Otkrivanje zajednica u grafu

Naglić, Luka

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Luka Naglić

OTKRIVANJE ZAJEDNICA U GRAFU

Diplomski rad

Voditelj rada:
prof. dr. sc. Zlatko Drmač

Zagreb, 2021.

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. _____

Posvećujem ovaj rad svim kolegama, prijateljima i obitelji koji su mi bila podrška za vrijeme studiranja te prijatelju Jurici koji mi je pomogao lektorirati ovaj rad. Posebne zahvale idu mentoru, koji mi je pomogao zaokružiti ovaj rad.

Sadržaj

Uvod

U današnjem svijetu okruženi smo prirodnim grafovima, bilo to kroz društvene mreže, mrežu interneta, gradski prijevoz ili bilo koje druge entitete između kojih postoji funkcionalna povezanost. Većina tih prirodnih grafova su veoma veliki, dosežu i do nekoliko milijardi entiteta, stoga moramo razvijati efikasne algoritme koji će u razumnom vremenu vraćati tražene rezultate.

U tim kompleksnim strukturama, mogu se pronaći manji dijelovi koji na neki način čine neku vrstu zajednice, tj. međusobno su bolje povezani nego što su s ostatkom grafa. Otkrivanje zajednica u grafovima od velike je važnosti i za ostala znanstvena područja, npr. sociologiju, neurologiju, biologiju i mnoge druge. Zbog same kompleksnosti problema, nije nimalo efikasno provjeravati sve moguće kombinacije, već moramo problemu pristupiti i na drugačije načine.

Možemo vidjeti kako su mnoge znanosti već profitirale od primjene algoritama otkrivanja zajednica. Naime, samo dvije godine prije nego što je osvojio Nobelovu nagradu, Lee Hartwell je tvrdio da se umjesto fokusiranja na pojedinačne gene znanost treba okrenuti k proučavanju grupa molekula koje zajedno obavljaju određenu staničnu funkciju [1], nakon čega se u [10] prvi put pokušala razviti sistematična identifikacija modula u metaboličkim mrežama. Zajednice također igraju ključnu ulogu u ljudskim bolestima, u radu [5] se pokazalo kako proteini koji se javljaju u istoj bolesti međusobno dolaze u interakciju. Iako se primjene otkrivanja zajednica mogu činiti beznačajnima, često mogu dovesti do značajnih rezultata. Za neke modele grafova, interpretaciju možemo odmah konstruirati iako za neke modele ne možemo bez da napravimo analizu otkrivanja zajednica. Samo zato što za neki koncept nemamo interpretaciju zašto se stvaraju zajednice, ne znači da se zajednice ne stvaraju.

U ovom radu obradit ćemo osnovne definicije teorije grafova koje će nam biti potrebne pri proučavanju odabranih algoritama otkrivanja zajednica. Algoritmi koje ćemo proučavati su hijerarhijskog tipa, što znači da njegov rezultat nije familija zajednica danoga grafa, već niz familija zajednica. Također ćemo proučiti ponašanje algoritama na poznatom skupu podataka jednog karate kluba [11]. Sami podaci pružaju mogućnost interpretacije zajednica u okviru teorija grupne dinamike, što može biti posebno korisno društvenim znanostima.

Poglavlje 1

Teorija grafova

Da bismo mogli razviti teoriju otkrivanja zajednica u grafu, prvo moramo formalno definirati neke pojmove koje ćemo koristiti. To je zapravo svedeno na definiciju grafa, imenovana svojstva grafa i funkcije nad grafovima. Sva teorija se služi jednostavnim konceptima te se više informacija o njoj može naći u [8].

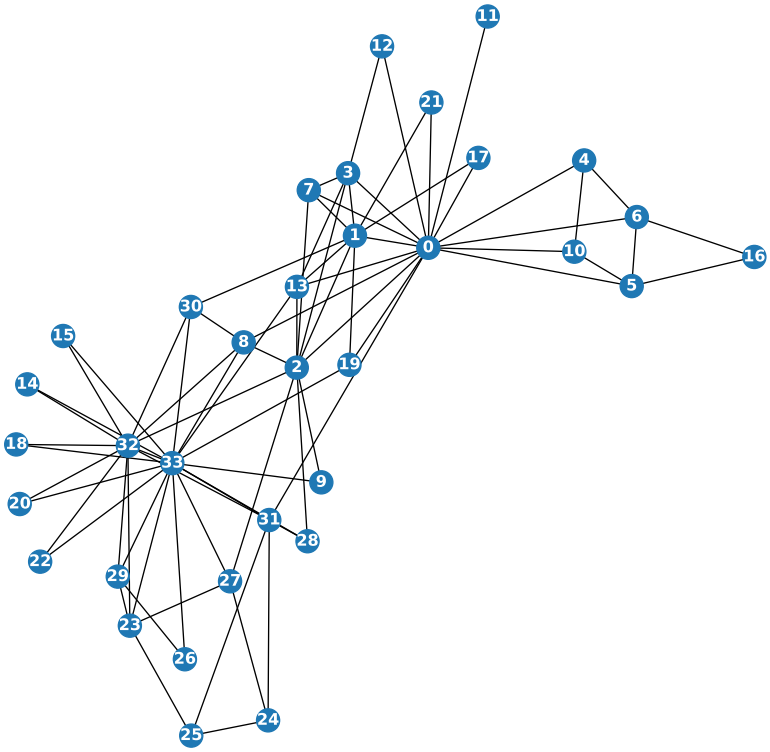
1.1 Osnovne definicije

Za početak ćemo definirati graf na što općenitiji način da bi mogli biti dobro definirani za korištene algoritme.

Definicija 1.1.1. *Neka je* $n \in \mathbb{N}$. *Neka je* $V = \{v_1, v_2, v_3, \ldots, v_n\}$, $E \subseteq V \times V$ *te neka je* $w : E \to \mathbb{R}$.
Graf G definiramo kao uređenu trojku (V, E, w).

Za zadani graf $G = (V, E, w)$, skup V nazivamo skupom čvorova, a E skupom bridova grafa G. Za zadani brid e, vrijednost $w(e)$ nazivamo težinom brida e. Grafički prikaz grafa G nazvat ćemo shemom grafa G.
Primijetite da smo zapravo definirali usmjereni težinski graf, ali definicija bez smanjenja općenitosti definira i neusmjerene grafove bez težina. U slučaju neusmjerenog grafa treba poistovjetiti uređene parove (v_x, v_y) i (v_y, v_x), a u slučaju netežinskog grafa postavimo da za svaki brid e vrijedi $w(e) = 1$, čime smo dobili neusmjereni graf bez težina.
Ako vrijedi $E = V \times V$, kažemo da je graf potpun.

Definicija 1.1.2. *Neka je* $G = (V, E, w)$ *te neka je* $v_x \in V$. *Skup odlaznih susjeda čvora* v_x *je skup* $Out(v_x) := \{v \in V : (v_x, v) \in E\}$. *Skup dolaznih susjeda čvora* v_x *je skup* $In(v_x) := \{v \in V : (v, v_x) \in E\}$. *Skup susjeda čvora* v_x *je skup* $N(v_x) := In(v_x) \cup Out(v_x)$.



Slika 1.1: Vizualizacija grafa karate kluba. Čvorovi su članovi kluba, a povezani su ako su bili u nekoj komunikaciji. Primjer je općepoznat u svijetu otkrivanja zajednica matematičkog grafa.

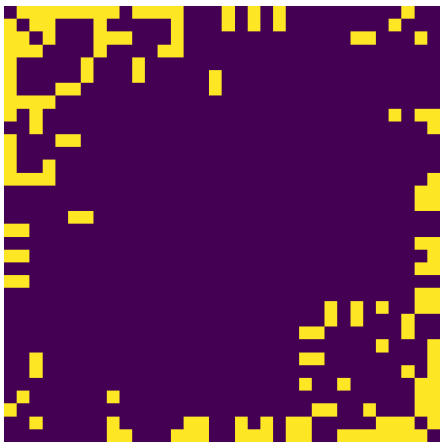
Definicija 1.1.3. *Neka je* $G = (V, E, w)$ *te neka je* $v \in V$*. Valenciju čvora* v *definiramo kao* $deg(v) := |N(v)|$.

Napomena 1.1.4. *U slučaju neusmjerenog grafa, za svaki čvor* $v \in V$ *vrijedi* $Out(v) =$

$In(v) = N(v)$.

Definicija 1.1.5. *Neka je* $G = (V, E, w)$, $|V| = n \in \mathbb{N}$. *Matricu* $A \in \{0,1\}^{n\times n}$ *nazivamo matricom incidencije grafa* G *ako vrijedi*

$$A_{i,j} = 1 \iff (v_i, v_j) \in E.$$



Slika 1.2: Matrica incidencije grafa karate kluba. Sliku možemo zamisliti kao prikaz s 34×34 piksela. Koristeći anonimizirane oznake čvorova, definiramo vrijednost piksela koji se nalazi na (i, j)-tom mjestu da bude žuta ako su čvorovi s oznakama i i j međusobno povezani, a ljubičasta ako nisu.

Definicija 1.1.6. *Neka je* $G = (V, E, w)$ *te neka je* $n \in \mathbb{N}$. *Neka su* $v_x, v_y \in V$. *Niz* $v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ *nazivamo šetnjom duljine* ℓ *između* v_x *i* v_y *ako vrijedi:*

1. $v_0 = v_x$, $v_n = v_y$
2. $\forall k \in \{1, 2, 3, \ldots, n\}$, $e_k = (v_{k-1}, v_k) \in E$
3. $l = \sum_{k=1}^{n} w(e_k)$.

Postojanje šetnje između v_x *i* v_y *kratko označavamo s* $v_x \equiv v_y$.

Napomena 1.1.7. *Radi jednostavnosti zapisa, šetnju duljine ℓ između v_x i v_y možemo označavati kao $v_x \rightsquigarrow_\ell v_y$.*

Propozicija 1.1.8. *Oznaka $\equiv$ nad čvorovima grafa G definira relaciju ekvivalencije.*

Definicija 1.1.9. *Klase ekvivalencije relacije $\equiv$ nad grafom G nazivamo komponentama povezanosti grafa G. Ako postoji samo jedna klasa ekivalencije, kažemo da je graf povezan.*

Definicija 1.1.10. *Neka je $G = (V, E, w)$ te neka je $v \rightsquigarrow_\ell v'$. Kažemo da je $v \rightsquigarrow_\ell v'$ minimalna šetnja između čvorova v i v' ako $\forall k \in \mathbb{R}$ t.d. $v \rightsquigarrow_k v'$ vrijedi $\ell \leq k$.*

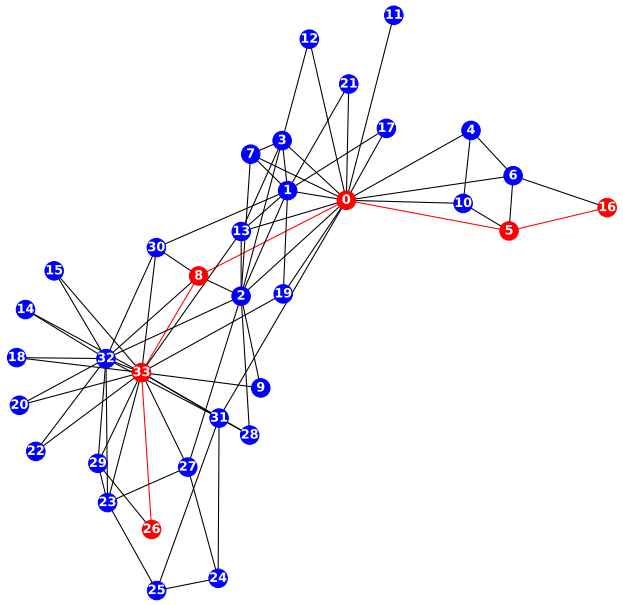
Definicija 1.1.11. *Neka je $G = (V, E, w)$ te neka su $s, t \in V$ te $e \in E$. Definiramo*

$$\sigma_{st} := |\{s \rightsquigarrow_\ell t : s \rightsquigarrow_\ell t \text{ je minimalna šetnja}\}|,$$

$$\sigma_{st}(e) := |\{s \rightsquigarrow_\ell t : s \rightsquigarrow_\ell t \text{ je minimalna šetnja koja prolazi bridom } e\}|.$$

Definicija 1.1.12. *Neka je $G = (V, E, w)$. Graf $G' = (V', E', w')$ nazivamo podgrafom grafa G ako vrijedi $V' \subseteq V$, $E' \subseteq E$ te $w' = w|_{E'}$.*

Definicija 1.1.13. *Neka je $G = (V, E, w)$ te $V' \subseteq V$. Graf $G' = (V', E', w')$ nazivamo grafom induciranim skupom čvorova V' nad G ako vrijedi $E' = \{(v_x, v_y) \in E : v_x, v_y \in V'\}$, te $w' = w|_{E'}$.*



Slika 1.3: Primjer minimalne šetnje između čvora s oznakom 16 i čvora s oznakom 26.

Poglavlje 2

Zajednice u grafovima

U ovom poglavlju obradit će se teorija potrebna za razumijevanje matematičkog modela zajednice, bit će opisani algoritmi koji za dani graf traže skupove zajednica i ocjenjuju koliko je koja podjela na zajednice dobra. Matematički model koji ćemo dati jednostavan je i nalikuje na particije skupa, iako nije jedini jer se ta teorija i dalje razvija.

2.1 Osnovna teorija

Formalnu definiciju dat ćemo odmah, iako nam ona ne govori kako je za dani graf moguće na dobar način odabrati koji čvorovi pripadaju kojoj zajednici, što se vidi i na primjeru na slici 2.1a.

Definicija 2.1.1. *Neka je* $G = (V, E, w)$ *te neka je* $k \in \mathbb{N}$. *Familija zajednica grafa* G *je* $\mathscr{C} = \{C_0, C_1, C_2, \ldots, C_k\}$ *za koju vrijedi sljedeće:*

1. $\forall C \in \mathscr{C}, \emptyset \neq C \subseteq V$
2. $\bigcup_{C \in \mathscr{C}} C = V.$

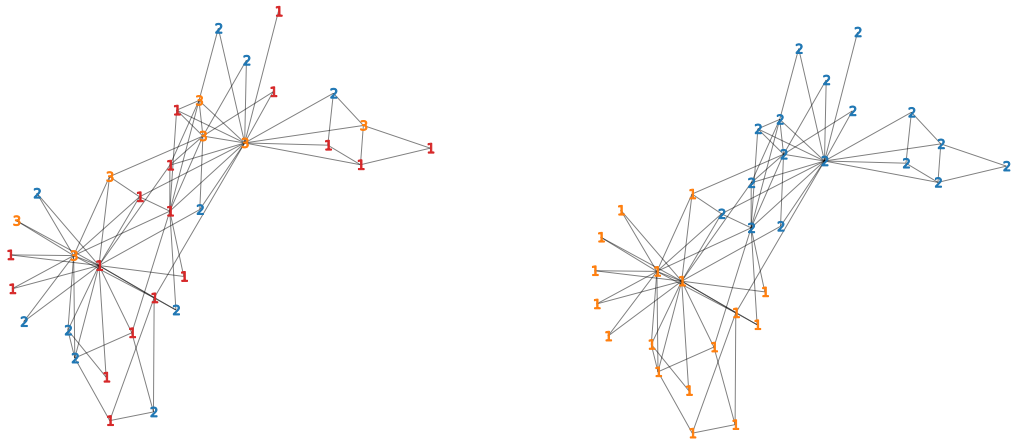
Elemente familije zajednica nazivamo zajednicama.

U prošloj definiciji ponekad ćemo zahtijevati i treći uvjet.

3. $\forall C_i, C_j \in \mathscr{C}, C_i \neq C_j \implies C_i \cap C_j = \emptyset.$

Svi algoritmi će vraćati familiju zajednica grafa koja je u skladu s definicijom 2.1.1, ali za neke će vrijediti i treći uvjet. To nije nužno bolje jer nas ograničava na uvjet da čvorovi smiju biti isključivo u jednoj zajednici, što u stvarnom svijetu nije nužno tako.

Napomena 2.1.2. *Primijetite da ta tri uvjeta zapravo definiraju particiju nad skupom* V.



(a) Primjer nasumičnog odabira familije zajednica grafa karate kluba. Familija se sastoji od 3 zajednice za koje se ne može reći da su odabrane s razlogom.

(b) Nakon raspada karate kluba osnovala su se dva nova kluba koji su na ovom grafu označeni različitim bojama. Za ovaj raspored čvorova u grafu, može se reći da je dobar.

Slika 2.1: Grafički prikaz familije zajednica grafa karate kluba.

Pojam odabira zajednice nije lagano formalno definirati u sklopu teorije grafova, stoga krećemo od nekih slutnji s kojima ćemo bolje razumjeti odabir zajednice grafa te formalno definirati algoritme koji će to raditi.

Slutnja 2.1.3. *Zajednica grafa G je jedinstveno kodirana u shemi grafa G.*

To zapravo znači da nije dovoljno promatrati graf G kroz neke njegove karakteristike, već je potrebno gledati graf kao cjelinu, a da ga ne reduciramo na nešto što bi potencijalno bilo lakše za izračunati.

Sljedeća slutnja nam govori što bi idejno trebala biti zajednica u grafu.

Slutnja 2.1.4. *Zajednica je lokalno gusti povezani podgraf grafa.*

Pojam *lokalno gustih* grafova nismo još formalno definirali. Razlog tomu je to što nam je ponekad potrebna stroža definicija toga pojma. Zajednice koje možemo uočiti u danom grafu ovise o tome kako točno definiramo *lokalno guste* grafove. Stoga ćemo proći nekoliko različitih definicija i pogledati kako koja definicija definira zajednice danoga grafa. Drugim riječima, želimo da su čvorovi zajednice međusobno dostižni (povezanost) te da

su s većom vjerojatnošću međusobno povezani unutar čvorova zajednice, a s manjom vjerojatnošu povezani s čvorovima koji ne pripadaju toj zajednici (lokalna gustoća).

2.1.1 Klike

Za početak ćemo krenuti s trivijalnim modelom zajednice, a to je upravo pojam k-klika.

Definicija 2.1.5. *Neka je $G = (V, E, w)$, $k \in \mathbb{N}$ te neka je $G' = (V', V' \times V', w')$ podgraf grafa G za kojeg vrijedi $|V'| = k$. Tada podgraf G' zovemo k-klikom grafa G.*

Klike su zapravo maksimalno povezani podgrafovi grafa, stoga su izvrsna polazna točka teorije zajednica u grafovima. Klike veoma jasno ispunjavaju obje slutnje koje zahtijevamo od pojma zajednice. Iz definicije klika slijedi da su one gusti povezani podgrafovi, a prikazom sheme klike vidi se da formiraju strukturu zajednice jer su svi čvorovi međusobno povezani.
Glavni problem klika je upravo trivijalnost same ideje klika, tj. zahtjev da to bude maksimalno povezan podgraf. U slučaju $k = 3$, za dani *prirodni* graf možemo pronaći puno 3-klika, ali za veće vrijednosti parametra k, rijetko pronalazimo k-klike.
U slučaju da uspijemo pronaći neku veću k-kliku određenog grafa, možemo sa sigurnošću znati da je to vrlo dobar kandidat za zajednicu toga grafa.

2.1.2 Jake i slabe zajednice

Da bismo proširili model zajednica u grafu tako da bi obuhvatio i podgrafove koji nisu nužno maksimalno povezani, proučavat ćemo kardinalitete skupova dolaznih i odlaznih susjeda.
Za početak ćemo definirati pojam koji će nam dati podatak o tomu koliko je podgraf dobro povezan sam sa sobom, a koliko sa svim ostalim čvorovima koji nisu u tom podgrafu.

Definicija 2.1.6. *Neka je $G = (V, E, w)$, $G' = (V', E', w')$ podgraf grafa G te neka je $v \in V'$. Definiramo unutarnju i vanjsku povezanost čvora v podgrafa G' kao:*

$$k_{int}(G, G', v) := |N(v) \cap V'|,$$

$$k_{out}(G, G', v) := |N(v) \cap (V \setminus V')|.$$

Definicija 2.1.7. *Neka je $G = (V, E, w)$ te neka je $G' = (V', E', w')$ podgraf grafa G. G' nazivamo jakom zajednicom ako vrijedi:*

$$\forall v \in V', k_{int}(G, G', v) > k_{out}(G, G', v).$$

Definicija 2.1.8. *Neka je $G = (V, E, w)$ te neka je $G' = (V', E', w')$ podgraf grafa G. G' nazivamo slabom zajednicom ako vrijedi:*

$$\sum_{v \in V'} k_{int}(G, G', v) > \sum_{v \in V'} k_{out}(G, G', v).$$

Kod jakih zajednica postoji strog zahtjev da svaki čvor bude bolje povezan sa svojom zajednicom nego s ostatkom grafa, dok kod slabih zajednica taj zahtjev nije toliko strog jer dopuštamo da postoje čvorovi koji nisu nužno bolje povezani sa svojom zajednicom nego sa ostatkom grafa.

Propozicija 2.1.9. *Neka je $G = (V, E, w)$, $G' = (V', E', w')$ podgraf grafa G' tada vrijedi:*

1. *Neka je $k \in \mathbb{N}$, G' je k-klika grafa $G \implies G'$ je jaka zajednica*
2. *G' je jaka zajednica $\implies G'$ je slaba zajednica.*

2.1.3 Složenost traženja zajednica grafa

Da bismo napravili efikasni algoritam koji za dani graf vraća skup njegovih zajednica, prvo moramo proučiti koliko različitih odabira skupa zajednica grafa koje bismo morali testirati postoji te zadovoljavaju li oni uvjete zajedništva koje tražimo. Analizirat ćemo najjednostavniju varijantu, a to je problem bisekcije grafa. Problem bisekcije grafa G definiramo kao traženje 2-particije skupa čvorova na način da minimiziramo broj bridova koji povezuju te dvije particije.

Definicija 2.1.10. *Neka je $G = (V, E, w)$. Skup $\{V_1, V_2\}$ nazivamo bisekcijom grafa G ako vrijedi:*

1. *$V_1, V_2 \subseteq V$, $V_1 \cap V_2 = \emptyset$ i $V_1 \cup V_2 = V$*
2. $$\sum_{v \in V_1} k_{out}(G, G_1, v) + \sum_{v \in V_2} k_{out}(G, G_2, v) = \min_{V_x, V_y}\left(\sum_{v \in V_x} k_{out}(G, G_x, v) + \sum_{v \in V_y} k_{out}(G, G_y, v)\right).$$

Ispitujući sve moguće kombinacije, pokušavamo otkriti složenost algoritma koji vraća bisekciju grafa. Radi jednostavnosti analize, uzmimo bisekciju $\{V_1, V_2\}$ t.d. $k := |V_1| = |V_2| = \frac{|V|}{2}$. Broj različitih kombinacija je

$$\binom{|V|}{k}$$

n	10	100	1000
n^2	100	10000	$1 \cdot 10^6$
2^n	1024	$1.2 \cdot 10^{30}$	$1.1 \cdot 10^{301}$
B_n	115975	$4.7 \cdot 10^{115}$	$2.9 \cdot 10^{1926}$

Tablica 2.1: Vrijednost pojedinih funkcija za velike vrijednosti n. U slučaju da algoritam mora napraviti B_{1000} koraka, od kojih svaki traje jednu nanosekundu, algoritmu bi trebalo 100 milijardi puta duže nego što je svemir star.

pa koristeći Stirlingovu formulu $n! \approx \sqrt{2\pi n}(n/e)^n$ broj različitih kombinacija postaje:

$$e^{(|V|+1)ln(2)-\frac{1}{2}ln(|V|)}.$$

Pomoću toga znamo da ne možemo efikasno provjeravati svaku granu jer za grafove s jako puno čvorova broj kombinacija pojednostavljenog problema raste eksponencijalno. Važno je primijetiti da smo problem pojednostavili tj. zahtijevali smo pronaći 2-particiju čiji su elementi jednako veliki. U slučaju nepojednostavljenog problema broj kombinacija particioniranja grafa G t.d. $|V| = n$ je po definiciji n-ti Bellov broj tj.

$$B_n = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}.$$

Bellov broj raste brže nego eksponencijalna funkcija, zbog čega algoritmi particioniranja koji pretražuju sve moguće kombinacije nisu uopće efikasni, što se može vidjeti u tablici 2.1, stoga je problemu potrebno pristupiti na drugačiji način.

2.2 Kvaliteta zajednica

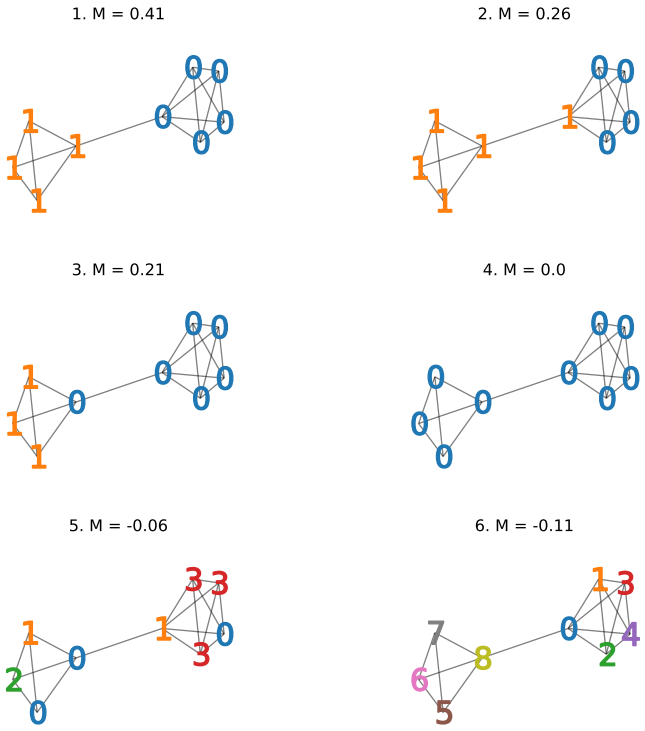
Da bismo mogli ocijeniti koliko je neka particija grafa dobra, moramo definirati neku mjeru koja to mjeri. Od te mjere očekujemo da za nasumično povezane grafove ne dobijemo značajne vrijednosti jer takvi grafovi inherentno nemaju nikakvu strukturu zajednica u sebi. Tu činjenicu ćemo opisati još jednom slutnjom.

Slutnja 2.2.1. *Nasumično povezani čvorovi nekog grafa nemaju strukturu zajednica.*

Ta slutnja nam daje jednostavan način na koji možemo provjeriti dane zajednice nekog grafa. Potrebno je maknuti sve bridove, a čvorove povezati na nasumičan način. Budući da su zajednice zapravo podskupovi skupa čvorova, možemo izmjeriti njihovu kvalitetu na originalnom grafu, ali i na nasumično povezanom grafu. Ta razlika između kvalitete originalnog i nasumičnog grafa bit će naša mjera kvalitete zajednica te ćemo ju nazivati modularnost.

2.2.1 Modularnost

Da bismo mogli definirati pojam modularnosti, prvo moramo definirati kako ćemo nasumično povezati skup čvorova da bismo barem zadržali neka svojstva.



Slika 2.2: Modularnost različitih familija zajednica nad istim grafom. Na prvoj slici se vidi da je modularnost najveća jer je upravo to evidentno najbolje rješenje particioniranja na zajednice. Modularnost je nula ako sve čvorove smjestimo u jednu zajednicu, što se vidi na četvrtoj slici. Modularnost je minimalna kada svaki čvor smjestimo u svoju zajednicu, što je vidljivo na šestoj slici.

Definicija 2.2.2. *Neka je* $G = (V, E, w)$ *te neka su* $v_i, v_j \in V$. *Vjerojatnost povezivanja čvorova* v_i *i* v_j *definiramo kao* $p(v_i, v_j) := \frac{deg(v_i) \cdot deg(v_j)}{2|E|}$.

Ono što zapravo kontroliramo je da očekivanje valencije svakog čvora ista kao i u originalnom grafu.

Definicija 2.2.3. *Neka je $G = (V, E, w)$ te neka je $C \subseteq V$ neka zajednica. Modularnost zajednice C u oznaci M_C definiramo kao*

$$M_C := \frac{1}{2 \cdot |E|} \sum_{v_i, v_j \in C} \left(A_{ij} - p_{ij} \right).$$

Napomena 2.2.4. *Gornja definicija može se svesti na sljedeću formulu.*

$$M_C := \frac{|E_C|}{|E|} - \left(\frac{deg(C)}{2 \cdot |E|} \right)^2,$$

gdje je E_C skup bridova grafa induciranog skupom čvorova C, a $deg(C) = \sum_{v \in C} deg(v)$.

Sada proširujemo pojam modularnosti na familiju zajednica.

Definicija 2.2.5. *Neka je $G = (V, E, w)$ te neka je $\mathscr{C} = \{C_0, C_1, C_2, \ldots, C_k\}$ familija zajednica grafa G. Modularnost familije zajednica definiramo kao*

$$M = \sum_{C \in \mathscr{C}} M_C.$$

Ponašanje modularnosti može se uočiti proučavanjem modularnosti nekih jednostavnih primjera. Na slici 2.2 pokazujemo ponašanje modularnosti nad različitim odabirom familija zajednica istog grafa. Budući da je graf načinjen od 4-klike te 5-klike povezane jednim bridom, vidimo da je modularnost najveća upravo kada su čvorovi klike u istoj zajednici, a najmanja kada stavimo svaki čvor u svoju zajednicu. Bitno je primijetiti da u slučaju jednočlane familije zajednica, gdje su svi čvorovi u skupu, modularnost iznosi 0, što je bilo i za očekivati iz definicije. Također, bitno svojstvo modularanosti je da za bolje odabire familija zajednica, modulranost je pozitivna i veća, dok je za nasumične odabire familije zajednica modularnost manja i negativna.

2.3 Hijerarhijsko klasteriranje

Budući da ne možemo efikasno provjeriti sve moguće kombinacije, moramo se zadovoljiti s nekim polinomijalnim algoritmom koji efikasno traži zajednice unutar grafa. Tu će nam uvelike pomoći familija algoritama koji spadaju pod hijerarhijsko klasteriranje. Takvi algoritmi se temelje na iterativnom mijenjanju familije zajednica s obzirom na neku mjeru. Mjere mogu biti različite i mjeriti različite entitete. Jednu mjeru smo već definirali, a to je modularnost. Popratni algoritam će uvijek odabrati sljedeći korak koji ima najveću modularnost dobivene familije zajednica među svim ostalim koracima. Također ćemo definirati i neke mjere čija je vrijednost veća za čvorove koji ne pripadaju zajednici u kojoj se nalaze, odnosno neke mjere povezanosti brida dviju zajednica.

Mjera, koja je u suštini heuristika, usmjerava algoritam da spoji dvije zajednice ili razdvoji postojeću zajednicu neke familije zajednica te tako definira sljedeću generaciju te familije zajednica. Skup svih generacija je zapravo hijerarhija tog algoritma. Postoje dva suprotna načina građenja hijerarhije, jedan koji grupira čvorove do željene zajednice, tzv. *aglomerativni* pristup, a drugi koji rastavlja grupe čvorova na više zajednica, tzv. raščlambeni pristup.

Oba ova načina ne vraćaju jednu familiju zajednica, već niz familija zajednica, zbog čega se i nazivaju hijerarhijskim algoritmima. Važno je istaknuti dvije relativno trivijalne familije zajednica koje će se pojavljivati u svakoj hijerarhiji. Prva je familija koja sadrži samo jednu zajednicu svih čvorova, koju ćemo nazivati **trivijalno punom familijom zajednica**, a druga u kojoj se svaki čvor nalazi sâm, u sebi jedinstvenoj zajednici zovemo **trivijalno praznom familijom zajednica**. Te dvije familije su posebne jer su upravo one polazišne i završne točke hijerarhijskih algoritama. Mehanizam algoritama će otkriti informacije o zajednicama promatrajući na koji način pojedini algoritmi kreću od jedne trivijalne familiju i pretvaraju je u drugu trivijalnu familiju.

2.3.1 Aglomerativni pristup

Aglomerativni pristup kreće od trivijalno prazne familije zajednica te kroz svaku iteraciju grupira čvorove dviju zajednica u jednu novu zajednicu. Završno stanje algoritma je trivijalno puna familija zajednica. Najpoznatiji aglomerativni algoritam je Ravaszov algoritam [10], koji će biti objašnjen kasnije, a prvo ćemo pokazati kako ćemo iskoristiti modularnost te definirati aglomerativni algoritam.

Modularnost - aglomerativni algoritam

Algoritam se temelji na isprobavanju svih mogućih kombinacija parova zajednica iz trenutne iteracije te spaja onu kombinaciju koja rezultira s maksimalnom modularnošću. Oz-

nakom $m(G, F)$ u pseudokodu označavat ćemo modularnost familije zajednica F grafa G.

Algorithm 1: Aglomerativna modularnost

```
Data: G = (V, E, w)
Result: familija zajednica danog grafa.
F ← trivijalno prazna familija zajednica
H ← { }
while |F| ≠ 1 do
    H ← H ∪ {F}
    (Ĉ1, Ĉ2, v) = (0, 0, ∞)
    foreach C1 ∈ F do
        foreach C2 ∈ F do
            if C1 = C2 then
                continue
            end
            x = m(G, (F \ {C1, C2}) ∪ {C1 ∪ C2})
            if x > v then
                (Ĉ1, Ĉ2, v) = (C1, C2, x)
            end
        end
    end
    F = F \ {Ĉ1, Ĉ2}) ∪ {Ĉ1 ∪ Ĉ2}
end
```

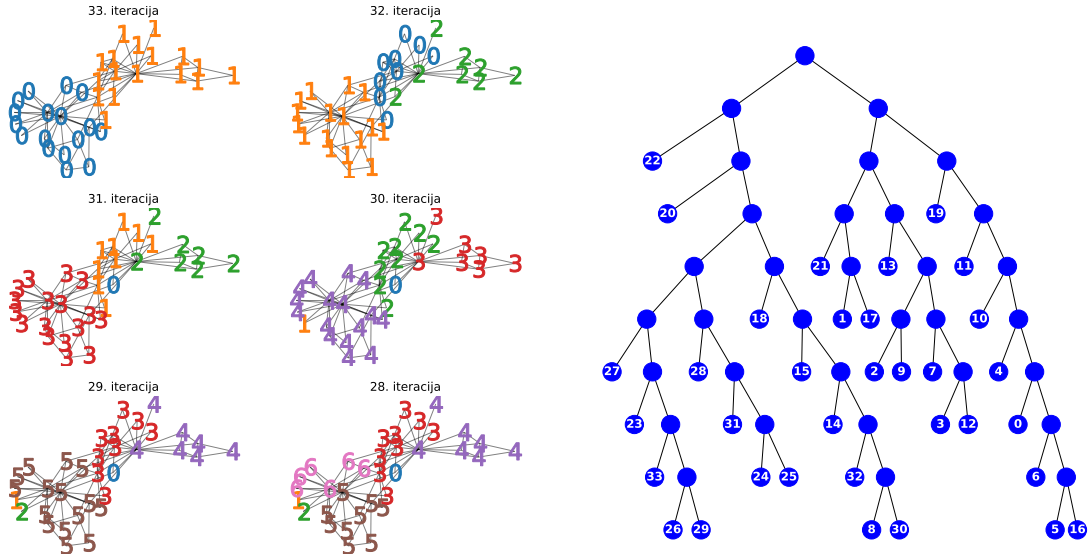
U algoritmu vektor s oznakom $(\hat{C}_1, \hat{C}_2, v)$ koristimo kao pokazivač na kombinaciju dvije zajednice čije spajanje uzrokuje najveću modularnost. Time na pohlepni način u svakom koraku pokušavamo optimizirati modularnost u nadi da će konačna familija sadržavati familiju zajednica s najvećom modularnošću. Na slici 2.3a mogu se proučiti sheme grafova na kojima su obojane pripadne familije zajednica iz posljednjih generacija algoritma nad grafom karate kluba. Slika 2.3b prikazuje dendrogram cijele hijerarhije.

Ravaszov algoritam

Da bismo opisali kako se definira sličnost među čvorovima, pa onda i među zajednicama, prvo ćemo definirati neke funkcije koje ćemo koristiti.

Definicija 2.3.1.

$$x \in \mathbb{R}, \Theta(x) := \begin{cases} 0, & x \leq 0 \\ 1, & x > 0. \end{cases}$$



(a) Posljednjih 6 iteracija algoritma. (b) Dendrogram cijele hijerarhije.

Slika 2.3: Grafički prikaz aglomerativnog algoritma koristeći modularnost.

Definicija 2.3.2. *Neka je* $G = (V, E, w)$ *te neka su* $v_x, v_y \in V$.

$$J(v_x, v_y) := |N(v_x) \cap N(v_y)|.$$

Sada možemo definirati topološki presjek dvaju čvorova, koji će nam biti temelj za računanje matrice sličnosti među zajednicama.

Definicija 2.3.3. *Neka je* $G = (V, E, w)$ *te neka su* $v_x, v_y \in V$. *Definiramo topološki presjek čvorova* v_x *i* v_y *kao*

$$\delta(v_x, v_y) = \frac{J(v_x, v_y)}{\min(deg(v_x), deg(v_y)) + 1 + \Theta(A_{xy})}.$$

Za dani graf bez smananjena općnitosti možemo proizvoljno poredati njegove čvorove. Za određeni poredak čvorova, pomoću topološkog presjeka dvaju čvora možemo definirati matricu sličnosti među čvorovima. Kada to napravimo moramo odlučiti kako ćemo za dvije proizvoljne zajednice odrediti koliko su one slične.

Definicija 2.3.4. *Neka je* $G = (V, E, w)$, δ *topološki presjek nad* $V \times V$ *te* $V_1, V_2 \subseteq V$ *t.d.* $V_1 \cap V_2 = \emptyset$. *Definiramo sličnost zajednica* V_1 *i* V_2 *kao*

$$\Delta(V_1, V_2, \delta) := \frac{1}{|V_1| \cdot |V_2|} \sum_{v_x \in V_1} \sum_{v_y \in V_2} \delta(v_x, v_y).$$

Algorithm 2: Ravaszov algoritam

```
Data: G = (V, E, w)
Result: familija zajednica danog grafa.
F ← trivijalno prazna familija zajednica
H ← { }
while |F| ≠ 1 do
    H ← H ∪ {F}
    (Ĉ1, Ĉ2, v) = (0, 0, ∞)
    foreach C1 ∈ F do
        foreach C2 ∈ F do
            if C1 = C2 then
                continue
            end
            x = Δ(C1, C2, δ)
            if x > v then
                (Ĉ1, Ĉ2, v) = (C1, C2, x)
            end
        end
    end
    F = F \ {Ĉ1, Ĉ2}) ∪ {Ĉ1 ∪ Ĉ2}
end
```

Navedeni algoritam vraća jednu zajednicu, ali ako zapisujemo u kojem trenutku su se spojile dvije zajednice te kolika je bila sličnost među njima, možemo odrediti na kojoj razini algoritma trebamo stati i za zajednice uzeti trenutno stanje algoritma. Složenost toga algoritma je polinomijalna, točnije $O(n^2)$, što je puno efikasnije od $O(2^n)$.

2.3.2 Raščlambeni pristup

Suprotno od aglomerativnog pristupa, raščlambeni pristup polazi od trivijalno pune familije zajednica koju iterativno pretvara u trivijalno praznu familiju zajednica. Najpoznatiji raščlambeni algoritam je Girvan-Newman algoritam [4, 9], a na sličan način ćemo definirati i pohlepni algoritam koji u svakom koraku teži optimizirati modularnost.

Modulanorst - raščlambeni algoritam

Algoritam traži onu zajednicu iz trenutne familije zajednica čije dijeljenje na dvije zajednice ostvaruje najveću modularnost. Da bismo izbjegli isprobavanje svih mogućih kombi-

nacija dijeljenja svake zajednice u familiji zajednica, ubrzat ćemo proces koristeći naivno svojstvo prirode problema. Pretpostavka na kojoj se ono temelji je da svaki graf induciran zajednicom posjeduje samo jednu komponentu povezanosti. Tako ćemo u induciranom grafu tražiti one bridove koji micanjem stvaraju dvije komponente povezanosti te će upravo skup dijeljenja koja su tako generirana biti naš prostor pretrage rješenja. Skup takvih bridova definiramo u sljedećoj definiciji.

Definicija 2.3.5. *Neka je $G = (V, E, w)$ povezan graf. Skup veznih bridova grafa G definiramo kao*

$$VB(G) = \{e \in E : G' = (V, E \setminus \{e\}, w) \textit{ nije povezan graf}\}.$$

Treba primijetiti da je u prethodnoj definiciji dovoljno zahtijevati nepostojanje šetnje između čvorova definiranih bridom e. Time se može konstruirati relativno efikasan algoritam koji za dani graf traži skup veznih bridova. Uz skup veznih bridova potreban je i skup komponenata povezanosti koje smo ranije definirali u definiciji 1.1.9, a ovdje ćemo taj skup za graf G označavati kao $KP(G)$.

Prilikom opisivanja algoritma u pseudokodu oznaka $VB(Z, G)$ označavat će listu veznih bridova grafa induciranog skupom čvorova Z nad G, a s $K_p(G)$ ćemo označavati listu skupova čvorova komponenata povezanosti, dok će oznaka $m(G, F)$ biti definirana isto kao i u prošlom pseudokodu. Također ćemo koristi oznaku nekog skupa čvorova kao graf induciran tim skupom čvorova.

Taj pristup zapravo briše bridove između čvorova koji bi trebali pripadati različitim zajednicama. Jedan od takvih algoritama je Girvan-Newmanov algoritam.

Girvan-Newmanov algoritam

Za razliku od prošlog algoritma, ovdje nećemo spajati zajednice koje bi potencijalno trebale biti skupa, već ćemo razdvajati one koje ne bi trebale biti skupa. Zbog toga želimo da matrica sličnosti zapravo bude matrica različitosti, tj. vrijednost x_{ij} treba biti veća između čvorova koji nikada ne bi trebali biti u istoj zajednici i obratno. Jedan primjer takve mjere je funkcija koja za dani brid vraća vrijednost koja je proporcionalna važnosti toga brida u danom grafu. Iako je definirana nad skupom bridova, funkcija se može definirati i nad parovima čvorova. U slučaju da u grafu ne postoji brid između čvorova s oznakama i, j, vrijednost funkcije u točki (i, j) bit će nula, a inače će biti vrijednost funkcije definirane na tom bridu.

Definicija 2.3.6. *Neka je $G = (V, E, w)$ te neka je $e \in E$. Kvaliteta povezanosti brida e, eng. edge betweenness centrality definiramo s*

$$g(e) := \sum_{s,t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}}.$$

Algorithm 3: Raščlamebna modularnost

Data: $G = (V, E, w)$
Result: familija zajednica danog grafa.
$F \leftarrow$ trivijalno puna familija zajednica
$H \leftarrow \{\ \}$
while $|F| \neq |V|$ **do**
  $H \leftarrow H \cup \{F\}$
  $(\hat{C}, \hat{e}, v) = (0, 0, -\infty)$
  foreach $C \in F$ **do**
    foreach $e \in VB(C)$ **do**
      $x = m(G, (F \setminus C) \cup K_p(C \setminus \{e\})$
      if $x > v$ **then**
        $(\hat{C}, \hat{e}, v) = (C_1, e, x)$
      end
    end
  end
  $F = (F \setminus \hat{C}) \cup K_p(\hat{C} \setminus \{\hat{e}\})$
end

Za potrebe algoritma bit će nam i potrebna

Napomena 2.3.7. *Ako fiksiramo s, t, argument sume je zapravo omjer broja najkraćih putova između s i t koji prolazi bridom e te brojem najkraćih putova između s i t uopće. Ideja ove mjere je da bridovi koji su spojnice dviju zajednica imaju veliku mjeru kvalitete povezanosti jer svi najkraći putovi između tih dviju zajednica moraju prolaziti tim bridom.*

Girvan-Newmanov algoritam

Kako ni u jednom trenutku nismo spomenuli zajednicu unutar algoritma, pretpostavlja se da je svaka komponenta povezanosti zapravo zajednica, što znači da u svakoj iteraciji, ako micanjem jednog brida povećamo broj komponenata povezanosti, time zapravo razdvajamo postojeću zajednicu na dvije. Na kraju algoritma ostajemo sa skupom čvorova originalnog grafa, ali bez bridova. Na neki je način Girvan-Newmanov algoritam dualan Ravaszovom algoritmu. Jedan polazi s pretpostavkom da je svaki čvor jedna zajednica, dok drugi završava u tom stanju, a onda dualno tome, jedan završava u stanju gdje postoji jedna zajednica, dok drugi kreće s tom pretpostavkom. Složenost algoritma je $O(|E|^2|V|)$, što je opet efikasnije od eksponencijalne složenosti.

Algorithm 4: Girvan-Newmanov algoritam

Data: $G = (V, E, w)$
Result: familija zajednica danog grafa.
$F \leftarrow$ trivijalno puna familija zajednica
$H \leftarrow \{\ \}$
while $|F| \neq |V|$ **do**
  $H \leftarrow H \cup \{F\}$
  $(\hat{C}, \hat{e}, v) = (0, 0, -\infty)$
  foreach $C \in F$ **do**
    foreach $e \in VB(C)$ **do**
      $x = g(e)$
      if $x > v$ **then**
        $(\hat{C}, \hat{e}, v) = (C_1, e, x)$
      end
    end
  end
  $F = (F \setminus \hat{C}) \cup K_p(\hat{C} \setminus \{\hat{e}\})$
end

2.3.3 Dendrogram

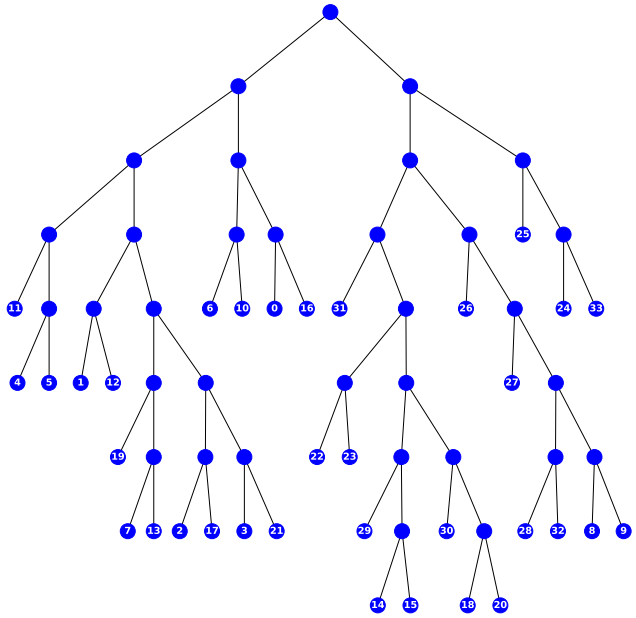
Kao što je već spomenuto, navedeni algoritmi vraćaju hijerarhiju tj. niz familija zajednica. Kako bismo na jednostavan način mogli vizualizirati cijelu hijerarhiju koristit ćemo dendrograme. Dendrogram je prikaz hijerarhije u obliku grafa koji je najlakše opisati preko algoritma koji hijerarhiju pretvara u graf.

Algorithm 5: algoritam za kreiranje dendrograma

Data: $\mathscr{F} = (\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_k)$
Result: dendrogram u obliku grafa.
$D \leftarrow$ prazan graf
for $i \leftarrow 0$ **to** k **do**
  Dodaj elemente familije $\mathscr{C}_i$ u skup čvorova grafa D
  Poveži, u grafu D, elemente familije $\mathscr{C}_i$ s elementima familije $\mathscr{C}_{i-1}$ ako su u međusobnoj relaciji podskupa
end

Tim postupkom hijerarhiju familija zajednica pretvaramo u usmjereni graf D, koji nam na grafički način pokazuje strukture unutar svake zajednice te kako je algoritam iterativno

prolazio kroz strukture samoga grafa. Primjer dendrograma je na slici 2.4. U tom grafu označili smo samo čvorove koji sadrže jednočlane zajednice, a zajednica svih čvorova nalazi se na vrhu grafa. Sve ostale čvorove jednostavno je konstruirati iz ovog grafa, za odabrani čvor treba pogledati sve listove do kojih se može doći šetnjom iz tog čvora.



Slika 2.4: Primjer dendrograma nad grafom karate kluba.

Poglavlje 3

Primjeri i primjena

U ovom poglavlju usporedit ćemo Girvan-Newmanov algoritam te algoritam pohlepne optimizacije modularnosti na jednom općepoznatom grafu, čije su zajednice znane, kao i na nasumično generiranim grafovima. Vidjet ćemo kako se oba algoritama ponašaju kroz svoje iteracije te koji od ta dva algoritma daje kvalitetnije rješenje po modularnosti i ostalim mjerama kvalitete. Također ćemo usporediti vremenske složenosti tih algoritama na način da ćemo algoritme izvrtiti na nasumičnim grafovima različitih veličina. Prikazat ćemo i kako iskoristiti Python biblioteku NetworkX spomenutu u [6]. Ta biblioteka ima implementirane razne algoritme za inicijalizaciju, manipulaciju, analiziranje te vizualizaciju grafova. Biblioteka je besplatna i jednostavna za koristiti. Sve pojedinosti u vezi biblioteke mogu se naći na [2].

3.1 Podaci

Analizu zajednica u grafu radit ćemo na primjeru podataka jednog karate kluba, koji je prvi put analiziran u radu [11]. Graf je zapravo društvena mreža sačinjena od članova karate kluba koji su međusobno povezani ako su imali nekakvu zajedničku interakciju van karate kluba. Motivacija za proučavanje toga grafa proizašla je iz interesa da se nađe neka stvarna primjena zajednica u grafu. Naime, zbog internih sukoba u klubu, klub se podijelio na dva manja kluba. Zanimljivo je bilo što je konačna podjela nakon raspada bila ekvivalentna rješenju problema otkrivanja dviju zajednica, tj. da se je s velikom preciznošću moglo predvidjeti kako će ta podjela izgledati. Shema grafa prikazana je na slici 1.1, iz koje se donekle može vidjeti da postoje dvije različite zajednice u tom klubu. Zanimljivo je da je ovaj skup podataka postao veoma popularan i citiran tek početkom 2000. godine, iako je službeni članak izašao još 1977.

Graf se sastoji od 34 čvora, te 78 bridova, pa je kao takav odličan za testiranje algoritama otkrivanja zajednica jer nije velik, zbog čega ti algoritmi, koji su inače kompleksni,

ne traju predugo. Osim toga, puno ga je lakše vizualizirati i prikazati njegovu shemu nego što bi to bilo kod nekih grafova čiji je broj čvorova znatno veći.

Osim podataka o karate klubu, nasumično ćemo generirati grafove po klasičnom modelu nasumičnog grafa, koji se naziva Erdös-Rényi modelom, prvi put opisanom u [3] te po modelu ratnih paktova opisanom u [7].

3.2 Implementacija

Algoritmi

Ovdje navodimo implementaciju samo jednog od algoritama. Svi algoritmi su relativno slični te su opisani popratni pseudokodovi. Korištena je i implementacija Girvan-Newmanovog algoritma iz biblioteke NetworkX, a dokumentacija se može naći u [2].

Ideja svake implementacije je koristiti varijablu `trenutna_particija`, koja će biti inicijalizirana odgovarajućom trivijalnom familijom zajednica. Također moramo inicijalizirati praznu listu `familija_particija`, koju ćemo kroz svaku iteraciju puniti s vrijednošću `trenutna_particija` iz te iteracije. Kroz svaku iteraciju tražimo najpogodniji čvor ili brid na kojem temeljimo našu pretvorbu varijable `trenutna_particija`.

Kod 3.1: Ravaszov algoritam

```
import networkx as nx
import numpy as np

def topoloski_presjek(G,v,w):
    adj = nx.linalg.graphmatrix.adjacency_matrix(G).todense()
    J = (adj[v]*adj[w].T)[0,0]
    min_deg = min(G.degree(v), G.degree(w))
    return J / (min_deg + 1 + adj[v,w])

def topoloska_vrijednost(G, Z1, Z2):
    suma = 0
    for v in Z1:
        for w in Z2:
            suma += topoloski_presjek(G,v,w)
    return suma / (len(Z1) * len(Z2))

trenutna_particija = [[node] for node in list(G.nodes)]
familija_particija = []
while len(trenutna_particija) != 1:
```

```
    familija_particija.append(trenutna_particija.copy())
    max_c1 = None; max_c2 = None; max_vrijednost = -np.inf
    for c1 in trenutna_particija:
        for c2 in trenutna_particija:
            if c1==c2:
                continue
            tmp = topoloska_vrijednost(G,c1,c2)
            if tmp > maxi_vrijednost:
                max_vrijednost=tmp; max_c1 = c1; max_c2 = c2
    trenutna_particija.remove(max_c1)
    trenutna_particija.remove(max_c2)
    trenutna_particija.append(max_c1+max_c2)
```

Nasumični grafovi

Implementacija Erdös-Rényi nasumičnog grafa dana je u biblioteci NetworkX. Sam algoritam je poprilično jednostavan. Za dani broj čvorova n te vjerojatnost stvaranja brida p prvo se kreira graf samo od n čvorova. Ukupno postoji $n \cdot (n-1)/2$ pozicija na kojima može postojati brid jer je upravo toliko neuređenih parova čvorova. Za svaki neuređeni par čvorova, generiramo brid između njih s vjerojatnošću p. Na kraju ostajemo s grafom koji ima n čvorova te otprilike $p \cdot n \cdot (n-1)/2$ bridova.

Model ratnih-paktova opisan je kodom 3.2. Ulazni podaci algoritma su broj čvorova n i broj bridova m. Algoritam inicijalizira početni graf kao m bridova koji nisu međusobno povezani. Takav graf onda sadrži $2 \cdot m$ čvorova. U svakoj iteraciji na nasumičan način biraju se dva čvora koja se miču iz grafa te se ubacuje novi čvor koji je povezan sa svim čvorovima s kojima su bili povezani odabrani čvorovi. Taj postupak se ponavlja $2 \cdot m - n$ puta tj. sve dok ne ostane n čvorova, nakon čega algoritam staje.

Kod 3.2: Algoritam grafa ratnih-paktova

```
import random

def ratn_paktovi(n,m):
    cvorovi = list(range(2*m))
    bridovi = [(2*i,2*i+1) for i in range(m)]
    brojac = 2*m
    while brojac > n:
        n1 = nodes.pop(random.randint(0,brojac-1))
        n2 = nodes[random.randint(0,brojac-2)]
        l = lambda e: (e[0],n2) e[1] == n1 else (n2,e[1])
        bridovi = list(map(l, bridovi))
```

```
            brojac -= 1
    return cvorovi, bridovi
```

vizualizacija

Biblioteka NetworkX podržava crtanje grafova preko biblioteke Matplotlib. U kodu 3.3 opisan je način na koji se vizualiziraju grafovi i familije zajednica nad njima. Ovisno o kontekstu postoje male promjene, ali načelno je sve opisano kodom. Ideja je svakoj zajednici pridružiti oblik čvora, boju čvora te prikazati to pomoću NetworkX funkcija.

Kod 3.3: Algoritam grafa ratnih-paktova

```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import networkx as nx

fig, ax = plt.subplots(1, 1, figsize=(15,15))

boje = [k for k in mcolors.TABLEAU_COLORS.keys()]
oblici = [f'${i}$' for i in range(100)]

pozicije = nx.spring_layout(graf)
for j,community in enumerate(partition):
    nx.draw_networkx_nodes(
        graf, pozicije, node_color=boje[j],
        nodelist=zajednica, ax=ax, node_shape=oblici[j]
    )
nx.draw_networkx_edges(
    graf, pozicije,
    width=1.0, alpha=0.5, ax=ax
)
```

3.3 Rezultati

Kao što je već spomenuto, algoritme ćemo isprobati na nasumično generiranim grafovima. Za početak ćemo generirati Erdös-Rényi grafove različitih veličina. Kako model uzima dva parametra, broj čvorova n i vjerojatnost stvaranja brida p, konstruirat ćemo grafove sa svim mogućim kombinacijama $n \in \{10, 25, 50, 75, 90\}$ i $p \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$.

3.3.1 Modularnost rezultata

Vrijednosti najveće modularnosti oba algoritma mogu se vidjeti na tablicama 3.1 i 3.2. Tablica 3.1 prikazuje rezultate modularnosti najbolje zajednice Girvan-Newmanovog algoritma nad nasumično generiranom Erdös-Rényi grafu s različitim parametrima, dok tablica 3.2 prikazuje rezultate modularnosti najbolje zajednice pohlepnog algoritma aglomerativnog pristupa za isti nasumično generirani graf kao i u prvoj tablici.
Zanimljivo je primijetiti da je u većini slučaja algoritam pohlepne optimizacije dao bolje rješenje po modularnosti nego što je to vraćao Girvan-Newmanov algoritam. Algoritam pohlepne optimizacije ipak mora proći kroz sve moguće kombinacije pa je tako vremenski složeniji od Girvan-Newmanovog što je vjerojanto i rezultiralo bolje odabranim zajednicama. Iako daje bolje rezultate, u tablici 3.3 te u tablici 3.4 mogu se usporediti vremena izvršavanja algoritma. U tim tablicama se vidi da je pohlepni algoritam i do šest puta sporiji nego Girvan-Newmanov.

n\p	0.1	0.25	0.5	0.75	0.9
10	-0.500	0.280	0.033	0.002	-0.011
25	0.542	0.118	0.031	0.002	0.002
50	0.306	0.065	0.021	0.004	-0.001
75	0.267	0.047	0.017	0.005	-0.001
90	0.122	0.034	0.013	0.007	0.001

Tablica 3.1: Vrijednosti najveće modularnosti Girvan-Newmanovog algoritma za nasumično generiran Erdös-Rényi graf s parametrima n, p.

n\p	0.1	0.25	0.5	0.75	0.9
10	0.000	0.235	0.118	0.077	-0.001
25	0.542	0.202	0.110	0.049	0.012
50	0.344	0.15	0.089	0.041	-0.016
75	0.320	0.149	0.078	0.035	0.017
90	0.241	0.128	0.070	0.037	0.018

Tablica 3.2: Vrijednosti najveće modularnosti pohlepnog algoritma optimizacije za isti nasumično generiran Erdös-Rényi graf s parametrima n, p.

n\p	0.1	0.25	0.5	0.75	0.9
10	0.001	0.002	0.004	0.006	0.008
25	0.010	0.060	0.161	0.290	0.334
50	0.267	1.290	3.474	6.530	7.369
75	1.240	6.410	23.91	45.14	49.57
90	4.483	17.02	62.09	110.6	126.9

Tablica 3.3: Vrijeme izvršavanja Girvan-Newmanovog algoritma za isti nasumično generiran Erdös-Rényi graf s parametrima n, p.

n\p	0.1	0.25	0.5	0.75	0.9
10	0.018	0.021	0.027	0.030	0.033
25	0.633	0.822	1.117	1.500	1.657
50	11.39	17.28	25.92	36.04	40.93
75	60.87	100.7	174.8	257.4	293.6
90	149.5	246.7	429.0	644.4	745.1

Tablica 3.4: Vrijeme izvršavanja pohlepnog algoritma optimizacije za isti nasumično generiran Erdös-Rényi graf s parametrima n, p.

Model ratnih paktova

Istu analizu napravit ćemo koristeći model nasumičnog grafa ratnih paktova. Kako je model poseban, te vremenski složen za grafove velikih gustoća, radit ćemo na manjim grafovima. Vrijednost broja čvorova bit će uzeta iz $\{10, 25, 50, 75, 90\}$, dok će vjerojatnost stvaranja brida biti uzeta iz skupa $\{0.1, 0.15, 0.2\}$. Rezultati modularnosti Girvan-Newmanovog algoritma nad tim grafovima prikazana je u tablici 3.5, a u tablici 3.6 se mogu vidjeti vrijednosti modularnosti pohlepnog aglomerativnog algoritma nad istim grafovima. Isto kao i u prošloj analizi, vidimo da pohlepni algoritam vraća bolje modularnosti. Uspoređena je i vremenska složenost, koja je komparativno ista kao i usporedba vremenske složenosti nad Erdös-Rényi grafom.

3.4 Zaključak

Otkrivanje zajednica nije lagan pothvat. Na početku postoje mnoge nedoumice kako uopće definirati zajednicu, a onda na koji način pojednostaviti rješavanje NP-teškog problema relativno efikasnim algoritmom, a da se pritom zadrže sve potrebne informacije o zadanom grafu. Na kraju dana algoritmi hijerarhijskog pretraživanja jesu polinomijlane složenosti,

n\p	0.1	0.15	0.2
10	0.620	0.438	0.335
25	0.470	0.342	0.204
50	0.239	0.116	0.041
75	0.084	0.034	0.137
90	0.046	0.023	0.031

Tablica 3.5: Vrijednost modularnosti Girvan-Newmanovog algoritma na nasumično generiranom grafu ratnih paktova s parametrima n, p.

n\p	0.1	0.15	0.2
10	0.800	0.438	0.343
25	0.515	0.315	0.271
50	0.312	0.247	0.190
75	0.252	0.209	0.161
90	0.232	0.174	0.158

Tablica 3.6: Vrijednost modularnosti pohlepnog aglomerativnog algoritma na istom nasumično generiranom grafu ratnih paktova s parametrima n, p.

ali sama količine podataka u grafovima koji se mogu modelirati iz prirode uzrokuje da je i polinomijalna složenost ponekad prevelika, pogotovo ako je riječ o polinoma više stupnja.

Iz rezultata se da naslutiti da algoritmi pohlepnog optimiziranja vraćaju zajednice koje imaju veću modularnost nego npr. Girvan-Newmanov algoritam. Razlog je jednostavan, Girvan-Newanov koristi jednostavnu heuristiku pa tako ne mora provjeravati modularnost za sve moguće kombinacije spajanja zajednica. To uzrokuje da će vrijeme izvršavanja biti puno kraće, što se da vidjeti u tablicama vremena izvršavanja, ali se zato odričemo kvalitete rješenja. Pohlepni algoritmi iako duže traju nad grafom nego što traje Girvan-Newmanov algoritam nad tim istim grafom, ali barem zato dobivamo kvalitetnije rješenje.

U oba slučaja, analiza se mogla provesti samo na relativno malim primjerima zbog same kompleksnosti algoritma, ako usporedimo kako se vrijeme izvršavanja ponaša za različite vrijednosti parametra n nasumičnog grafa, a fiksiramo parametar p, vidimo da vrijeme izvršavanja raste jako brzo. Algoritmi jesu efikasniji nego provjeravanje apsolutno svih kombinacija, ali smanjenje složenosti očito nije bilo dovoljno da se rade brze analize nad velikim grafovima.

Bibliografija

[1] A. L. Barabási, *Network Science*, 2015., http://networksciencebook.com, Zadnji put pogledano 04. Rujna 2021.

[2] NetworkX Developers, *NetworkX Library*, 2021., https://networkx.org/, Zadnji put pogledano 18. Kolovoza 2021.

[3] P. Erdös i A. Rényi, *On random graphs I.*, 1958., https://www.renyi.hu/~p_erdos/1959-11.pdf, Zadnji put pogledano 02. Rujna 2021.

[4] M. Girvan i M. E. J. Newman, *Community structure in social and biological networks*, Proceedings of the National Academy of Sciences **99** (2002), br. 12, 7821–7826, ISSN 0027-8424, https://www.pnas.org/content/99/12/7821.

[5] Kwang Il Goh, Michael E. Cusick, David Valle, Barton Childs, Marc Vidal i Albert László Barabási, *The human disease network*, Proceedings of the National Academy of Sciences **104** (2007), br. 21, 8685–8690, ISSN 0027-8424, https://www.pnas.org/content/104/21/8685.

[6] Aric A. Hagberg, Daniel A. Schult i Pieter J. Swart, *Exploring Network Structure, Dynamics, and Function using NetworkX*, Proceedings of the 7th Python in Science Conference (Pasadena, CA USA) (Gaël Varoquaux, Travis Vaught i Jarrod Millman, ur.), 2008, str. 11 – 15.

[7] Luka Naglić i Lovro Šubelj, *War pact model of shrinking networks*, PLOS ONE **14** (2019), br. 10, 1–14, https://doi.org/10.1371/journal.pone.0223480.

[8] Ivica Nakić, *Diskretna matematika*, 2011./2012., https://web.math.pmf.unizg.hr/nastava/komb/predavanja/predavanja.pdf, Zadnji put pogledano 02. Rujna 2021.

[9] M. E. J. Newman i M. Girvan, *Finding and evaluating community structure in networks*, Physical Review E **69** (2004), br. 2, ISSN 1550-2376, http://dx.doi.org/10.1103/PhysRevE.69.026113.

[10] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai i A. L. Barabási, *Hierarchical Organization of Modularity in Metabolic Networks*, Science **297** (2002), br. 5586, 1551–1555, ISSN 0036-8075, https://science.sciencemag.org/content/297/5586/1551.

[11] Wayne W. Zachary, *An Information Flow Model for Conflict and Fission in Small Groups*, Journal of Anthropological Research **33** (1977), br. 4, 452–473, ISSN 00917710, http://www.jstor.org/stable/3629752.

Sažetak

U ovom radu obrađeni su osnovni koncepti polinomijalnih algoritama otkrivanja zajednica u grafu, tj. particije skupa čvorova danog grafa da bi posjedovao nekakvu interpretaciju.

Obrađena je osnovna teorija grafova potrebna za razumijevanje opisanih algoritama te ostalih koncepata otkrivanja zajednica. Svi opisani algoritmi temelje se na hijerarhijskom pretraživanju. Glavna ideja hijerarhijskog pretraživanja je iz particije skupa čvorova generirati novu particiju čvorova. Dva su pristupa hijerarhijskom pretraživanju: aglomerativni i raščlambeni pristup. Aglomerativni je pristup koji u svakoj iteraciji povećava broj particija za jedan, a raščlambeni onaj koji u svakoj iteraciji smanjuje broj particija za jedan. Opisani su najpoznatiji algoritmi oba pristupa, a to su Ravaszov i Girvan-Newmanov. Također je opisan i pohlepni algoritam optimiziranja modularnosti, koji se može prilagoditi obama pristupima. Rezultat algoritama hijerarhijskog pretraživanja je niz famlija zajednica. Nizovi familija zajednica prikazani su dendrogramima.

Efikasnost neoptimiziranih, ali ekvivalentnih *Python* implementacija svih opisanih algoritama, kao i kvaliteta rezultata vrednovana modularnošću, napravljena je na nasumično generiranim grafovima uz pomoć *NetworkX* biblioteke.

Summary

This article deals with polynomial algorithms for community detection in networks which satisfy some useful interpretation when applied.

The basic theory necessary for the understanding of community detection algorithms is explained, as well as some community detection algorithms and properties. The algorithms described are part of a concept called *hierarchical clustering*. The main focus of hierarchical clustering is transforming of a vertex partition into a new vertex partition, thus creating a hierarchy. There are two approaches, one of which is agglomerative, and the other divisive. The agglomerative approach attempts to increase the number of partitions by one at each iteration, whereas the divisive approach attempts to do the opposite, decreasing the number of partitions by one at each iteration. We have described the best known examples of algorithms for each approach – the Ravasz and the Girvan-Newman algorithm, respectively. Apart from that, we have described a greedy algorithm for optimizing modularity which can be implemented in both approaches. The result of hierarchical clustering is a family of community series, which we have visualized by using dendrograms.

The efficiency of the non-optimal, but equivalent Python implementation of the algorithms decribed above, as well as the quality of the results measured by modularity has been conducted on randomly generated networks, with the help of a Python library called NetworkX.

Životopis

Luka Naglić rođen je 1995. godine u Zagrebu. Pohađao je Osnovnu školu Gustava Krkleca nakon koje je upisao opći smjer u Prvoj gimnaziji u Zagrebu. Tokom svog srednjoškolskog obrazovanja sudjelovao je na raznim školskim i županijskim natjecanjima iz prirodoslovo-matematičkih područja te debate. U drugom razredu srednje škole, kvalificira se na državno natjecanje iz debate te na državno natjecanje iz informatike, na kojem osvaja 5. mjesto u B kategoriji. Nakon završene gimnazije upisuje preddiplomski studij matematike pri Matematičkom odsjeku, Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu. Nakon preddiplomskog studija na istom odsjeku upisuje diplomski studij računarstva i matematike. Na prvoj godini studija osvaja stipendiju Erasmus+ projekta te pohađa jedan semestar na Matematičkom odsjeku Fakulteta za matematiku i fiziku Sveučilišta u Ljubljani. Od početka 2020. godine, stalno je zaposlen u Nanobitu kao analitičar podataka za marketing.