Algoritmi za particioniranje grafova

Iveković, Ana

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ana Iveković

ALGORITMI ZA PARTICIONIRANJE GRAFOVA

Diplomski rad

Voditelj rada:
Doc. dr. sc. Zvonimir Bujanović

Zagreb, 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Hvala mojoj obitelji na strpljenju i toleranciji. Posebno hvala Zvonku i Josipu na optimizmu, smirenosti i svim odgovorima na moja pitanja. Hvala i Ivi, Mirni i Ani na prijateljstvu i podršci. Neizmjerno hvala Igoru što mi je cijelo vrijeme bio veliki oslonac i motivacija. Hvala i mentoru Zvonimiru Bujanoviću na izdvojenom vremenu, strpljivosti, korektnosti i poučnim komentarima.

Sadržaj

Uvod

Istraživanje particioniranja grafova započelo je u ranim 70-im godinama prošloga stoljeća. Cilj je za dani graf pronaći particiju skupa vrhova određenog kardinaliteta takvu da je zadovoljen neki uvjet optimalnosti. Mnogi se problemi iz stvarnog svijeta mogu svesti na problem pronalaženja optimalne particije grafa. Neki su primjeri transportne mreže, segmentiranje slika i redukcija dimenzije prilikom dizajna integriranih krugova. Također, danas se sve više i više govori o istraživanju društvenih navika, posebno putem društvenih mreža. Kako su ljudi na društvenim mrežama povezani relacijama prijateljstva (Facebook), praćenja (Instagram) i slično, moguće je za takvu mrežu modelirati graf u kojima su ljudi čvorovi, a relacije među njima bridovi. Particioniranjem takvog grafa moguće je detektirati i okarakterizirati neke skupine. Zato je ovo područje vrlo interesantno i neprestano se razvija.

S druge strane, radi se o sve većim i većim grafovima pa razvoj metoda za particioniranje traži sve više poboljšanja u odnosu na standardno korištene metode. U ovom ćemo radu opisati matematičku stranu problema particioniranja grafa. Na početku ćemo dati sve potrebne definicije i definirati sam problem, a nakon toga ćemo promotriti neke poznate metode rješavanja, spektralno particioniranje i višerazinsku metodu. Vidjet ćemo kako još postoje brojni problemi i nedostaci u opisanim algoritmima, što će najbolje prikazati potrebu za daljnjim istraživanjem. Na kraju ćemo implementirati neke od opisanih algoritama i usporediti dobivene rezultate.

Poglavlje 1

Pojmovi i oznake

Rješavanje problema particioniranja grafova spaja nekoliko matematičkih područja. U radu će nam biti potrebni pojmovi iz teorije grafova, neki pojmovi iz područja diskretne matematike te optimizacije. Zato počinjemo definiranjem pojmova koje ćemo koristiti. Poglavlje je podijeljeno na dva dijela. U prvom je dijelu dana teorija koja se tiče grafova, a u drugom se nalaze ostali pojmovi s kojima ćemo se često susretati, npr. pojam particije.

1.1 Grafovi

Definicija 1.1.1. *(Graf) Neka je V neprazan skup i E skup koji sadrži uređene parove iz $V \times V$. Graf G jest uređeni par (V, E). Elemente skupa V nazivamo vrhovima grafa G, a elemente multiskupa E nazivamo bridovima grafa G. Graf G je konačan ako su V i E konačni, a inače je beskonačan.*

U ovom ćemo se radu susretati samo s konačnim grafovima pa odsad to nećemo posebno napominjati. Sljedeća definicija daje vrlo važnu podjelu grafova.

Definicija 1.1.2. *(Usmjeren i neusmjeren graf) Neka je $G = (V, E)$ graf. Ukoliko za svaki brid $(v_1, v_2) \in E$ vrijedi $(v_2, v_1) \in E$, kažemo da je graf G neusmjeren. U suprotnom kažemo da je graf G usmjeren.*

Neusmjerenim bismo grafovima mogli prikazati mrežu prijatelja na Facebooku. Naime, ako jedna osoba ima drugu za prijatelja, tada je relacija prijateljstva simetrična, tj. i druga osoba ima prvu na listi prijatelja. U nekim drugim slučajevima nije isto. Na društvenoj mreži Instagram moguće je da jedna osoba prati drugu, ali da suprotan smjer ne vrijedi. Tada bismo za prikaz mreže trebali usmjereni graf. U ovom ćemo se radu susretati samo s neusmjerenim konačnim grafovima.

Definicija 1.1.3. *Neka je $G = (V, E)$ graf te neka je $e = (v_1, v_2) \in E$. Kažemo da su vrhovi v_1 i v_2 krajevi od e, da su međusobno susjedni te da su incidentni s e. Za dva brida e_1 i e_2 kažemo da su susjedni ako imaju zajednički kraj, tj. $e_1 = (v, v_1)$ i $e_2 = (v, v_2)$.*

Počevši od sljedeće definicije, u radu će se često koristiti pojam kardinaliteta skupa. Ako je S neki skup, njegov ćemo kardinalitet označiti s $|S|$.

Definicija 1.1.4. *(Stupanj vrha) Neka je $G = (V, E)$ graf. Stupanj vrha $v \in V$ jest broj bridova kojima je on kraj:*

$$deg(v) = |\{(v, v') \in E, v' \in V\}|. \tag{1.1}$$

Definicija 1.1.5. *(Petlje) Brid čiji se krajevi podudaraju naziva se petlja.*

Definicija 1.1.6. *(Jednostavan graf) Neusmjereni je graf jednostavan ako nema petlji.*

Definicija 1.1.7. *(Regularan graf) Za neusmjereni graf kažemo da je regularan stupnja R ako svi vrhovi imaju isti stupanj R.*

U radu ćemo se baviti jednostavnim i regularnim grafovima. Još jedno svojstvo grafa koje ćemo spominjati jest povezanost.

Definicija 1.1.8. *(Put i povezan graf) Neka je $G = (V, E)$ graf i $n \in \mathbb{N}$, $n \leq |V| - 1$. Put na grafu G jest niz bridova $(v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)$, $v_i \in V$, takav da za svaki par i i j takav da je $i \neq j$ vrijedi $v_i \neq v_j$. Za graf G kažemo da je povezan ako za svaki par vrhova postoji put između njih.*

Definicija 1.1.9. *(Potpuno povezan graf) Za graf G kažemo da je potpuno povezan ili potpun ako je svaki par vrhova spojen bridom.*

Nastavljamo s definicijama sparivanja, koje će biti vrlo važne u opisu višerazinske metode u četvrtom poglavlju.

Definicija 1.1.10. *(Sparivanje na grafu) Sparivanje (engl. matching) na grafu $G = (V, E)$ jest skup bridova $M \subseteq E$ u kojemu nijedan par bridova nema zajedničkih vrhova, odnosno u kojemu ne postoje susjedni bridovi. Kažemo da su vrhovi u i v spareni u M ako su u i v krajevi nekog brida u M.*

Definicija 1.1.11. *(Najveće sparivanje) Sparivanje M na grafu G je najveće (engl. maximum matching) ako ne postoji sparivanje s većim brojem bridova.*

Definicija 1.1.12. *(Maksimalno sparivanje) Sparivanje M na grafu G je maksimalno (engl. maximal matching) ako se ne može proširiti još nekim bridom.*

Primijetimo da je svako najveće sparivanje ujedno i maksimalno. Obratno ne mora vrijediti. Nadalje, graf može imati više najvećih sparivanja, a time i više maksimalnih sparivanja.

Slika 1.1: Razlika između maksimalnog (lijevi graf: $M = \{e_2\}$) i najvećeg (desni graf: $M = \{e_1, e_3\}$) sparivanja, preuzeto iz [2].

Bridovi u grafu označavaju povezanost vrhova, a u primjenama često tu povezanost želimo dodatno označiti dodavanjem određenih vrijednosti, težina.

Definicija 1.1.13. *(Težinski graf) Za graf $G = (V, E)$ kažemo da je težinski ako je svakom bridu $e \in E$ pridružena realna vrijednost $w(e)$, koju nazivamo težina brida e. Paru vrhova $(v_1, v_2) \in V^2$ za koji vrijedi $(v_1, v_2) \notin E$ pridružujemo težinu $w(v_1, v_2) = 0$.*
Težina podskupa X skupa bridova E jest suma težina elemenata od X:

$$w(X) = \textstyle\sum_{e \in X} w(e).$$

Napomena: Težinu brida $e = (v_i, v_j)$ označavamo s $w(e)$, $w(v_i, v_j)$ ili w_{ij}.

Također, postoje problemi u kojima je potrebno vrhovima dodijeliti težine, no ovdje uzimamo da su vrhovi težine jedan, a bridovima dodjeljujemo različite vrijednosti, koje su u literaturi najčešće, a u ovom radu uvijek, strogo pozitivne cjelobrojne.

Ukoliko se radi o težinskom grafu, redefiniramo stupanj vrha $v \in V$ kao sumu težina svih njemu incidentnih bridova:

$$deg(v) = \sum_{(v,v') \in E} w(v, v'). \tag{1.2}$$

Netežinske grafove možemo promatrati kao težinske u kojima svi vrhovi i bridovi imaju težinu jednaku jedan. Nadalje, grafovima pridružujemo matrice u kojima su zapisana neka njihova svojstva poput stupnjeva vrhova i težina bridova. Iz tih matrica možemo pročitati mnoge korisne informacije, što ćemo kasnije vidjeti, posebno u primjeru spektralnog particioniranja grafa.

Definicija 1.1.14. *(Matrica susjedstva) Neka je $G = (V, E)$ jednostavan graf. Matricu $A \in \mathbb{R}^{|V| \times |V|}$ takvu da za sve $(i, j) \in \{1, 2, ..., |V|\}^2$ vrijedi*

$$A_{ij} = \begin{cases} 0, & ako\ i = j, \\ w_{ij}, & inače, \end{cases}$$

nazivamo matrica susjedstva grafa G.

Definicija 1.1.15. *(Matrica stupnjeva) Neka je G = (V, E) graf. Matricu* $D \in \mathbb{R}^{|V|\times|V|}$ *takvu da za sve* $(i, j) \in \{1, 2, ..., |V|\}^2$ *vrijedi*

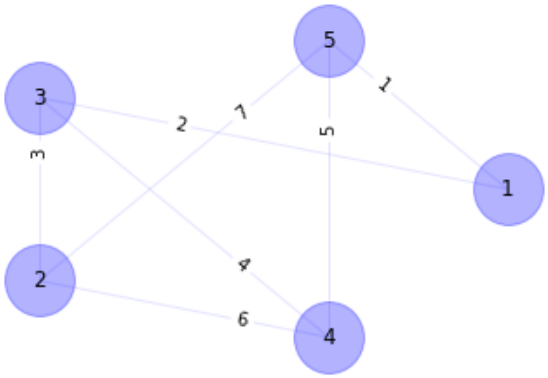
$$D_{ij} = \begin{cases} deg(i) = \sum_{k=1}^{|V|} w_{ik}, & ako\ i = j, \\ 0, & inače, \end{cases}$$

nazivamo matrica stupnjeva vrhova grafa G.

Definicija 1.1.16. *(Laplaceova matrica) Neka je graf G = (V, E) jednostavan graf, a matrice A i D redom pripadajuće matrice susjedstva i stupnjeva vrhova grafa G. Matricu*

$$L = D - A$$

nazivamo Laplaceova matrica grafa G.



$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 13 \end{pmatrix}, A = \begin{pmatrix} 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 3 & 6 & 7 \\ 2 & 3 & 0 & 4 & 0 \\ 0 & 6 & 4 & 0 & 5 \\ 1 & 7 & 0 & 5 & 0 \end{pmatrix}, L = \begin{pmatrix} 3 & 0 & -2 & 0 & -1 \\ 0 & 16 & -3 & -6 & -7 \\ -2 & -3 & 9 & -4 & 0 \\ 0 & -6 & -4 & 15 & -5 \\ -1 & -7 & 0 & -5 & 13 \end{pmatrix}$$

Slika 1.2: Primjer neusmjerenog težinskog grafa i njemu pripadajuće matrice stupnjeva, matrice susjedstva te Laplaceove matrice.

1.2 Ostali pojmovi

Definicija 1.2.1. *(Particija skupa) Neka je $X \neq \emptyset$. Kažemo da je familija skupova $\{X_i : i \in I\}$ particija skupa X ako vrijedi:*

1. *za sve $i \in I$ je $X_i \subseteq X$,*
2. *za sve $i \in I$ je $X_i \neq \emptyset$,*
3. *za sve $i, j \in I, i \neq j$, vrijedi $X_i \cap X_j = \emptyset$,*
4. *$\cup_{i \in I} X_i = X$.*

Broj elemenata particije, odnosno kardinalitet particije, označavamo s k.

Broj particija skupa nazivamo Bellov broj. S B_n ćemo označavati n-ti Bellov broj, odnosno broj particija skupa kardinaliteta n. Po definiciji je $B_0 = 1$, a Bellovi brojevi dalje zadovoljavaju rekurziju:

$$B_n = \sum_{k=1}^{n} \binom{n-1}{k-1} B_{n-k}, n \in \mathbb{N}. \tag{1.3}$$

Ako neki skup ima n elemenata, onda broj njegovih particija koje se sastoje od točno k skupova nazivamo Stirlingov broj druge vrste i označavamo sa $S_{n,k}$. Vrijedi:

$$S_{n,k} = \sum_{i=1}^{k} (-1)^i \frac{(k-i)^n}{i!(k-i)!}. \tag{1.4}$$

Također, vrijedi:

$$B_n = \sum_{i=1}^{n} S_{n,i}. \tag{1.5}$$

Definicija 1.2.2. *(Kombinatorni optimizacijski problem) Kombinatorni optimizacijski problem jest uređena trojka (S, p, f) takva da:*

- *S je diskretan skup koji nazivamo prostor rješenja,*
- *p je predikat na S, odnosno funkcija koja elemente skupa S preslikava na skup {true, false},*
- *f: $S \to \mathbb{R}$ je funkcija koja svakom elementu x skupa S pridružuje neku težinu f(x).*

Funkciju f nazivamo ciljnom funkcijom problema. Predikat p generira skup $S_a = \{x \in S : P(x) = true\}$ koji nazivamo skupom dopustivih rješenja problema. Svaki element skupa S naziva se rješenje problema, a svaki element skupa S_a dopustivo rješenje problema.

Rješavanje kombinatornog optimizacijskog problema svodi se na traženje elementa $\tilde{x} \in S_a$ ili podskupa skupa S_a koji minimizira ciljnu funkciju f:

$$f(\tilde{x}) = min_{x \in S_a} f(x). \tag{1.6}$$

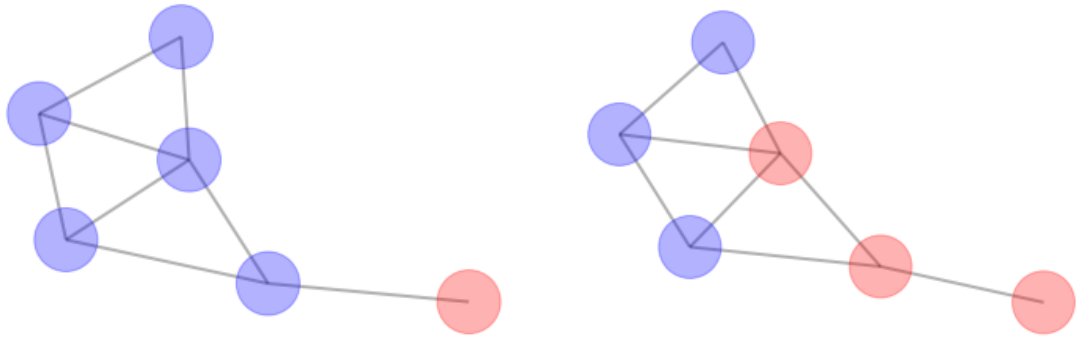
U ovom su poglavlju dane osnovne definicije i oznake koje će biti potrebne u nastavku rada. Prije nego što prijeđemo na algoritme, potrebno je pobliže opisati problem particioniranja grafova i s njime povezane pojmove, poput ravnoteže, reza i drugih.

Poglavlje 2

Problem particioniranja grafa

U definiciji 1.2.1 dana je definicija particije nekog skupa. No, u definiciji 1.1.1 graf smo definirali kao par skupa vrhova i skupa bridova pa nije odmah jasno na koji se od tih skupova problem particioniranja grafa odnosi. U ovom ćemo radu promatrati particioniranje skupa vrhova.

Particija skupa vrhova grafa definira se kao u definiciji 1.2.1, uz $X = V$. Neka je $P_k = \{V_1, ..., V_k\}$ neka takva particija skupa V. Broj k je kardinalnost particije i u ostatku rada ćemo, ukoliko tražimo particiju kardinalnosti k, govoriti o k-particioniranju. Skupove $V_i, i \in I$, zovemo dijelovima particije.



Slika 2.1: Prikazane su dvije različite particije istog grafa. Vrhovi koji pripadaju istom dijelu particije označeni su istom bojom.

2.1 Ravnoteža particije

Bellov broj za $n = 6$ iznosi 203 (vidi (1.3)). Dakle, graf na slici 2.1 mogli smo particionirati na 203 načina. Prikazan je primjer particije u kojoj je jedan vrh odvojen te primjer u kojem oba dijela particije imaju jednak broj vrhova. Kako je smisao particioniranja grafa često baš

one vrhove koji su međusobno povezaniji svrstati u iste skupove, to bi i situacija s prvog dijela slike mogla biti česta pojava. Naime, odvojeni je vrh povezan samo još s jednim vrhom dok ostali imaju barem dva susjeda. Jedna je od primjena particioniranja grafova istraživanje društvenih mreža pa uzmimo sličan primjer. Vikendom se često održavaju sportska natjecanja pa se roditelji angažiraju i svojim osobnim automobilima prevoze grupu djece, npr. neku kadetsku nogometnu ekipu. Kao i u svakoj skupini, neka su djeca više međusobno povezana, neka manje. Želimo ih grupirati po automobilima tako da bolji prijatelji budu zajedno, no ne želimo da neki automobil bude pretrpan, a neki drugi gotovo prazan. Tada algoritmu za particioniranje grafova, osim toga da zajedno stavi onu djecu koja su međusobno u boljim odnosima, dajemo i uvjet da grupe budu podjednako velike (i, u slučaju podjele po automobilima, ne veće od četiri elementa). Opisano se u problemu particioniranja grafova ostvaruje pojmom ravnoteže particije.

Definicija 2.1.1. *(Ravnoteža particije) Neka je $G = (V, E)$ graf i $P_k = \{V_1, ..., V_k\}$ k-particija grafa G. Prosječna težina dijela pri k-particiji P_k od V jest*

$$w_{avg} = \left\lceil \frac{w(V)}{k} \right\rceil.$$

Ravnoteža $bal(P_k)$ particije P_k jest kvocijent dijela particije P_k najveće težine i prosječne težine particije:

$$bal(P_k) = \frac{\max_i w(V_i)}{w_{avg}}. \tag{2.1}$$

Particija je savršeno uravnotežena kada joj je ravnoteža jednaka 1. Ranije smo rekli kako ćemo uzimati da su težine vrhova grafa jednake jedan pa je vrijednost $w(V)$ zapravo jednaka kardinalitetu skupa V, $|V|$. Dakle, $w_{avg} = \lceil \frac{|V|}{k} \rceil$. Ukoliko je $|V|$ djeljivo s k, vrijednost ravnoteže k-particije bit će 1 ako i samo ako su svi dijelovi particije težine $\frac{|V|}{k}$. Ako pak pri dijeljenju $|V|$ s k dobivamo ostatak koji nije nula, vrijednost ravnoteže 1 postiže se ako $|V|$ *mod* k dijelova ima $\lceil \frac{|V|}{k} \rceil$ elemenata, a ostali jedan manje. Znači, pojmovi prosječne težine i ravnoteže particije pomažu pri konstrukciji k-particije s dijelovima gotovo identičnih težina/kardinaliteta.

2.2 Funkcije particioniranja

Prema [2], ciljne funkcije particioniranja grafa opisuju razlike između dijelova particije ili sličnosti unutar dijelova particije. U težinskom će grafu veća sličnost dvaju vrhova biti označena većom težinom brida između njih.

Definicija 2.2.1. *(Funkcija reza) Neka je $G = (V, E)$ graf i V_a i V_b neka dva podskupa od V. Definiramo rez između skupova V_a i V_b:*

$$rez(V_a, V_b) = \sum_{u \in V_a} \sum_{v \in V_b} w(u, v). \tag{2.2}$$

Neka je $P_k = \{V_1, ..., V_k\}$ neka particija skupa V na k dijelova. Najjednostavnija funkcija u problemima particioniranja grafova jest rez particije, koji označava ukupnu težinu bridova koji spajaju vrhove različitih dijelova particije P_k.

Definicija 2.2.2. *(Rez particije)*

$$\begin{aligned} rez(P_k) &= \sum_{i<j} rez(V_i, V_j) \\ &= \frac{1}{2} \sum_{i=1}^{k} rez(V_i, V \setminus V_i) \end{aligned} \tag{2.3}$$

Funkcija koja opisuje omjer između reza i težine za svaki dio particije zove se razmjerni rez. U literaturi postoji više definicija ove funkcije, no ovdje se držimo definicije kao u [2].

Definicija 2.2.3. *(Razmjerni rez particije)*

$$Rrez(P_k) = \sum_{i=1}^{k} \frac{rez(V_i, V \setminus V_i)}{w(V_i)} \tag{2.4}$$

Još jedna funkcija koja se često pojavljuje u particioniranju grafova jest normalizirani rez, koji za svaki dio particije opisuje omjer između reza i sume težina svih bridova s kojima je incidentan barem jedan vrh iz tog dijela. Drugim riječima, opisuje omjer između težine bridova koji povezuju vrh iz dijela particije s vrhom koji nije u tom dijelu i svih bridova koji izlaze iz vrhova tog dijela. Ako su vrhovi razvrstani po particiji po sličnosti, onda je u svakom dijelu više onih bridova čija su oba vrha u tom dijelu, a manje je onih bridova koji spajaju različite dijelove particije. Dakle, normalizirani rez je manji.

Definicija 2.2.4. *(Normalizirani rez particije)*

$$\begin{aligned} Nrez(P_k) &= \sum_{i=1}^{k} \frac{rez(V_i, V \setminus V_i)}{rez(V_i, V)} \\ &= \sum_{i=1}^{k} \left(1 - \frac{rez(V_i, V_i)}{rez(V_i, V)}\right) \end{aligned} \tag{2.5}$$

Napomena: Ako su težine u grafu pozitivne, što je najčešći slučaj u problemima particioniranja grafova, onda su i gornje funkcije pozitivne.

2.3 Formalna definicija i podjela

Ranije je definiran kombinatorni optimizacijski problem (definicija 1.2.2). U ovom ćemo odjeljku formalno definirati problem particioniranja grafa kao kombinatorni optimizacijski problem i uvest ćemo podjelu s obzirom na to postoji li uvjet na ravnotežu particije ili ne postoji. Osvrnut ćemo se i na ranije definirane funkcije reza, razmjernog reza i normaliziranog reza u različitim vrstama problema.

Definicija 2.3.1. *(Generalizirani optimizacijski problem k-particioniranja grafa)*
Neka je $G = (V, E)$ *graf i* $k \in \mathbb{N}$, $k \geq 2$. *Definiramo trojku* (S, p, f) *koja karakterizira generalizirani optimizacijski problem k-particioniranja grafa:*

- *prostor rješenja S je skup svih mogućih particija od V, kardinaliteta od 1 do* $|V|$,
- *neka je* p' *predikat takav da za* $P \in S$ *vrijedi* $p'(P) = true$ *ako i samo ako* $|P| = k$; *predikat p generaliziranog problema definiramo tako da* $p(P) = true \implies p'(P) = true$,
- $f : S \to \mathbb{R}$ *je ciljna funkcija.*

Skup dopustivih rješenja problema definiran je sljedećim:

$$S_a = \{P \in S : p(P) = true\}.$$

Generalizirani optimizacijski problem svodi se na pronalazak particije $\tilde{P}_k \in S_a$ *koja minimizira ciljnu funkciju f:*

$$f(\tilde{P}_k) = min_{P_k \in S_a} f(P_k). \tag{2.6}$$

Primijetimo da se u definiciji spominju dva predikata. Predikat p' zahtijeva samo da je kardinalitet particije u skupu dopustivih rješenja jednak k. Predikat p može biti restriktivniji. Upravo se to događa kada uvedemo uvjet na ravnotežu particije. Predikat p' napravit će skup k-particija, a predikat p iz tog će skupa izbaciti sve one particije koje ne poštuju uvjet ravnoteže. Tako se dobiva skup dopustivih rješenja na kojemu se onda traži minimum neke ciljne funkcije, npr. reza. Kao što je već napomenuto na početku ovog odjeljka, postojanje/nepostojanje uvjeta na ravnotežu probleme particioniranja grafa dijeli na dvije velike skupine.

Problem particioniranja grafa s uvjetom na ravnotežu

Ravnoteža particije definirana je u definiciji 2.1.1. Da bi neka k-particija bila rješenje ovakvog problema, nije dovoljno samo da minimizira ciljnu funkciju, već mora poštivati dani uvjet na ravnotežu. Najčešće je uvjet dan kao maksimalna dozvoljena ravnoteža i, prema [2], iznosi između 1.00 i 1.05. Naime, ukoliko nije nužno da je particija doista savršeno uravnotežena, nekad se malim odstupanjem može dobiti poboljšanje na minimizaciji ciljne funkcije.

Što se ciljnih funkcija tiče, u ovom se tipu problema najčešće minimizira funkcija reza. U [2] je pokazano da je za probleme u kojima su dijelovi particije jednakih kardinaliteta rez jednak razmjernom rezu, do na multiplikativnu konstantu. Dakle, minimizacija reza u ovom je slučaju ekvivalentna minimizaciji razmjernog reza. Normalizirani rez se, opet prema [2], u ovim problemima ne susreće. Naime, funkcija normaliziranog reza izolira međusobno povezane vrhove od ostalih, što je često u kontradikciji s nalaženjem jednakih dijelova. Slijedi formalna definicija problema.

Definicija 2.3.2. *(Particioniranje grafa s uvjetom na ravnotežu)*
Neka je $G = (V, E)$ graf, k kardinalitet particije i bal_max maksimalna ravnoteža particije. Neka je S prostor rješenja definiran generaliziranim problemom particioniranja. Skup dopustivih rješenja S_a dan je sljedećim:

$$S_a = \{P_i \in S : |P_i| = k \; i \; bal(P_i) \leq bal\_max\}.$$

Problem particioniranja grafa s uvjetom na ravnotežu svodi se na pronalaženje $\tilde{x} \in S_a$ koji minimizira ciljnu funkciju f:

$$f(\tilde{x}) = min_{x \in S_a} f(x). \tag{2.7}$$

Problem particioniranja grafa bez uvjeta na ravnotežu

Definicija 2.3.3. *(Particioniranje grafa bez uvjeta na ravnotežu)*
Neka je $G = (V, E)$ graf i k kardinalitet particije. Neka je S prostor rješenja definiran generaliziranim problemom particioniranja. Skup dopustivih rješenja S_a dan je sljedećim:

$$S_a = \{P_i \in S : |P_i| = k\}.$$

Problem particioniranja grafa bez uvjeta na ravnotežu svodi se na pronalaženje particije $\tilde{x} \in S_a$ koja minimizira ciljnu funkciju f:

$$f(\tilde{x}) = min_{x \in S_a} f(x). \tag{2.8}$$

Slijedi nekoliko primjera primjene particioniranja grafa na različitim područjima.

2.4 Primjene particioniranja grafova

Primjer 1. Segmentiranje slika

Segmentiranje slike jest podjela slike na neki konačan broj regija. Regije su obično različiti objekti na slici ili pozadina i ostatak slike. Cilj segmentiranja je najčešće izdvajanje nekog objekta od ostatka slike. Slike možemo promatrati kao neusmjerene grafove kojima su pikseli čvorovi, a bridovi povezuju svaki piksel s njemu susjednim pikselima. Segmentiranje slike moguće je provesti particioniranjem grafa piksela. Svaki je piksel opisan vektorom koji sadrži tri vrijednosti: razinu crvene, razinu zelene te razinu plave boje (RGB vrijednosti piksela). Tako se za svaki par vrhova može izračunati sličnost po bojama. Pikseli koji tvore isti objekt vjerojatno će biti slični po bojama. Također, za svaki se piksel iz RGB vrijednosti može izračunati razina sive boje. Nadalje, pikseli koji pripadaju istom objektu uglavnom će biti na susjednim mjestima. Particioniranjem grafa piksela slični se pikseli, dakle oni koji vjerojatno pripadaju istom objektu, svrstavaju u isti dio particije. Težina bridova treba označiti sličnost među pikselima. Jedna je mogućnost za dodjelu težina bridovima funkcija predložena u [2] sljedeće:

$$w_{ij} = \left(\frac{|p_i - p_j|}{dist(i,j)} + \beta \right)^{\alpha}, \forall e \in E, \tag{2.9}$$

gdje su $\alpha, \beta \in \mathbb{N}$ neki parametri, p_i i p_j su razine sive dvaju susjednih vrhova, odnosno piksela, a $dist(i,j)$ je neka mjera udaljenosti među pikselima, npr. euklidska udaljenost. Kako se radi o particioniranju bez uvjeta na ravnotežu (ne možemo garantirati da npr. objekt na slici zauzima jednak broj piksela kao pozadina), agoritam particioniranja minimizira jednu od triju opisanih ciljnih funkcija. Najčešće je to funkcija normaliziranog reza, koja je, prema [2] i kreirana kako bi u segmentiranju slika particioniranjem grafova zamijenila razmjerni rez.

Primjer 2. Izdvajanje interesnih skupina na temelju komentara na forumu

U [1] je opisana primjena particioniranja grafova na izdvajanje skupina na temelju komentara na forumu. Komunikacija se na forumu odvija tako da se na nečiju poruku odgovara referencom na njegovu poruku i zatim slijedi tekst odgovora. Tako su autori komentara povezani u graf u kojem su oni čvorovi, a brid između njih označava relaciju „odgovoriti na poruku". Dakle, u ovom se slučaju radi o usmjerenom grafu. Karakteristika skupina od koje autori rada polaze jest da će osoba češće odgovoriti na nečije mišljenje ako se s njime ne slaže. Zato ovdje imamo malo drugačiju situaciju nego dosad jer zapravo želimo razdvojiti vrhove među kojima postoji brid, što znači da želimo da rez bude što veći. Također, nemamo uvjet na ravnotežu particije jer ne možemo unaprijed reći da će polovina autora biti jednog, a polovina drugog mišljenja. Particioniranje grafova tako će dati podjelu autora na skupine *za* i *protiv* po temama abortusa, kontrole oružja i imigracija. Kako su poruke na

forumima često kratke i iz ključnih riječi nije lako odrediti radi li se o slaganju ili protivljenju jer su ključne riječi uglavnom slične, ovaj je pristup dao bolje rezultate nego analiza metodama za obradu teksta.

Primjer 3. Planiranje ruta

Particioniranje grafa u geografskom smislu općenito može poslužiti za podjelu velike i kompliciranе geografske mreže na manje cjeline koje je lakše analizirati. Planiranje ruta problem je s kojim se svakodnevno susreću brojne kompanije koje moraju prevesti neki teret na različita odredišta. U interesu im je teret prevesti u što kraćem roku sa što manjim troškom. Ako odredišta promatramo kao čvorove, a puteve između njih kao bridove, cilj je pronaći optimalan redoslijed dostave tereta. Bridovi dobivaju težine koje ne označavaju samo udaljenosti između vrhova, već i neke druge faktore, poput npr. kvalitete ceste, postojanje benzinske pumpe ili nešto drugo što je kompaniji važno u njezinom poslovanju. Particioniranjem takvog grafa i minimizacijom reza između particija utvrđuje se koja su područja bolje povezana u opisanom smislu te se to dalje koristi u planiranju realizacije dostave. Ukoliko kompanija ima više vozila, particioniranje grafa odredišta može i dati područje dostave za svako od tih vozila. U prvom slučaju nema uvjeta na ravnotežu, cilj je dobiti što kvalitetnije particije, bez obzira na broj odredišta u dijelovima. U drugom slučaju uvjet može postojati. Neka kompanija može zahtijevati da svaki vozač preveze jednak teret.

Na kraju ovog poglavlja napomenimo još da je problem particioniranja grafa tipično NP-težak. Obrazloženje se može naći u [2, poglavlje 1, sekcija 1.10] ili u [4, poglavlje, sekcija 2.1]. Vjeruje se da za rješavanje problema iz te klase ne postoje polinomijalni algoritmi pa im se pristupa heurističkim i aproksimacijskim algoritmima.

Poglavlje 3

Spektralno particioniranje grafa

U ovom poglavlju opisujemo metodu spektralnog particioniranja grafova. Prvo definiramo pojmove iz linearne algebre potrebne za razumijevanje metode te navodimo nekoliko bitnih svojstava ranije definiranih matrica. Zatim na jednostavnom primjeru objašnjavamo osnovnu ideju. Nastavljamo s biparticioniranjem i k-particioniranjem s uvjetom na ravnotežu particije. Na kraju dajemo pseudokod za opisani algoritam.

3.1 Osnovni pojmovi

Definicija 3.1.1. *(Svojstvena vrijednost i svojstveni vektor)*
Svojstveni vektor kvadratne matrice $M \in \mathbb{R}^{n\times n}$ jest svaki vektor $v \in \mathbb{R}^n$, $v \neq \overrightarrow{0}$, za koji postoji skalar $\lambda \in \mathbb{R}$ takav da vrijedi

$$Mv = \lambda v.$$

Pripadni skalar λ zovemo svojstvena vrijednost matrice M. Skup svih svojstvenih vrijednosti λ matrice M zovemo spektar od M.

Definicija 3.1.2. *(Pozitivno semidefinitna matrica)*
Za realnu simetričnu matricu M reda n kažemo da je pozitivno semidefinitna ako zadovoljava jedno od sljedećih (ekvivalentnih) svojstava:

1. *za svaki vektor-stupac $v \in \mathbb{R}^n, v \neq \overrightarrow{0}$, vrijedi $v^T Mv \geqslant 0$,*
2. *sve su svojstvene vrijednosti matrice M nenegativne.*

Propozicija 3.1.3. *Laplaceova matrica L neusmjerenog težinskog grafa $G = (V, E)$ s pozitivnim težinama bridova pozitivno je semidefinitna.*

Dokaz. (kao u [2])

Jednostavno je vidjeti da je Laplaceova matrica neusmjerenog grafa simetrična. Naime, vrijedi $L_{ij} = -w_{ij} = -w_{ji} = L_{ji}$, za $i \neq j$. Uzmimo sada neki vektor $x \in \mathbb{R}^n \setminus \{\overrightarrow{0}\}$, gdje je $n = |V|$. Za svaki $i \in \{1, ..., n\}$ vrijedi:

$$\begin{aligned}(x^T L)_i &= x_i deg(i) - \sum_{j=1}^{n} x_j w_{ij} \\ &= x_i \sum_{j=1}^{n} w_{ij} - \sum_{j=1}^{n} x_j w_{ij} \\ &= \sum_{j=1}^{n} (x_i - x_j) w_{ij}.\end{aligned} \tag{3.1}$$

Dakle,

$$x^T L x = \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i^2 - x_i x_j) w_{ij}. \tag{3.2}$$

Zbog neusmjerenosti grafa vrijedi $w_{ij} = w_{ji}$ te u sumi za svaki par indeksa (i, j) imamo sljedeće:

$$(x_i^2 - x_i x_j)\, w_{ij} + (x_j^2 - x_j x_i)\, w_{ji} = (x_i - x_j)^2\, w_{ij}. \tag{3.3}$$

Budući da je $w_{ij} = 0$ ukoliko $(v_i, v_j) \notin E$, izraz u (3.2) možemo pisati ovako:

$$x^T L x = \sum_{(v_i, v_j) \in E} (x_i - x_j)^2 w_{ij}. \tag{3.4}$$

Kako su težine bridova pozitivne, to je i gornji izraz nenegativan pa je matrica L pozitivno semidefinitna. □

Propozicija 3.1.4. *Najmanja svojstvena vrijednost Laplaceove matrice jest $\lambda_1 = 0$, a pripadajući je svojstveni vektor $\mathbb{1} = [1, ..., 1]^T$.*

Dokaz. Iz prethodne propozicije znamo da je Laplaceova matrica pozitivno semidefinitna, a tada, iz definicije pozitivno semidefinitnih matrica, slijedi da su joj sve svojstvene vrijednosti nenegativne.

Nadalje, u i-tom retku na dijagonali Laplaceove matrice nalazi se stupanj i-tog vrha grafa, a na ostalim mjestima istog retka nalaze se težine bridova iz i-tog vrha pomnožene s -1. Kako je stupanj vrha zapravo zbroj svih težina bridova koji izlaze iz tog vrha, to je zbroj elemenata u retku jednak 0. No, zbrajanje svih elemenata retka jednako je množenju retka s vektorom $\mathbb{1}$. Dakle, vrijedi:

$$L\mathbb{1} = 0 = \lambda_1 \mathbb{1}.$$

□

Sada dajemo teorem na kojemu se temelji algoritam spektralnog particioniranja grafa i po kojem je i dobio ime. Realna simetrična matrica u teoremu je Laplaceova matrica, a rastav koji je dan teoremom kasnije koristimo u povezivanju funkcija particioniranja grafa sa spektrom Laplaceove matrice. Na taj način problem particioniranja grafa svodimo na rješavanje odgovarajućeg generaliziranog svojstvenog problema.

Teorem 3.1.5. *(Spektralni teorem)*
Matrica $M \in \mathbb{R}^{n\times n}$ je realna simetrična ako i samo ako postoje realna ortogonalna matrica $U \in M_n$ i realna dijagonalna matrica $\Lambda \in M_n$ takve da je $M = U\Lambda U^T$.

Sljedeću definiciju samo navodimo jer ćemo definirani pojam koristiti pri minimizaciji funkcije normaliziranog reza u particioniranju bez uvjeta na ravnotežu.

Definicija 3.1.6. *(Težinska normalizirana Laplaceova matrica)*
Neka je D matrica stupnjeva. Težinska normalizirana Laplaceova L_{norm} matrica definira se kao $L_{norm} = D^{\frac{-1}{2}} L D^{\frac{-1}{2}}$, odnosno:

$$(l_{norm})_{ij} = \begin{cases} 1 - \frac{w_{jj}}{d_j}, & za\ i = j, \\ -\frac{w_{ij}}{\sqrt{d_i d_j}}, & za\ i \neq j. \end{cases}$$

Primijetimo da je i matrica L_{norm} simetrična.

3.2 Osnovna ideja algoritma spektralnog particioniranja

U prošlom su odjeljku definirane matrice koje sadrže informacije o grafu. Sada svojstva tih matrica želimo iskoristiti za particioniranje istog. Za prikaz ideje spektralnog particioniranja iskoristit ćemo jedan od jednostavnijih slučajeva, biparticioniranje s ciljnom funkcijom reza. Neka je, dakle, $G = (V, E)$ graf sa skupom vrhova V i skupom bridova E. Želimo ga podijeliti na dva dijela V_1 i V_2, i to tako da rez među njima bude što je moguće manji.

Neka je $x \in \mathbb{R}^n$, gdje je $n = |V|$, takav da za svaki $v_i \in V$ vrijedi:

$$x_i = \begin{cases} 1, & \text{ako } v_i \in V_1, \\ -1, & \text{ako } v_i \in V_2. \end{cases} \tag{3.5}$$

Također, vrijedi:

$$rez(V_1, V_2) = \sum_{v_i \in V_1, v_j \in V_2} w_{ij} = \frac{1}{2} \sum_{(v_i, v_j) \in E} (x_i - x_j)^2 w_{ij}. \tag{3.6}$$

Sada iz (3.4) slijedi:

$$rez(V_1, V_2) = \frac{1}{2} x^T L x. \tag{3.7}$$

Dakle, minimizacija reza biparticije može se postići pronalaženjem vektora x s elementima iz skupa $\{-1, 1\}$ koji minimizira izraz $x^T L x$. Matrica L ima n nenegativnih realnih svojstvenih vrijednosti, a najmanja je, po propoziciji 3.1.4, jednaka 0. Dakle, svojstvene vrijednosti matrice L možemo sortirati i zapisati ovako:

$$0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n. \tag{3.8}$$

Po spektralnom teoremu postoje ortogonalna matrica $U \in \mathbb{R}^{n \times n}$ i dijagonalna matrica $\Lambda \in \mathbb{R}^{n \times n}, \Lambda = diag(\lambda_1, ..., \lambda_n)$ takve da je $L = U \Lambda U^T$. Sada za svaki $x \in \mathbb{R}^n$ vrijedi:

$$x^T L x = x^T (U \Lambda U^T) x = (U^T x)^T \Lambda (U^T x) = \sum_{i=1}^{n} \lambda_i |(U^T x)_i|^2. \tag{3.9}$$

Budući da je svaki $|(U^T x)_i|^2$ nenegativan, slijedi:

$$\lambda_1 \sum_{i=1}^{n} |(U^T x)_i|^2 \leq x^T L x = \sum_{i=1}^{n} \lambda_i |(U^T x)_i|^2. \tag{3.10}$$

Dakle, trivijalni svojstveni vektor minimizira traženi izraz (jer je tada $Lx = \lambda_1 x = 0$ pa je i $x^T L x = 0$), no ne daje zadovoljavajuće rješenje. Međutim, kako je $\lambda_1 = 0$, jednakost s desne strane izraza (3.10) možemo zapisati i ovako:

$$x^T L x = \sum_{i=2}^{n} \lambda_i |(U^T x)_i|^2 \tag{3.11}$$

pa vrijedi

$$\lambda_2 \sum_{i=2}^{n} |(U^T x)_i|^2 \leq \sum_{i=2}^{n} \lambda_i |(U^T x)_i|^2 = x^T L x. \tag{3.12}$$

Ovime je početni problem sveden na problem nalaženja drugog najmanjeg svojstvenog vektora Laplaceove matrice grafa, poznatog i pod nazivom drugi Fiedlerov vektor. Elementi tog vektora općenito ne poprimaju vrijednosti iz skupa $\{-1, 1\}$. Stoga particiju određujemo pronalaženjem medijana elemenata Fiedlerovog vektora te njihovom podjelom na one koji su manji i one koji su veći od medijana. Na primjer, uzmimo Fiedlerov vektor x nekog grafa G sa šest vrhova takav da je:

$$x^T = [-0.14 \quad -0.15 \quad -0.4 \quad -0.06 \quad 0.87 \quad -0.08].$$

Medijan ovog vektora je -0.11 pa ćemo x, odnosno x^T diskretizirati na skup $\{-1, 1\}$ tako da na one elemente na kojima je vrijednost manja od -0.11 stavimo -1, a na ostala 1. Dobivamo sljedeće:

$$x^T = [-1 \ -1 \ -1 \ 1 \ 1 \ 1].$$

Ako je, dakle, skup vrhova grafa G $V = \{1, 2, 3, 4, 5, 6\}$, prva će tri vrha pripasti jednom dijelu particije (jer je na prva tri mjesta u diskretiziranom Fiedlerovom vektoru vrijednost -1), a druga će tri vrha pripasti drugom dijelu particije.

3.3 Spektralno biparticioniranje s uvjetom na ravnotežu

U slučaju da je pri particioniranju zadan uvjet na ravnotežu, postupak je vrlo sličan onome iz prošlog odjeljka. Razlika je samo u diskretizaciji vektora x. Po [2], dovoljno je uzlazno sortirati vrijednosti u x i odabrati granični indeks i tako da vrijedi: $x_i < 0$ i dio particije postiže maksimalnu težinu dozvoljenu uvjetom na ravnotežu ili $x_i > 0$ i dio particije postiže minimalnu težinu dozvoljenu uvjetom na ravnotežu. Dakle, ako je zadano da vrijednost ravnoteže particije ne smije biti veća od 1.05, iz izraza 2.1.1 možemo dobiti maksimalnu dozvoljenu veličinu dijela particije. Vektor x sortiramo uzlazno i prvom dijelu particije dodajemo vrhove po redu, onako kako su sortirani po pripadajućim vrijednostima u vektoru x, i tako sve dok ne postignemo maksimalnu težinu, tj. kardinalitet takav da je zadani uvjet na ravnotežu zadovoljen.

3.4 Spektralno k-particioniranje s uvjetom na ravnotežu

Prema [2] postupak spektralnog biparticioniranja moguće je generalizirati i na 2^i- particioniranje grafa G. Za pronalaženje 2^i-particije potrebno je prvih $i + 1$ Fiedlerovih vektora. Svaki se od njih diskretizira na skup $\{-1, 1\}$, kao i u slučaju biparticioniranja, i iz takvih se diskretiziranih vektora stvaraju odgovarajuće bisekcije. Međutim, kako nove bisekcije trebaju čuvati ravnotežu, u praksi se, opet prema [2] pokazalo da je postupak moguć za particioniranje na 2, 4 ili 8 dijelova. Za više od toga najčešće se koristi tehnika rekurzivnog biparticioniranja.

3.5 Spektralno particioniranje bez uvjeta na ravnotežu

Ukoliko rješavamo problem particioniranja grafa bez uvjeta na ravnotežu, prema [2] treba razlikovati dva slučaja: ciljna je funkcija normalizirani rez i ciljna je funkcija razmjerni rez.

Normalizirani rez

U [3] je pokazano kako je za nalaženje minimuma u slučaju normaliziranog reza potrebno riješiti spektralni problem težinske normalizirane Laplaceove matrice L_{norm} definirane u definiciji 3.1.6, koji glasi ovako:

$$L_{norm}z = \lambda z.$$

Nadalje, pokazano je kako iz gornje formulacije, uz $y = D^{\frac{-1}{2}}z$, slijedi $D^{-1}Ly = \lambda y$. Tako dobivamo generalizirani spektralni problem za Laplaceovu matricu L:

$$(D - A)y = \lambda Dy, y = D^{\frac{-1}{2}}z. \tag{3.13}$$

I u ovom se slučaju minimizacija postiže drugim Fiedlerovim vektorom. Budući da nema uvjeta na ravnotežu, projekcija x na bisekciju $P_2 = \{V_1, V_2\}$, prema [2], vrši se na sljedeći način: ako je x_i negativan, vrh v_i stavljamo u prvi dio V_1, inače u V_2.

Razmjerni rez

Ako je ciljna funkcija razmjerni rez, minimizacijski se proces sastoji od rješavanja sljedećeg generaliziranog svojstvenog problema:

$$Lx = \lambda Ax. \tag{3.14}$$

Ostatak procesa izvršava se analogno slučaju normaliziranog reza.

3.6 Algoritam spektralnog particioniranja

Slijedi pseudokod za algoritam spektralnog biparticioniranja. Ukoliko je traženi broj particija veći od dva, problem možemo riješiti rekurzivnim biparticioniranjem.

Algoritam spektralnog biparticioniranja

Ulaz: graf $G = (V, E)$
Izlaz: biparticija (V_1, V_2)

1: Generiraj matrice D i A grafa G.
2: Iz matrica D i A generiraj Laplaceovu matricu L grafa G.
3: Riješi odgovarajući generalizirani svojstveni problem i izračunaj drugi Fiedlerov vektor x.
4: Izračunaj medijan vektora dobivenog u prošlom koraku.
5: Diskretiziraj vektor x na skup $\{-1, 1\}$ s obzirom na postojanje uvjeta na ravnotežu.
6: Za sve $i \in \{1, ..., n\}$, ukoliko je $x_i > 0$, vrh v_i svrstaj u V_1, inače u V_2.

Poglavlje 4

Višerazinsko particioniranje

Višerazinsko particioniranje (engl. *multilevel method*) jedna je od najkorištenijih metoda za rješavanje problema particioniranja grafa, posebno s uvjetom na ravnotežu. Naziv je dobila po tome što se provodi u tri razine: sažimanje (engl. *coarsening*), particioniranje i profinjavanje (engl. *refinement*). U nastavku ćemo detaljnije opisati svaku od navedenih razina. Također, prikazat ćemo neke algoritme unutar razina ove metode. Mnogi od njih mogu se koristiti i samostalno.

Glavna ideja višerazinske metode jest brže particioniranje velikih grafova, tj. grafova s velikim brojem vrhova. Umjesto da se prolazi po svakom vrhu i da se za svaki vrh provodi neki račun, u početku se skup vrhova sažme do željenog broja, tada se particionira takav sažeti graf i na kraju se vrhovi opet prošire do početnog oblika. Neka je $G = (V, E)$ graf.

Sažimanje

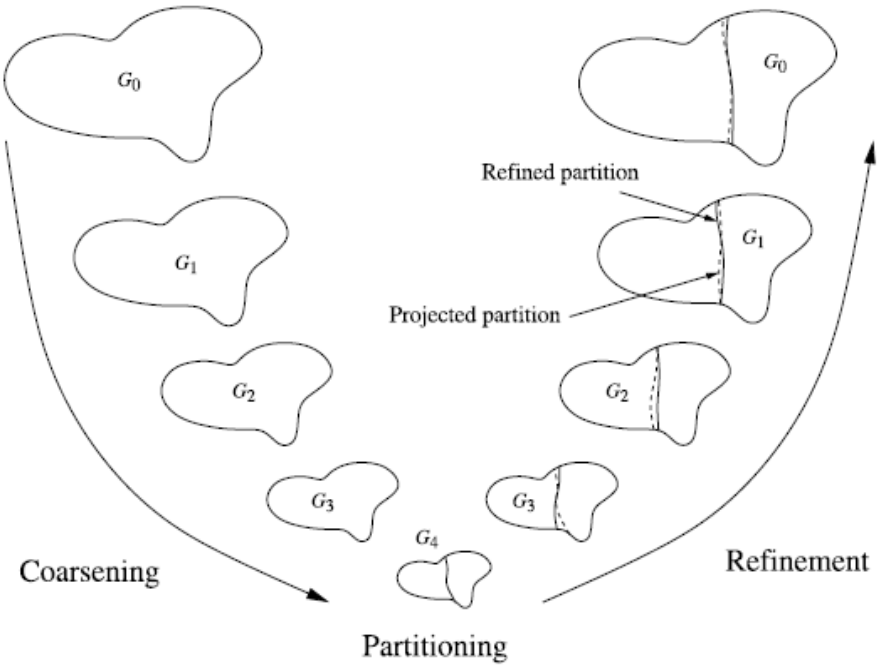
Sažimanje se provodi iterativno, i to tako da se u svakoj iteraciji od vrhova grafa iz prethodne iteracije formiraju grupe. Te su grupe vrhova onda vrhovi grafa nove iteracije. Dakle, u ovoj se razini generira niz grafova $\{G_1, ..., G_n\}$, $G_1 = G$, tako da u svakom grafu G_{i+1} svaki vrh predstavlja grupu vrhova prethodnog grafa G_i. Iteracije staju kada generirani graf ima zadovoljavajuće mali broj vrhova ili kad razlika broja vrhova grafa iz prethodne iteracije i novog grafa postane premala. Bitno je napomenuti da se vrhovi ne grupiraju nasumično, nego se nastoji što je moguće više zadržati početni oblik grafa.

Particioniranje

U fazi particioniranja se, koristeći neku heuristiku (kasnije dajemo nekoliko primjera), kreira particija P_k^n grafa G_n na k dijelova.

Projekcija i profinjavanje

U završnoj fazi višerazinske metode potrebno je dobivenu particiju P_k^n projicirati na početni graf G. Direktno proširenje na početni skup vrhova uglavnom ne daje najbolje rezultate, nego je potrebno nakon svakog proširenja grupe vrhova na vrhove grafa prijašnje iteracije profiniti particiju, zamijeniti neke vrhove u dijelovima particije. To nisu velike promjene, particija će otprilike zadržati svoj oblik, samo ne identičan, već poboljšan. Dakle, particija P_k^n grafa G_n dobivenog u prvoj fazi projicira se na graf G_{n-1}. Zatim se profinjavanjem dobiva particija P_k^{n-1} grafa G_{n-1}. Proces se ponavlja do zadnjeg profinjenja, kojim se dobiva završno rješenje, particija P_k^1 grafa $G_1 = G$. Završna je particija dobro rješenje i globalno i lokalno.



Slika 4.1: Prikaz triju faza višerazinske metode ([2]).

Prije upuštanja u detaljno opisivanje faza ove metode, dajemo pseudokod za algoritam particionirajna grafa njome. Algoritam je općenit i nije navedena konkretna metoda ni za jednu od faza te treba uzeti u obzir da će učinkovitost uvelike ovisiti o njihovom izboru.

Algoritam particioniranja grafa višerazinskom metodom

Ulaz: graf $G = (V, E)$, broj particija k
Izlaz: particija P_k^1 grafa G

1: $G_1 = G$
2: $i = 1$
3: **ponavljaj**
4: $\quad i = i + 1$
5: $\quad G_i$ = sažimanje(G_{i-1})
6: **dok** G_{i+1} nije dovoljno malen i broj vrhova se dovoljno razlikuje od prethodne razine
7: $P_k^i = k$-particija od G_i
8: **za** $j = i - 1$ **do** 1
9: $\quad \tilde{P}_k^j$ = projekcija(P_k^{j+1}, G_j)
10: $\quad P_k^j$ = profinjenje($\tilde{P}_k^j$)
11: **vrati** P_k^1

Nastavljamo s detaljnijim opisima pojedinih razina ove metode. Budući da se uglavnom koristi za rješavanje problema s uvjetom na ravnotežu, njezino ćemo djelovanje prikazivati na takvim problemima.

4.1 Sažimanje grafa

Cilj početne razine višerazinske metode jest smanjenje broja vrhova početnog grafa kako bi particioniranje u drugoj fazi bilo što brže i učinkovitije. Od ključne je važnosti da graf sa smanjenim brojem vrhova zadrži globalne karakteristike početnog grafa, inače ne možemo očekivati da particija koja je dobra za jedan graf bude dobra i za drugi.

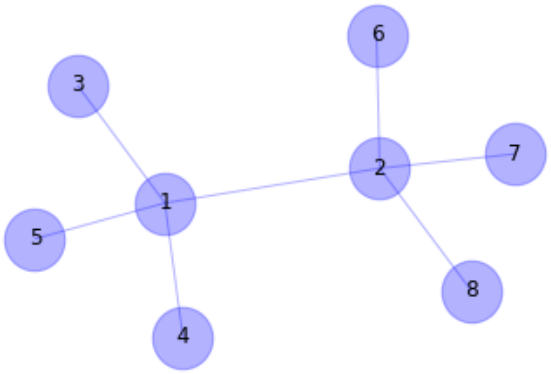
Ranije je navedeno da se se u ovoj fazi generira niz grafova $\{G_1, ..., G_n\}$, uz $G_1 = G$. Vrhovi grafa G_{i+1} dobivaju se grupiranjem vrhova grafa G_i. Ako je v vrh u G_{i+1}, označimo se V_i^v podskup skupa vrhova od G_i čijim je sažimanjem dobiven vrh v. Težina vrha v jednaka je zbroju težina vrhova u V_i^v. Zatim, ako je (v_1, v_2) neki brid u grafu G_{i+1}, tada je njegova težina jednaka zbroju težina svih bridova koji spajaju vrhove iz $V_i^{v_1}$ i $V_i^{v_2}$. Time je omogućeno očuvanje reza i ravnoteže particije u kasnijem povratku prema početnom grafu, tj. u projiciranju particije grafa G_n do particije grafa G_1.

Svrha particioniranja grafova jest podjela grafova na dijelove u kojima su međusobno povezani, slični vrhovi. Zato je logično da ćemo i u sažimanju grafa u iste grupe staviti vrhove među kojima postoje bridovi veće težine. Tako se i olakšava minimiziranje funkcije reza jer nesažeti ostaju bridovi manje težine. Postoji više načina provođenja postupka sažimanja skupa vrhova grafa. Ako je broj vrhova grafa G_i u skupu V_i^v koji tvore vrh v grafa G_{i+1} najviše dva, govorimo o sažimanju sparivanjem. Inače govorimo o sažimanju klaste-

riranjem. U problemima particioniranja grafa s uvjetom na ravnotežu koristi se sparivanje vrhova pa ćemo dalje opisivati algoritme za taj postupak.

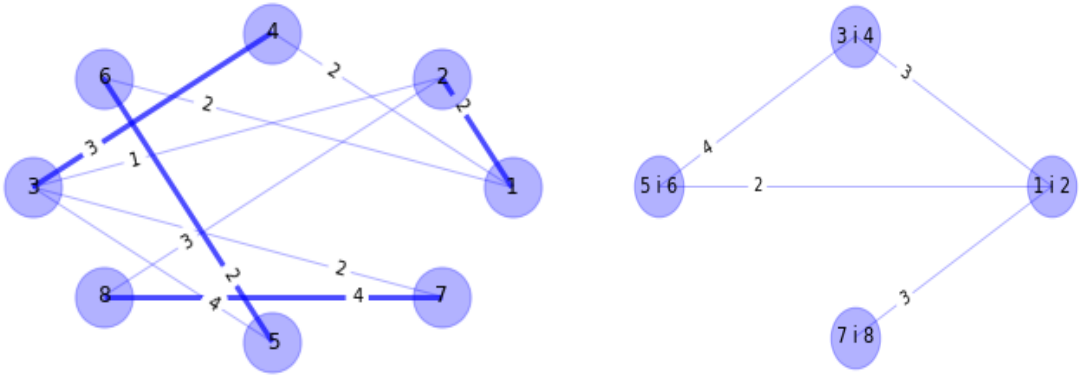
Sažimanje sparivanjem provodi se nalaženjem maksimalnog sparivanja svakog grafa G_i i grupiranjem parova vrhova nađenog sparivanja (vidi definicije 1.1.10, 1.1.11 i 1.1.12). Bichot i Siarry u [2] navode kako bi traženje najvećeg sparivanja bilo vremenski preskupo pa se, iako bi moglo sadržavati više bridova, ne provodi.

Cilj sažimanja jest smanjiti broj vrhova grafa kako bi sljedeća faza bila što brža i efikasnija. Ako je G neki graf i $n = |V|$ broj vrhova grafa G, pronalaženjem maksimalnog sparivanja dobivamo novi graf G', označimo njegov broj vrhova s n'. Vrijedi $n' \geq n/2$. Međutim, broj vrhova u maksimalnom sparivanju ovisi i o povezanosti grafa (prisjetimo se da sparivanje ne uzima susjedne bridove). Pogledajmo graf na sljedećoj slici.



Slika 4.2: Prikaz strukture grafa u kojoj ne možemo dobiti maksimalno sparivanje s upola manje vrhova.

U višerazinskoj metodi krećemo sažimanjem grafa G_1 do željene veličine. Neka je h ta željena veličina. Da bi se konstruirao graf s h vrhova od početnog, potrebno je $O(log_2(|V|/h))$ sažimanja, gdje je $|V|$ broj vrhova grafa G_1. No, u praksi se često događa da se proces sažimanja skrati. Na slici 4.2 pokazano je da nije uvijek moguće upola smanjiti broj vrhova. Dakle, može se dogoditi da maksimalno sparivanje grafa G_i ima puno manje od $|V_i|/2$ vrhova. Tada će i omjer kardinaliteta skupova V_i i V_{i+1} biti puno manji od dva, dakle sažimanjem ne dobivamo veliku razliku. Ako razlike u broju vrhova grafova u različitim iteracijama postanu nedovoljno velike, proces sažimanja se prekida i prelazi se na particioniranje. Ipak, metoda ostaje učinkovita jer će zadnji dobiveni graf G_n uglavnom imati manje vrhova od početnog grafa $G_1 = G$.



Slika 4.3: Na lijevoj je slici prikazan graf s označenim maksimalnim sparivanjem, a na desnoj graf dobiven sažimanjem označenim sparivanjem (težine vrhova na prvoj slici iznose 1, a na drugoj 2).

Hendrickson-Lelandov algoritam

Hendrickson-Lelandov algoritam za sažimanje grafova analogan je jednom od jednostavnijih algoritama za pronalaženje maksimalnog sparivanja grafa. Nasumično se odabere brid (v_1, v_2) grafa G_i i njegovi krajevi formiraju jedan vrh u grafu G_{i+1}. Nakon toga se iz skupa bridova grafa G_i brišu svi bridovi kojima su v_1 ili v_2 krajevi i od njih se formiraju bridovi u G_{i+1}. Postupak se ponavlja, kao i prije, dok broj vrhova u G_{i+1} ne postane zadovoljavajuće mali ili dok sažimanje daje dovoljno dobre rezultate. Složenost ovog algoritma je $O(|E_i|)$. Slijedi pseudokod opisanog algoritma.

Hendrickson-Lelandov algoritam za sažimanje grafa (H-L)

Ulaz: graf $G_i = (V_i, E_i)$
Izlaz: sažeti graf $G_{i+1} = (V_{i+1}, E_{i+1})$

1: $E = E_i$
2: $E_{i+1} = E_i$
3: $V_{i+1} = V_i$
4: **dok** $E \neq \emptyset$
5: nasumično odaberi brid (v_1, v_2) iz E
6: izbriši brid (v_1, v_2) iz E i E_{i+1}
7: izbriši vrhove v_1i v_2 iz V_{i+1}
8: dodaj novi vrh $v = \{v_1, v_2\}$ u V_{i+1}
9: $w(v) = w(v_1) + w(v_2)$
10: prilagodi težine bridova novonastalom vrhu v (ako su v_1 i v_2 imali zajednički susjedni vrh $u \in V_{i+1}$, novi brid ima (v, u) težinu jednaku $w(v, u) = w(v_1, u) + w(v_2, u)$; ako pak u nije zajednički susjedni vrh, već je susjedan samo s npr. v_1, novi brid (v, u) ima težinu jednaku $w(v_1, u)$, analogno za (v_2, u))
11: izbriši sve bridove iz E kojima su v_1 ili v_2 krajevi
12: **vrati** $G_{i+1} = (V_{i+1}, E_{i+1})$

Algoritam sparivanja najtežim bridom

Algoritam sparivanja najtežim bridom (engl. *heavy edge matching algorithm*, skraćeno HEM) još je jedna metoda za sažimanje grafa koja se temelji na traženju maksimalnog sparivanja. Od prošlog se algoritma razlikuje u tome što ne bira nasumično brid, već vrh, dakle jedan kraj brida. Drugi kraj brida bira ovisno o težini. Naime, cilj je pronaći maksimalno sparivanje koje minimizira rez. Ako je $G_i = (V_i, E_i)$ neki graf i M_i sparivanje na G_i kojim ćemo generirati novi graf G_{i+1}, onda će suma težina bridova grafa G_{i+1} biti jednaka sumi bridova grafa G_i, ali bez bridova koji su u M_i jer su ti bridovi spajali vrhove koji su u novom grafu jedan vrh. Dakle, vrijedi:

$$w(E_{i+1}) = w(E_i) - w(M_i). \tag{4.1}$$

Što je manja ukupna težina bridova nekog grafa, to je manji i rez pa minimiziranje ukupne težine maksimiziranjem težine maksimalnog sparivanja ima smisla. Opišimo sada ukratko algoritam.

HEM algoritam provodi se dok je skup netaknutih vrhova neprazan. Taj je skup u početku jednak skupu vrhova grafa G_i koji želimo sažeti. Nasumično se iz njega izabire neki vrh te se traži brid maksimalne težine kojemu je taj vrh kraj. Pronađeni se brid dodaje sparivanju M_i i njegovi se krajevi stapaju u novi vrh grafa G_{i+1}. Kao i u prošlom

algoritmu, potrebno je prilagoditi skup bridova novonastalom vrhu. Ovaj je algoritam složenosti $O(|E_i|)$.

Algoritam sparivanja najtežim bridom (HEM)

Ulaz: graf $G_i = (V_i, E_i)$
Izlaz: sažeti graf $G_{i+1} = (V_{i+1}, E_{i+1})$

1: $V = V_i$
2: $E_{i+1} = E_i$
3: $V_{i+1} = V_i$
4: **dok** $V \neq \emptyset$
5: nasumično odaberi vrh $v_1 \in V$
6: pronađi brid $(v_1, v_2) \in E_i$ maksimalne težine
7: izbriši v_1 i v_2 iz V
8: dodaj novi vrh $v = \{v_1, v_2\}$ u V_{i+1}
9: $w(v) = w(v_1) + w(v_2)$
10: prilagodi težine bridova novonastalom vrhu v kao u liniji 10 H-L algoritma
11: **vrati** $G_{i+1} = (V_{i+1}, E_{i+1})$

Postoji i učinkovitija varijanta upravo opisanog algoritma, engleski je naziv *sorted heavy edge matching* ili, kraće, SHEM. Ona primjenjuje HEM na listu vrhova sortiranih po njihovim stupnjevima.

Nakon primjene nekog algoritma za sažimanje grafa, dobiven je graf G_n s manjim brojem vrhova od početnog grafa. Sažeti grafovi uglavnom imaju do nekoliko stotina vrhova. Sljedeći je korak particioniranje grafa G_n.

4.2 Particioniranje

U ovoj je fazi višerazinske metode cilj pronaći k-particiju sažetog grafa G_n dobivenog u prošloj fazi. Rezultat će biti inicijalna particija koja će se onda poboljšavati u sljedećoj, posljednjoj fazi. Particioniranje se provodi ili rekurzivnim biparticioniranjem ili direktnim k-particioniranjem, a odabir jednog od tih dvaju postupaka odredit će mogućnosti u trećoj fazi.

Dobivanje inicijalne particije može se dobiti i sažimanjem početnog grafa dok mu broj vrhova ne postane jednak željenom broju particija k. Tada je sažeti graf sam po sebi inicijalna particija pa se razina particioniranja može preskočiti. Međutim, takav pristup ima nekoliko nedostataka. Jedan od njih je da se sažimanjem dobivaju vrhovi čije se težine mogu uvelike razlikovati. Tada algoritam u trećoj fazi ima teži posao i mora biti učinkovitiji kako bi se mogao poštovati uvjet ravnoteže. Više o ovom pristupu može se pronaći u [2,

poglavlje 2, sekcija 2.4], gdje su navedeni i neki radovi koji ga koriste te neki radovi koji nude poboljšanja. U ovom je radu koncentracija na više korištenim metodama. Jedna je od njih spektralno particioniranje grafa, koje je detaljno opisano u prošlom poglavlju. Opisat ćemo još jednu grupu metoda, takozvane metode izrastanja područja.

Metode izrastanja područja

Metode izrastanja područja (engl. *region growing methods*) ili izrastanja grafova (engl. *graph growing*) naziv je za brojnu skupinu pohlepnih metoda particioniranja, od kojih sve slijede isti obrazac. Na početku se odabire nekoliko vrhova koji predstavljaju dijelove particije. U daljnjim koracima novi vrh može biti nadodan dijelu particije samo ako je susjedan barem jednom od vrhova iz tog dijela. Metode iz ove grupe jednostvne su za implementaciju i učinkovite u rješavanju manjih problema (do nekoliko stotina vrhova). U rješavanju većih daju znatno slabije rezultate ([2]) pa se uglavnom koriste unutar višerazinske metode. Predstavit ćemo dvije metode čiji je rezultat biparticija grafa. Za $k > 2$ do k-particije se dolazi postupkom rekurzivnog biparticioniranja.

Algoritam particioniranja izrastanjem grafa

Neka je $G = (V, E)$ graf. Algoritam particioniranja izrastanjem grafa (engl. *graph growing partitioning algorithm*, GGP) iterativno generira skup vrhova V' grafa G čija težina iznosi približno polovinu od ukupne težine vrhova. Vrhovi grafa podijeljeni su u tri skupa: skup V', skup vrhova susjednih onima iz V', označimo ga s U, koji zovemo granicom od V' te skup ostalih vrhova R. U svakoj se iteraciji u V' dodaju njemu susjedni vrhovi, tj. $V' = V' \cup U$. Algoritam staje kada težina skupa V' dostigne polovinu ukupne težine vrhova grafa G. Dobivamo biparticiju čiji je jedan dio V', a drugi $V \setminus V'$.

Nedostatak ovakvog algoritma je što kvaliteta dobivene particije ovisi o odabiru početnog vrha, koji se bira nasumično. No, kako broj vrhova sažetog grafa nije prevelik, algoritam možemo izvršiti više puta, svaki put s drugim početnim vrhom. Na kraju ćemo kao rješenje uzeti particiju koja je imala najmanji rez. Ovako dobivena particija može se značajno poboljšati u sljedećoj fazi.

Pohlepni algoritam particioniranja izrastanjem grafa

Pohlepni algoritam particioniranja izrastanjem grafa (engl. *greedy graph growing partitioning algorithm*, GGGP) poboljšana je varijanta prethodno opisanog GGP algoritma. Isto se provodi iterativno i koristi skupove V', U i R, no vrhovi su u U sortirani prema dobiti, odnosno vrijednosti za koju bi rez bio smanjen kad bi vrh prešao iz U u V'. Dakle, algoritam započinje inicijalizacijom skupa V' nasumičnim odabirom nekog vrha iz V. Skupovi U i R također su inicijalizirani kao u prethodnom algoritmu. Sada u svakoj iteraciji umjesto

cijelog skupa U dodajemo samo vrh $v \in U$ s najvećom dobiti. Nakon toga se vrhovi iz R susjedni vrhu v prebacuju u U i računaju im se dobiti. Vrhovima iz U susjednima v dobiti se preračunavaju i započinje sljedeća iteracija. Algoritam završava kad zbroj težina vrhova u V' dostigne polovinu ukupne težine od V.

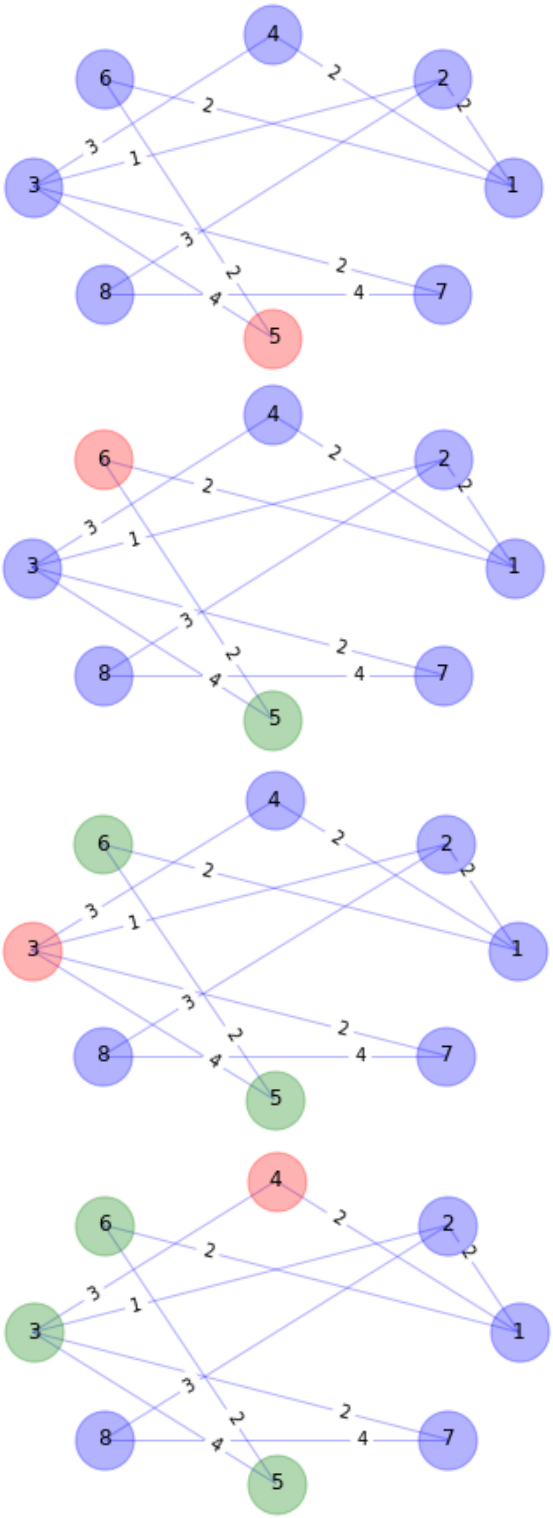
Iako učinkovitost ovog algoritma ne ovisi toliko o izboru početnog vrha kao učinkovitost prethodnog algoritma, za poboljšanje je korisno pokrenuti ga više puta. Bichot i Siarry u [2] predlažu četiri pokretanja, dok za GGP predlažu deset. Navode i da je GGGP u tom slučaju jednako brz kao i GGP, ali daje kvalitetnije bisekcije. Još jedna mogućnost poboljšanja koju navode jest korištenje strukture podataka Fiduccia-Mattheyses, uz neke prilagodbe. Navedenu ćemo strukturu promatrati u opisivanju sljedeće razine pa ne navodimo detalje ovdje. Slijedi pseudokod za GGGP.

Pohlepni algoritam particioniranja izrastanjem grafa (GGGP)

Ulaz: graf $G_i = (V_i, E_i)$
Izlaz: skup V' takav da je $(V', V \setminus V')$ bisekcija grafa G

```
1:  nasumično odaberi vrh v_0 ∈ V
2:  V' = {v_0}
3:  U = {v ∈ R : (v, v_0) ∈ E}
4:  R = V \ {{v_0} ∪ U}
5:  izračunaj dobiti vrhova u U
6:  dok w(V') < (1/2) w(V)
7:      izaberi vrh v_i ∈ U najveće dobiti
8:      premjesti v_i iz U u V'
9:      za svaki brid (v, v_i) ∈ E
10:         ako v ∈ U
11:             ažuriraj dobit od v
12:         inače ako v ∈ R
13:             dodaj v u U
14:             izračunaj dobit od v
15: vrati V'
```

Na sljedećoj je slici dan primjer rada GGGP algoritma na grafu koji smo već koristili za prikaz maksimalnog sparivanja na slici 4.3. U svakom je koraku odabrani vrh označen crvenom bojom i napisane su vrijednosti skupova V', U i R nakon dodavanja označenog vrha u V'. Vrijednosti u skupu U zapisane su u obliku uređenog para u kojemu prvi element označava vrh, a drugi vrijednost dobiti od premještanja tog vrha u drugi dio particije. Na početku su svi vrhovi u istom dijelu particije, a u prvom je koraku nasumično odabran vrh s oznakom 5.



$V' = \{5\}$
$U = \{(6, 0), (3, -2)\}$
$R = \{1, 2, 4, 7, 8\}$
$|V'| = 1$

$V' = \{5, 6\}$
$U = \{(3, -2), (1, -2)\}$
$R = \{1, 4, 7, 8\}$
$|V'| = 2$

$V' = \{3, 5, 6\}$
$U = \{(4, 1), (1, -2), (7, -2), (2, -4)\}$
$R = \{8\}$
$|V'| = 3$

$V' = \{3, 4, 5, 6\}$
$|V'| = 4$
STOP

Slika 4.4: Na slici je prikazan tijek GGGP algoritma za graf s osam vrhova. Algoritam se zaustavlja kad broj vrhova u V' dostigne polovinu ukupnog broj vrhova. Završna je particija oblika $P_2 = (V', V \setminus V')$ i vrijednost reza iznosi 7.

4.3 Projekcija i profinjavanje

U posljednjoj fazi višerazinske metode s inicijalne se particije sažetog grafa vraćamo na početni graf. Imamo vrhove koji zapravo predstavljaju skupove vrhova i bridove koji u sebi sadrže više bridova. Bichot i Siarry u [2] navode autore koji su pokazali da lokalno dobra particija grafa G_{i+1} dobivenog sažimanjem grafa G_i ne mora biti jednako kvalitetna na grafu G_i. Dakle, kada bismo samo vrhove i bridove vratili u prvobitno stanje, kvaliteta particije mogla bi se značajno urušiti. Zbog toga se u ovoj fazi korak po korak dolazi do početnog grafa. Krećući od inicijalne particije P_k^n grafa G_n, u svakom se koraku particija P_k^{i+1} projicira na graf G_i, tj. svi se vrhovi i bridovi grafa G_i sažeti u prvoj razini vraćaju u prvobitno stanje. Zatim se, prije daljnjeg projiciranja, algoritmom profinjavanja traži particija koja će smanjiti rez i biti lokalno dobra na G_i. Postupak završava projekcijom particije P_k^2 na $G_1 = G$ i njezinim profinjavanjem. Završna particija P_k^1 zbog cjelokupne je višerazinske metode dobra globalno, a zbog postupka profinjavanja i lokalno.

Načina za provođenje ove razine višerazinske metode ima mnogo, a ovdje ćemo opisati jedan od najpoznatijih i najkorištenijih, algoritam Kernighana i Lina. Bavit ćemo se i poboljšanjem tog algoritma, strukturom Fiduccia i Mattheysesa. Navedene su metode razvijene za poboljšavanje bisekcija grafa. Višerazinska metoda može u koraku particioniranja koristiti neki algoritam koji ne daje bisekciju, već direktno k-particionira graf. U tom slučaju algoritam profinjavanja treba prilagoditi. Opisat ćemo neke mogućnosti.

Algoritam Kernighana i Lina

Algoritam Kernighana i Lina (skraćeno: K-L) lokalni je optimizacijski algoritam koji služi za profinjavanje bisekcije grafa. Može se provoditi i samostalno, izvan višerazinske metode. Njegova brzina ovisi o kvaliteti inicijalne biparticije. U višerazinskoj metodi to nije prepreka jer se u fazi particioniranja koriste algoritmi koji teže smislenim i kvalitetnim biparticijama.

Algoritam radi u takozvanim prolazima. Prolazi su zapravo iteracije, a u svakoj je od tih iteracija cilj u bisekciji pronaći podskupove dijelova čijom bi se zamjenom reducirala vrijednost reza. Prolazi se zaustavljaju kada više nije moguće naći takve podskupove.

Pretpostavimo da želimo naći savršeno uravnoteženu biparticiju grafa u kojem su svi vrhovi jednake težine. Neka je $G = (V, E)$ graf te $P_2 = (V_1, V_2)$ biparticija grafa G. Definiramo pojmove koji opisuju povezanost vrha v_i s ostalim vrhovima unutar istog dijela biparticije te s vrhovima iz drugog dijela. Unutarnja povezanost $I(v_i)$ vrha $v_i \in V_i$ (engl. *internal cost*) jest zbroj težina svih bridova koji povezuju vrh v_i s nekim vrhom unutar istog dijela:

$$I(v_i) = \sum_{v_i' \in V_i} w(v_i, v_i'). \tag{4.2}$$

Vanjska povezanost $E(v_i)$ vrha $v_i \in V_i$ (engl. *external cost*) jest zbroj težina svih bridova koji povezuju vrh v_i s nekim vrhom iz drugog dijela particije:

$$E(v_i) = \sum_{v_i' \in V \setminus V_i} w(v_i, v_i'). \tag{4.3}$$

S $D(v_i)$ označimo razliku vanjske i unutarnje povezanosti vrha v_i:

$$D(v_i) = E(v_i) - I(v_i). \tag{4.4}$$

Ukoliko $D(v_i)$ poprima pozitivnu vrijednost, vrh v_i više je povezan s vrhovima iz drugog dijela particije P_2. To bi već mogao biti indikator da v_i treba premjestiti. Ipak, tražimo da u dijelovima particije bude jednak broj vrhova pa, ako želimo v_i premjestiti u drugi dio, trebamo pronaći neki vrh v_i' iz drugog dijela koji bi zauzeo njegovo mjesto. Tada treba uzeti u obzir i težinu brida (v_i, v_i'), ako on postoji. Zato se algoritam ne koncentrira na razliku vanjske i unutarnje povezanosti, već na funkciju dobiti. Dobit (engl. *gain*), u oznaci g, označava ukupnu promjenu na vrijednosti reza koja bi se postigla zamjenom vrhova $v_1 \in V_1$ i $v_2 \in V_2$:

$$g(v_1, v_2) = D(v_1) + D(v_2) - 2w(v_1, v_2). \tag{4.5}$$

Opišimo sada jedan prolaz K-L algoritma. Neka je (V_1, V_2) trenutna particija grafa G. Cilj je, kao što je ranije rečeno, pronaći podskupove od V_1 i V_2 čijom bi se zamjenom smanjila vrijednost reza biparticije. Prolaz se provodi iterativno, u svakoj se iteraciji vrši jedna zamjena elemenata, i to ona koja donosi maksimalnu dobit g. Promotrimo algoritam u q-toj iteraciji. Za svaki se vrh v_i računa vrijednost $D(v_i)$ i dobivene se vrijednosti razvrstavaju u skupove D_1 ako $v_i \in V_1$ i D_2 ako $v_i \in V_2$. Skupovi D_1 i D_2 se zatim silazno sortiraju:

$$\begin{aligned} &D_1 : d_{1,1} \geq d_{1,2} \geq ... \geq d_{1,n/2}, \\ &D_2 : d_{2,1} \geq d_{2,2} \geq ... \geq d_{2,n/2}. \end{aligned} \tag{4.6}$$

Kako su skupovi D_1 i D_2 sortirani, to algoritam ne mora razmotriti sve parove $v_1 \in V_1$ i $v_2 \in V_2$. Fiksiramo li element $d_{1,i} \in D_1$, ako nađemo element $d_{2,j} \in D_2$ takav da je $d_{1,i} + d_{2,j}$ manje ili jednako trenutnoj maksimalnoj dobiti, računanje za $d_{1,i}$ se zaustavlja. Naime, izračunatom se zbroju još oduzima dvostruka vrijednost brida između odgovarajućih vrhova, koja je nenegativna, pa cijeli izraz sigurno ostaje manji ili jednak trenutnoj dobiti.

Označimo nađene vrhove koji maksimiziraju dobit u q-toj iteraciji s v_1^q i v_2^q te pripadajuću dobit s g^q. Vrhovi v_1^q i v_2^q se zaključavaju, što znači da se do kraja prolaza više ne razmatraju za razmjenu, preračunavaju se vrijednosti $D(v_i)$ za nezaključane vrhove i postupak se ponavlja u sljedećoj iteraciji na novoj bisekciji $(V_1 \setminus \{v_1^q\}, V_2 \setminus \{v_2^q\})$. Primijetimo još

da se nove vrijednost $D(v_i)$ ne moraju računati tako da se ponovno zbrajaju težine bridova. Neka je v_i neki vrh iz $V_1 \setminus \{v_1^q\}$. Računamo novu vanjsku povezanost E' od v_i:

$$E'(v_i) = E(v_i) - w(v_i, v_2^q) + w(v_i, v_1^q)$$

jer sada više v_1^q nije u istom dijelu particije, a v_2^q jest. Za unutarnju povezanost I' vrijedi:

$$I'(v_i) = I(v_i) - w(v_i, v_1^q) + w(v_i, v_2^q).$$

Dakle, nova je razlika:

$$D'(v_i) = E'(v_i) - I'(v_i) = D(v_i) + 2w(v_i, v_1^q) - 2w(v_i, v_2^q). \tag{4.7}$$

Analogno za $v_i \in V_2 \setminus \{v_2^q\}$. Dobit od prolaza u kojem su zamijenjeni podskupovi $\{v_1^1, ..., v_1^q\}$ i $\{v_2^1, ..., v_2^q\}$ dana je sljedećim izrazom:

$$G(q) = \sum_{k=1}^{q} g^k, \tag{4.8}$$

gdje je $g^k = D(v_1^k) + D(v_2^k) - 2w(v_1^k, v_2^k)$.

Nakon $|V|/2$ iteracija u prolazu dobit je jednaka nuli jer je kompletni dio V_1 zamijenjen s V_2 i obratno. Dakle, nakon $|V|/2$ iteracija imamo skup dobiti $\{G(1), ..., G(|V|/2)\}$. Tražimo $\tilde{q} \in \{1, ...|V|/2\}$ za koji je $G(\tilde{q})$ maksimalna. Ukoliko je $G(\tilde{q})$ veća od nule, vršimo zamjenu $\{v_1^1, ..., v_1^{\tilde{q}}\}$ i $\{v_1^2, ..., v_2^{\tilde{q}}\}$, završavamo prolaz i prelazimo na novi. U suprotnom je algoritam gotov, a rez završne bisekcije jednak je razlici početnog reza i sume dobiti svakog prolaza algoritma.

Algoritam Kernighana i Lina

Ulaz: graf $G = (V, E)$, bisekcija $P_2 = (V_1, V_2)$ grafa G
Izlaz: profinjena bisekcija $P_2 = (V_1', V_2')$, nova vrijednost reza

1: $rez = rez(P_2)$
2: **dok** $G(q) > 0$ **ponavljaj** (prolaz)
3: izračunaj dobiti vrhova iz V_1 i V_2 i razvrstaj ih u D_1 i D_2
4: sortiraj D_1 i D_2 silazno
5: $S_1 = \{\}$; $S_2 = \{\}$
6: **za** $q = 1$ **do** $|V|/2$
7: nađi v_1^q i v_2^q čija zamjena maksimizira dobit g^q
8: $S_1 = S_1 \cup \{v_1^q\}$; $S_2 = S_2 \cup \{v_2^q\}$; $G(q) = G(q-1) + g^q$
9: zaključaj v_1^q i v_2^q
10: ažuriraj D_1 i D_2
11: izaberi q koji maksimizira $G(q)$
12: **ako** $G(q) > 0$
13: $rez = rez - G(q)$
14: zamijeni prvih q elemenata iz S_1 s prvih q elemenata iz S_2
15: **vrati** (P_2, rez)

Najveći nedostatak K-L algoritma njegova je vremenska složenost. Računanje vanjske i unutarnje povezanosti složenosti je $O(|E^2|)$, a nalaženje dvaju podskupova za zamjenu tijekom prolaza složenosti je $O(|E|^2 log|E|)$. Dakle, cijeli je algoritam složenost $O(|E|^2 log|E|)$ (vidi [2]). Autori algoritma predlažu varijantu algoritma u kojoj skupovi D_1 i D_2 nisu sortirani, nego se u svakom od njih traži najveći element i mijenjaju se vrhovi kojima te vrijednosti pripadaju. Takva je varijanta složenosti $O(|V|^2)$, ali ne uzima u obzir težinu brida između zamijenjenih vrhova pa ne donosi nužno maksimalnu dobit. Nakon još nekih napomena o K-L algoritmu opisat ćemo jedno učinkovito poboljšanje, algoritam Fiduccie i Mattheysesa.

Dijelovi različitih veličina

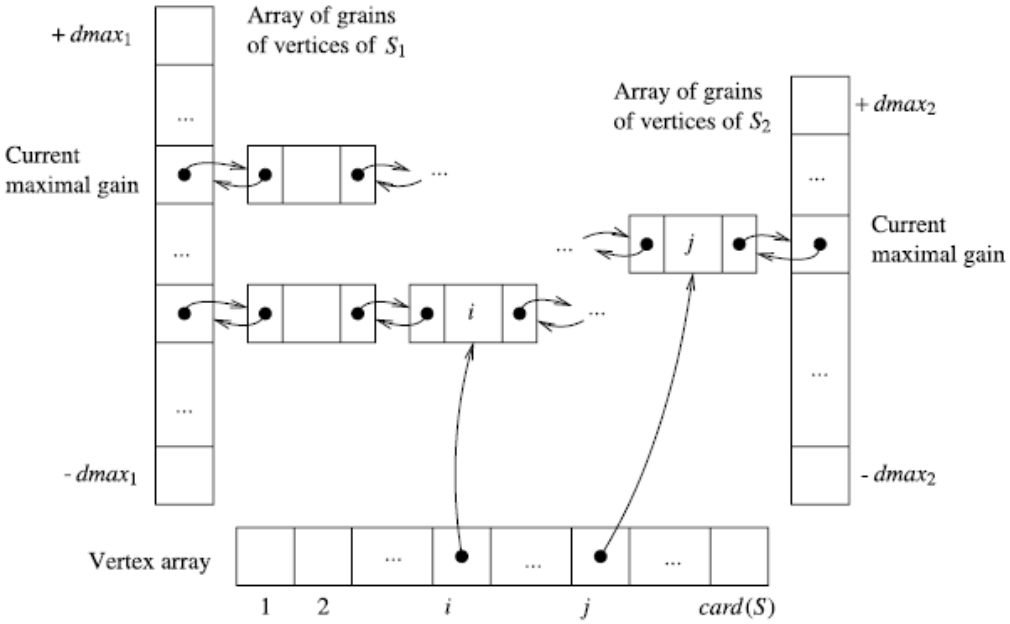
Neka je $G = (V, E)$ graf i $P_2 = (V_1, V_2)$ neka bisekcija grafa G, gdje je $|V_1| = n_1$, a $|V_2| = n_2$. Uzmimo da je $n_1 < n_2$. Želimo profiniti danu bisekciju tako da nova ima jedan dio kardinaliteta najmanje n_1, a drugi kardinaliteta najviše n_2. Opisujemo prilagodbu koju predlažu autori algoritma (vidi [2]). Dijelu kardinaliteta n_1 dodaje se $n_2 - n_1$ virtualnih vrhova težine jednake ostalim vrhovima, ali bez bridova kojima bi bili povezani s nekim drugim vrhom. Dobit od njihovog premještanja tada je jednaka nuli. U svakom se prolazu može zamijeniti najviše n_1 vrhova. Nakon završetka algoritma virtualni se vrhovi uklanjaju. Tako se dobiva tražena bisekcija.

Vrhovi različitih težina

Na početku opisivanja K-L algoritma napomenuli smo kako promatramo vrhove jednakih težina. Sada kratko opisujemo prilagodbu algoritma na slučaj vrhova nejednakih težina. Prilagodbu su također predložili autori algoritma (vidi [2]). Svaki se vrh težine $p > 1$ zamjenjuje s p vrhova jedinične težine koji su svi međusobno povezani. Međutim, takvim pristupom broj vrhova može vrlo brzo narasti, a i može se dogoditi da vrhovi dobiveni rastavljanjem istog vrha budu razdvojeni, što nema smisla. Prema [2], takvo se rješenje na većinu problema ne može primijeniti.

Algoritam Fiduccie i Mattheysesa

Poboljšanje koje algoritam Fiduccie i Mattheysesa (kraće: F-M algoritam) donosi u odnosu na K-L algoritam očituje se u strukturi podataka korištenoj za ažuriranje vrijednosti $D = E - I$ vrhova, jednoj vrsti prioritetnog reda (engl. *bucket priority queue*). Za svaki dio particije postoji sortirana tablica dobiti vrhova tog dijela. Svaka ćelija tablice ima pripadajuću dvostruko povezanu listu čiji su elementi vrhovi čija je vrijednost D jednaka indeksu ćelije tablice. Također, postoji i tablica koja sadrži sve vrhove grafa. Svaki vrh iz te tablice pokazuje na svoje mjesto u odgovarajućoj dvostruko povezanoj listi. Dakle, svakom se vrhu u osnovnoj tablici koja sadrži sve vrhove može pristupiti u vremenu $O(1)$. Svojstvo korištenog prioritetnog reda jest da se svaki element može unijeti u odgovarajuću dvostruko povezanu listu, kao i izbrisati iz nje, također u konstantnom vremenu. Dakle, inicijalizacija dobiti vrši se u vremenu $O(|E|)$. Kako je ažuriranje dobiti u konstantnom vremenu, to je je jedan prolaz algoritma vremenske složenosti $O(|E|)$. Cijeli je algoritam tada linearne složenosti, što je poboljšanje u odnosu na K-L, koji je složenosti $O(|E|^2 log|E|)$.



Slika 4.5: Implementacija tablice s vrijednostima D u algoritmu Fiduccie i Mattheysesa, preuzeto iz [2].

Slijedi pseudokod opisanog algoritma.

Algoritam Fiduccie i Mattheysesa za problem particioniranja grafa

Ulaz: graf $G = (V, E)$, bisekcija $P_2 = (V_1, V_2)$ grafa G
Izlaz: profinjena bisekcija $P_2 = (V_1', V_2')$, nova vrijednost reza

```
1:  rez = rez(P_2)
2:  dok G > 0 ponavljaj          (prolaz)
3:      izračunaj vrijednosti D(v) za sve vrhove
4:      razvrstaj vrijednosti D(v) vrhova iz V_1 u liste reda Q_1, a vrhova iz V_2 u liste reda Q_2
5:      uređena lista L = Ø
6:      odblokiraj sve vrhove
7:      dok postoji neblokirani vrh
8:          naizmjence izaberi red Q ∈ {Q_1, Q_2}
9:          v je vrh maksimalne vrijednosti D(v) u Q
10:         premjesti v u drugi blok, dodaj ga na kraj liste L, blokiraj ga i izbriši iz Q
11:         ažuriraj vrijednosti dobiti za sve vrhove susjedne vrhu v
12:     pronađi maksimalnu dobit G
13:     ako G > 0
14:         izvrši promjene
15:         rez = rez − G
16: vrati (P_2, rez)
```

Upravo opisana struktura podataka poslužila je i u globalnom algoritmu Kernighana i Lina, koji autori u [2] predlažu za direktno k-particioniranje danog grafa. Kratko opisujemo spomenuti algoritam.

Globalni algoritam Kernighana i Lina

Kao i ranije opisani algoritam Kernighana i Lina, i ovaj se provodi iterativno. U svakoj iteraciji, odnosno prolazu, pronalazi se skup vrhova čije bi premještanje u druge dijelove reduciralo ukupni rez k-particije. Također, algoritam koristi strukturu Fidducie i Mattheysesa, ali čuva samo jednu tablicu dobiti. Na početku prolaza računa se dobit svakog vrha prema formuli (4.4). Budući da se radi o k-particioniranju, postoji više dijelova particije za koje treba gledati razliku vanjskog i unutarnjeg reza. Za svaki se vrh posebno gleda odnos sa svakim susjednim dijelom particije, dakle posebno se računa razlika vanjskog i unutarnjeg reza za sve dijelove u koje postoji brid iz trenutnog vrha. Završna dobit vrha jest maksimalna dobit dobivena ovim računom. Ukoliko završna dobit nije pozitivna, vrh se zaključava i više ga se do kraja prolaza ne uzima u obzir za razmjene. Time se algoritam ubrzava, ali se otežava postizanje lokalnog minimuma.

Kad je pripremljena lista nezaključanih vrhova i tablica pripadnih dobiti, započinju iteracije unutar prolaza. Svaka od tih iteracija u početku odabire vrh v s najvećom dobiti. Za

izabrani vrh traži se neki susjedni dio particije takav da je dobit od premještanja v u taj dio što je moguće veća, a da je uravnoteženost particije očuvana. Kada je v premješten u novi dio particije, postaje zaključan i ažuriraju se dobiti njemu susjednih nezaključanih vrhova te kreće nova iteracija. Prolaz završava kada više nema nezaključanih vrhova ili kada smo određen broj puta zaredom premještanjem odabranog vrha dobili nezadovoljavajuće poboljšanje. Na kraju prolaza provode se zamjene koje smanjuju rez particije.

Cijeli algoritam završava kada je proveden zadani broj prolaza ili kada rezultati prolaza više ne reduciraju rez particije. Najveći je nedostatak ovakvog pristupa to da je vrlo teško profiniti particije koje već same po sebi nisu uravnotežene.

Globalni algoritam Kernighana i Lina

Ulaz: graf $G = (V, E)$, k-particija grafa P_k grafa G, *maxbal* [broj prolaza, maksimalna duljina niza negativnih dobiti]
Izlaz: profinjena k-particija P_k

dok $g > 0$ i nije postignut zadani broj prolaza **ponavljaj** (prolaz)
    kreiraj tablicu dobiti i skup nezaključanih vrhova V_v
    dok $V_v \neq \emptyset$ i nemamo niz (unaprijed zadane duljine) negativnih dobiti
        odaberi vrh v najveće dobiti, neka je $V_j \in P_k$ takav da je $v \in V_j$
        za sve $V_i \in P_k \setminus \{V_j\}$ takve da $V_i \cup \{v\}$ čuva *maxbal*
            izračunaj dobit premještanja v u V_i
        odaberi dio $V_i \in P_k$ maksimalne dobiti g za vrh v
        zapamti premještanje vrha v u dio V_i i odgovarajuću dobit g
        za sve nezaključane vrhove v' susjedne v
            ažuriraj tablicu dobiti za v'
        zaključaj v
    odaberi skup vrhova čijim bi se premještanjem ostvarila najveća dobit g
    ako $g > 0$
        izvrši premještanja
vrati P_k

Poglavlje 5

Implementacija

U ovom poglavlju opisat ćemo implementaciju nekih od opisanih algoritama i usporediti performanse na računalu s Intelovim dvojezgrenim procesorom i3 frekvencije 1.80GHz s 4 GB RAM-a. Kodovi su vrlo slični danim pseudokodovima, uz manje prilagodbe koje ne mijenjaju cjelokupni smisao algoritma. Tako, na primjer, u opisu GGGP algoritma spominjemo tri skupa, V', U i R, pri čemu R sadrži sve one vrhove koji nisu u V' i U. U samoj implementaciji taj skup R ne znači za račun pa ga nema.

Implementirani su algoritam spektralnog particioniranja, pohlepni algoritam particioniranja izrastanjem grafa (GGGP) i algoritam Kernighana i Lina, svi za biparticioniranje s uvjetom na ravnotežu. Kodovi su pisani u Pythonu 3.6 koristeći modul networkx, koji omogućava spremanje svih potrebnih informacija o grafu, listu vrhova i bridova te njihove težine. Također, sadrži metodu za računanje Fiedlerova vektora, što je vrlo bitno za brzinu i efikasnost spektralnog particioniranja. Još jedna vrlo korisna funkcionalnost jest ugrađeno računanje matrice susjedstva grafa. Njezino korištenje bilo je ključno u ubrzavanju računanja reza te vanjske i unutarnje povezanosti.

Grafovi na kojima su algoritmi testirani potpuno su povezni i imaju paran broj vrhova. Spremljeni su u tekstualnu datoteku tako da svaki brid zauzima jednu liniju i zapisan je u obliku $u\ v\ d$, gdje su u i v vrhovi, a d težina brida među njima. Radi bolje kontrole, grafovi su generirani tako da su vrhovi u prvoj polovici međusobno bolje povezani i isto tako i vrhovi u drugoj polovici. Dakle, težine bridova između vrhova u istoj polovici poprimaju slučajne cjelobrojne vrijednosti između 3 i 6, a bridovi koji spajaju vrhove u različitim polovicama poprimaju slučajne cjelobrojne vrijednosti između 1 i 3.

Usporedba algoritama

Navedene smo algoritme pokretali na grafovima s različitim brojevima vrhova. Algoritam Kernighana i Lina u višerazinskoj se metodi koristi za profinjavanje pa smo ga tako i ovdje pokretali. Nakon provođenja spektralnog particioniranja i GGGP-a, dobiveni smo

rezultat pokušali profiniti algoritmom Kernighana i Lina. Tablica za svaki od provođenih postupaka i svaki broj vrhova prikazuje vrijeme trajanja algoritma u sekundama. Algoritam GGGP provođen je u četiri iteracije, kako je ranije savjetovano, pa smo u stupac s vremenom upisali prosječno vrijeme izvršavanja jedne od tih iteracija. Također, dodan je stupac (Iteracije) u kojemu piše u kojoj je iteraciji algoritam pronašao najbolji, odnosno najmanji rez.

		Spektralno particioniranje		Spektralno particioniranje + K-L		GGGP			GGGP + K-L	
Broj vrhova grafa	Optimalan rez	Vrijeme (s)	Rez	Vrijeme (s)	Rez	Vrijeme (s)	Rez	Iteracija	Vrijeme (s)	Rez
100	5004	0.04	5098	15.20	5004	36.75	5004	1	7.00	5004
200	19953	0.28	19953	116.35	19953	769.13	20420	2	297.03	19953
500	125320	1.75	125320	4409.63	125320	5120	125320	2		
1000	500171	6.37	501339							

Slika 5.1: Tablica rezultata dobivenih pri pokretanju algoritama na grafovima različitih brojeva vrhova.

U tablici 5.1 vidi se povećanje razlike u učinkovitosti spektralnog algoritma i GGGP-a pri porastu broja vrhova. Algoritam GGGP na grafu od 1000 vrhova nismo pokretali jer je već na 500 vrhova izvršavanje trajalo predugo u odnosu na spektralno particioniranje. Na manjim grafovima nema velikih razlika u rezultatu, posebno nakon provođenja K-L algoritma. Ipak, vremenska je razlika značajna čak i na grafu od 100 vrhova. To, naravno, može biti zbog same implementacije i značajki računala na kojem je sve testirano, no spektralni je algoritam vrlo jednostavno implementirati, što mu je sigurno velika prednost. Algoritam Kernighana i Lina zadnje smo pokrenuli nakon spektralnog particioniranja grafa od 500 čvorova i vidimo da je trajao više od jednog sata, a nije bilo poboljšanja.

U radu smo spomenuli i da je za učinkovitost algoritma Kernighana i Lina vrlo bitna kvaliteta početne particije. Usporedbe radi, algoritam smo pokrenuli na grafu od 200 čvorova, istom onom na kojem smo testirali i ostale algoritme, no u početku smo mu dali nasumično generiranu particiju, čiji je početni rez iznosio 32443. Izvršavanje je trajalo više od 40 minuta.

Dakle, spektralno je biparticioniranje jednostavnije za implementaciju i, budući da su najbitnije funkcionalnosti već ugrađene u Python, one su optimizirane pa je smanjen utjecaj autora koda na efikasnost koda. Za GGGP su očekivani slabiji rezultati na većim grafovima, u [2] i piše da se zbog toga uglavnom koristi unutar višerazinske metode jer je broj vrhova u fazi sažimanja smanjen. Algoritam Kernighana i Lina ubrzala bi implementacija pomoću strukture podataka Fiduccie i Mattheysesa opisane u četvrtom poglavlju.

Bibliografija

[1] R. Agrawal, S. Rajagopalan, R. Srikant i Y. Xu, *Mining newsgroups using networks arising from social behavior*, WWW '03 Proceedings of the 12th international conference on World Wide Web (2003), 529 – 535.

[2] C. Bichot i P. Siarry, *Graph Partitioning*, Wiley, 2011.

[3] H. Pelin, *Diplomski rad: Spektralno klasteriranje*, (2016), `http://digre.pmf.unizg.hr/5263/1/diplomski.pdf`.

[4] C. Schulz, *Course Notes: Graph Partitioning and Graph Clustering in Theory and Practice*, Karlsruhe Institute of Technology, 2016.

Sažetak

Particioniranje grafa optimizacijski je problem na koji je moguće svesti velik broj problema iz stvarnog svijeta, na primjer optimizaciju transportne mreže, segmentiranje slika i analizu društvenih mreža. U ovom je radu opisan sam problem, s naglaskom na težinske grafove u kojima su svi vrhovi jednake težine. Definirani su pojmovi usko povezani s problemom, poput sparivanja na grafu, reza, vanjske i unutarnje povezanosti vrhova te ravnoteže particije.

Kako se radi o NP-teškom problemu, rješavanje se provodi heurističkim i aproksimacijskim algoritmima. U radu je opisan algoritam spektralnog particioniranja te višerazinska metoda, unutar koje su opisani algoritmi iterativno sažimanje grafa te profinjavanje particije s viših razina na niže. Među tim se algoritmima nalazi i jedan od poznatijih algoritama za biparticioniranje, algoritam Kernighana i Lina, te njegovo poboljšanje, algoritam Fiduccie i Mattheysesa. U opisima je dan osvrt na probleme u kojima postoji uvjet na ravnotežu završne particije te na probleme u kojima takav uvjet ne postoji.

U završnom dijelu rada implementirani su neki od opisanih algoritama i uspoređene su njihove performanse na potpuno povezanim težinskim grafovima s jednakom težinom vrhova.

Summary

Graph partitioning is an optimization problem which can be used to solve many real-world problems, such as transportation network optimization, image segmentation and social networks analysis. This thesis focuses on partitioning of weighted graphs with equally weighted vertices. The key notions of graph matching, cut function, external and internal cut and balance of a partition are defined and explained.

Graph partitioning problems are typically NP-hard, therefore, in order to solve them one needs to use heuristic and approximation algorithms. In this thesis several such algorithms are described, namely spectral partitioning algorithm and multilevel method. For the multilevel method, we describe algorithms for iterative coarsening of the graph, as well as algorithms for refining the partition obtained at the lower level to the higher level. Some of the well known bipartitioning algorithms can be found there, such as the Kernighan-Lin algorithm and its improvement, the Fiduccia-Mattheyses algorithm. Solving techniques are described both from constrained and unconstrained partitioning point of view.

In the final chapter an implementation for some of the mentioned algorithm is described. Their performance is compared on fully connected weighted graphs with vertices of equal weight.

Životopis

Rođena sam 10.10.1993. u Zagrebu, gdje sam završila osnovnu školu i Gimnaziju Lucijana Vranjanina, jezični smjer. Tijekom srednjoškolskog obrazovanja uspješno sam se natjecala u poznavanju hrvatskog i njemačkog jezika te u šahu. Godine 2012. upisujem preddiplomski studij matematike na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu. Isti završavam 2016. te upisujem diplomski studij Računarstvo i matematika. Za vrijeme studija predstavljala sam Fakultet i Sveučilište na brojnim šahovskim natjecanjima. Sa ženskom ekipom Sveučilišta u Zagrebu pet godina nastupala sam na državnim sveučilišnim natjecanjima, gdje smo svake godine bile vrlo uspješne. Također, 2016. godine predstavljala sam Sveučilište na europskom natjecanju održanom u Zagrebu i Rijeci.