Algoritam za preporuke u društvenim mrežama

Buday, Lovro

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni diplomski studij matematike,
smjer: Matematika i računarstvo

Lovro Buday

Algoritam preporuke u društvenim mrežama

Diplomski rad

Osijek, 2023

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni diplomski studij matematike,
smjer: Matematika i računarstvo

Algoritam preporuke u društvenim mrežama

Diplomski rad

Mentor:

izv. prof. dr. sc.
Snježana Majstorović Ergotić

Kandidat:

Lovro Buday

Osijek, 2023

Sadržaj

1 | Uvod

Već od 18. stoljeća teorija grafova je promatrana u više znanstvenih grana, od biologije, kemije do matematike i računarstva. Jedna od njenih najčešćih primjena je u sociologiji, gdje se proučavaju odnosi između ljudi. Poslovni, prijateljski ili neki drugi odnosi analizirani su metodama iz teorije grafova i kompleksnih mreža. U ovome radu promatrat ćemo odnose pratitelja i praćenika u društvenim mrežama. Analizom pratitelja i praćenika promatanog korisnika društvene mreže pokušat ćemo zaključiti koju sljedeću osobu on želi pratiti, s konačnim ciljem da mu tu osobu preporučimo za praćenje. Temeljni dio našeg algoritma za preporuku bit će algoritam za otkrivanje zajednica kojima korisnik pripada. Stvaranjem ocjene koja govori koliko jako korisnik pripada kojoj zajednici, te zajednice ćemo rangirati. Na osnovu ranga zajednice, dobit ćemo preporuku za sljedećeg pratitelja. Ukoliko neki korisnik *jako* pripada nekoj zajednici, onda je veća vjerojatnost da će zapratiti neku osobu iz te zajednice koju do sada nije pratio.

2 | Teorija grafova

2.1 Porijeklo teorije grafova

Teorija grafova je grana diskretne matematike s korijenima iz 18. stoljeća kada se pojavio rad švicarskog matematičara Leonharda Eulera. Eulerov rad o poznatom problemu sedam mostova Königsbera, objavljen 1736. godine široko je prihvaćen kao početak teorije grafova [5]. Od tog vremena, teorija grafova je evoluirala u bogato i opširno polje s primjenama u mnogim disciplinama, uključujući računarstvo, analizu društvenih mreža, biologiju, kemiju, geografiju.

Problem sedam mostova Königsberga odnosio se na pitanje možemo li prošetati gradom Königsberg tako da prijeđemo svaki od sedam mostova koji se nalaze u gradu točno jednom i vratimo se na mjesto s kojeg smo krenuli. Eulerovo pronicljivo rješenje, koje je pokazalo da je takva šetnja nemoguća, iznijelo je temelje teorije grafova tako da je predstavilo koncept grafa u matematici kao modela koji predstavlja odnos između objekata.

Od Eulerovog vremena do sada, teorija grafova je postala moćan alat za modeliranje i analiziranje kompleksnih sustava. Kompleksni sustavi su sustavi načinjeni od raznih komponenata koje su u međusobnoj interakciji. Jedno od najvažnijih svojstava kompleksnih sustava je njihova nelinearnost, tj. kada promatramo procese u kompleksnim sustavima, može se dogoditi da malena promjena u sustavu može rezultirati drastičnom promjenom stanja sustava, dok velike promjene u sutavu ne moraju znanto utjecati na njegovo stanje. To je vidljivo u raznim društvenim i transportnim mrežama, biološkim sustavima i računalnim mrežama. Jedan od glavnih razloga za široku primjenu teorije grafova je činjenica da nam dozvoljava da reprezentiramo kompleksne sustave na jednostavan i intuitivan način, tako da apstrahiramo detalje sustava i fokusiramo se na njegova osnovna svojstva.

U promatranju društvenih mreža teorija grafova je široko primjenjiva. Vrhovi u društvenim mrežama su ljudi ili grupe, a bridovi među njima predstavljaju odnose među tim ljudima, poput prijateljstava, komunikacije ili poslovne suradnje. Istraživanje društvenih mreža s ciljem razumijevanja društvenih interakcija, ponašanja i drugih utjecaja među ljudima i grupama naziva se analiza društvenih mreža.

Ovaj rad je fokusiran na analizu podgrafa u društvenoj mreži. No, najprije ćemo navesti temeljne pojmove iz teorije grafova, a potom u trećem poglavlju objasniti pojmove povezanosti i particioniranja grafa te njihovu primjenu. U četvrtom poglavlju ćemo naveste neke osnovne metode particioniranja te istaknuti njihove prednosti i nedostatke. Peto poglavlje se bavi Louvain metodom koja će biti kori-

štena u izgradnji algoritma za preporuku prijatelja, a koji je detaljno objašnjen u šestom poglavlju.

2.2 Temeljni pojmovi teorije grafova

Definicija 1. *Neusmjereni graf G je uređena trojka $G = (V(G), E(G), \psi_G)$ koja se sastoji od nepraznog skupa $V = V(G)$, čiji su elementi vrhovi od G, skupa $E = E(G)$ disjunktnog s $V(G)$, čiji su elementi bridovi od G i funkcije incidencije ψ_G koja svakom bridu od G pridružuje neuređeni par vrhova u i v od G. Pišemo $e = \{u, v\}$ ili kraće $e = uv$ ili $e = vu$.*

Ako je svakom bridu grafa pridružen smjer, onda se takav graf zove usmjereni graf ili digraf. U ovome radu ćemo se fokusirati na neusmjerene grafove, ali će biti potrebno znati prepoznati i usmjereni graf. Slijedi definicija usmjerenog grafa.

Definicija 2. *Usmjereni graf ili digraf D je uređena trojka $(V(D), A(D), v_D)$ koja se sastoji od nepraznog skupa $V(D)$ vrhova, skupa $A(D)$ lukova (ili usmjerenih bridova) i funkcije incidencije v_D koja svakom luku a pridružuje uređeni par(ne nužno različitih) vrhova u, v. Vrh u je početni, a v je krajnji vrh od a i pišemo $a = (u, v)$.*

Kažemo da je vrh v incidentan s bridom e u G ako je v jedan kraj brida e. Za vrhove koji su krajevi istog brida kažemo da su susjedni. Ako su u grafu G dva vrha u i v spojena s više bridova, onda kažemo da postoji višestruki brid između u i v. Ako brid spaja vrh sa samim sobom, onda ga zovemo petljom. Graf je jednostavan ako ne sadrži višestruke bridove ni petlje. U ovome radu ćemo se baviti isključivo jednostavnim grafovima.

Definicija 3. *Stupanj vrha ili valencija vrha grafa G, u oznaci $d_G(v)$ je broj bridova u G incidentnih s v.*

Intuitivno, stupanj vrha u geometrijskoj reprezentaciji grafa G je broj sjecišta male kružnice oko nacrtanog vrha s linijama koje izlaze iz njega. U usmjerenom grafu, razlikujemo ulazni i izlazni stupanj vrha. Ulazni stupanj vrha je broj bridova čiji je taj vrh krajnji, a izlazni stupanj je broj bridova čiji je taj vrh početni.

Stupanj vrha može biti neophodan faktor u razumijevanju povezanosti grafa jer je za vrhove većeg stupnja veća vjerojatnost da pripadaju većoj komponenti povezanosti.

Najveći stupanj vrha u grafu G označavamo s $\Delta(G)$, tj. $\Delta(G) = \max_{v \in V(G)} d_G(v)$.

Najmanji stupanj grafa G označavamo s $\delta(G)$, tj. $\delta(G) = \min_{v \in V(G)} d_G(v)$.

Prosječni stupanj grafa G označavamo s $d(G)$, odnosno $d(G) = \frac{1}{|V(G)|} \sum_{v \in V(G)} d_G(v)$.

Navedimo definicije dva najčešće korištena grafa.

Definicija 4. *Put P_n s n vrhova je jednostavan graf sa skupovima vrhova i bridova redom $V(P_n) = \{v_1, v_2, \ldots, v_n\}$ i $E(P_n) = \{v_i v_{i+1} : i = 1, \ldots, n-1\}$.*

Definicija 5. *Ciklus s n vrhova C_n je jednostavan graf sa skupom vrhova i bridova redom $V(C_n) = \{v_1, v_2, \ldots, v_n\}$ i $E(C_n) = \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1\}$.*

Definicija 6. *Šetnja u grafu G je netrivijalan konačan niz $W = v_0e_1v_1e_2v_2 \dots e_kv_k$ čiji su članovi naizmjence vrhovi v_i i bridovi e_i, tako da su krajevi od e_i vrhovi v_{i-1} i v_i za svako i, $1 \leq i \leq k$.*

Kažemo da je W $(v_0, v_k)-$šetnja ako su vrhovi v_0 i v_k redom početak i kraj šetnje W, a $v_1, \dots, v_{k-1}$ su njeni unutarnji vrhovi. Za šetnju kažemo da je zatvorena ako je $v_0 = v_k$. Šetnja koja se sastoji od međusobno različitih vrhova (posljedično i bridova), naziva se put.

Za vrhove u i v grafa G kažemo da su povezani ako postoji $(u, v)-$ put u G. Za $(u, v)-$put kažemo da je trivijalan ako vrijedi $u = v$.

Definicija 7. *Udaljenost $d_G(u, v)$ dvaju vrhova $u, v \in V(G)$ u grafu G je duljina najkraćeg $(u, v)-$puta u G.*

Ako ne postoji put između u i v u G, stavljamo $d_G(u, v) = \infty$.

Definicija 8. *Graf G je povezan ako $d_G(u, v) < \infty \, \forall u, v \in V(G)$. U suprotnom kažemo da je G nepovezan.*

Kod usmjerenih grafova povezanost dijelimo na jaku i slabu. Za usmjereni graf D kažemo da je jako povezan ako se iz svakog vrha može doći do svih ostalih vrhova diputom (usmjerenim putom) u D. Usmjereni graf D je slabo povezan ako mu je pripadni neusmjereni graf G povezan. (Pripadni neusmjereni graf G usmjerenog grafa D je graf koji dobijemo zamjenom usmjerenih bridova s neusmjerenim bridovima.)

Da bismo deifnirali dijametar i radijus grafa, najprije moramo deifnirati ekscentricitet vrha u grafu. Ekscentricitet $e(v)$ vrha v povezanog grafa G je $\max\limits_{u \in V(G)} d(v, u)$. Radijus $r(G)$ od G je $\min\limits_{v \in V(G)} e(v)$, dok je dijametar $diam(G)$ od G jednak $\max\limits_{v \in V(G)} e(v)$. Za vrh v kažemo da je centralni vrh mreže G ako vrijedi $e(v) = r(G)$. Za vrh v kažemo da je periferni vrh mreže G ako vrijedi $e(v) = diam(G)$.

Definicija 9. *Težinski graf G^α je graf čijim su bridovima pridruženi realni brojevi, odnosno postoji težinska funkcija $\alpha : E(G) \to \mathbb{R}$, pri čemu broj $\alpha(e)$ zovemo težinom brida $e \in E(G)$.*

U primjenama su težine bridova najčešće pozitivni brojevi, a dogovorno uzimamo $\alpha(e) = 0$ ako $e \notin E(G)$.

Definicija 10. *Gustoća ρ grafa G je realan broj iz intervala $[0, 1]$ koji predstavlja omjer broja postojećih bridova i ukupnog broja svih mogućih bridova u grafu:*

$$\rho(G) = \frac{m}{\binom{n}{2}} = \frac{d}{n-1}.$$

Za usmjereni graf D vrijedi $\rho(G) = \frac{m}{n(n-1)}$. Graf je gust ako $\lim\limits_{n \to \infty} \rho = K$, $K = const$, a rijedak ako $\lim\limits_{n \to \infty} \rho = 0$, odnosno ako d konvergira k nekoj konstanti.

Definicija 11. *Graf H je podgraf od G, u oznaci $H \subseteq G$, ako je $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, a $\psi_H = \psi_{G|E(H)}$, odnosno ψ_H je restrikcija ψ_G na $E(H)$.*

Graf H je pravi podgraf od G ako vrijedi $H \subseteq G$ i $H \neq G$. Pišemo $H \subset G$. Ako je $H \subseteq G$, tj. ako je H podgraf od G onda je G nadgraf od H.

Najjednostavniji primjeri podgrafa su grafovi dobiveni izbacivanjem jednog vrha ili jednog brida iz G. Tako s $G - v$ označavamo podgraf od G dobiven izbacivanjem vrha v (i svih bridova incidentnih s v) iz G, dok s $G - e$ označavamo podgraf od G dobiven uklanjanjem brida e iz G.

Definicija 12. *Podgraf H od G za koji je $V(G) = V(H)$ zove se podgraf koji razapinje G ili razapinjujući podgraf od G.*

Definicija 13. *Neka je $\emptyset \neq V' \subseteq V(G)$. Podgraf od G čiji je skup vrhova V', a skup bridova je podskup skupa bridova od G čija su oba kraja u V' zove se podgraf induciran s V' i označava s $G[V']$. Kažemo još da je $G[V']$ inducirani podgraf od G.*

Poseban slučaj induciranih podgrafova je klika, a da bismo ju pravilno definirali, potrebno je navesti definiciju potpunog grafa.

Definicija 14. *Potpun graf K_n je jednostavan graf s n vrhova u kojemu je svaki par vrhova spojen bridom.*

Definicija 15. *Klika C u grafu G je inducirani podgraf od G koji je potpun.*

Maksimalna klika je klika koja nije sadržana u niti jednoj kliki s većim brojem vrhova, odnosno dodavanjem nekog vrha ona prestaje biti klika. Najveća klika je klika koja ima najveći broj vrhova. Ona često implicira postojanje klastera.

Primjetimo da je klika inducirani podgraf od G koji je potpun.

Definicija 16. *Za neusmjereni graf G i vrh $v \in V(G)$, neka je $N(v)$ skup susjeda vrha v tj. $N(v) = \{u \in V | uv \in E(G)\}$. Ego graf vrha v, u oznaci $ego(v)$ je inducirani podgraf od G, definiran pomoću skupa vrhova $U = \{v\} \cup N(v)$ i skupa bridova $F \subseteq E(G)$ tako da F sadrži sve bridove od $E(G)$ koji spajaju vrhove u U.*

Vrh v iz prethodne definicije zovemo *ego*, a njegove susjede *altere*. Ego graf možemo proširiti tako da ne uzimamo u obzir samo direktne susjede, nego i sve vrhove koji su od v udaljeni za neku zadanu vrijednost. Primjerice, reći ćemo da promatramo ego graf udaljenosti 2 ako se on sastoji od ega, altera i svih vrhova koji su susjedi alterima.

Navedimo još i tvrdnju o stupnjevima vrhova u grafu koja je u literaturi poznata pod nazivom *lema o rukovanju*, a jedna je od najosnovnijih tvrdnji iz teorije grafova.

Teorem 1. ([15]) *U svakom grafu je broj vrhova neparnog stupnja paran broj.*

Ova lema je ključna u razumijevanju distribucije stupnjeva u grafu, koja nam može približiti strukturu i povezanost grafa. Primjerice, graf s neuravnoteženom distribucijom stupnjeva, tj. graf u kojemu mali broj vrhova ima jako veliki stupanj, može imati intrigantnu strukturu koja ukazuje na pojavljivanje podskupova vrhova u grafu s posebnim svojstvima, za razliku od grafa čija je distribucija stupnjeva uniformna.

3 | Povezanost i particioniranje grafa

Od samog nastanka teorije grafova, povezanost grafa je jedan od prvih koncepata koji je definiran radi boljeg razumijevanja strukture grafa. S vremenom, tema povezanosti grafa postala je temeljni dio moderne teorije grafova. Koncept povezanosti su razvijali matematičari i računalni znanstvenici, što je dovelo do raznih algoritama i tehnika particioniranja grafova ([1, 10]).

Proučavanje povezanosti u grafu temeljna je tema u teoriji grafova koja nam pruža pogled u strukturu realnih grafova: kompleksnih mreža te nam dopušta da bolje razumijemo odnose među elementima raznih kompleksnih sustava.

Ranije smo naveli da je graf povezan ako postoji put između bilo koja dva njegova vrha, dok je graf nepovezan ako postoji barem jedan par vrhova koji nisu spojeni putom.

Ako je graf povezan, tada će nas zanimati koliko jako je povezan, kakva mu je distribucija stupnjeva itd., što će nam pomoći u izboru povoljnog algoritma za particioniranje.

U ovom poglavlju ćemo istražiti koncept povezanosti i njegov odnos s problemom particioniranja grafa. Istaknut ćemo važnost komponenata povezanosti, njihova svojstva i primjene u raznim područjima poput računalnih znanosti, analizi društvenih mreža, biologiji itd.

3.1 Komponente povezanosti grafa

Povezanost nam daje uvid u strukturu grafa, što nam dozvoljava da identificiramo različite grupe, klastere unutar grafa. Komponente povezanosti su najveći povezani podgrafovi grafa te se koriste u analizi grafa. U mnogim realnim primjenama, poput analize društvenih mreža i bioloških mreža, komponente povezanosti mogu indicirati značajna grupiranja ili klasteriranja koja pružaju vrijedne informacije o bazičnoj strukturi grafa.

Formalno, proizvoljnom grafu G možemo particionirati skup vrhova V na skupove $V_1, V_2, V_3 \ldots, V_k$ tako da su dva vrha $u, v \in V$ povezana u G ako i samo ako postoji $i \in 1, \ldots, k$ tako da vrijedi $u, v \in V_i$. Grafove $G[V_1], G[V_2], \ldots, G[V_k]$ zovemo komponentama povezanosti grafa G. Broj komponenata povezanosti od G označavamo s $c(G)$. S obzirom da su komponente povezanosti inducirani podgrafovi, ako je $V \subset G$ komponenta povezanosti u G i postoji barem jedan vrh $v \in G$ za koji vrijedi $v \notin V$, tada graf $G[V \cup v]$ nije povezan. Povezani grafovi imaju samo jednu komponentu povezanosti.

U usmjerenim grafovima razlikujemo slabe i jake komponente povezanosti koje se definiraju kao i slabo i jako povezani usmjereni grafovi. U primjenama je najčešći slučaj da postoji ili samo jedna velika slaba komponenta povezanosti u grafu ili samo jedna velika jaka komponenta povezanosti. Primjerice, aciklički usmjereni grafovi (ne sadrže usmjerene cikluse) nemaju jaku komponentu povezanosti.

Komponentu povezanosti grafa G koja ima znatno veći broj vrhova od ostalih zovemo *gigantskom komponentom*. Preciznije, za gigantsku komponentu s m vrhova u grafu G s n vrhova vrijedi $m = \mathcal{O}(n)$, odnosno postoji pozitivna realna konstanta K tako da vrijedi $m = K \cdot n$. Stoga, povećanjem broja vrhova grafa G, povećava se i broj vrhova giganstke komponente.

Mnogi grafovi u primjenama nisu povezani, nego se sastoje od jedne gigantske komponente koja često sadrži više od 90% vrhova grafa, dok se ostatak grafa najčešće sastoji od jako malih komponenata povezanosti koje nisu zanimljive za proučavanje.

3.2 Particioniranje grafa

Particioniranje grafa je metoda koja pruža profinjeniji pogled u strukturu grafa jer omogućuje podjelu vrhova u manje odvojene grupe. Najčešći cilj particioniranja grafa je napraviti takvu particiju vrhova grafa u kojoj je suma težina bridova između grupa vrhova minimizirana, dok je suma težina bridova unutar grupa maksimizirana. Ovaj proces može otkriti odnose između objekata predstavljenih vrhovima koji možda nisu očiti kada uzmemo u obzir cijeli graf. Particioniranje grafa ima mnogo praktičkih primjena, poput balansiranja opterećenja u paralelnim izračunima, dizajniranje strujnih krugova, klasteriranje podataka [11] itd. Treba naglasiti da se particioniranje grafa može odvijati i prema nekom drugom kriteriju, ovisno o problemu koji želimo riješiti koristeći graf. Dakle, particija grafa G se odnosi na podjelu vrhova grafa u izvjesne neprazne, međusobno disjunktne podskupove vrhova, čija unija odgovara skupu vrhova grafa G, koristeći pritom neki unaprijed zadani kriterij particioniranja.

Grafovi u kojima se pojavljuju podskupovi vrhova spojeni izrazito velikim brojem bridova zovemo heterogenim grafovima. No, važno je spomenuti da neke grafove nema smisla particionirati, odnosno u njima se ne pojavljuju gusti podskupovi vrhova (inducirani pografovi s velikim brojem bridova) pa particioniranje ne daje rezultate koji se mogu smisleno interpretirati.

Povezanost grafa i njegovo particioniranje se isprepliću. Primjerice, nepovezan graf možemo prirodno particionirati u njegove komponente povezanosti.

3.3 Jakost povezanosti i jakost particije grafa

Kao što smo ranije napomenuli, u primjenama je najčešći cilj particioniranja grafa maksimizirati broj bridova koji spajaju vrhove unutar pojedinih podskupova particije, a minimizirati broj bridova koji spajaju vrhove koji pripadaju različitim podskupovima particije. Ukoliko je graf težinski, onda se minimizira suma težina bridova unutar podskupova particije, a maksimizira suma težina bridova koji spajaju

vrhove različitih podskupova particije. Ako tako gledamo proces particioniranja, možemo govoriti o tome koliko jako određeni vrh pripada određenom podskupu particije. Također, usko vezano za snagu particije je koncept snage povezanosti koja se može utvrditi na više načina. Jedan od načina kako ustanoviti jakost povezanosti nekog grafa jest da se usredotočimo na rezne vrhove i rezne bridove.

Definicija 17. *Vrh v grafa G zovemo **rezni vrh** ako vrijedi $c(G-v) > c(G)$.*

Definicija 18. *Brid e grafa G zovemo **rezni brid** ili **most** ako vrijedi $c(G-e) > c(G)$.*

Treba napomenuti da se uklanjanjem reznog brida broj komponenata povezanosti grafa poveća za točno jedan, dok se uklanjanjem reznog vrha broj komponenata povezanosti može proizvoljno povećati, ovisno o strukturi grafa.

Teorem 2. *Neka je $|V(G)| \geq 3$. Vrh v povezanog grafa G je rezni vrh ako i samo ako postoje vrhovi u i w $(v \neq u, w)$ tako da v pripada svakom (u,w)-putu u G.*

Dokaz. $\Rightarrow$ Neka je v rezni vrh. Očito je $G-v$ nepovezan graf. Ako su u i w vrhovi iz različitih komponenti grafa $G-v$, tada nema (u,w)-puteva u $G-v$. Obzirom da je G povezan, postoje (u,w)-putevi u G. Stoga, svaki (u,w)-put u G sadrži v.
$\Leftarrow$ Neka postoje vrhovi u i w u G takvi da v pripada svakom (u,w)-putu u G. Tada nema (u,w)-puteva u $G-v$, a to znači da je $G-v$ nepovezan pa je v rezni vrh. □

Slično se dokazuje analogna tvrdnja za mostove povezanog grafa G pa ćemo ju navesti bez dokaza.

Teorem 3. *Brid e povezanog grafa G je most ako i samo ako postoje vrhovi u i w takvi da e pripada svakom (u,w)-putu od G.* □

Slijedi tvrdnja koja povezuje mostove i cikluse povezanog grafa.

Teorem 4. *Brid e povezanog grafa G je most ako i samo ako e ne pripada niti jednom ciklusu u G.*

Dokaz. $\Rightarrow$ Dokažimo da ako brid e leži u nekom ciklusu u G, tada on nije most. Bez smanjenja općenitosti pretpostavit ćemo da je G povezan. Neka $e = uv$ leži u ciklusu C. Nadalje, neka su w_1 i w_2 proizvoljni različiti vrhovi u G. Ako e ne pripada (w_1, w_2)-putu u G, tada je (w_1, w_2)-put cijeli sadržan u $G-e$. Ako e pripada (w_1, w_2)-putu u G, tada zamjenom brida e s (u,v)- ili (v,u)-putem u C koji ne sadrži e daje (w_1, w_2)-šetnju u $G-e$. No, (w_1, w_2)-šetnja u $G-e$ sadrži (w_1, w_2)-put u $G-e$. Stoga su w_1 i w_2 povezani u $G-e$ pa e nije most.
$\Leftarrow$ Pretpostavimo suprotno. Neka e nije most u G. Slijedi $G-e$ je povezan. Stoga postoji (u,v)-put u $G-e$. No, takav put zajedno s e čini ciklus u G koji sadrži e, što ne može biti. □

Svi navedeni teoremi vrijede i ako G nije povezan, ali se tada dokazi odnose na komponente povezanosti u kojima je prisutan rezni vrh ili most.

3.3.1 $K-$povezanost grafa

U nekim grafovima je potrebno ukloniti više od jednog brida ili vrha da se on raspadne u više komponenata povezanosti. Zato je često važno znati koliko jako je neki graf povezan, tj. hoće li se raspasti uklanjanjem samo jednog vrha ili brida, ili će biti potrebno ukloniti više vrhova ili bridova.

Definicija 19. *Vršni rez (separator, separacijski skup) od G je podskup $V' \subseteq V(G)$ tako da je $G - V'$ nepovezan ili trivijalan. K-vršni rez je vršni rez s k elemenata, tj. $|V'| = k$. Slično definiramo bridni rez i k-bridni rez od G.*

Slično definiramo bridni rez i k-bridni rez od G.

Definicija 20. *Povezanost ili vršna povezanost $\kappa(G)$ grafa G je najmanji broj vrhova čijim izbacivanjem graf prestaje biti povezan ili postaje trivijalan, tj.*

$$\kappa(G) = \min\{|V'| \, : \, V' \text{ vršni rez od } G\}.$$

Kažemo da je G k-povezan ako je $\kappa(G) \geq k$. Dakle, $k-$povezanost vrha je generalizacija koncepta povezanosti vrha i pruža nam preciznije razumijevanje povezanosti grafa. Graf G je $k-$povezan ako mu je povezanost vrha barem k. Drugim riječima, G je $k-$povezan ako može podnijeti uklanjanje $k-1$ vrhova bez da izgubi svojstvo povezanosti.

Definicija 21. *Bridna povezanost $\kappa'(G)$ je najmanji broj bridova čijim izbacivanjem graf prestaje biti povezan ili postaje trivijalan, tj.*

$$\kappa'(G) = \min\{|F'| \, : \, F' \text{ bridni rez od } G\}.$$

Kažemo da je G k-bridno povezan ako je $\kappa'(G) \geq k$.

Kažemo da je G k-bridno povezan ako je $\kappa'(G) \geq k$, odnosno G je $k-$bridno povezan ako je uklanjanjem manje od k bridova on i dalje povezan graf.

Sada je jasno da povezanost vrha u grafu kvantificira otpornost grafa na gubitak vrha. Primjerice, graf s velikom povezanosti vrha smatra se robustnim jer može podnijeti gubitak nekoliko vrhova bez da izgubi svojstvo povezanosti.

Kao i povezanost vrha, povezanost brida je mjera otpornosti grafa na gubitak brida. Graf visoke bridne povezanosti može izdržati gubitak nekoliko bridova bez da izgubi svojstvo povezanosti.

Povezanost vrha i povezanost brida igraju ključnu ulogu u particioniranju grafa. Graf s visokom povezanošću često zahtjeva sofisticiranije metode particioniranja da bi se njegovi vrhovi učinkovito odvojili u disjunktne skupove. S druge strane, graf s malom povezanošću može se lako particionirati jer postoji manje putova između vrhova.

3.4 Praktične primjene povezanosti i particioniranja

Slijede neke od značajnijih primjena povezanosti i particioniranja grafa.

• Analiza društvenih mreža.
U proučavanju društvenih mreža, razumijevanje povezanosti i particioniranja pomaže nam u prepoznavanju zajednica, utjecajnih ljudi i načina komuniciranja unutar mreže (grafa) (Newman, 2006). Ove informacije mogu biti iskorištene u svrhu razvijanja marketinških strategija, predviđanja širenja informacija i poboljšanja razumijevanja društvenih dinamika.

• Biologija.
U biologiji, teorija grafova se koristi za proučavanje proteinskih mreža, mreža regulacije gena i metaboličkih mreža. Analiza povezanosti i particioniranja ovih mreža može nam pomoći u otkrivanju funkcionalnih modalnosti, otkriti organizaciju bioloških procesa i identificirati potencijalne farmaceutske mete.

• Transport i logistika.
U transportu i logistici, teorija grafova se koristi za modeliranje i analiziranje transportnih mreža poput cestovnih mreža, avionskih i brodskih ruta. Proučavanje povezanosti i particioniranje ovih mreža može pomoći u optimiziranju transportnih ruta, identificiranju kritične infrastrukture i poboljšanju učinkovitosti transportnih sistema.

Ovi primjeri ilustriraju svestranost primjene povezanosti i particioniranja u analizi grafova, s naglaskom na važnost razumijevanja ovih kocepata u kontekstu stvarnog svijeta.

4 | Metode particioniranja i njihova svojstva

S ciljem da utvrdimo najbolju metodu particioniranja za našu upotrebu, pogledat ćemo neke od često korištenih metoda i usporediti ih. Navest ćemo njihove prednosti i nedostatke.

4.1 Spektralno particioniranje

Spektralna teorija grafova je moćan alat u proučavanju strukture grafa i njegovih svojstava, posebice povezanosti i heterogenosti. Bazirana je na analizi svojstvenih vrijednosti i svojstvenih vektora matrice susjedstva, Laplaceove matrice [3] ili neke druge matrice pridružene grafu.

Spektralno particioniranje je jedna od primjena spektralne teorije grafova koja koristi svojstvene vrijednosti i svojstvene vektore za potrebe particije skupa vrhova grafa. Jedna od popularnih metoda particioniranja zove se Fiedler metoda, koja koristi drugu najmanju svojstvenu vrijednost grafove Laplaceove matrice, poznata kao Fiedlerova vijednost, i njen odgovarajući svojstveni vektor, Fiedlerov vektor. Fiedlerova metoda se pokazala efektivnom za particioniranje grafova visoke povezanosti jer može identificirati podjele unutar grafa s obzirom na njegova spektralna svojstva. Štoviše, spektralne metode particioniranja mogu biti kombinirane s drugim tehnikama particioniranja, poput multi-level particioniranja, kako bi poboljšali učinkovitost i efektivnost.

Prednosti spektralnog particioniranja su particije visoke kvalitete s minimalnim brojem bridova između skupova particija, što ga čini prikladnim za detekciju zajednica u društvenim mrežama. Ova tehnika je također relativno jednostvna za implementirati i može biti kombinirana s drugim metodama particioniranja za poboljšanje rezultata. Nedostatak ove metode je visoka računalna složenost kod većih grafova. Dodatno, spektralne metoda nekad proizvedu particije s nebalansiranim veličinama, što zna biti loše kod nekih primjena.

4.2 Multi-level particioniranje

Multilevel particioniranje podrazumijeva hijerarhijski pristup particioniranju grafa. Graf biva aproksimiran nizom manjih grafova. Najmanji graf u nizu se tada particionira koristeći spektralne ili neke druge metode, a zatim se rekonstruira origi-

nalan graf te se dobivena particija profinjuje [9].

Prednosti multi-level particioniranja je velika učinkovitost, odnosno takav pristup rezultira dobrim particijama u relativno kratkom vremenu. Ova metoda je posebno primjenjiva u velikim grafovima, što je čini popularnim izborom za prepoznavanje zajednica (grupa vrhova) u društvenim mrežama. Nedostatak metode jest osjetljivost na izbor inicijalne metode particioniranja. Štoviše, ova tehnika također zna proizvesti particije s nebalansiranim veličinama.

4.3 Kernighan-Lin Algoritam

Kernighan-Lin (KL) algoritam koristi pohlepan pristup kako bi iterativno poboljšao kvalitetu danih particija. Ovaj algoritam kreće od neke inicijalne balansirane particije te iterativno zamjenjuje dva vrha iz dviju različitih skupova particije u svrhu minimiziranja bridnog reza.

Prednost KL algoritma jest proizvodnja particije s malim bridnim rezom, što je poželjno u detekciji zajednica u društvenim mrežama. Također ga je lako implementirati, a može se kombinirati s drugim metodama particioniranja. Nedostatak KL algoritma je visoka računalnu složenost, što ga čini manje prikladnim za velike grafove. Također, KL metoda može proizvesti particije koje nisu optimalne.

4.4 Louvain metoda

Louvain metoda je bazirana na modularnosti grafa. Ona koristi pohlepnu optimizaciju kako bi detektirala strukture zajednica u grafu [2]. Algoritam iterativno poboljšava particije tako da im optimizira modularnost (veličinu koja mjeri kvalitetu particije) i time povećava kvalitetu strukture zajednica.

Prednosti Louvain metode su velika učinkovitost i visoko kvalitetne particije s relativno malom računalnom složenosti. Ova tehnika je također primjenjiva u prepoznavanju zajednica u grafovima različitih gustoća. Nedostatak metode je velika osjetljivost na inicijalnu konfiguraciju, te može proizvesti drugačije rezultate za različite prolaske na istom grafu.

4.5 Girvan-Newman algoritam

Girvan-Newman algoriman je baziran na bridove grafa. On iterativno uklanja bridove koji se nalaze na najvećem broju putova koji povezuju vrhove grafa (kažemo da ti bridovi imaju visoku međupoloženost). Uzastopnim uklanjanjem bridova graf se razdvaja na manje zajednice koje čine particiju grafa [14]. Nakon svakog uklanjanja brida, algoritam računa modularnost.

Prednost Girvan-Newman algoritma je mogućnost otkrivanja hijerarhijske strukture zajednica uklanjanjem bridova visoke međupoloženosti. Glavni nedostatak ove metode je njena računalna složenost i nemogućnost detektiranja malih zajednica (ovo potonje može biti i prednost metode). Također, algoritam se oslanje na računanje međupoloženosti bridova, što često nije prikladno svojstvo bridova za

kvalitetno detektiranje zajednica.

Svaka od ranije navedenih tehnika ima niz prednosti i nedostataka, posebice kada je cilj prepoznati zajednice u velikim društvenim mrežama. Zajednički nedostatak većine spomenutih metoda je visoka računalna složenost, a to je potrebno izbjeći. Stoga treba težiti metodi čija je računalna složenost relativno malena, a takva metoda je Louvain metoda. Osim male računalne složenosti, takva metoda osigurava ravnotežu između učinkovitosti i kvalitete particioniranja, a sposobna je prepoznati strukture u grafovima visoke gustoće, što je posebno poželjno kada se radi o društvenim mrežama.

U nastavku ćemo se baviti Louvain metodom te ćemo navesti njenu primjenu u problemu koji promatramo: odnos pratitelja i praćenika u društvenim mrežama.

5 | Louvain metoda

Iako postoji velik broj algoritama za detektiranje zajednica u kompleksnim mrežama, mi ćemo se voditi razmatranjima iz prethodnog poglavlja te koristiti samo jedan, Louvain algoritam.

5.1 Povijest metode

Louvain metoda je prvi put predstavljena 2008. godine u radu nazvanom "Fast unfolding of communities in large networks" [2]. Algoritam je razvijen kao odgovor na sve veću potrebu za metodama koje učinkovito detektiraju zajednice u velikim mrežama, pogotovu u kontekstu društvenih mreža. Algoritam je dobio ime po sveučilištu na kojem je nastao, *Université catholique de Louvain*, a inspiriran je problemom detekcije zajednica u mrežama koristeći modularnost. Metoda se pokazala uspješnom na mrežama sa 100 milijuna vrhova i milijardu bridova.

5.2 Modularnost

Modularnost je veličina koja se odnosi na strukturu kompleksnih mreža, a mjeri jakost podjele mreže u grupe (zajednice ili module). Mreže s velikom modularnosti imaju guste zajednice, tj. zajednice s velikim brojem bridova, dok je broj bridova između vrhova koji pripadaju različitim zajednicama malen. Modularnost se često koristi u optimizacijskim metodama za otkrivanje zajednica u kompleksnim mrežama. Biološke mreže, uključujući neuronske mreže životinja, imaju visoku modularnost. Međutim, maksimiziranje modularnosti nije uvijek statistički konzisentno jer se može dogoditi da se pomoću nje u potpuno slučajnim grafovima otkriju zajednice. Također, otpimiziranjem modularnosti nećemo moći otkriti male zajednice.

Modularnost se pojavila još 1970. godine, kada je Mark Granovetter u svome radu [7] predstavio ideju "snage slabih veza" kako bi objasnio važnost slabih komponenata povezanosti u društvenim mrežama. To je stvorilo temelje za rad [14] u kojemu su Newman i Girvan predstavili modularnost kao formalnu mjeru koja služi za identifikaciju i evaluaciju struktura zajednica u kompleksnim mrežama.

Modularnost Q definirana je kao udio broja bridova koji se nalaze unutar zajednica u mreži umanjen za očekivani udio bridova u tim zajednicama kada bi oni bili raspoređeni slučajno. Vrijednost modularnosti nalazi se u intervalu $[-1/2, 1]$, a pozitivna je ako broj bridova unutar zajednica premašuje očekivani broj bridova

u zajednicama kada su oni u mreži slučajno raspoređeni. Formula za modularnost je

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

gdje je

- Q modularnost;
- A_{ij} je element matrice susjedstva mreže (u jednostavnoj mreži iznosi 1 ako postoji brid među vrhovima i i j, inače je jednak 0);
- k_i i k_j su redom stupnjevi vrhova i i j, a ukoliko je graf težinski, to su sume težina svih bridova incidentnih s vrhovima i, odnosno j;
- m je suma stupnjeva (suma težina bridova) u mreži;
- c_i i c_j predstavljaju zajednice kojima redom pripadaju vrhovi i i j;
- $\delta(c_i, c_j)$ je Kronecker delta funkcija koja je jednaka 1 ako $c_i = c_j$, inače je 0 (jednaka je 1 ako i i j pripadaju istoj zajednici, inače je 0).

Vidimo da se modularnost dobije sumiranjem razlike među stvarnim brojem bridova između vrhova A_{ij} i očekivanim brojem bridova $\frac{k_i k_j}{2m}$ za sve parove vrhova i, j. Kronceker delta funkcija $\delta(c_i, c_j)$ osigurava da jedino bridovi unutar iste zajednice doprinose modularnosti.

Mnogi algoritmi koriste modularnost za otkrivanje zajednica u mrežama. Dva često korištena algoritma su Newman-Girvan algoritam i Louvain metoda. Newman-Girvan algoritam računa modularnost tako da iterativno uklanja bridove s najvećom centralnosti međupoloženosti dok ne dođe do željenog broja zajednica. Louvain metoda koristi pohlepan algoritam tako da najprije traži male zajednice lokalnom optimizacijom modularnosti u mreži, a zatim spaja vrhove koji pripadaju istoj zajednici i gradi mrežu čiji su vrhovi te zajednice. U sljedećem odjeljku ćemo pomnije objasniti kako radi Louvain metoda.

5.3 Koraci metode

Kao što smo ranije istaknuli, Louvain algoritam je pohlepni optimizacijski algoritam koji se bazira na maksimizaciji modularnosti. Ovaj algoritam je iterativan i sastoji se od dva glavna koraka koji se ponavljaju sve dok modularnost ne konvergira do svoje maksimalne vrijednosti. Dizajniran je da radi na težinskim grafovima.

1. Lokalna optimizacija: Inicijalno, svaki vrh mreže je pridužen jedinstvenoj zajednici tako da je ukupan broj vrhova mreže jednak ukupnom broju zajednica. Zatim se svaki vrh pomiče u susjednu zajednicu tako da uzrokuje najveće moguće povećanje u modularnosti. To možemo lakše zamisliti ako svaka zajednica ima svoj indeks, vrh se fizički ne pomiče nigdje nego se indeks zajednice koji mu je dodijeljen promijeni. Nakon te promjene računa se modularnost cijele mreže i uspoređuje s modularnosti mreže prije promjene. Indeks promatranog vrha možemo promijeniti samo u indekse zajednica koje su mu susjedne. Kada isprobamo

sve moguće promjene, biramo onu koja daje najveću pozitivnu promjenu u modularnosti. Ovaj proces se ponavlja za sve vrhove sve dok više ne možemo povećati modularnost.

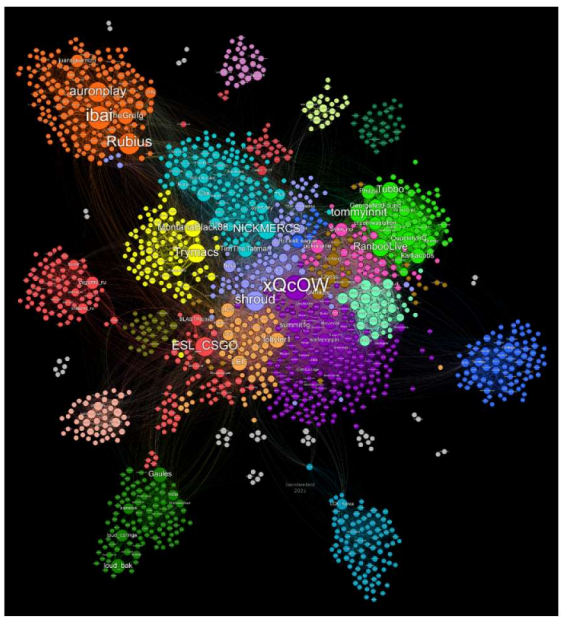
2. Agregacija zajednica: Nakon koraka lokalne optimizacije, svaku zajednicu tretiramo kao jedan vrh, te izgradimo novi graf od tih agregiranih vrhova. Težina bridova među novim vrhovima je dobivena sumom težina bridova među zajednicama. Ako graf nije težinski onda je ta suma jednaka broju bridova među određenim zajednicama. Na novom grafu ponavljamo korake 1 i 2, opet kreiramo novi graf i nastavljamo ponavljati korake 1 i 2 sve dok nema povećanja u modularnosti.

5.4 Prednosti i primjer

Louvain metoda ima nekoliko prednosti nad drugim metodama particioniranja grafa:

1. Učinkovitost: Louvain algoritam ima nisku računsku složenost, $O(n\ log(n))$, što znači da je vrlo učinkovit na velikim mrežama. Ovo svojstvo je ključno kod analiziranja društvenih mreža i drugih velikih kompleksnih sustava.
2. Prilagodljivost: Može se primijeniti na više vrsta grafova: usmjerenim, neusmjerenim, težinskim ili ne težinskim.
3. Modularnost: Algoritam je fokusiran na maksimizaciju modularnosti, što osigurava da će njegov rezultat biti relevantan i dati dobar uvid u organizaciju grafa.

Kako bi bolje demonstrirali prednost modularnosti, pogledajmo primjer grafa na kojemu je primijenjena Louvain metoda i analizirajmo njegova svojstva.



Slika 5.1: Particioniran graf društvene mreže Twitch [16].

Na slici 5.1 je prikazana mreža korisnika zvana Twitch. To je društvena mreža u kojoj su vrhovi korisnici, a bridovi spajaju korisnike koji su u prijateljskom odnosu. Korisnici ove mreže objavljuju videosnimke sebe. Najčešće su videosnimke vezane uz jednu temu, primjerice igranje videoigara, politika, kulinarstvo itd. Svaki korisnik je na danoj slici obojen bojom zajednice kojoj pripada, a zajednice su dobivene Louvain algoritmom. Iako nije dokumentirano na slici, kada bi istraživali svakog korisnika i kakve videosnimke on izrađuje, ubrzo bismo shvatili da ih je algoritam grupirao po kategorijama njihovih videosnimaka iako mu te informacije nisu bile dostupne. Korisnici čije su videosnimke fokusirane oko igranja neke igre grupirani su s korisnicima koji igraju istu tu igru. Neki korisnici znaju imati videosnimke koji obuhvaćaju više kategorija ili se znaju pojaviti u videosnimkama drugih korisnika ili izrađuju videosnimke s drugim korisnicima koji ne pripadaju savršeno u njihovu kategoriju. Ali čak i ti korisnici su dobro grupirani. Najčešće su grupirani u zajednice s ljudima s kojima često surađuju.

Louvain algoritam je uspješno pronašao zajednice unutar mreže iako podjela mreže na prvi pogled nije bila očita. Uz to, ovaj algoritam radi iznimno brzo, pogotovo u usporedbi s algoritmima slične primjene. Zato ćemo u sljedećem poglavlju izgraditi algoritam za preporuku prijatelja koji će koristiti upravo Lovuain metodu.

6 | Izgradnja algoritma za preporuku prijatelja

U ovome poglavlju ćemo se baviti algoritmom za preporuku baziranom na Louvain algoritmu. Temeljna ideja našeg algoritma je promatrati ego mrežu (ego graf) nekog korisnika koji će biti centralni vrh te mreže, te ju podijeliti u zajednice pomoću Louvain metode i generirati preporuke prijatelja s obzirom na to koliko jako korisnik pripada kojoj zajednici. Ego mreža će biti udaljenosti 2.

6.1 Ego mreže i pripadanje zajednici

Prvi korak našeg algoritma je izdvajanje ego mreže udaljenosti 2 ciljanog korisnika iz neke društvene mreže. S obzirom da razmatramo ego mrežu udaljenosti 2, ne uzimamo u obzir samo susjede korisnika nego i susjede njegovih susjeda. Takva persprektiva nam pomaže u razumijevanju šireg društvenog konteksta povezanog s našim ciljanim korisnikom. Ciljanog korisnika, odnosno centralnog vrha u našoj ego mreži biramo tako da on bude takozvani hub. Hub je vrh čiji je stupanj puno veći od prosječnog stupnja mreže.

Pojasnimo zašto je pogodno uzeti u obzir ego mrežu udaljenosti 2. Više je razloga. Jedan od razloga je previše mogućih kandidata za izbor prijatelja. U poznatom radu [12] analizirano je svojstvo takozvanog *malog svijeta* koje kaže da su svi ljudi šest ili manje socijalnih kontakata "udaljeni" jedan od drugog. Iz toga možemo zaključiti da povećavanjem dijametra naše ego mreže, broj vrhova u toj mreži raste eksponencijalno. Već za dijametar 3, broj mogućih kandidata za preporuku prlazi 10000. Toliki izbor kandidata je prevelik za praktičku primjenu, a također stvara preveliki šum kod evaluiranja podataka. Eksperimentom je utvrđeno [6] da postoji takozvani *horizont promatranja*. Sposobnost promatranja osobe je najčešće ograničena na ljude koji su u direktnom kontaktu ili koji imaju barem jedan kontakt zajedno. O ljudima koji su više udaljeni znatno je teže saznati čak i osnovne informacije. Zadnje svojstvo koje nam je posebno važno kod preporuke prijatelja je takozvano trijadsko zatvaranje (eng. Triadic closure) [7]. To svojstvno nam govori da ako je osoba A prijatelj sa osobom B, te je osoba B prijatelj s osobom C, to vrlo vjerojatno može rezultirati u uspostavi odnosa između osobe A i C. Taj koncept možemo proširiti na takozvano snažno trijadsko zatvaranje koje kaže da ako je čovjek B prijatelj s A i C, onda su A i C barem poznanici.

Nakon što smo izdvojili ego mrežu iz društvene mreže, koristimo Louvain algoritam za njenu particiju u disjunktne zajednice. Ove zajednice reprezentiraju

grupe vrhova koji su gušće povezani unutar grupe nego što su povezani s vrhovima izvan grupe. U kontekstu našeg algoritma za preporuku, ove zajednice mogu biti interpretirane kao razni društveni krugovi ili grupe kojima ciljani korisnik možda pripada, direktno ili indirektno kroz svoja poznanstva.

6.2 Mjerenje stupnja pripadanja zajednicama

Nakon što smo identificirali zajednice unutar ego mreže, potrebno je odrediti u kojoj mjeri ciljani korisnik pripada svakoj zajednici. Ova mjera će nam omogućiti da prioritiziramo preporuke bazirane na korisnikovoj pristranosti prema određenim zajednicama.

Da bismo izračunali razinu pripadanja tj. stupanj pripadanja, koristimo jednadžbu za optimiziranje modularnosti koja se koristi u Louvain algoritmu. Otprije znamo da je modularnost određene particije određena pomoću formule

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j).$$

Ciljanog korisnika uvrstimo redom u svaku zajednicu te zapamtimo vrijednost ukupne modularnosti grafa. U ovom koraku niti jednom drugom vrhu ne mijenjamo zajednicu osim centralnom (našem ciljanom korisniku). Primjerice, ako stavimo da centralni vrh pripada nekoj zajednici x, računanjem nove vrijednosti Q zaključujemo o afinitetu našeg centralnog vrha da pripada zajednici x.

6.3 Rangiranje korisnika unutar zajednica

Nakon što smo odredili do koje razine ciljani korisnik pripada kojoj zajednici, moramo rangirati korisnike unutar ovih zajednica kako bismo identificirali najrelevantnije prijatelje za preporuku. Uzimamo u obzir koliko prijatelja korisnici dijele s ciljanim korisnikom jer ovo može služiti kao idikator potencijalnog prijateljstva i dijeljenog interesa.

Za svakog korisnika unutar zajednice računamo ocjenu prijateljstva baziranu na broju zajedničkih prijatelja koji dijele ta osoba i naš ciljani korisnik. Tada normaliziramo ovu ocjenu tako da je podijelimo s ukupnim brojem prijatelja koje promatrani korisnik ima unutar te zajednice.

6.4 Kombiniranje ocjene prijateljstva i zajednice

Kako bismo generirali krajnju ocjenu preporuke, kombiniramo ocjenu zajednice (stupanj pripadanja nekoj zajednici) s normaliziranom ocjenom prijateljstva. Razmatranjem ova dva faktora prioritet dajemo korisnicima koji pripadaju istim zajednicama kao i ciljani korisnik i imaju veliki udio dijeljenih prijatelja unutar te zajednice.

Jedan od mogućih pristupa kombiniranja ovih ocjena je da izračunamo njihovu

prosječnu težinu, Primjerice, ako ocjena zajednice ima težinu α, a ocjena prijateljstva ima težinu $(1-\alpha)$, njihova kombinirana ocjena može biti izračunata pomoću formule:

$$\text{kombinirana ocjena} = \alpha \times \text{ocjena prijateljstva} + (1-\alpha) \times \text{ocjena zajednice}.$$

Optimalna vrijednost veličine α može biti ustanovljena empirijski, tako da testiramo razne vrijednosti i odaberemo one koje maksimiziraju rezultat našeg algoritma.

6.5 Evaluacija algoritma

U svrhu evaluiranja našeg algoritma za preporuku prijatelja, možemo koristiti unakrsnu provjeru (eng. cross-validation) izostavljenih korisnika. Nasumično uklonimo 10% prijatelja ciljanog korisnika, tj samo uklonimo bridove koji ih spajaju s našim ciljanim korisnikom. Vrhove izbacujemo jedino ako više nisu povezani s gigantskom komponentom povezanosti, tj. ako nemaju put do centralnog vrha. Zatim pokrenemo algoritam na modificiranoj ego mreži i generiramo preporuke prijatelja. Potom računamo omjer uklonjenih prijatelja koje algoritam uspješno preporuči tako što iterativno testiramo razne kombinacije ocjena zajednice i prijateljstva (prilagodbom vrijednosti faktora α). Možemo optimizirati algoritam tako da maksimiziramo broj uklonjenih prijatelja koje on preporuči. Ovaj proces osigurava prioritizaciju korisnika koji su relevantniji kao preporuke s obzirom na odabir ciljanog korisnika.

6.6 Programski kod

Kako bi proveli eksperiment i testirali rad našeg algoritma, odlučili smo odabrati programski jezik Python te u njemu provesti cijeli eksperiment. Za rad s grafovima odlučili smo koristiti okvir zvan NetworkX [13]. To je programski paket koji služi za stvaranje, analizu i obradu kompleksnih mreža. Funkcije i objekte koje ćemo koristiti iz tog paketa, objasnit ćemo na mjestu gdje ćemo ih i koristiti. Za primjenu Louvain algoritma koristit ćemo paket Community [4], koji se ponaša kao proširenje NetworkX paketa. Pakete učitavamo s kodom:

```
1 import networkx as nx
2 from community import community_louvain
```

Pomoću ovog koda pozivamo NetworkX s aliasom *nx* te iz paketa *Community* unosimo *community_Louvain* što je implementirani Louvain algoritam.

Kod će biti prezentiran redom kojim bi se algoritam izvršavao. Za svaku veću cjelinu algoritma smo napravili posebnu funkciju koju ćemo kasnije primjenjivati na graf njenim pozivanjem.

6.6.1 Induciranje ego grafa

Ovaj dio koda je jako jednostavan. Njegova cijela implementacija se nalazi unutar paketa NetworkX te ga samo moramo primijeniti pozivanjem.

```
G = nx.generators.ego_graph(G, center_node, radius=2)
```

Primijetimo da se funkcija za induciranje ego grafa nalazi na putu *.generators.ego_graph* unutar *nx*-a. Ona prima tri argumenta: graf *G*, indeks ciljanog korisnika *center_node* i veličinu *radius* koja predstavlja radijus naše ego mreže. Nakon izvršavanja funkcije, novi ego graf zamjenjuje naš početni graf u varijabli *G*.

6.6.2 Odvajanje prijatelja

Prije nego počnemo s particioniranjem i daljnjom obradom grafa, moramo zapamtiti prijatelje našeg korisnika radi lakše evaluacije i analize u narednim koracima. Odvajanje prijatelja radimo na jednostavan način:

```
Friends = [n for n in G.neighbors(center_node)]
```

U ovome kodu koristimo jednu od metoda iz NetworkX zvanu *neighbors*. Metoda prima jedan argument, indeks nekog vrha, a vraća sve susjede tog vrha. Sve susjede stavljamo u varijablu *friends*. No, to nisu jedini prijatelji koje želimo zapamtiti. Sjetimo se da radi kasnije evaluacije, moramo nasumično ukloniti 10% prijatelja korisnika. Zato stvaramo dvije funkcije. Jedna je *decision* koja odlučuje hoćemo li zadržati prijatelja, a druga funkcija je *remove_random_friends* koja koristi funkciju *decision* i pomoću nje uklanja nasumične prijatelje.

Slijedi jednostavna funkcija koja koristi paket *random*. Taj je paket jedan od ugrađenih Python paketa za generiranje slučajnih vrijednosti:

```
import random

def decision(probability):
  return random.random() < probability
```

Iz *random* paketa koristimo funkciju *random* koja generira slučajan broj između 0 i 1. Taj broj usporedimo sa željenom vjerojatnošću i vračamo *True* ako je manji od vjerojatnosti i *False* ako nije.

Slijedi funkcija nasumičnog uklanjanja prijatelja.

```

import math

def remove_random_friends(G, center_node, percentage=0.1):

```

```
6   list_of_edges = G.out_edges([center_node])
7   num_of_edges = len(list_of_edges)
8
9   RandomSample = random.sample(list(list_of_edges), math.floor(
      num_of_edges*percentage))
10  G.remove_edges_from(RandomSample)
11
12  Friends_after_removal = [n for n in G.neighbors(center_node)]
13
14  return G, Friends_after_removal
```

Ovu funkcija prima tri argumenta: graf *G*, indeks središnjeg vrha *center_node* i postotak susjeda koje želimo izbaciti *percentage*. Da bismo dobili prijatelje našeg ciljanog korisnika, koristimo funkciju *out_edges* iz NetworkX-a koja vraća sve usmjerene bridove koji počinju s danim vrhom. Zatim uzimamo slučajan uzorak liste bridova koju nam je ta funkcija dala, ponovno koristeći paket *random* i njegovu funkciju *sample*. Funkcija *sample* prima listu i broj elemenata koje treba slučajno odabrati. Taj broj dobijemo tako da duljinu liste pomnožimo s vrijednosti *percentage*. Kako rezultat te operacije ne mora biti cijeli broj, na njega primijenimo funkciju $math.floor(x)$ koja je ekvivalentna dobro poznatoj funkciji $\lfloor x \rfloor$.

Nakon što znamo slučajno odabrane susjede, koristimo metodu *remove_edges_from* koja prima listu bridova koje uklanja iz grafa na kojemu je pozvana. Zatim, na isti način na koji smo zapamtili prijatelje prije nasumičnog uklanjanja, pamtimo prijatelje nakon uklanjanja. Prijatelje spremimo u varijablu $Friends\_after\_removal$. Funkcija vraća modificirani graf i varijablu *Friends_after_removal*.

6.6.3 Particioniranje grafa

Kao što smo spomenuli, za primjenu Louvain algoritma koristimo paket *Community* tj. koristimo klasu unutar njega koja se zove *community_Louvain*.

```
1
2 def partition_the_graph(G, center_node):
3
4   saved_edges = list(G.edges(center_node))
5   G.remove_node(center_node)
6
7   G = G.subgraph(max(nx.weakly_connected_components(G), key=len)).
      copy()
8   G = G.to_undirected()
9   partition = community_louvain.best_partition(G)
10
11  partition[center_node] = 0
12  G.add_edges_from(saved_edges)
13
14  return G, partition
```

Prvo moramo privremeno ukloniti promatranog korisnika iz grafa kako ne bi utjecao na detekciju zajednica. Da bismo ga ponovno vratili, moramo zapamtiti

sve bridove koji su bili uklonjeni zajedno s njim. Te bridove spremamo u varijablu *saved_eges* pa uklanjamo centralni vrh. Kako uklanjanjem tih bridova postoji mogućnost da neki vrhovi više nisu povezani s ostatkom grafa, njih izbacujemo tako da uzimamo samo najveću slabu komponentu povezanosti grafa i zamjenjujemo naš graf s njom. To radimo pomoću tri funkcije: *nx.weekly_connected_components* koja vraća sve slabe komponente povezanosti u grafu, *max* koja bira najveću od tih komponenti s obzirom na veličinu koju označavamo s *len* i *subgraph* koja prima vrhove nekog grafa i vraća podgraf induciran tim vrhovima.

Sljedeći korak je primjena Louvain algoritma. Njega primijenimo pozivanjem funkcije *best_partition* koja vraća particiju grafa *partition*. Objekt *partition* je zapravo lista uređenih parova (v, c) u kojima je v indeks vrha i c indeks zajednice kojoj taj vrh pripada. Particija je optimizirana Louvain algoritmom na način koji smo naveli u prijašnjem poglavlju.

Zadnji korak je vraćanje središnjeg vrha u graf. S obzirom da je graf particioniran, središnji vrh moramo staviti u zajednicu. Odlučujemo se za onu čiji je indeks 0 (jer za nju znamo da će postojati). Zatim koristeći funkciju *add_edges_from* vraćamo u graf sve bridove iz varijable *saved_edges*, zatim vraćamo modificirani graf i particiju.

6.6.4 Ocjenjivanje zajednica

Kao što smo ranije naveli, zajednice ocjenjujemo tako da u svaku zajednicu redom dodamo centralni vrh i računamo modularnost.

```
1
2  def number_of_communities(partition):
3    comm_num = 0
4
5    for elem in partition.items():
6      if elem[1] > comm_num:
7        comm_num = elem[1]
8
9    return comm_num
10
11 def get_modularity_scores(G, partiton, center_node):
12
13   comm_num = number_of_communities(partition)
14
15   modularity_list = []
16
17   for i in range(comm_num+1):
18     partition[center_node] = i
19
20     modularity_list.append(community_louvain.modularity(partition,
       G))
21
22   modularity_list_original = modularity_list.copy()
23   modularity_list = [x for x in enumerate(modularity_list)]
```

```
24
25    modularity_list.sort(key=lambda tup: -tup[1])
26
27    return modularity_list
```

Indeksi zajednica zadani su redom od 0 do nekog broja pa moramo prvo pobrojati broj zajednica u particiji. To radimo funkcijom *number_of_communities*. Zatim incijaliziramo listu *modularity_list* u koju ćemo spremati izračunate modularnosti. Redom *center_node* dodajemo u particiju s proizvoljnim indeksom zajednice i računamo modularnost. To radimo pomoću metode *modularity* koja prima particiju i graf kojemu računa modularnost. Kad je modularnost izračunata, spremamo ju u *modularity_list*.

Nakon što smo izračunali modularnost dodavajući centralni vrh svakoj od dobivenih zajednica, u listu dodajemo njihov indeks koristeći funkciju *enumerate*. Potom listu modularnosti sortiramo silazno po modularnosti i vraćamo *modularity_list*.

6.6.5 Biranje prospekata

Kako bi ocijenili moguće kandidate za preporuku, prvo trebamo znati koje kandidate ocjenjujemo. Primjerice, ne trebamo ocjenjivati trenutne prijatelje (oni koji su ostali nakon uklanjanja 10% nasumičnih).

```
1  def choose_prospects(G, partition, Friends_after_removal):
2
3    nodes_to_evaluate = dict([(x, 0) for x in G.nodes])
4
5    for node in G.nodes:
6    if not (node in Friends_after_removal):
7      for fren in Friends_after_removal:
8        if (G.has_edge(node,fren) or G.has_edge(fren,node)) \
9      and partition[node] == partition[fren]:
10       nodes_to_evaluate[node] += 1
11
12   else:
13     del nodes_to_evaluate[node]
14
15   return list(nodes_to_evaluate.items())
```

Svrha funkcije *choose_prospects* je da iz grafa izvuče sve vrhove koji nisu susjedni s ciljanim korisnikom te da sumira sve prijatelje koje oni dijele s ciljanim korisnikom. Prvo inicijaliziramo varijablu *nodes_to_evaluate* na listu uređenih parova $(vrh, 0)$ u kojima *vrh* označava indeks vrha. Indekse dobijemo iz *G.nodes*. Zatim za svaki vrh u grafu provjeravamo je li on prijatelj našeg ciljanog korisnika. Ako nije, obrišemo ga iz liste. Ako jest, sumiramo sve zajedničke prijatelje unutar zajednice i stvorimo ranije navedeni uređeni par. Sumiramo tako da za svakog prijatelja centralnog vrha gledamo ima li isti indeks zajednice kao trenutno promatrani vrh i ima li barem jedan brid koji dijeli s njim. Vraćamo listu *nodes_to_evaluate*.

6.6.6 Rangiranje prospekata

Sljedeći korak je kombiniranje ocjena iz naše *modularity_list* liste s normaliziranim brojem dijeljenih prijatelja svakog vrha unutar zajednice. Za ovaj dio napravili smo funkciju *get_ranking* koja prima listu modularnosti i vraća listu rangiranih korisnika/vrhova.

```
1
2 def get_ranking(G, modularity_list, nodes_to_evaluate, alpha = 0.1)
    :
3
4   min_modularity = min(modularity_list, key = lambda t: t[1])[1]
5   new_modularity = [(x, math.log(y/min_modularity)) for x, y in
     modularity_list]
6
7   modularity_list.sort(key=lambda tup: tup[0])
8   modularities_by_node = [y for x, y in new_modularity]
9
10  follower_ranking = []
11  for x, y in nodes_to_evaluate.items():
12    score = (y/(sum([1 for _ in G.neighbors(x)])+1))*alpha \
13      + modularities_by_node[partition[x]]*(1-alpha)
14  follower_ranking.append((x, score))
15
16  follower_ranking.sort(key=lambda tup: -tup[1])
17  return follower_ranking
```

S obzirom da su vrijednosti modularnosti međusobno vrlo bliske, moramo ih nekako istaknuti. Najprije sve modularnosti dijelimo s najmanjom modularnosti u listi. To radimo jer se za drugačije grafove ocjena modularnosti zna puno razlikovati. Ovako ih sve svedemo na broj koji je bliže 1. Zatim, da naglasimo razlike logaritmiramo dobivenu vrijednost. Listu modularnosti sortiramo po indeksima zajednica. Inicijaliziramo praznu listu rangiranih korisnika ($follower\_ranking$) i iteriramo kroz prospekte za evaluaciju. Za svakog prospekta primijenimo ranije navedenu formulu

$$\text{kombinirana ocjena} = \alpha \times \text{ocjena prijateljstva} + (1-\alpha) \times \text{ocjena zajednice}$$

i u listu $follower\_ranking$ dodajemo uređeni par $(vrh, kombinirana\_ocjena)$. Nakon što prođemo svaki od prospekata, listu $follower\_ranking$ sortiramo silazno po ocjeni. Ta sortirana lista je rezultat našeg algoritma.

6.6.7 Rezultati

Nakon što smo algoritam nekoliko puta pokrneuli na nekoliko slučajno odabranih korisnika društvene mreže, pokazalo se da je optimalna vrijednost faktora α oko 0.1.

Kako bismo evaluirali rad algoritma, moramo ograničiti količinu prijatelja koje planiramo preporučiti, tj. moramo uzeti samo nekoliko najboljih vrijednosti iz rangirane liste. Ovo radimo zato što želimo provjeriti koliko rano će naš algoritam

preporučiti jednog od uklonjenih prijatelja. Odlučili smo se ograničiti na prvih 100 rezultata i tesitranjem u 10 ego grafova dobili smo vrijednosti:

uklonjeno	preporučeno	ukupno
24	9	2295
17	5	1988
22	6	2119
29	6	2331
14	7	1382
14	7	1382
12	6	1197
18	7	1785
26	5	2326
16	9	1896

Tablica 6.1: Najboljih 100 rezultata (koristeći $\alpha = 0.1$).

Stupac *uklonjeno* predstavlja broj prijatelja koje smo nasumično uklonili. Stupci *preporučeno* i *ukupno* su brojevi koji označavaju koliko je uklonjenih prijatelja bilo preporučeno i koliki je ukupni broj kandidata za preporuku. Kao što vidimo, dobar dio prijatelja je preporučen unutar prvih 100 rezultata. Kada bi preporuke rangirali samo preko sume zajedničkih prijatelja (u cijelom grafu), najčešće u najboljih 100 preporuka ne bi bio niti jedan uklonjeni prijatelj.

7 | Zaključak

Ovaj rad je prezentirao uspješnu implementaciju Louvain algoritam u sustavu za preporuku prijatelja. Pokazana je njegova prednost nad pristupom koji gleda samo broj zajedničkih prijatelja. Možemo zaključiti da je algoritam učinkovit u primjenama, pogotovo u slučajevima kada su kontekstualni podatci ograničeni, tj. kada imamo samo običan graf bez težina ili klasifikacije vrhova. Kako god, važno je naglasiti da društvene mreže često sadrže bogatije kontekstualne informacije o korisnicima i njihovim vezama. Takve informacije možemo iskoristiti da razvijemo još sofisticiraniji pristup koji možemo iskoristiti na toj specifičnoj primjeni. Potrebno je dodatno istraživanje kako bi se otkrio potencijal inkorporiranja ovih dodatnih podataka u poboljšanje performanse ovog sustava.

Literatura

[1] K. Andreev, H. Räcke, *Balanced Graph Partitioning*, Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (2004), 120–124.

[2] V. D. Blondel, J-L. Guillaume, R. Lambiotte, E. Lefebvre, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment **2008** (2008).

[3] Fan R. K. Chung, *Spectral Graph Theory*, American Mathematical Society **92** (1997).

[4] Community proširenje za NetworkX dokumentacija
`https://python-louvain.readthedocs.io/en/latest/`

[5] L. Euler, *Solutio problematis ad geometriam situs pertinentis*, Commentarii academiae scientiarum Petropolitanae **8** (1736), 128–140.

[6] N. E. Friedkin, *Horizons of Observability and Limits of Informal Control in Organizations*, Social Forces **62** (1983), 54–77.

[7] M. Granovetter, *The Strength of Weak Ties*, American Journal of Sociology **78** (1973), 1360–1380.

[8] J. J. Hein, *Example 3: The Handshaking Problem p. 703*, Discrete Structures, Logic, and Computability, Jones & Bartlett Publishers 2015.

[9] B. Hendrickson, R. Leland, *A multilevel algorithm for partitioning graphs*, Proceedings of the 1995 ACM/IEEE conference on Supercomputing. ACM. p. 28. (1995).

[10] B. W. Kernighan, S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Technical Journal **49** (1970), 291–307.

[11] H. Meyerhenke, P. Sanders, C. Schulz, *Partitioning Complex Networks via Size-Constrained Clustering*, Experimental Algorithms **8504**, 2014.

[12] S. Milgram, *The Small World Problem*, Psychology Today by Ziff-Davis Publishing Company **2** (1967), 60–67.

[13] NetworkX dokumentacija
`https://networkx.org/documentation/stable/index.html`.

[14] M. E. J. Newman, M. Girvan, *Finding and Evaluating Community Structure in Networks*, Physical review E **69**(2004), 26113.

[15] D. Veljan, *Kombinatorna i diskretna matematika*, Algoritam, 2001.

[16] Vizualizacija twitch zajednica
`https://github.com/KiranGershenfeld/VisualizingTwitchCommunities`.

Sažetak

U ovom radu razvili smo algoritam za preporuku prijatelja koristeći Louvain metodu. Najprije smo naveli osnovne pojmove u teoriji grafova, a zatim smo detaljno proučili povezanost i particioniranje grafova. Pokazali smo kako funkcionira algoritam preporuke i pokazali njegovu učinkovitost u generiranju preporuka prijatelja u zadanoj mreži.

Ključne riječi

ego mreža, preporuka prijatelja, particioniranje grafa, modularnost, Louvain algoritam.

Recommendation algorithm in social networks

Summary

In this paper, we developed a friend recommendation algorithm utilizing the Louvain method. Initially, we provided an overview of fundamental concepts in graph theory, followed by an in-depth examination of connectedness and graph partitioning. Building on this foundation, we designed the proposed algorithm and demonstrated its effectiveness in generating friend recommendations.

Keywords

ego network, friend recommendation, graph partitioning, modularity, Louvain algorithm.

Životopis

Rođen sam 12. siječnja 1999. godine u Vinkovcima gdje sam i završio osnovnu školu Ivana Mažuranića. Zatim sam se upisao u Tehničku školu u Vinkovcima. Nakon srednje škole, 2017. godine upisao sam se na preddiplomski studij Matematike i računarstva na Odjelu za matematiku. Završio sam studij izradom završnog rada "Analiza sentimenta" kod mentora doc. dr. sc. Domagoja Ševerdije. Odmah nakon završetka preddiplomskog studija, upisao sam diplomski studij Matematike i računarstva na istome odjelu. Na diplomskom studiju odradio sam stručnu praksu za tvrtku TalentLyft.