. *Algorithmic aspects of graph connectivity*, volume 123 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2008.

[15] Dalit Naor, Dan Gusfield, and Charles Martel. A fast algorithm for optimally increasing the edge connectivity. *SIAM Journal on Computing*, 26(4):1139–1165, 1997.

[16] Dalit Naor and Vijay V Vazirani. Representing and enumerating edge connectivity cuts in rnc. In *Workshop on Algorithms and Data Structures*, pages 273–285. Springer, 1991.

[17] Ján Plesník. Minimum block containing a given graph. *Archiv der mathematik*, 27(1):668–672, 1976.

[18] Krishnaiyan Thulasiraman, Subramanian Arumugam, Andreas Brandstädt, and Takao Nishizeki, editors. *Handbook of graph theory, combi-*

natorial optimization, and algorithms. Chapman & Hall/CRC Computer and Information Science Series. CRC Press, Boca Raton, FL, 2016.

[19] Shuji Tsukiyama, Keiichi Koike, and Isao Shirakawa. An algorithm to eliminate all complex triangles in a maximal planar graph for use in vlsi floorplan. In *Algorithmic Aspects Of VLSI Layout*, pages 321–335. World Scientific, 1993.

[20] Toshimasa Watanabe and Akira Nakamura. Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96–144, 1987.

[21] Robin J. Wilson and John J. Watkins. *Uvod v teorijo grafov*. Društvo matematikov, fizikov in astronomov Slovenije, Ljubljana, 1997.