SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1188

ALGORITMI OTKRIVANJA ZAJEDNICE U MODELU GRAF BAZE PODATAKA ZA SUSTAV PREPORUKE

Lovro Malojčić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1188

ALGORITMI OTKRIVANJA ZAJEDNICE U MODELU GRAF BAZE PODATAKA ZA SUSTAV PREPORUKE

Lovro Malojčić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 1188

Pristupnik:	**Lovro Malojčić (0036532080)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Mario Brčić

Zadatak: **Algoritmi otkrivanja zajednice u modelu graf baze podataka za sustav preporuke**

Opis zadatka:

Sustavi preporuke sadržaja imaju općeprisutnu primjenu u digitalnim platformama te im je glavna namjena ponuditi personalizirani sadržaj prilagođen profilu korisnika. Kako bi se takav sustav implementirao potrebno je prikupiti mnoštvo podataka o korisnicima i njihovim navikama pri čemu se javlja problem pohrane i dohvata te količine podataka. U sustavima preporuke sa kolaborativnom filtriranju (engl. collaborative filtering recommender system) potrebno je, iz skupa podataka o korisnicima, identificirati slične korisnike te donositi preporuke sukladno ponašanju sličnih korisnika. Za otkrivanje sličnih korisnika, odnosno zajednica korisnika, koriste se algoritmi za otkrivanje zajednica (engl. community detection algorithms). U ovom će radu biti napravljena usporedba različitih algoritama za otkrivanje zajednica nad podacima koji su pohranjeni u graf bazi podataka. Za provedbu ove usporedbe bit će potrebno prikupiti odgovarajući skup podataka o korisnicima te provesti niz transformacija kako bi se isti pohranili u graf bazu podataka. Nakon toga će se implementirati različiti algoritmi otkrivanja zajednice te će se napraviti detaljna analiza njihovih performansi.

Rok za predaju rada: 9. lipnja 2023.

Sadržaj

Uvod

Graf se u teoriji grafova definira kao skup čvorova i veza koji opisuju odnose između objekata. Pojam zajednice u grafu označava skup čvorova koji je po nečemu grupiran, najčešće po vezama, kada su čvorovi unutar zajednica više povezani nego s drugim zajednicama, ili po svojstvima čvora. Neke uporabe zajednica su u sustavima za preporučivanje sadržaja/proizvoda (traženjem sličnih korisnika ili proizvoda), detektiranje ekstremističkih zajednica ili drugi slučajevi gdje je potrebno grupiranje korisnika/čvorova. Zajednice u grafovima su pogodne za istraživanje društvenih mreža, zato što možemo promatrati zajednice kao meta-čvorove, koji nam olakšavaju proučavanje cjelokupnog grafa [1].

Jedna od podjela algoritama za otkrivanje zajednica je dijeljenje na statičke i dinamičke. Statički se izvode na nepromjenjivim grafovima, tj. moraju se ponovo izvoditi u slučaju promjena u grafu. Dinamički algoritmi mogu pratiti promjene u grafovima, bez da moraju ponovo razmatrati cijeli graf [2]. U ovom radu proučavat će se statični algoritmi.

Algoritmi za statične grafove dijele se na tri vrste: algoritmi za zajednice bez preklapanja, s preklapanjem i neizrazite zajednice, ovisno o pripadnosti čvorova zajednicama. U algoritmima za zajednice bez preklapanje (engl. *crisp*) svaki čvor pripada isključivo jednoj zajednici. Algoritmi za zajednice koje se preklapaju (engl. *overlapping*) omogućavaju čvorovima pripadnost više od jedne zajednice. Odlika te dvije vrste algoritama je da je pripadnost zajednicama binarna, čvor može ili pripadati zajednici ili ne. Za razliku od toga, kod algoritama za neizrazite zajednice (engl. *fuzzy*) čvorovi mogu s određenom količinom neizvjesnosti pripadati nekoj zajednici [2]. U ovom radu proučavat će se algoritmi za zajednice bez preklapanja.

U poglavlju 1 opisana je programska podrška korištena u radu, u poglavlju 2 opisani su grafovi u kojima su tražene zajednice, u poglavlju 3 definirani su korišteni algoritmi, u poglavlju 4 opisane su metrike korištene za evaluaciju pronađenih zajednica, u poglavlju 5 definirana je implementacija algoritama, a u poglavlju 6 opisani su i interpretirani dobiveni rezultati traženja zajednica.

1. Programska podrška

U ovom poglavlju bit će predstavljena programska podrška korištena u radu. Ona je korištena za izvršavanje pronalaženja zajednica, evaluaciju dobivenih rezultata i prikaz tih rezultata.

1.1. Programski jezik i biblioteke

Za programski jezik odabran je *Python*, zbog lakoće pisanja koda i dostupnih biblioteka. Biblioteka odabrana za implementacije algoritama za pronalaženje zajednica je bila *CDLib* [3], zato što posjeduje mnogo različitih algoritama i metrika za ispitivanje pronađenih zajednica te omogućava uvoženje i generiranje sintetičkih grafova pogodnih za ispitivanje kvalitete algoritama.

Za grafove korištena je biblioteka *NetworkX* [4], zbog svoje široke uporabe, što olakšava pronalaženje literature i unaprijed generiranih grafova te zato što je dobro integrirana s *CDLib*-om i ostalim korištenim bibliotekama. Za crtanje grafova korištena je biblioteka *Matplotlib* [5]. Za mjerenje vremena izvođenja algoritama korištena je standardna *Python* biblioteka *timeit*. *Timeit* je odabran (umjesto biblioteke *time*) zato što koristi najprecizniji mjerač vremena, prilagođeno sustavu na kojem se izvodi, te zato što prilikom mjerenja onemogućava izvršavanje nevezanih naredbi, zbog čega je korisniji za potrebe preciznog mjerenja vremena [6].

Jedan graf je bio uvezen iz graf baze podataka *Memgraph* [7], pa se za spajanje na tu bazu i izvođenje potrebnih upita koristila biblioteka *GQLAlchemy* [8].

1.2. Virtualna okruženja u Docker-u

Vrlo rano u radu postalo je očito da će biti potrebno kod izvršavati u više verzija *Python*-a, zbog konfliktnih potreba različitih biblioteka. Za tu svrhu korišteni su *Docker* kontejneri u kojima je lagano odabrati željenu verziju [9].

S obzirom na to da postoje službene verzije kontejnera za *Python*, one su i korištene. No, postoji više različitih verzija za svaku verziju *Pythona* [10]. Najraširenija verzija

kontejnera koristi *Alpine Linux* distribuciju, zato što zahtjeva iznimno malo resursa, tj. operacijski sustav nema puno mogućnosti, ali je vrlo efikasan zato što ne mora izvoditi mnogo pozadinskih procesa. Iako je *Alpine* verzija vrlo dobra za većinu slučajeva, u zadanom nije odgovarala zato što nedostaju potrebne binarne datoteke za instalaciju pojedinih biblioteka.

Na kraju je odabran *Buster* kontejner za *Python 3.8*. *Buster* je ime 10. verzije distribucije *Debian*, a pogodan je zato što posjeduje sve potrebne binarne datoteke i ne zahtjeva mnogo resursa.

Tijekom implementacije, kontejneri su bili na *Windows 10* računalu s *AMD Ryzen* procesorom, koristeći *Ubuntu WSL*, i na *MacOS* laptopu s M1 procesorom. Konačni rezultati su dobiveni na *MacOS*-u. Na *MacOS*-u kontejner nije došao sa unaprijed instaliranim *CMake*-om, pa se prije instalacije biblioteke *leidenalg*, koja služi za implementaciju jednog od algoritama, morao ručno dobaviti [11]. Crtanje grafova izvršeno je izvan *Docker* kontejnera, koristeći *Python 3.10*.

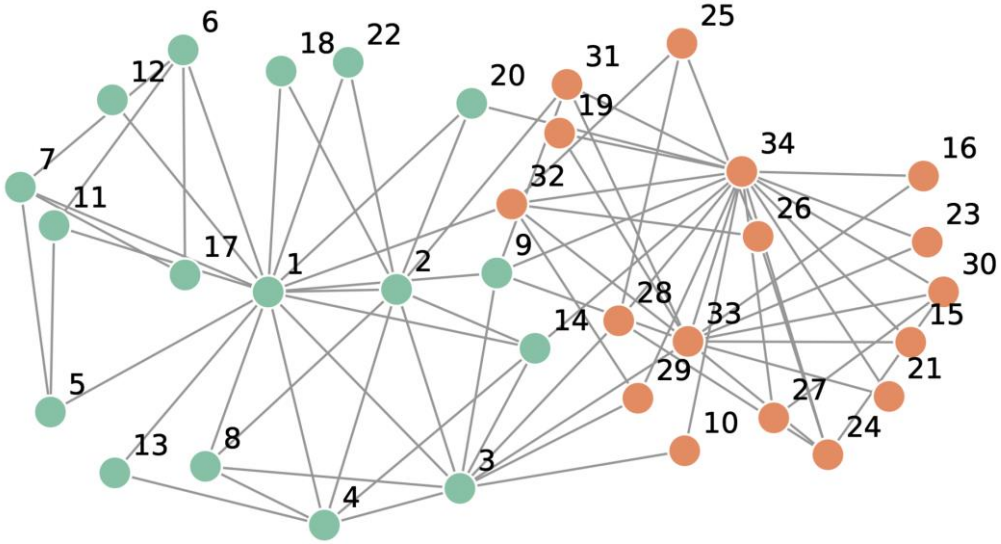
2. Grafovi

U ovom poglavlju bit će opisani grafovi korišteni u radu. Grafovi se dijele na stvarne, koji opisuju podatke uzete iz stvarnog svijeta, i sintetičke, koje računalo generira u svrhu testiranja algoritama za pronalaženje zajednica.

2.1. Stvarni grafovi

2.1.1. Karate klub

Ovaj graf je prikaz mreže poznanstava u jednom sveučilišnom karate klubu između 1970. i 1972. Graf je definirao William W. Zachary u svom radu iz 1977. [12] Njegov rad bavio se raskolom u klubu nakon svađe administratora i učitelja, tj. predviđanjem u koju zajednicu će se svrstati članovi nakon svađe. Koristeći Ford-Fulkerson algoritam, pohlepni algoritam koji maksimizira protočnost grafa, Zachary je točno predvidio kako će se opredijeliti svi osim jednog člana. Zachary-jeva hipoteza može se vidjeti na slici (Sl. 2.1).



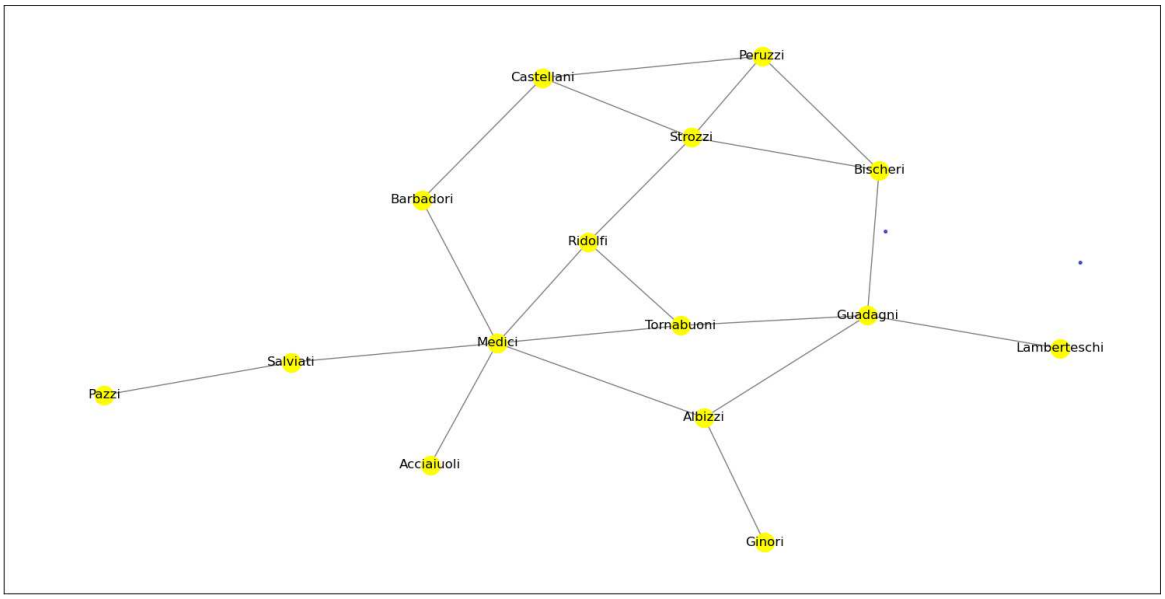
Sl. 2.1 Zacharyjevo predviđanje ishoda raskola [13]

Graf je postao popularan kao primjer zajednica nakon što su ga u radu iz 2002. koristili Michelle Girvan i Mark Newman [14]. Kasnije je graf postao toliko raširen, da danas postoji tzv. *Karate club club* koji dodjeljuje članstvo u klubu i pehar istraživaču koji prvi koristi taj graf kao primjer na nekoj konferenciji o grafovima.

Graf se sastoji od 34 člana/čvora, koji su povezani sa 77 ili 78 veza, ovisno o verziji, zbog nejasnoća u originalnom radu. Korištena implementacija ima 78 veza.

2.1.2. Graf obitelji u Firenci

Ovaj graf sastoji se od 15 obitelji iz Firence, gdje su obitelji povezane ako postoji brak između njihovih članova. Graf je napravljen na bazi podataka koje su sakupili Breiger i Pattison [15]. Prikaz grafa može se vidjeti na slici (Sl. 2.2).

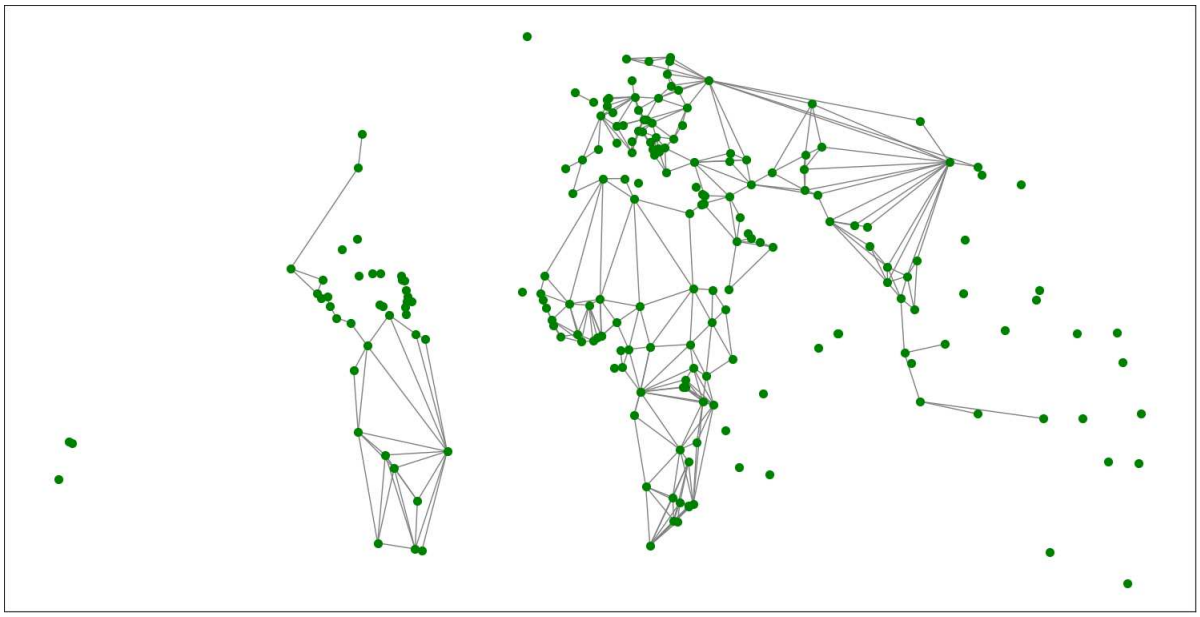


Sl. 2.2 Prikaz grafa obitelji

Cilj traženja zajednica na ovom grafu je predviđanje koje bi obitelji mogle pobijediti u borbi za političku kontrolu grada, ukoliko pretpostavimo da ukoliko su povezane da je vjerojatnije da će se međusobno podržavati. Graf se sastoji od 15 čvorova i 20 veza između njih.

2.1.3. Graf glavnih gradova

Graf je preuzet iz graf baze podataka *Memgraph* [7]. Sastoji se od glavnih gradova država, koji su povezani ako su unutar iste države (zbog država koje imaju više glavnih gradova) ili ako države imaju kopnenu granicu. Prikaz grafa vidljiv je na slici (Sl. 2.3).



Sl. 2.3 Prikaz grafa glavnih gradova

Mogući nedostatak grafa je da grafovi otočnih država nisu povezani, što bi moglo uzrokovati naočigled besmisleno grupiranje, ukoliko koristimo algoritme koji koriste veze u grafu kako bi definirali zajednice. Graf se sastoji od 212 gradova/čvorova i 369 veza između njih.

2.1.4. Amazon graf

Za primjer većeg grafa iz stvarnog svijeta odabran je graf sličnih proizvoda na Amazonu. Graf je napravljen koristeći „*Customers Who Bought This Item Also Bought*“ opciju na stranici pojedinog proizvoda, gdje su proizvodi povezani ako su prikazani u tom dijelu izbornika [16]. Graf se sastoji od 334863 proizvoda/čvora i 925872 veza između njih.

2.2. Sintetički grafovi

2.2.1. LFR grafovi

U ovom radu korišteni su Lancichinetti–Fortunato–Radicchi (nadalje LFR) grafovi mjerila za testiranje kvalitete pronađenih zajednica. Ti grafovi su pogodni zato što dolaze s unaprijed određenim idealnim zajednicama pa se algoritmi mogu uspoređivati po tome koliko su slični tim zajednicama. Algoritam za stvaranje tih grafova definirali su Lancichinetti, Fortunato i Radicchi u radu iz 2008. [17].

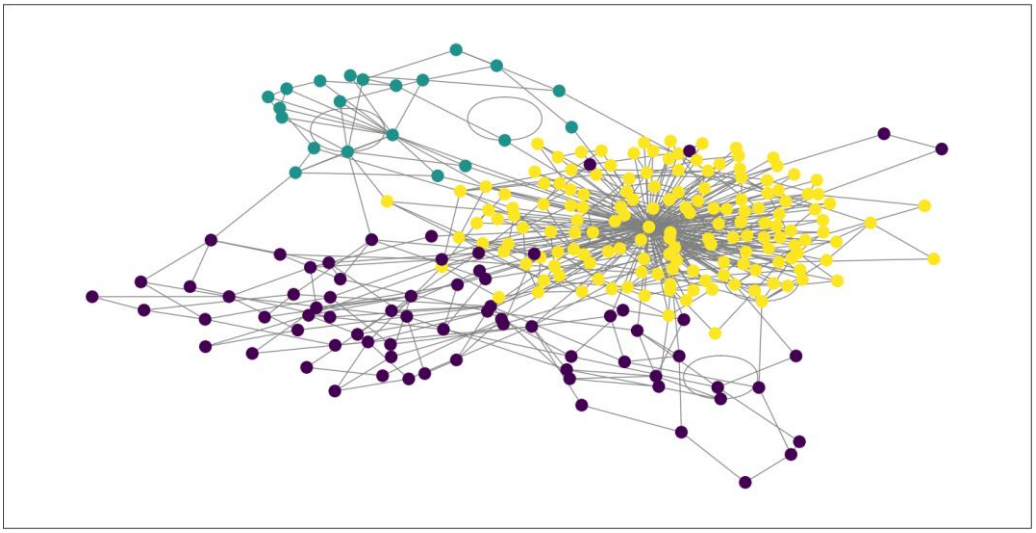
Na početku se definiraju broj čvorova (N), prosječni stupanj čvora (k) i parametar miješanja (μ). Parametar miješanja predstavlja prosječni postotak susjeda nekog čvora koji ne pripadaju zajednici tog čvora, tj. predstavlja količinu buke ili odstupanja od idealnih zajednica.

U prvom koraku generiraju se čvorovi po eksponencijalnoj distribuciji s unaprijed definiranim eksponentom (γ ili *tau1*), tako da odgovaraju definiranom k. Ovisno o implementaciji, mogu se odrediti minimalni i maksimalni stupanj čvora koji dodatno ograničavaju distribuciju stupnjeva.

U drugom koraku generiraju se veličine zajednica, isto po eksponencijalnoj distribuciji s unaprijed definiranim eksponentom (β ili *tau2*).

Zatim se svaki čvor nasumično dodjeli nekoj zajednici, ako definirane veličine zajednica i stupnja to dopuštaju, inače čvor ostaje „beskućnik“. Ako bi se dodavanjem čvora u neku zajednicu premašila zadana veličina zajednice, nasumično se izbacuje čvor iz zajednice. Iteracija po čvorovima prestaje kada su sve zajednice pune/svi čvorovi pripadaju jednoj zajednici.

U zadnjem koraku definiraju se veze između čvorova tako da njihovi stupnjevi ostanu isti, a da postotak veza izvan zajednice za svaki čvor bude približno μ. Primjer LFR grafa s $N=250$, $tau1=3$, $tau2=1.5$, $\mu=0.1$ i $k=5$ vidi se na slici (Sl. 2.4).



Sl. 2.4 Primjer LFR grafa

3. Algoritmi za pronalaženje zajednica

U ovom poglavlju bit će opisano pet algoritama za pronalaženje zajednica bez preklapanja, koji su kasnije u radu implementirani. Tih pet algoritama su: Louvain, Leiden, širenje oznaka, *infomap* i *Constant Potts Model*.

3.1. Louvain

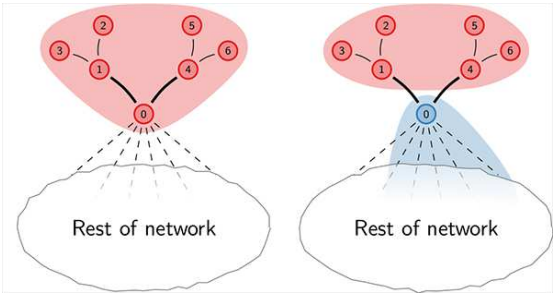
Ovaj algoritam razvijen prvi put je opisan u radu istraživača sa sveučilišta u Louvainu 2008. [18]. Cilj algoritma je maksimalizacija modularnosti grafa (više o modularnosti u poglavlju 4). Ukratko, modularnost računa koliko su jako zajednice unutar grafa povezane.

Algoritam se sastoji od dva koraka koji se ponavljaju dok se modularnost može povećati. U prvom koraku, na početku svaki čvor predstavlja zasebnu zajednicu. Zatim se svaki čvor stavlja u zajednicu svakog od njegovih susjeda i računa se promjena modularnosti. Taj se proces ponavlja dok god je moguće povećati modularnost. Kada to više nije moguće, onda se prelazi na drugi korak.

U drugom koraku gradi se novi graf, u kojemu se svi čvorovi u istoj zajednici spajaju u jedan čvor, veze unutar zajednice pretvaraju se u vezu sa samim sobom, a veze izvan u veze sa čvorovima koji predstavljaju druge zajednice. Broj veza (ako je graf bestežinski) ili zbroj težina postaje težina nove veze.

3.2. Leiden

Tijekom godina uočen je jedan bitan problem s Louvain algoritmom. Zbog načina na koji funkcionira maksimalizacija modularnosti i agregacija čvorova često se može dogoditi da algoritam pronađe loše spojene zajednice. Primjer toga vidi se na slici (Sl. 3.1).



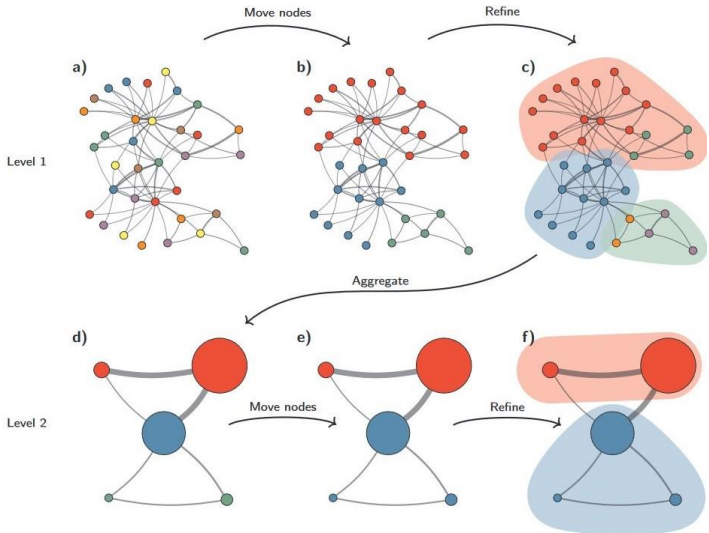
Sl. 3.1 Primjer greške kod Louvain algoritma [19]

Tu grešku dobro su opisali Traag, Waltman i van Eck u radu iz 2018. [19]. Iako je čvor 0 most između dva dijela zajednice, moguće je da se lokalnom optimizacijom modularnosti taj čvor pridijeli nekoj drugoj zajednici. U tom slučaju bilo bi optimalno prvu zajednicu podijeliti na dva dijela, ali Louvain nema mehanizam koji bi to detektirao ili izvršio. Većim brojem iteracija algoritma taj problem se pogoršava. Dani primjer je najgori slučaj, ali manje očite greške slične ovoj događaju se dovoljno često da budu značajne.

Kako bi riješio taj problem, Leiden algoritam omogućuje i razdvajanje zajednica. Baziran je na *Smart Local Moving* algoritmu koji su ranije razvili isti autori. Leiden algoritam sastoji se od tri koraka:

1. Lokalno grupiranje čvorova
2. Poboljšavanje nađenih zajednica
3. Agregacija grafa po poboljšanim zajednicama, koristeći prve zajednice kao početne zajednice za novu iteraciju algoritma

Primjer izvođenja moguće je vidjeti na slici (Sl. 3.2).



Sl. 3.2 Primjer izvođenja Leiden algoritma [19]

Osim ispravljanja grešaka, Leiden algoritam je efikasniji od Louvain-a, zato što, umjesto da u svakom koraku algoritma provjerava mogućnosti za svaki čvor, provjerava mogućnosti samo za „nestabilne“ čvorove. Nakon što jednom prođe sve čvorove, Leiden ponovo posjećuje samo susjede čvorova koje je prilikom prvog prolaza prebacio u nove zajednice, za razliku od Louvain-a koji posjećuje sve čvorove dok za sve ne može povećati modularnost. Detalji izvođenja ovog algoritma su izvan dosega ovog rada, no izvoran rad može se naći u literaturi [19].

3.3. Algoritam širenja oznaka

Algoritam širenja oznaka (engl. *label propagation*) je algoritam koji širi inicijalne oznake koristeći isključivo strukturu grafa. Korištena *CDLib* implementacija izvodi se na sljedeći način:

1. Na početku, svaki čvor dobije jedinstvenu oznaku (identifikator zajednice)
2. U svakoj iteraciji svaki čvor poprima oznaku koju ima najveći broj njegovih susjeda, a u slučaju da to nije jedinstveno, poprima nasumično odabranu oznaku (iz tog skupa)
3. Drugi korak ponavlja se dok svi čvorovi nemaju oznaku koju ima većina njegovih susjeda

Prednosti algoritma širenja oznaka su kratko vrijeme izvođenja, zbog manjka kompliciranih izračuna, te mogućnost izvođenja bez drugih informacija osim strukture grafa. Nedostatci su da algoritam, pogotovo na početku, ima veliku količinu nasumičnosti, te ne provjerava pronađene zajednice, kao što npr. Louvain i Leiden koriste modularnost.

3.4. Infomap

Infomap je baziran na konceptima iz teorije informacija. Algoritam pokušava minimizirati funkciju troška. Koristi kodirane rezultate nasumičnih šetnji kao zamjenu za tok informacija u stvarnom sustavu. Mreža se zatim rastavlja u module/zajednice smanjujući količinu informacija u nasumičnim šetnjama. *Infomap* su prvi razvili Rosvall i Bergstrom 2008. [20].

3.5. Constant Potts model

Constant Potts model razvili su 2011. godine Traag i Van Dooren [21]. Poput Leiden algoritma, ovaj algoritam razvijen je da riješi problem pronalaska malih zajednica koji postoji s algoritmima koji maksimiziraju modularnost, poput Louvain-a. Model je baziran na *first principle Potts model*-u koji su razvili Reichardt i Bornholdt [22]. Funkcija koju model pokušava minimizirati je (1).

$$Q = \sum_{ij} (A_{ij} - \gamma)\delta(\sigma_i, \sigma_j) \tag{1}$$

Gdje je A matrica susjedstva, γ parametar rezolucije, a δ funkcija koja govori jesu li dva čvora u istoj zajednici.

Kontroliranjem parametra γ možemo birati veličinu zajednica koje će algoritam pronalaziti, gdje će unutarnja gustoća zajednica u grafu uvijek biti veća, a vanjska gustoća uvijek manja od γ.

Ukratko, model pokušava maksimizirati količinu veza unutar zajednica, a održava manje zajednice zbog parametra rezolucije.

4. Metrike evaluacije zajednica

U ovom poglavlju detaljnije će se opisati četiri korištene metrike za evaluaciju zajednica. Korištene su dvije unutarnje metrike, koje daju vrijednost iz svojstava pronađenih zajednica. U ovom radu to su modularnost i provodnost. Korištene su i dvije vanjske metrike, koje vrijednosti dobivaju usporedbom s ispravnim zajednicama. U ovom radu to su normalizirani F1 i normalizirani uzajamni sadržaj informacije.

4.1. Modularnost

Modularnost je jedna od ključnih metrika za evaluaciju pronađenih zajednica. Modularnost računa koliko su jako zajednice unutar grafa povezane, što je već spomenuto u poglavlju 3.1. Modularnost može poprimiti vrijednosti [-0.5, 1] za bestežinske i neusmjerene grafove. Veća modularnost je poželjna. Modularnost se računa tako da se dobije razlika postotka veza unutar zajednica u danom grupiranju i postotak veza unutar zajednica kada bi veze bile nasumično raspoređene.

Postoje različiti pristupi za računanje nasumično raspoređenog grafa. Najčešće korišteni pristup održava stupanj svakog čvora. Jedan drugi pristup uzima postotak i pretpostavlja da su svi čvorovi nasumičnog grafa povezani s tom vjerojatnosti.

Odabrana biblioteka koristi formulu (2) za računanje modularnosti zajednica bez preklapanja.

$$Q(S) = \frac{1}{m} \sum_{c \in S} (m_s - \frac{(2m_s + l_s)^2}{4m}) \tag{2}$$

Gdje je S zajednica, m ukupan broj veza u grafu, m_s broj veza unutar zajednice, a l_s broj veza koje idu izvan S. Na kraju se zbroje vrijednosti za sve zajednice. Tu formulu definirali su Newman i Girvan [23]. Probleme s tom formulom možemo naići kod primjene na velikim grafovima, kada nije realno očekivati da čvorovi mogu biti spojeni sa svakim drugim čvorom u grafu, pa će algoritmi koji maksimiziraju samo modularnost težiti velikim zajednicama, čak i kada one nemaju smisla po ostalim metrikama.

4.2. Provodnost

Provodnost zajednice (engl. *conductance*) definira se kao udio veza koje su povezane s čvorovima izvan zajednice. Provodnost grafa dobiva se uprosjećivanjem vrijednosti provodnosti zajednica. Odabrana implementacija računa provodnost zajednice formulom (3).

$$f(S) = \frac{c_s}{2m_s + c_s} \tag{3}$$

Gdje je *S* zajednica, c_s broj čvorova u zajednici, a m_s broj veza u zajednici. Poželjna je što manja provodnost. Prednost provodnosti je da se može koristiti na zajednicama sa i bez preklapanja.

4.3. Normalizirani uzajamni sadržaj informacije

Normalizirani uzajamni sadržaj informacije (engl. *normalized mutual information*) je metrika koja uspoređuje dva različita grupiranja istog grafa. Metrika je normalizirana na raspon između 0, kada ne postoji nikakva korelacija između grupiranja, do 1, kada se nađene zajednice savršeno preklapaju. Normalizacija se obavlja po formuli (4), gdje je *H* entropija skupa podataka.

$$NMI = MI * \sqrt{H(ispravne) * H(pronađene)} \tag{4}$$

Ova metrika je iznimno korisna ako postoji idealni raspored zajednica. To možemo dobiti koristeći stvarne (engl. *ground truth*) zajednice, odnosno zajednice dobivene istraživanjem podataka, ili grafove koji dolaze s ugrađenim zajednicama, poput LFR grafova.

4.4. F1 metrika

F1 metrika je razvijena kako bi se riješio glavni problem s NMI metrikom, a to je da je vrlo skupa za izvođenje. Cilj F1 rezultata je isti kao i za NMI, odrediti sličnost dva seta zajednica, što koristimo u kombinaciji sa stvarnim zajednicama kako bi odredili kvalitetu pronađenih zajednica.

Za svaki set zajednica X koji pronađe algoritam, svaki čvor se označi s se sa zajednicom kojoj pripada u setu stvarnih zajednica Y. Onda za svaki par (x, y), gdje su x i y oznake zajednica iz setova X i Y, mogu izračunati *precision* (P), po formuli (5), i *recall* (R), po formuli (6).

$$P = \frac{|x \cap y|}{|x|} \tag{5}$$

$$R = \frac{|x \cap y|}{|y|} \tag{6}$$

Za svaki par se onda može izračunati F1 rezultat po formuli (7).

$$F1 = 2\frac{P * R}{P + R} \tag{7}$$

Uprosjećivanjem vrijednosti za svaki par, možemo dobiti rezultat za cijeli graf.

5. Implementacija traženja zajednica

U ovom poglavlju bit će definiran točan postupak dobivanja rezultata za stvarne i sintetičke grafove. Za LFR grafove će biti definirane vrijednosti parametara.

5.1. Stvarni grafovi

Kako bi se dobili točniji rezultati, za grafove karate kluba, firentinskih obitelji i gradova traženje zajednica provedeno je deset tisuća (10000) puta. Za svake pronađene zajednice izračunate su modularnost i protočnost, koje će biti prikazane u rezultatima uz standardnu devijaciju. Vrijeme će biti prikazano kao zbroj izmjerenih vremena pronalaženja zajednica.

Za karate klub rezultati svakog od algoritama su uspoređeni sa stvarnim slučajem, koristeći NMI i NF1. To nije bilo moguće za ostale stvarne grafove, zato što ne postoje nužno ispravni podatci.

Rezultati za Amazon graf su dobiveni iz jednog pokretanja algoritama, zato što je vrijeme izvođenja bilo preveliko za izvođenje dovoljan broj puta da bi se dobili uprosječeni rezultati.

5.2. LFR grafovi

LFR grafovi uvezeni su iz baze podataka iz koje *CDLib* može direktno uvoziti grafove i njihove zajednice [24]. To je tako napravljeno zato što je generiranje tih grafova dugotrajno, a cilj je bio testirati algoritme na što većem broju grafova.

Svojstva preuzetih LFR grafova su sljedeća:

- Broj čvorova (N): 1000, 5000, 10000, 50000, 100000
- Prosječni stupanj čvora (k): 5
- Minimalna veličina zajednice: 50
- Parametar miješanja (μ): 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
- *tau1*: 3

- *tau2*: 1.5

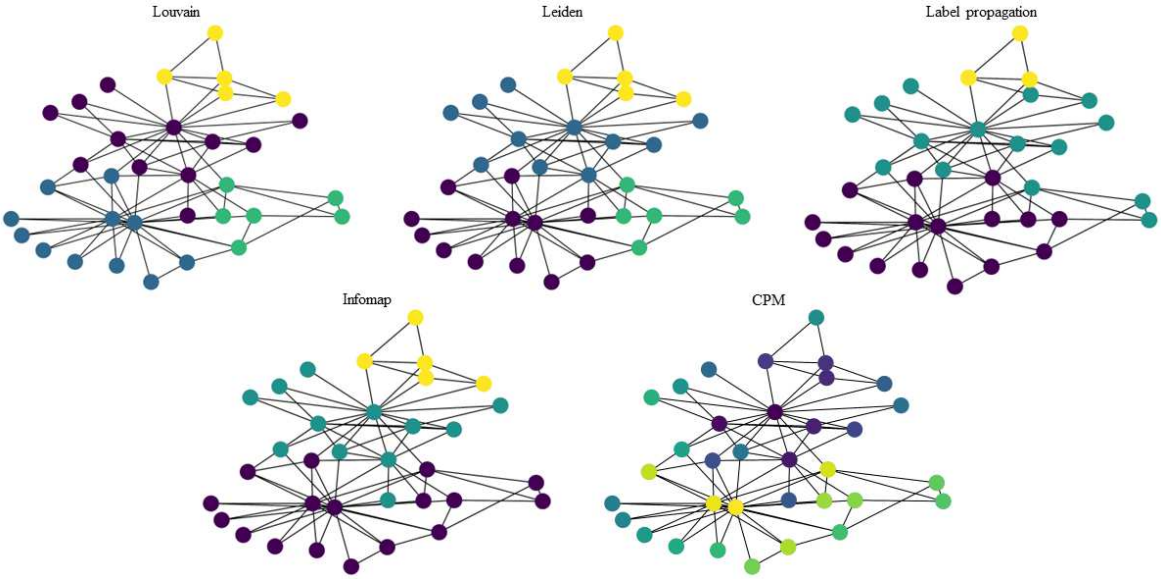
Na svakom grafu svaki algoritam pokrenut je jednom, zbog vremena izvođenja.

6. Rezultati provođenja algoritama

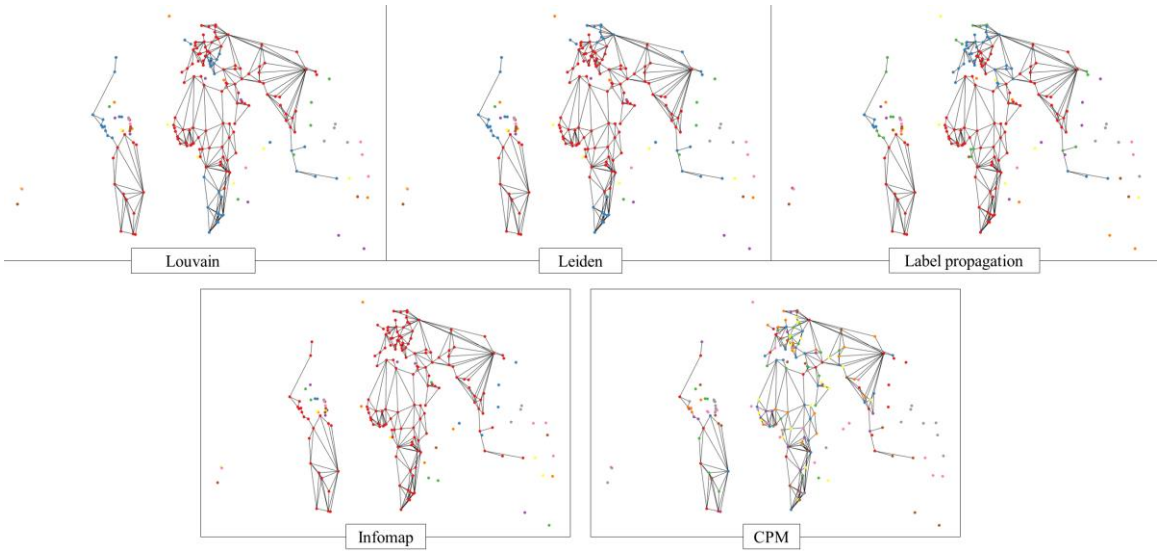
U ovom poglavlju bit će prikazane neke od dobivenih zajednica, opisani rezultati metrika za stvarne i sintetičke grafove. Na kraju će se dati pregled rezultata po algoritmima i prijedlozi za daljnje istraživanje.

6.1. Stvarni grafovi

Primjeri pronađenih zajednica mogu se vidjeti na slikama (Sl. 6.1) i (Sl. 6.2).

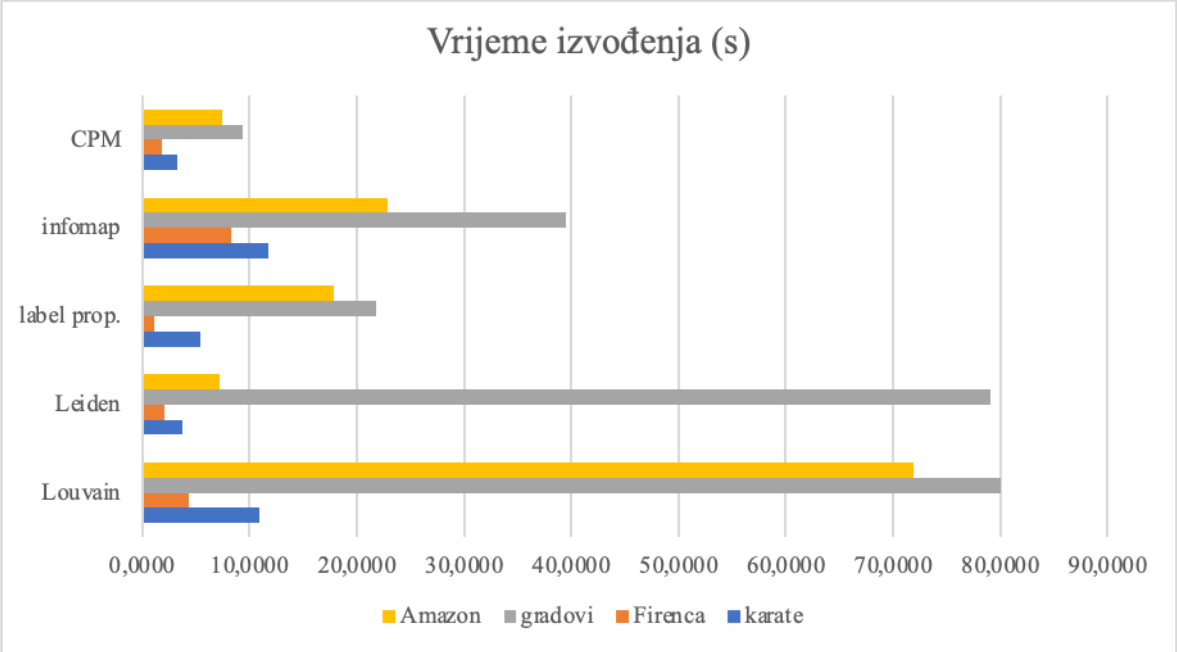


Sl. 6.1 Zajednice na karate grafu



Sl. 6.2 Zajednice na grafu glavnih gradova

Već se mogu vidjeti neka od svojstava algoritama. Vidi se da CPM radi daleko najmanje zajednice, što je u skladu s njegovom definicijom. Isto se može vidjeti da algoritmi Louvain i Leiden daju slične rezultate, no razlikuju se po tome da Leiden ima na nekim mjestima manje zajednice, što također odgovara teoriji tih algoritama. Vrijeme izvođenja za pojedinih algoritama na ranije definiranim grafovima vidljivo je na slici (Sl. 6.3).

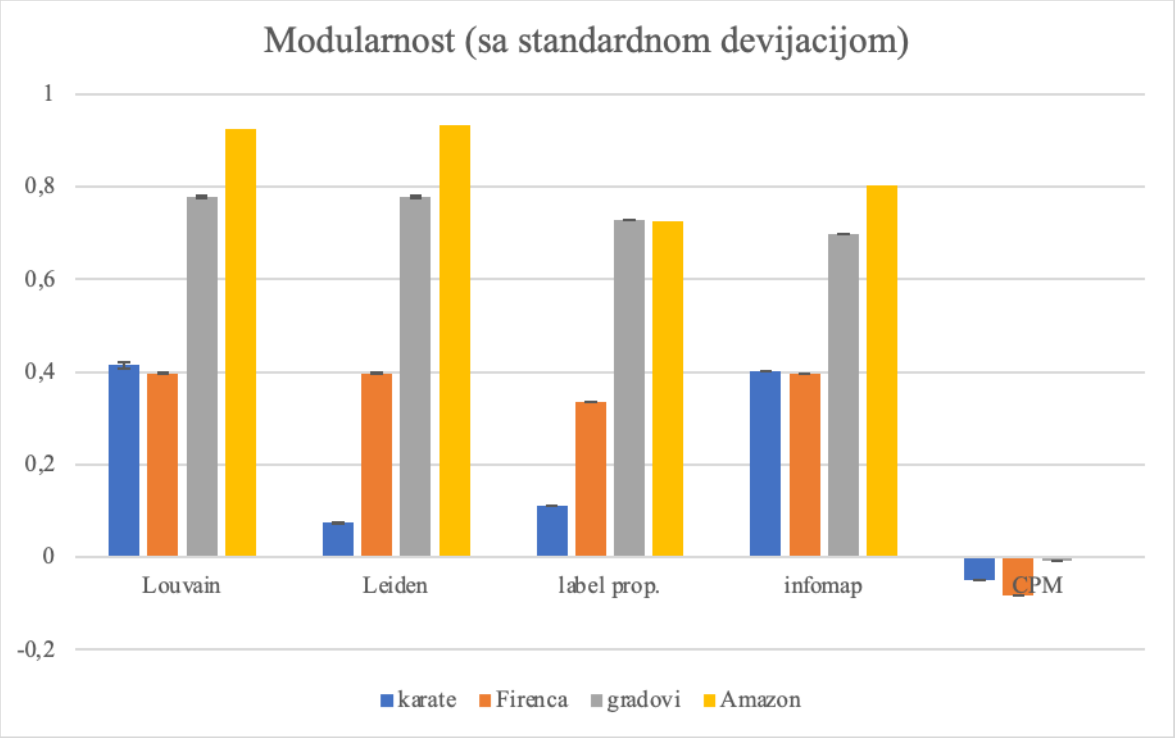


Sl. 6.3 Vrijeme izvođenja algoritama

Najsporiji za manje grafove (karate i Firenca) je algoritam *infomap*. No, na većim grafovima Louvain se izvršava vrlo sporo. Zanimljivo je primijetiti da Leiden, usprkos poboljšanjima zbog kojih se u tri od četiri grafa brže izvršava, za pronalaženje zajednica na

grafu glavnih gradova treba mu gotovo isto vrijeme kao i kod Louvain-a. Gledajući definiciju algoritama, mogu se uočiti dva moguća uzroka: potreba za čestim razdvajanjem zajednica i postojanje velikih brojeva „nestabilnih“ čvorova. Proučavajući dobivene zajednice, može se vidjeti da se te zajednice ne razlikuju značajno, tj. Leiden nije na mnogo mjesta trebao razdvajati zajednice, pa se može pretpostaviti da graf glavnih gradova ima mnogo „nestabilnih“ čvorova.

Rezultati mjerenja modularnosti za dobivene zajednice mogu se vidjeti na slici (Sl. 6.4).

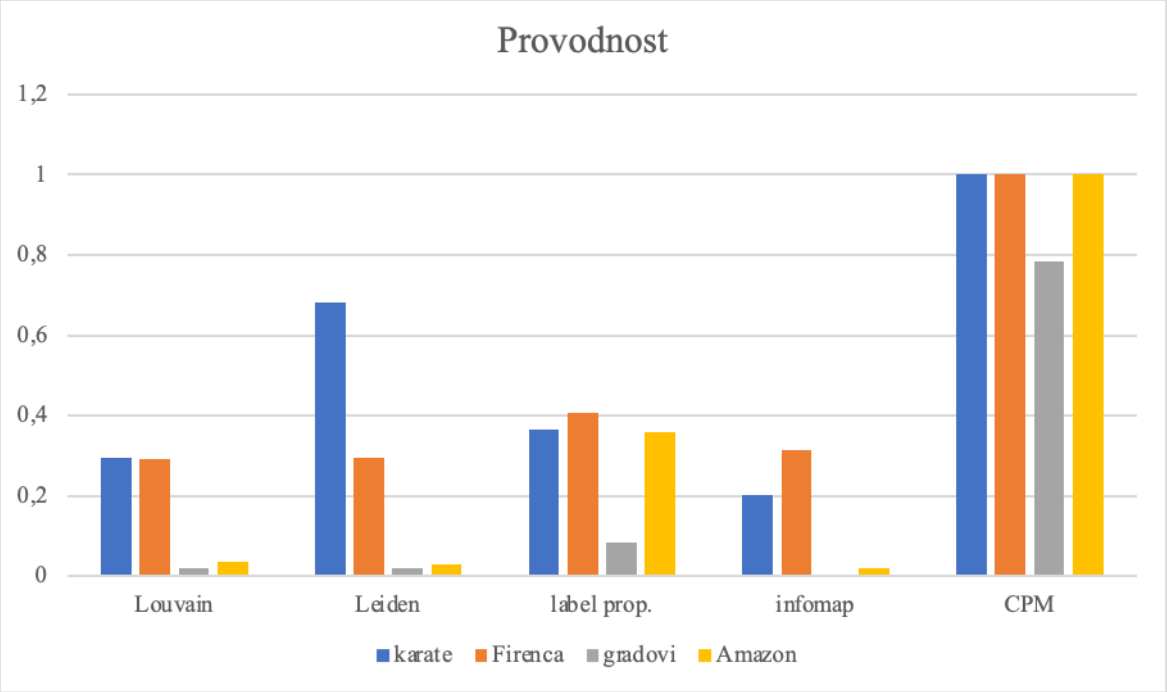


Sl. 6.4 Graf rezultata mjerenja modularnosti

Prva stvar koja se može uočiti da (za grafove nad kojima su algoritmi više puta provedeni) je standardna devijacija modularnosti bliska nuli. Iz toga može se zaključiti da, usprkos nasumičnim elementima u algoritmima, svi proučavani algoritmi nalaze vrlo slične zajednice pri višestrukim pokretanjima na istom grafu. Louvain ima najbolje rezultate na gotovo svim grafovima, zato što iako *infomap* ima slične rezultate na karate i firentinskom grafu, kaska na dva veća grafa. To je očekivani rezultat, zato što Louvain optimizira zajednice na bazi modularnosti. Leiden ima manje vrijednosti na manjim grafovima, gdje odluka da se neke zajednice razdvoje ima veći utjecaj na modularnost, ali na većim grafovima ima slične vrijednosti kao Louvain, što opet odgovara očekivanjima. Što se tiče algoritama *infomap* i širenje oznaka, oni imaju dobre rezultate, ne znatno manje od Louvain-a, no na većim grafovima, pogotovo na grafu Amazon proizvoda, može se vidjeti

da ti algoritmi ne pokušavaju optimizirati vrijednost modularnosti. Za CPM mogu se vidjeti jako loši rezultati, što je očekivano, zato što taj algoritam pronalazi vrlo male zajednice, što znači da će te zajednice imati malen broj veza unutar zajednice.

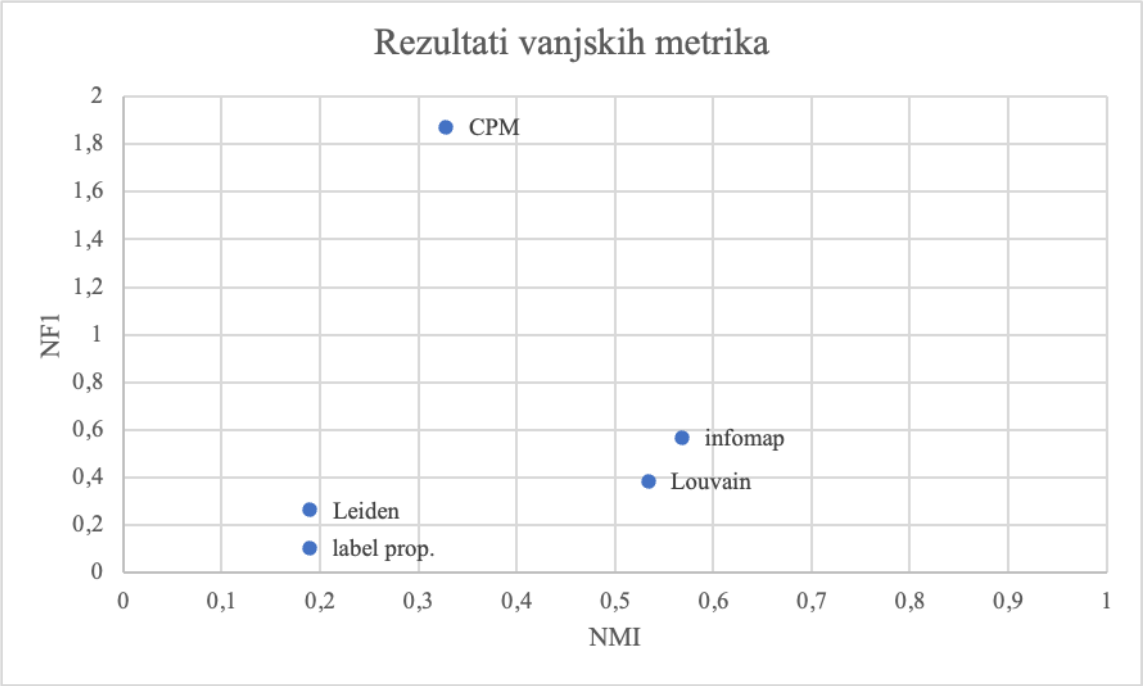
Rezultati mjerenja provodnosti za stvarne grafove mogu se vidjeti na slici (Sl. 6.5).



Sl. 6.5 Graf rezultata mjerenja provodnosti

Rezultat koji se najviše ističe su visoke vrijednosti za algoritam CPM. Nailazi se na sličan problem kao kod loših vrijednosti modularnosti. Činjenica da algoritam pronalazi male zajednice uzrokuje nepovoljne vrijednosti metrika koje mjere povezanost zajednica. Može se primijetiti i da konstrukcija grafa daju donju granicu provodnosti, zato što grafovi s manjim brojem veza ne mogu imati male vrijednosti, zbog načina računanja. Za sve grafove osim Firence, zajednice koje pronalazi *infomap* imaju najmanju provodnost, dok za Firencu nabolje rezultate daje Louvain.

Prikaz rezultata za vanjske metrike (NF1 i NMI) dobivene uspoređivanjem sa stvarnim zajednicama može se vidjeti na slici (Sl. 6.6).

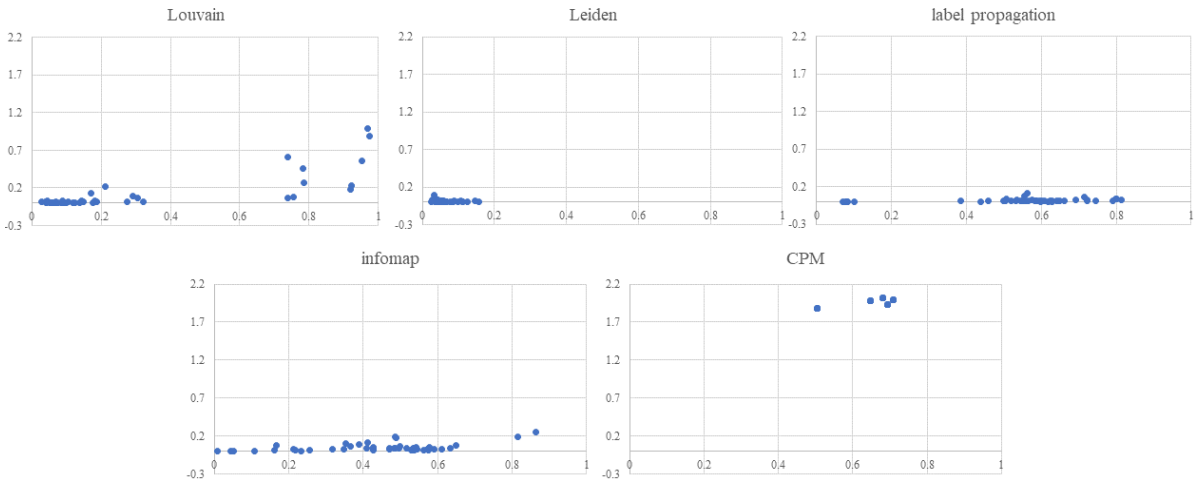


Sl. 6.6 Rezultati za vanjske metrike na karate grafu

Za rezultate algoritama Louvain, Leiden, širenje oznaka i *infomap* vidimo slične rezultate za vrijednosti NF1 i NMI. To je rezultat koji se mogao očekivati, zato što te metrike zapravo pokušavaju izmjeriti istu stvar, sličnost zajednica. Može se vidjeti da za karate graf Louvain i infomap daju bolje rezultate nego Leiden i širenje oznaka. To se može pripisati svojstvu Louvain-a i infomap-a da rade manje većih zajednica nego Leiden i širenje oznaka, a ispravna podjela je na dvije zajednice, što je objašnjeno u poglavlju 2.1.1 Karate klub. Može se uočiti i značajno odstupanje vrijednosti NF1 za algoritam CPM. Postoje dva moguća uzroka tog odstupanja: greška kod računanja F1 rezultata ili greška kod normalizacije. Uspoređivanjem izračuna F1 za ostale algoritme i CPM, dobivaju se vrijednosti istog reda veličine, iz čega se može zaključiti da je do problema došlo zbog načina na koji je normalizacija implementirana u korištenoj biblioteci.

6.2. Sintetički grafovi

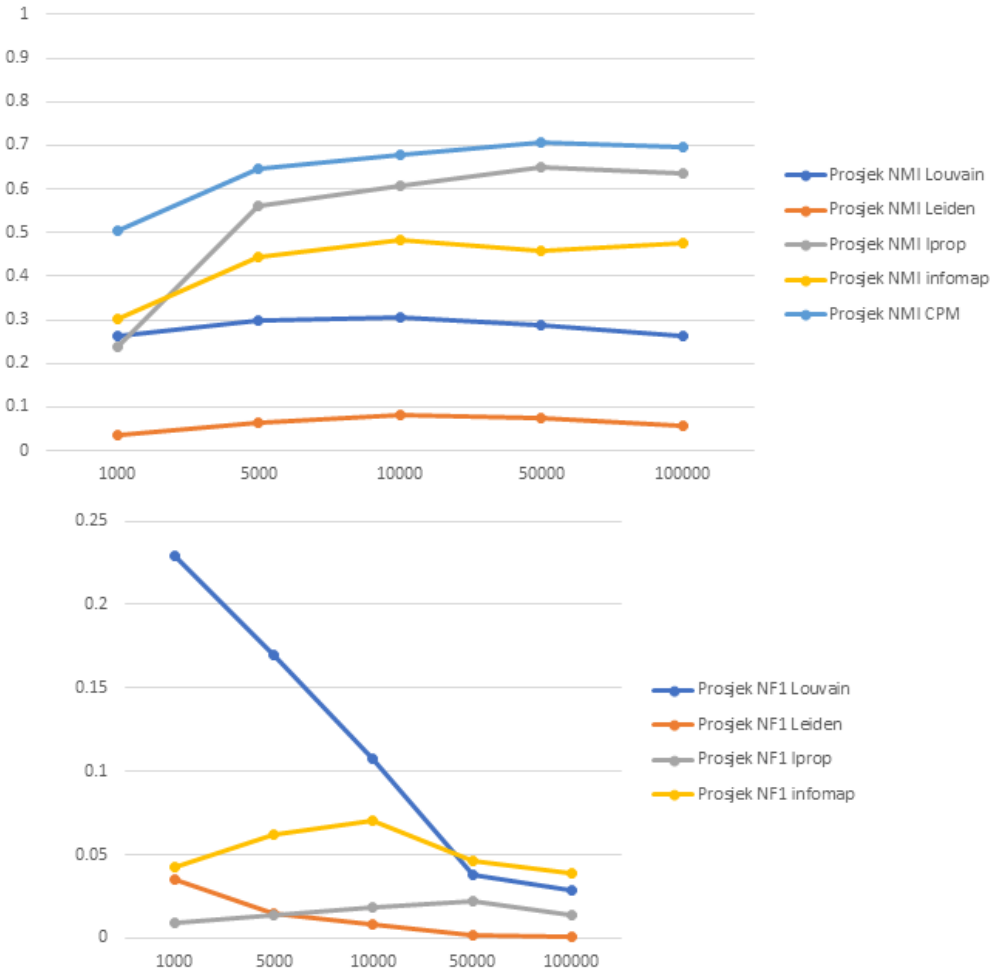
Rezultati vanjskih metrika za zajednice pronađene na LFR grafovima mogu se vidjeti na slici (Sl. 6.7).



Sl. 6.7 Rezultati vanjskih metrika

Na grafovima *x* os predstavlja vrijednost NMI, a *y* os vrijednost NF1. Opet se vidi isti problem za algoritam CPM, to jest prevelike vrijednost NF1 uzrokovane lošom normalizacijom. No, ovdje se mogu vidjeti i niski rezultati NF1 za gotovo sve algoritme. Leiden daje dosta loše rezultate za sve proučavane grafove. *Infomap* i širenje oznaka daju bolje rezultate, ali *infomap* ipak lošije od ta dva. CPM, usprkos grešci kod računanja NF1, daje jako dobre rezultate za NMI. Vrijednosti za Louvain su jedine kod kojih se može vidjeti jasnija korelacija između NMI i NF1. Te su vrijednosti ujedno i podijeljene na dvije grupe, tj. algoritam daje ili vrlo dobre ili loše rezultate.

Rezultati u ovisnosti za broj čvorova u grafu mogu se vidjeti na slici (Sl. 6.8).



Sl. 6.8 Rezultati u odnosu na broj čvorova

Iz ovih grafova može se vidjeti da u puno slučajeva broj čvorova nema osobiti utjecaj na vrijednosti vanjskih metrika. Za NMI se vidi blagi porast za algoritam širenja oznaka, *infomap* i CPM, dok za NF1 se vidi blagi pad za Leiden, no s obzirom na to da su to sve vrijednosti manje od 0.05, utjecaj zapravo nije značajan. Može se uočiti i značajan pad NF1 za Louvain, no dubljim proučavanjem rezultata može se primijetiti velika standardna devijacija (u nekim slučajevima veća nego vrijednost prosjeka), pa ne možemo ništa konkretno zaključiti iz vrijednosti u grafu. Slične se vrijednosti mogu dobiti ako se pokuša dobiti ovisnost o parametru miješanja.

6.3. Pregled rezultata za algoritme

Louvain daje vrlo dobre rezultate za većinu grafova, u unutarnjim i vanjskim metrikama. Iz tih rezultata lako je razumljivo zašto se taj algoritam smatrao zlatnim standardom

godinama nakon što je definiran. No, cijena tih rezultata je dulje vrijeme izvođenja, pogotovo za velike grafove.

Leiden daje slične rezultate kao i Louvain, na kojem se bazira, za unutarnje metrike, no za usporedbe sa stvarnim zajednicama daje slabije rezultate. No, nije provedeno dovoljno istraživanja da bi se mogao dobiti definitivan zaključak oko usporedbe rezultata za vanjske metrike. Optimizacije vremena izvođenja za Leiden najjasnije su vidljive za Amazon graf, gdje mu je potrebno otprilike 10% vremena koje je potrebno Louvain-u.

Algoritam širenja oznaka i *infomap* daju dobre rezultate za stvarne grafove, no širenje oznaka daje slabe vrijednosti za provodnost za veće grafove, za razliku od *infomap*-a koji za to daje iznimno dobre rezultate. Za sintetičke grafove oba algoritma daju raznolike rezultate, bez otkrivene ovisnosti.

Constant Potts Model (CPM) daje vrlo specifične rezultate, zbog toga što stvara mnogo vrlo malih zajednica. S obzirom na to da proučavane unutarnje metrike mjere unutarnju povezanost zajednica, logično je da CPM daje jako loše rezultate za te metrike, što se lako može vidjeti u rezultatima. Do problema dolazi i kod normalizacije za F1, što otežava proučavanje vanjskih metrika. No, CPM daje dobre rezultate za NMI za sve proučavane sintetičke grafove.

6.4. Daljnje istraživanje

Iz rezultata lako se može primijetiti potreba za daljnjom i detaljnijom usporedbom algoritama. Kako bi se algoritmi bolje usporedili na stvarnim grafovima, potrebno bi bilo pronaći više stvarnih grafova različitih veličina i različitih stvari koje prikazuju, da bi se moglo npr. povezati tip grafa s algoritmom koji najbolje nalazi zajednice. Rezultati bi se isto trebali uprosjećiti za veće grafove, za što bi bilo potrebno jače računalo nego što je bilo dostupno za ovaj rad.

Za sintetičke grafove potrebno bi bilo generirati veću količinu LFR grafova koji variraju po većem broju parametara, kako bi se mogle istražiti ovisnosti koje nisu u ovom radu proučene, zbog manjka grafova. Potrebno bi bilo i višestruko izvođenje algoritama na danim grafovima kako bi se dobili precizniji rezultati. Kako bi se to moglo provesti, potrebni su značajniji računalni resursi nego što su bili dostupni.

Moguće je na isti način proširiti usporedbu i na više algoritama za zajednice bez preklapanja. Na sličan način može se obaviti i usporedba za zajednice s preklapanjem, no bilo bi potrebno prilagoditi izračune metrika postojanju preklapanja u zajednicama.

Zaključak

Traženje zajednica u grafovima je kompliciran problem koji nema točno, optimalno rješenje. Ovaj rad pokušava dati pregled nekih često korištenih algoritama i usporediti rezultati odabranih metrika na stvarnim i sintetičkim grafovima.

U ovom radu proučavano je pet algoritama: Louvain, Leiden, širenje oznaka, *infomap* i *Constant Potts Model* (CPM). To su algoritmi za traženje zajednica bez preklapanja na statičnim grafovima. Stvarni korišteni grafovi su graf karate kluba, graf obitelji u Firenci, graf glavnih gradova i graf proizvoda s Amazona. Sintetički grafovi korišteni su Lancichinetti–Fortunato–Radicchi (LFR) grafovi, koji dolaze s ugrađenim zajednicama koje se mogu koristiti za uspoređivanje sa zajednicama koje algoritmi pronalaze. Metrike koje su korištene za evaluaciju algoritama su sljedeće: modularnost, provodnost, normalizirani uzajamni sadržaj informacije i normalizirana F1 metrika.

Najbolje rezultate modularnosti davali su Louvain i Leiden, što je očekivano zato što ti algoritmi pokušavaju maksimizirati modularnost. Ta dva algoritma, uz *infomap*, daju i najbolje vrijednosti provodnosti. Za vanjske metrike, Leiden daje najgore rezultate, Louvain, *infomap* i širenje oznaka daju raspon rezultata. CPM daje dobre rezultate za NMI, ali dolazi do greške kod računanja NF1, pa te rezultate ne možemo usporediti.

Neka od mogućih poboljšanja su višestruko izvođenje na svim grafovima kako bi se dobile prosječne vrijednosti i koristiti veću količinu LFR grafova koji variraju po većem broju parametara kako bi mogli ispravnije odrediti ovisnosti rezultata o tim parametrima. Također je moguće koristiti ovakav postupak za ispitivanje većeg broja algoritama.

Literatura

[1] Newman, M.E.J. Finding community structure in networks using the eigenvectors of matrices, Physical Review E 74, 036104 (2006)

[2] Rossetti, G., *Community Discovery algorithms*, CDLib, Poveznica: https://cdlib.readthedocs.io/en/latest/reference/algorithms.html; pristupljeno 6. lipnja 2023.

[3] Rossetti, G. et al, *CDLib*, Poveznica: https://cdlib.readthedocs.io/en/latest/; pristupljeno: 5. lipnja 2023.

[4] *NetworkX*, Poveznica: https://networkx.org/; pristupljeno 5. lipnja 2023.

[5] *Matplotlib*, Poveznica: https://matplotlib.org/; pristupljeno 5. lipnja 2023.

[6] Campbell, S. *Python Timeit() with Examples*, Guru99, (2023, svibanj). Poveznica: https://www.guru99.com/timeit-python-examples.html; pristupljeno 1. lipnja 2023.

[7] *Memgraph*, Poveznica: https://memgraph.com/; pristupljeno 1. lipnja 2023.

[8] *GQLAlchemy*, Memgraph. Poveznica: https://memgraph.com/gqlalchemy; pristupljeno 5. lipnja 2023.

[9] Donohue, T. *How I Created a Python Development Environment with Docker*, tutorialworks, (2022, siječanj). Poveznica: https://www.tutorialworks.com/python-develop-container/; pristupljeno 1. lipnja 2023.

[10] *Python – Official Image*, DockerHub. Poveznica: https://hub.docker.com/_/python; pristupljeno 1. lipnja 2023.

[11] Buzdar, K. *How to Install CMake on Debian 11*, linuxhint, (2022, studeni). Poveznica: https://linuxhint.com/install-cmake-on-debian/#2; pristupljeno 1. lipnja 2023.

[12] Zachary, W. W. *An Information Flow Model for Conflict and Fission in Small Groups,* Journal of Anthropological Research Volume 33, Number 4 (1977)

[13] *Zachary karate club social network*, (2022, rujan). CC BY-SA 4.0, Poveznica: https://en.wikipedia.org/w/index.php?curid=71789482; pristupljeno 1. lipnja 2023.

[14] Girvan, M., Newman, M.E.J. *Community structure in social and biological networks*, Proceedings of the National Academy of Sciences (2002)

[15] Breiger, R.L., Pattison, P.E., Cumulated social roles: The duality of persons and their algebras, 1 Social Networks, Volume 8, Issue 3 (1986)

[16] Leskovec, J. *Amazon product co-purchasing network and ground-truth communities*, Poveznica: https://snap.stanford.edu/data/com-Amazon.html; pristupljeno 1. lipnja 2023.

[17] Lancichinetti, A., Fortunato, S., Radicchi, F. *Benchmark graphs for testing community detection algorithms*, Physical Review E78, 046110 (2008)

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

[19] Traag, V. A., Waltman, L., van Eck, N. J. *From Louvain to Leiden: guaranteeing well-connected communities*, Scientific Reports, vol. (9): 5233 (2019)

[20] Rosvall, M., Bergstrom, C. T. *Maps of random walks on complex networks reveal community structure*, Proc Natl Acad SciUSA 105(4):1118–1123 (2008)

[21] Traag, V. A., Van Dooren, P., Nesterov, Y. *Narrow scope for resolution-limit-free community detection*, Phys. Rev. E 84, 016114 (2011)

[22] Reichardt, J., Bornholdt, S. *Partitioning and modularity of graphs with arbitrary degree distribution*, Phys Rev E 76, 015102 (2007)

[23] Girvan, M., Newman, M.E.J. *Finding and evaluating community structure in networks*, Physical Review E 69, 26113 (2004)

[24] Rossetti, G. *CDlib datasets*, (2020, prosinac). Poveznica: https://github.com/GiulioRossetti/cdlib_datasets; pristupljeno: 1. lipnja 2023.

Sažetak

Sustavi preporuke temeljeni na kolaborativnom filtriranju (engl. *collaborative filtering recommender system*) za identifikaciju sličnih proizvoda koriste algoritme za otkrivanje zajednica (engl. *community detection algorithms*) u grafovima. S tim ciljem je u ovom radu provedena analiza tih algoritama za pronalaženje zajednica bez preklapanja. Analizirani algoritmi su: Louvain, Leiden, širenje oznaka, *infomap* i *Constant Potts Model*. Algoritmi su provedeni nad stvarnim grafovima od kojih su neki pohranjeni u graf bazu podataka, a to su karate klub, firentinske obitelji, glavni gradovi svijeta i Amazon proizvodi te nad sintetičkim LFR grafovima. Dobivene zajednice uspoređene su po unutrašnjim metrikama modularnosti i provodnosti te vanjskim metrikama normalizirane F1 vrijednosti i normaliziranom uzajamnom sadržaju informacija.

Ključne riječi: teorija grafova, detekcija zajednica, analiza algoritama, Python, CDLib

Summary

Collaborative filtering recommender systems use algorithms for community detection in graphs to identify similar products. This study analyzes community detection algorithms for crisp communities for this purpose. The analyzed algorithms include Louvain, Leiden, label propagation, infomap, and Constant Potts Model. These algorithms were applied to real-world graphs, some of which were stored in a graph database, including the karate club, Florentine families, world capitals, Amazon products, as well as synthetic LFR graphs. The obtained communities were compared using internal metrics such as modularity and conductance, as well as external metrics such as normalized F1 score and normalized mutual information.

Keywords: graph theory, community detection, algorithm analysis, Python, CDLib

Skraćenice

CPM	*Constant Potts Model*	algoritam pronalaženja zajednica
LFR	*Lancichinetti–Fortunato–Radicchi*	vrsta sintetičkih grafova
NF1	*Normalized F1*	vanjska metrika
NMI	*Normalized Mutual Information*	vanjska metrika