Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Hľadanie komunít s využitím Absorbujúcich Markovovských Reťazcov

Bakalárska práca

2022
Andrej Vašek

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Hľadanie komunít s využitím Absorbujúcich Markovovských Reťazcov

Bakalárska práca

Študijný program:	Aplikovaná Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Peter Náther, PhD.

Bratislava, 2022
Andrej Vašek

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Andrej Vašek
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Hľadanie komunít s využitím Absorbujúcich Markovovských Reťazcov
Detection of communities using Absorbing Markov Chains

Anotácia: Cieľom práce je implementovať navrhnutý algoritmus, ako rozšírenie existujúceho pluginu pre platformu GEPHI a porovnať jeho vlastnosti s už implementovanými algoritmami

Literatúra:
[1] Brandes et al, ”On finding Graph Clusterings with Maximum Modularity”, http://www-i1.informatik.rwth-aachen.de/en/mhoefer/Forschung/05-07/Brandes07Modularity.pdf

[2] ”Communitydetectioningraphs”,http://snap.stanford.edu/class/cs224w-readings/fortunato10community.pdf

[3] http://web.stanford.edu/class/cs224w/projects2013/cs224w-065-final.pdf.

Vedúci: Mgr. Peter Náther, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 02.10.2014

Dátum schválenia: 03.11.2020 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

Bakalárska práca Hľadanie komunít s využitím absorbujúcich markovovských reťazcov sa venuje implementácii a testovaniu zhlukovacieho algoritmu. Tento algoritmus bol navrhnutý so zámerom dosahovať dobré rozdelenia do zhlukov pri nízkej časovej zložitosti.

Algoritmus je implementovaný ako sučasť modulu do platformy Gephi, slúžiacej na vizualizíciu a prácu s grafmi. Obsahom práce bola aj aktualizácia daného modulu na súčasnú verziu platformy.

Zhlukovací algoritmus bol hodnotený pri použití na dátových množinách so známou správnou klasterizáciou. Výsledky boli hodnotené priamo pomocou funkcií ako Davies-Bouldin index alebo hodnotením modularity výsledku. Spolu s ostatnými algoritmami pluginu bol hodnotený aj externe podľa metrík Normalized Mutual Information, Rand index a F-measure.

Pri hodnotení dosahuje algoritmus dobré riešenia, porovnateľné s výsledkami zvyšných zo sady algoritmov. Práca nakoniec rozoberá limitácie algoritmu a navrhuje jeho vylepšenia.

Kľúčové slová: zhlukovanie, markovovský reťazec

Abstract

The final bachelor thesis, Detection of communities using Absorbing Markov Chains, implements and evaluates a cluster analysis algorithm. The algorithm was proposed with aim to reach good community structure within polynomial time complexity.

The algorithm has been implemented as a component of module Community Detection for platform Gephi - a graph visualization tool. The goal of the thesis include actualization of the Community Detection module for the current version of the platform.

The clustering algorithm was evaluated on data sets with known correct clustering. The resulting clustering were evaluated internally with function such as Davies-Bouldin index and modularity of resulting clustering. Along with other algorithms of the plugin, the algorithm was evaluated externally based on Normalized Mutual Information, Rand index and F-measure.

The algorithm reaches good solutions, comparable to those of the other algorithms of the plugin. The thesis proceeds to analyse limitations of the algorithm and proposes improvements.

Keywords: clustering, markov chain

Obsah

Úvod

Na analýzu dát sa často používajú zhlukovacie algoritmy. Tie sa snažia správne rozdeliť dáta do skupín (zhlukov) tak, aby prvky, ktoré sa spolu nachádzajú v jednom zhluku boli podobné. To, akým spôsobom sa prvky rozdeľujú závisí od konkrétneho algoritmu a prístupov je mnoho.

Zhluková analýza je užitočná preto, že umožnuje nájsť štruktúru v dátach, ktoré sú rozsiahle a ich manuálna analýza by bola nepraktická. Na základe nájdenej štruktúry sa potom dá ľahšie porozumieť jednotlivým častiam a nakoniec aj celému systému.

To umožňuje do istej miery urýchliť analýzu, pretože s menším zapojením priamej ľudskej práce môže napredovať rýchlejšie. Veľmi dobré procesy zhlukovej analýzy by potenciálne mohli ľudské zapojenie aj plne nahradiť a získavať tak z dát informácie samostatne. Takéto nástroje by veľmi ulahčili a zrýchlili procesy skúmania.

Zhlukovacie algoritmy sa často stretajú s problémom, že pre získanie dobrých zhlukovaní sa algoritmy stávajú zložitými a potom sú príliš pomalé na vyhodnocovanie veľkých dátových množín. Preto sa zvykne hľadať kompromis medzi rýchlosťou a kvalitou výsledného zhlukovania.

Cieľom tejto práce je implementovať a otestovať algoritmus na hľadanie komunít využívajúci absorbujúce markovovské reťazce. Tento algoritmus bol navrhnutý v práci A.Kumara [11]. Jeho zámerom bolo vytvoriť algoritmus produkujúci dobré rozdelenia, ktérý by dosahoval iba polynomiálnu zložitosť.

Algoritmus má byť implementovaný ako súčasť pluginu Community Detection vytvorenom v diplomovej práci M.Šmitalu [23].

S využitím funkcii tohto pluginu sa potom algoritmus môže otestovať a porovnať s ostatnými implementovanými algoritmami.

Rozdelenie práce

V kapitole 1 sa opisuje súčasný stav oblasti. V odstavcoch 1.1 až 1.5 sa rozoberá oblasť spracovania a analýzy dát s ohľadom na hľadanie komunít (zhlukov). Prehľad je postavený najmä na prácach A review of clustering techniques and developments [18], Machine Learning [13] a Community detection in graphs [6]. V časti 1.6 opisuje

predošlé práce, ktoré s aktuálnou súvisia, alebo na ne nadväzuje. Časť 1.7 opisuje technológie používané pri realizácii práce.

Kapitola 2 definuje ciele práce aj ich čiastkové ciele.

V kapitole 3 sa špecifikuje popis algoritmu, navrhuje sa postup implementácie a opisuje sa implementácia algoritmu. Okrem toho je navrhnutý aj postup aktualizácie pluginu Community Detection a popisuje sa priebeh tejto migrácie. V častiach 3.6 a 3.7 sú popísané dátové množiny, ktoré sa na testovanie používaju a spôsob hodnotenia algoritmu.

Kapitola 4 opisuje výsledky zhlukovania algoritmov a diskutuje o nich.

Kapitola 1

Súčasný stav oblasti

Rozsah dát, ktoré vznikajú počas experimentov, pri meraniach, digitalizácii, finančných transakciách alebo používateľskej interakcii stále narastá.[5, 14] Aby sa iba bez úžitku nehromadili v databázach, je nutné ich spracúvať porovnateľným tempom (alebo túto rýchlosť neskôr dosiahnuť). Tým, čo ich vznik natoľko urýchlilo bola automatizácia a digitalizácia. Ak teda chceme s produkciou udržať krok, je vhodným nápadom využiť na to podobné technológie. Na to je nutné automatizovať a optimalizovať stále väčšiu časť s tým súvisiacich procesov.

Pre množstvo dát a s tým spojenú výpočtovú zložitosť môžeme pri skutočných problémoch iba ťažko dosahovať perfektné výsledky. Výpočet všetkých možností jednoducho nemusí byť dosiahnuteľný.[6] Preto sa využívajú heuristiky (odhadovacie algoritmy) a inteligentné procesy. Tie za cenu zníženej presnosti dosahujú štatisticky dobré výsledky omnoho rýchlejšie[8].

1.1 Strojové učenie (Machine learning)

Strojové učenie je oblasť venujúca sa algoritmom, ktoré sa postupne vylepšujú (učia), ako problémy riešiť.[13] Na to, aby sa mohli učiť, získavajú znalosti z prostredia. Využívajú sa napríklad pre odporúčacie algoritmy, riadenie autonómnych vozidiel, optimalizáciu dopravy alebo filtrovanie spamu.

Pri analýze dát niekedy vieme, čo sa snažíme zistiť alebo máme vhodné doménové znalosti. To je pri riešení problému prospešné - pretože vieme lepšie ohodnotiť dosiahnuté výsledky, vybrať vhodnejší algoritmus, lepšie ho nakonfigurovať a podobne.

Môže sa ale stať, že vopred nevieme, čo máme v dátach hľadať a aké pravidelnosti čakať. Alebo bude manuálna práca odborníka plytvaním na spracovanie menej dôležitých zdrojov dát. Alebo to nebude pre rýchlosť generácie a ich objem jednoducho možné. Vtedy potrebujeme všeobecnejšie riešenie, ktoré sa nebude môcť spoliehať na priamu ľudskú expertízu a expertov.

Podľa prístupu k učeniu sa strojové učenie delí na 2 základne druhy:

- Učenie s učiteľom

  Algoritmus má prístup ku označkovaným dátam. Vďaka tomu si vie overiť, nakoľko sú jeho výsledky správne. Popri tom, ako si buduje svoj model problému a ho teda môže jeho učiteľ kontrolovať a usmerňovať. Napríklad, algoritmus na rozpoznávanie obrazu si vie skontrolovať, či správne určil objekty na fotografii.

- Učenie bez učiteľa

  Pri tomto prístupe algoritmus nemá k dispozícií správne výsledky. Snaží sa model problému vytvoriť podľa vzťahov alebo závislostí, ktoré sám objaví. Dôvodom, prečo algoritmus nemal k dispozícii známe výsledky môže byť napríklad potreba množstva ľudskej práce na vyhodnocovanie nahrávok ľudskej reči, ale aj príliš komplikované alebo nenápadné vzory v záplave astronomických dát. Nedostatok predspracovaných dát potom môže spôsobiť, že sa ťažko dokazuje alebo rozhoduje správnosť rozhodnutí algoritmu.

1.2 Získavanie znalostí (Data mining)

Oblasťou, ktorá takisto využíva inteligentné algoritmy je dolovanie informácií. Cieľom je získať z dát nové poznatky. Môžu to byť pravidelnosti, anomálie alebo aj vzťahy medzi objektmi. Časté problémy pri dolovaní v dátach sú [1]:

- Detekcia anomálií

  Hľadanie neobvyklých objektov. Môže sa jednať o chyby alebo dôležité prvky systému. To sa využíva napríklad na detekciu finančných podvodov[16] alebo chýb v texte.

- Analýza asociácií

  Hľadanie asociačných pravidiel medzi prvkami. Tým je možné nájsť rôzne skupiny, štruktúry alebo závislosti. Napríklad, objekty A sa vyskytuje vždy nejaký čas pred objektami B a C. Toto je zvyčajne užitočné pri analýze používateľov na webe, alebo spotrebiteľov v obchode.

- Zhlukovanie

  Proces zatrieďovania objektov do skupín, v rámci ktorých by si mali byť prvky čo najpodobnejšie (najbližšie) nejakými vlastnosťami. Napríklad polohou, farbou, rozmermi alebo ľubovoľnou kombináciou iných atribútov.

- Klasifikácia

  Rozdeľovanie objektov podľa ich vlastností do príslušných kategórií. Oproti predošlému zhlukovaniu je pri klasifikácií vopred známe, do akých kategórií objekty môžu patriť. To sa môže využiť na rozpoznávanie v obraze pre autonómne vozidlá alebo rozlišovanie hlasu v zvukovej nahrávke.

- Regresia

  Snaha nájsť funkciu, ktorá by modelovala dáta čo najpresnejšie. Na základe toho sa potom dá odhadovať nové vzťahy a závislosti medzi objektmi. Môže to byť užitočné na ich komprimáciu alebo aj na detekciu chýbajúcich dát.

- Sumarizácia

  Zhrnutie daných dát menšou reprezentatívnou vzorkou, pri zachovaní dôležitých informácii. Pre text to môžu byť najdôležitejšie vety, vo videu najdôležitejšie zábery.

  Sumarizácia sa ďalej rozlišuje podľa toho, či sa pôvodný materiál iba cituje, parafrázuje, alebo sa výsledok použije ako základ pre ľudskú sumarizáciu.

Pretože sa snažíme procesy spracovania dát automatizovať (aby sa dali škálovať), musíme stále viac uplatňovať autonómne postupy. Opätovne sa vyskytujúci problém je samostatná analýza dát. Medzi hlavné spôsoby, ako to dosiahnuť patrí zhlukovanie. [12]

1.3 Zhlukovanie

Zhlukovanie je proces, ktorým sa objekty triedia do zhlukov (komunít). Cieľom je nájsť v dátach (grafe, množine dátových bodov, ...) skupiny bez toho, aby sme vopred poznali ich štruktúru.[18] To je užitočné na analýzu týchto systémov – môžeme tak empiricky zistiť, aké druhy objektov systém obsahuje, ako spolu súvisia a aké črty daný systém má.

Ku zhlukovaniu existuje veľké množstvo prístupov, pretože pojem zhluk nie je presne definovaný. Je určený v závislosti od konkrétnej domény a situácie [6]. Často je dokonca zhluk vymedzený iba algoritmicky – teda ako výstup daného algoritmu.

Zhlukovanie sa dá podľa prístupu rozdeliť na:

- Hierarchické zhlukovanie

  To je založené na spájaní prvkov (aglomeratívny postup) alebo delení celku (divizívny postup). Algoritmus postupne delí celok na menšie časti, pokým nedosiahne

dostatočné rozdelenie, prípadne až pokým nie je každý prvok osamotený. V opačnom smere tak každý prvok začína a postupne sa pospájajú do jedného zhluku. Výsledkom takéhoto zhlukovania je dendrogram – stromový graf popisujúci ako sa zhluky postupne pohlcovali / delili. Jednotlivé stupne zhlukovania sa potom nachádzajú na rezoch (vrstvách) dendrogramom.

- Fuzzy zhlukovanie

  Objekty sa dajú prideliť do zhlukov jednoznačne. Vtedy sa stráca informácia, ako prvky súvisia s ostatnými skupinami, do ktorých nepatria. Okrem toho existujú aj fuzzy zhlukovania – objekty sa pri nich môžu priradiť do viacerých zhlukov zároveň a v rôznej miere. Výsledkom je menej jednoznačné rozdelenie, ktoré ale zachováva komplexnejšie vzťahy medzi prvkami alebo aj celými zhlukmi.

- Zhlukovanie založené na ťažiskách

  Pri takomto prístupe je snaha dosiahnuť zhlukovanie v ktorom sú body zoskupované tak, aby boli čo najbližšie k ťažisku svojho zhluku. Algoritmus K-means to dosahuje tak, že body najprv náhodne poprideľuje k zhlukom. Následne každému zhluku vypočíta ťažisko a preznačí body podľa najbližšieho zhluku. Takéto posúvanie ťažísk opakuje pokým sa výsledok neustáli.

  Nevýhodou je, že počet zhlukov k musí byť pre algoritmus vopred určený. To sa dá prekonať tak, že sa postupne skúšajú rôzne hodnoty k. Výsledky sa porovnávajú a na grafe vyhodnocovacej funkcie sa hľadá tzv. lakeť - tu sa ohodnotenie prestane výrazne zlepšovať, potom sa ako ideálna hodnota k vyberie tá, pri ktorej sa dosiahol takýto "lakeť", pretože ďalšie iterácie už prinášajú iba zanedbateľné zlepšenia.

  Nevýhodou tohto prístupu je, že "lakeť"sa nemusí zreteľne objaviť a voľba najlepšieho k je potom subjektívna.

- Zhlukovanie založené na hustote

  Tu sú zhluky tvorené podľa hustoty pokrytia bodmi, body, ktoré majú dostatok susedov sa nachádzajú v zhluku a tie s nedostatkom susedov sa vyhodnotia ako šum. Príkladom je DBSCAN. [4]

- Zhlukovanie založené na distribúcií

  Dáta sú modelované rôznymi štatistickými modelmi (napríklad Gaussova distribúcia). Algoritmus sa potom snaží nájsť také výsledné zhlukovanie, aby čo najlepšie zodpovedalo danej distribúcií.

- Spektrálne zhlukovanie

Využíva maticu podobnosti objektov a potom hľadá zhlukovanie na grafe ktorý reprezentuje. Účinnosť toho prístupu závisí od spôsobu akým sa podobnosť určuje. Výsledné zhlukovanie môže upriamovať pozornosť na rôzne vlastnosti, alebo pri nesprávnej príprave dát vôbec nedávať zmysel.

Oblasť zhlukovania v grafe (sieti) môže využívať aj mnoho prístupov z oblasti teórie grafov. Napríklad zhlukovanie podľa stupňa vrcholu, hustoty hrán alebo hľadaním kompletných podgrafov.

Okrem toho sa pri zhlukovaní využívajú aj simulácie fyzikálnych alebo biologických procesov. Metódy sú často kombinované alebo prispôsobené aktuálnemu problému. Pretože zhlukovanie nemá presne určený postup, prístupov je naozaj veľa. [9]

Po vykonaní zhlukovania nasleduje ďalší problém - ako ho ohodnotiť.

Používané metódy evaluácie zhlukovania sú:

Interná evaluácia

Je vyhodnocovaná čisto na základe nájdeneho zhlukovania pre danú dátovú množinu. Pretože ale nepoznáme správny výsledok, môžeme iba odhadovať, nakoľko je dosiahnuté zhlukovanie dobré.

Príklady hodnotenia kvality:

- Davis-Bouldin index

  Hodnotí zhlukovanie podľa toho, ako sú rozptýlené body zhluku od jeho ťažiska.

- Silhouette coefficient

  Porovnáva priemernú vzdialenosť objektov v zhluku s priemernou vzdialenosťou objektov v rôznych zhlukoch.

Externá evaluácia

Je vyhodnocovaná na základe dát, ktoré sa nepoužívali na zhlukovanie - vopred známych priradeniach alebo testovaním algoritmu na iných dátových množín, ktoré sú vopred vyhodnotené.

Príklady hodnotení:

- Rand index

  Počíta pomer správnych zaradení (True positive + True negative) ku všetkým zaradeniam. Správne rozloženie sa získa z vopred známeho zhlukovania týchto dát.

- Jaccard index

Hľadá prienik výsledného zhlukovania pre vyhodnocovaný algoritmus a známe zhlukovanie.

Ďalšou možnosťou je manuálne hodnotenie, ktoré je ale podobne subjektívne.

Najreálnejším spôsobom evaluácie zhlukovania je, či z výsledku vyplynie nejaký úžitok.

1.4 Teória grafov

Pod pojmom graf rozumieme nejakú neprázdnu množinu vrcholov (objektov), ktoré môžu byť poprepájané hranami (spojnicami).[17]

Takýto graf sa dá reprezentovať napríklad diagramom. V prípade komplikovanejších grafov sa ale takéto zobrazenie ľahko stane neprehľadným.

Ďalším spôsobom zobrazenia grafu je matica susednosti. Táto matica je štvorcová, rozmeru $N \times N$, kde N je počet vrcholov. Každý prvok matice v stĺpci A na riadku B potom reprezentuje susedstvo dvoch vrcholov A a B. Takéto znázornenie ale tiež môže byť nepraktické, pretože sa v ňom značia všetky potenciálne susednosti (N^2 prvkov matice). Všetky susednosti sa však vyskytujú jedine v úplných grafoch. Pre všetky ostatné grafy by sa tak zaznamenávala aj každa neexistujúca hrana. Riešením je graf reprezentovať ako zoznam vrcholov a zoznam hrán. Vtedy sa v grafe vyskytujú práve tie hrany, ktoré sú vymenované.

Grafy sa delia do viacerých (nie vždy disjunktných) kategórií podľa svojich vlastností:

- Orientovanosť hrán

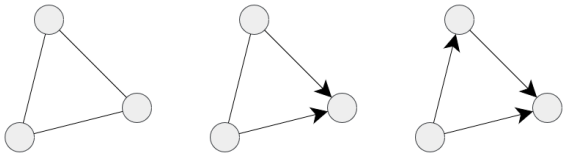
  Hrany grafu môžu byť orientované. Vtedy susednosť dvoch vrcholov neplatí obojsmerne – teda hrana spájajúca vrchol A s vrcholom B nespája tieto vrcholy v opačnom smere. Môže ich ale spájať iná hrana z B do A.

  V závislosti od toho, aké druhy hrán graf obsahuje môže graf byť:

  - orientovaný, všetky hrany takéhoto grafu sú orientované
  - migraf (mixed graph), obsahuje orientované aj neorientované hrany
  - neorientovaný, všetky jeho hrany sú neorientované [viď obr.1.4]

- Násobnosť hrán

  Ak graf obsahuje viacnásobné hrany (viacero hrán spájajúcich ten istý vrchol s tým istým ďalším vrcholom) - jedná sa o multigraf. Násobnou hranou nie je dvojica orientovaných hrán, ktoré spájajú dva vrcholy v opačných smeroch - vtedy sa jedná o dve rôzne hrany, nie jednu dvojnásobnú.



Obr. 1.1: Na diagrame sú odľava: neorientovaný graf, migraf a orientovaný graf

- Slučky

  V grafe sa môžu vyskytovať hrany, ktoré spájajú niektorý vrchol sám so sebou. Takéto hrany sa označujú ako slučka.

- Ohodnotenie (váha)

  Hranám alebo vrcholom môže byť priradená nejaká hodnota. Tá môže reprezentovať cenu za návštevu vrcholu, prechodu po hrane, jej dĺžku, pravdepodobnosť prechodu, alebo iné vlastnosti.

  Ak graf obsahuje ohodnotené prvky, nazýva sa ohodnoteným grafom.

- Cesta

  Cestou je v grafe taká postupnosť hrán, v ktorej po sebe nasledujúce hrany zdieľajú vrchol a táto postupnosť neprechádza žiadnou hranou ani vrcholom viackrát.

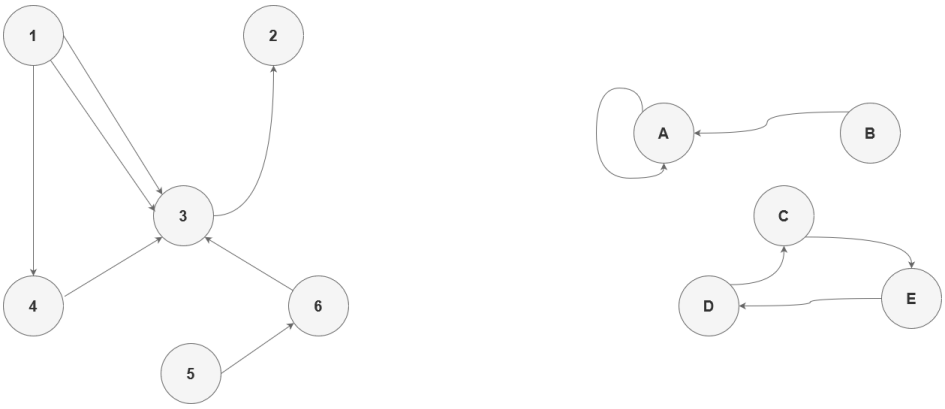
- Prepojenosť grafu

  Graf je súvislý, ak neobsahuje žiadne izolované vrcholy ani skupiny vrcholov. Pre neorientovaný graf vtedy platí, že medzi každými dvoma vrcholmi existuje aspoň jedna cesta. Pri orientovanom grafe sa považuje za súvislý taký graf, v ktorom by pre každú dvojicu vrcholov existovala aspoň jedna spájajúca cesta, ak by pre každú hranu existovala aj jej protihrana (hrana medzi rovnakými vrcholmi, v opačnom smere).

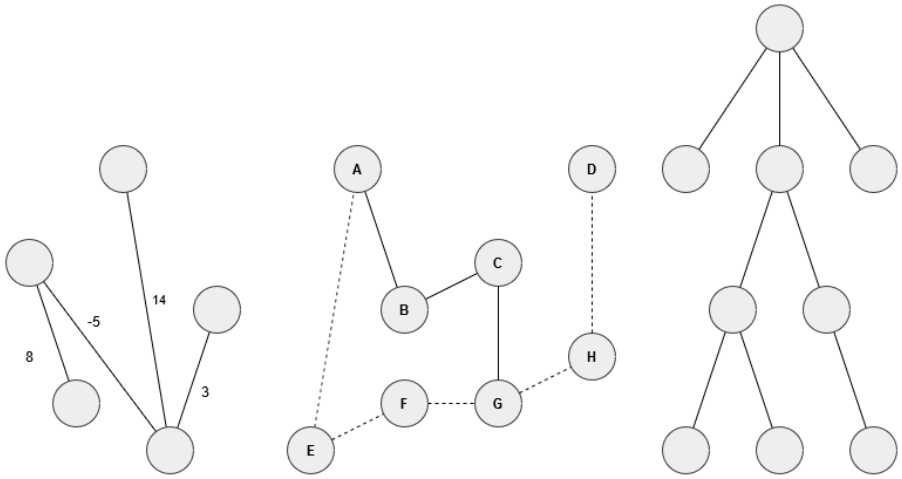
- Strom

  Strom je taký spojitý neorientovaný graf, pri ktorom každú dvojicu vrcholov spája práve jedna cesta. Z toho vyplýva, že v stromovom grafe sa nemôžu vyskytovať cykly. [viď obr.1.3]

Na reprezentáciu takýchto grafov sa v matici susednosti využíva pri označení existencie hrany namiesto čísla 1 priamo jej váha. Na rozlíšenie orientácie hrany sa použije prvok v riadku x a stĺpci y pre jeden smer hrany a prvok v riadku y a stĺpci x pre druhý smer hrany.



Obr. 1.2: Na diagrame je vľavo (s číselnými vrcholmi) súvislý multigraf, a napravo (vrcholy označené písmenami) nesúvislý graf so slučkou.



Obr. 1.3: Na diagrame sú odľava: ohodnotený graf, graf so zvýraznenou cestou cez vrcholy A-E-F-G-H-D, stromový graf.

1.5 Markovovský model

Markovovský model reprezentuje stochastický proces. Teda proces založený na náhodnom alebo pravdepodobnostnom princípe. Takýto model musí spĺňať podmienku, že model nemá pamäť (má tzv. Markov property). To znamená, že jeho história nemá žiadny vplyv na jeho ďalšie správanie. Každá ďalšia zmena alebo nasledujúca konfigurácia závisí jedine od aktuálneho stavu modelu.

To je veľmi užitočné pri modelovaní stochastických procesov, pretože vplyv histórie na ďalšie správanie by pridával modelu rýchlo pribúdajúcu komplexitu.

Markovovský model je typicky reprezentovaný stochastickou (pravdepodobnostnou) maticou. To je štvorcová matica rozmeru $N \times N$, ktoré obsahuje kladné hodnoty. Okrem toho sa musí skladať zo stochastických vektorov (vektor nezáporných reálnych čisel, ktorých súčet je 1). Podľa toho môže byť stochastická matica typu:

- Pravá stochastická matica Súčet hodnôt v každom jej riadku je rovný 1. Matica sa skladá z N stochastických vektorov nad sebou.

- Ľavá stochasticka matica Súčet hodnôt v každom je stĺpci je rovný 1.

- Dvojito stochastická matica. Matica spĺňa zároveň podmienky na pravú aj ľavú stochastickú maticu.

Takýto model sa dá veľmi jednoducho reprezentovať grafom, zodpovedajúcim tejto stochastickej matici. A to ako skupinu stavov (vrcholov), s rôznymi prechodmy (hranami), ktoré majú istú pravdepodobnosť (váhu).

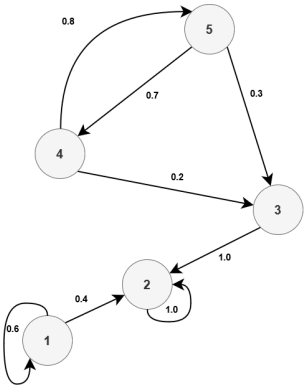
Tento model môže byť statický alebo dynamický vzhľadom na čas - udržiavať si rovnakú konfiguráciu nezávisle na plynutí času, alebo sa s behom času meniť (pridávať alebo odoberať stavy, prechody, meniť pravdepodobnosti prechodov...).

1.5.1 Markovovský reťazec

Markovovský reťazec je taký markovovský model, v ktorom je vždy plne pozorovateľný aktuálny stav. Teda má okrem pravdepodobností prechodov aj vybratý určitý stav, v ktorom sa systém práve nachádza.

Typicky sa za markovovské reťazce považujú také, kde čas plynie diskrétne a stavov je počítateľné množstvo. Existujú ale aj generalizácie, ktoré pracujú so spojitým časovým priebehom, alebo stavovým priestorom.

Ak je v markovovskom reťazci možné dosiahnuť stav, z ktorého už nemožno vyjsť, jedná sa o absorbujúci markovovský reťazec. Samotný reťazec je potom absorbujúcim markovovským reťazcom. [viď obr.1.4]



	1	2	3	4	5
1	0.6	0.4	0	0	0
2	0	1	0	0	0
3	0	1	0	0	0
4	0	0	0.2	0	0.8
5	0	0	0.3	0.7	0

Obr. 1.4: Príklad absorbujúceho markovovského reťazca s absorbujúcim stavom 2 (digram vľavo, pravá stoch. matica vpravo)

1.6 Predchádzajúce práce

1.6.1 Vyhľadávanie klastrov v grafoch a vizualizácia získaných dát - Matej Šmitala

V tejto práci sa autor zaoberá zhlukovacími algoritmami ktoré sa dajú využívať pre väčšie grafy (rádovo stovky tisíc vrcholov). Algoritmy boli vyberané tak, aby pokryly viacero rôznych prístupov ku zhlukovanie. Venuje sa ich implementácií, analýze a vizualizácií. Implementuje nasledujúce algoritmy:

- SCAN (Structural Clustering Algorithm for Networks)[21]

  Algoritmus, ktorý na pracuje na základe hustoty rozloženia. Vrcholy spája podľa množstva zdieľaných susedov. Dokáže rozoznávať aj mosty - prepojenia zhlukov a šum - vrcholy, ktoré do zhlukov nepatria.

- Link communities

  Tento zhlukovací algoritmus patrí k aglomeratívnym hierarchickým. Prebieha takto: najprv každú hranu priradí do vlastného zhluku. Potom vyrátava podobnosť pre všetky hrany, ktoré zdieľajú vrchol a postupne spája tie najpodobnejšie. Postupné spájanie zhlukov sa ukladá do stromu, ktorý sa dá reprezentovať dendrogramom. Výsledkom je hierarchia zhlukov, do ktorých sa postupne hrany nabaľovali.

- SLPA (Speaker-Listener label Propagation Algorithm)[20]

  SLPA hľadá komunity tak, že simuluje šírenie informácií medzi vrcholmi. Robí to tak, že na začiatku sa každému vrchol nastaví do pamäti iba jeho identifikátor. Potom sa pri každej iterácií vyberie poslucháč, ktorý si vypočuje od svojich susedov náhodne vybratý prvok z ich pamäti. Poslucháč si potom zapamätá najčastejšiu informáciu.

  Výsledkom je prekrývajúce sa zhlukovanie, s príslušnosťou danou tým, v akom pomere si vrcholy pamätali jednotlivé identifikátory.

Okrem implementácie sa autor venoval aj vizualizácií a evaluácií týchto algoritmov. Implementoval internú evaluáciu použitím Davies - Bouldin indexu a porovnávania počtu vnútornych s vonkajšimi hranami. Externú evaluáciu vykonal formou Rand indexu, F skóre a NMI. Výsledkom tejto práce je okrem zhodnotenia spomenutých algorimtov aj plugin Community Detection pre program Gephi.

Tento plugin bude pre moju prácu užitocný pri analýze a hodnotení navrhnutého algoritmu. Nevýhodou je, že plugin bol vytvorený pre staršiu verziu Gephi. V tom prišlo k zmene systému pre spravovanie projektov z Apache Ant na Apache Maven. Pre využitie pluginu v aktuálnej verzií Gephi je preto potrebná jeho migrácia.

1.6.2 MCL – Markov Cluster Algorithm

MCL je zhlukovací algoritmus objavený Holanďanom Stijn van Dongenom v roku 1997.[19] Je založený na simulácií náhodného procesu (stochastic flow) v grafe. Komunity by v takomto grafe mali byť prepojené častejšie vnútorne než s inými komunitami, preto by náhodné prechody mali najčastejšie chodiť práve po hranách v rámci komunity. Algoritmus v princípe prebieha opakovaním dvoch operácií : expanzie a inflácie.

Expanzia je vykonávaná umocnením matice grafu. To na grafe reprezentuje náhodné prechody cez viacero krokov.

Inflácia sa vykonáva ako umocnenie jednotlivých prvkov matice. Tá má v grafe posilniť cesty s vyššou pravdepodobnosťou a oslabiť tie menej pravdepodobné. Počas tejto operácie sa v matici zväčša vychýlia pravdepodobnosti prechodu a preto sa prvky musia normalizovať, aby sa navrátila stochastická vlastnosť matice – súčet prvkov v stĺpci má byť rovný 1.

Algoritmus sa dá opísať takto:

```
majme graf G
pridajme vrcholom G slucky

nastavme parameter r pre funkciu inflacie

majme M_1 ako maticu nahodnych prechodov G

opakuj pokym dochadza k zmene:
    M_2 = M_1 \times M_1
    M_1 = inflacia(M_2)
    zmena = rozdiel(M_1, M_2)

vysledne zhlukovanie je dane castami M_1
```

Výhodou tohto algoritmu je jeho jednoduchosť - má jediný parameter, ktorým sa dá ovplyvňovať granularita výsledného zhlukovania. Vďaka tomu sa dá používať aj bez potreby doménových znalostí. Jeho nevýhodou je, že sa zle vyrovnáva s grafmi, ktoré obsahujú cykly, alebo jednosmerné hrany.

Autor MCL neskôr úspešne použil na rozdelenie proteínových sekvencií do skupín, pričom jeho použitie klasifikáciu výrazne urýchlilo.[3]

1.6.3 Detecting Communities using Absorbing Markov Chains - Ankit Kumar

Autor v práci navrhuje a hodnotí algoritmus, ktorý by mal odhadovať dobré zhlukovania v polynomiálnom čase.

Algoritmus ku pôvodným vrcholom grafu pridáva nové absorbujúce vrcholy, ktoré pripojí hranami do pôvodného grafu. Rozdelenie grafu do zhlukov potom vyplýva z toho, do ktorého z pridaných vrcholov sú prvky pôvodného grafu najčastejšie absorbované.

Na spôsob počiatočného rozloženia týchto nových "jadier"použil viacero techník:

- Absorbujúce vrcholy sú prídávané na náhodné pozície, čo je vyvážené tým, že sa výpočet opakuje veľa krát a použije sa jeho najlepší výsledok (na to je potrebná vyhodnocovacia funkcia).

- Vhodné jadrá sa hľadajú ako pozície, ktoré sú vzájomne málo navštevované pri náhodných prechodoch (sú od seba "vzdialené"). Jadrá sa hľadajú ako ďalší najvzdialenejší vrchol.

- Hľadaním jadier pomocou algoritmu k-means podľa vzdialenosti vrcholov. K-means nevyužíva vlastnosti grafu ako súboru vrcholov a hrán, ale pracuje nad transformovanou reprezentáciou pôvodného grafu. Nájdené jadrá sa potom dajú použiť na štandardný beh algoritmu.

Z týchto možností bola najužitočnejšia metóda s použitím k-means.

Algoritmus je typu expectation-maximization. Teda opakovane odhaduje rozdelenie vrcholov v zhlukoch a obnovuje prepájanie pridaných jadier tak, aby dosahoval čo najväčsiu pravdepodobnosť, že vrcholy priradené do určitého zhluku sú pohlcované práve jeho jadrom. Cyklus odhadu rozdelenia a obnovy prepájania jadier sa opakuje, pokým sa výsledok neustáli.

Algoritmus umožnuje binárnu ale aj fuzzy klasifikáciu. Pri fuzzy klasifikícii dosahuje lepšie výsledky ale zároveň viac zachováva menšie zhluky. Tie nie sú absorbované väčšími jednoznačnými zhlukmi ale často vznikajú z osamotenejších vrcholov, ktoré do nich patria viac, než do väčších zhlukov.

Fuzzy klasifikácia okrem toho dobre reprezentuje, aké podobné sú si zhluky navzájom (môžu mať rôzne veľký prienik vrcholov).

Výpočtová zložitosť algoritmu je $O(n^3)$, ale operácie, ktoré túto zložitosť spôsobujú sa vyskytujú pomerne zriedka (maticová inverzia na začiatku a jeden súčin matíc počas každej iterácie).

Pri porovnávaní nad často používanými dátovými množinami dosahoval porovnateľné výsledky s algoritmami MCL [19] a WalkTrap [15].

Autor svoj algoritmus hodnotí ako úspešný, pretože dosahuje dobré výsledky s rozumnou časovou zložitosťou.

Nakoniec odporúča možné vylepšenia pre algoritmus, ktoré by mohli odstrániť potrebu vopred vedieť (alebo skúšať) hodnotu K - počet zhlukov. Tento algoritmus sa vo svojej práci pokúšam implementovať.

1.7 Použité technológie

1.7.1 Java

Java je objektovo orientovaný kompilovaný programovací jazyk. Je veľmi ľahko prenosný, pretože sa spúšta na Java virtálnom stroji (Java virtual machine), ktorý má mnoho implementácií pre rôzne architektúry zariadení. Kód je teda vytvorený univerzálne a program je vykonateľný na každom zariadení, ktoré podporuje platformu Java virtuálneho stroja. V tejto práci sa používa verzia Java 1.11.

1.7.2 Maven

Apache Maven je nástroj na automatizáciu zostavovania softvéru (software build automation tool). Pre projekt spravuje závislosti a umožnuje štandardizované vykonávanie obvyklých krokov pri tvorbe softvéru: validácia, kompilácia, testovanie, ... až po nasadenie na vzdialený repozitár.

1.7.3 Gephi

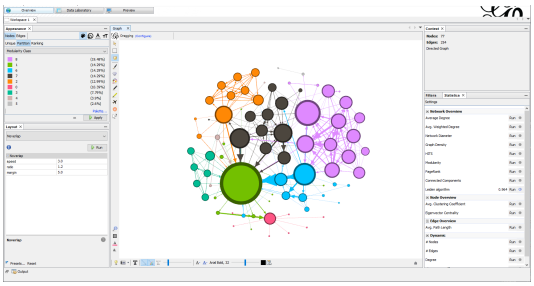
Gephi je open-source softvér určený na vizualizáciu grafov. Je vytvorený v jazyku Java na platforme NetBeans. Na vizualizáciu využíva rozhranie OpenGL. [2]

Podporuje import a export grafov v štandardných formátoch (napr. CSV, GML, GEXF) ale aj dopytom z relačnej databázy. Navyše je možné pridať vlastný importer ako rozširujúci plugin. Gephi podporuje aj generovanie grafov, pričom nové generátory môžu byť pridávané ako súčasť pluginov.

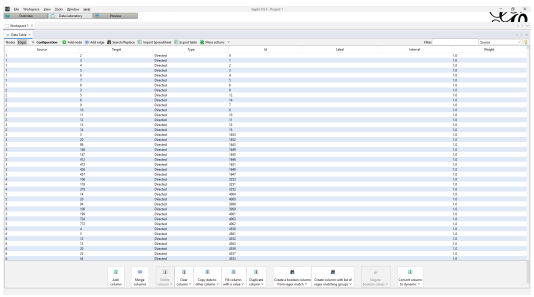
Gephi dokáže pracovať s grafmi, ktoré sú orientované, neorientované aj kombinované. Okrem toho môže byť graf aj dynamický, teda sa jeho konfigurácia v čase mení. Každý vrchol alebo hrana grafu môže mať priradené ľubovoľné atribúty.

S grafmi je možné manipulovať v reálnom čase a upravovať atribúty vrcholom aj hranám alebo inak zasahovať do grafu v 3 rôznych prostrediach.

V základnom náhľade [viď obr. 1.5] je možné pracovať s vizualizáciou grafu. Graf je zobrazený v centrálnom paneli. S týmto zobrazením je možné priamo manipulovať (presúvať vrcholy po ploche, zvýrazňovať ich, pridávať nové hrany alebo aj vrcholy.



Obr. 1.5: Obrazovka interaktívnej vizualizácie v Gephi



Obr. 1.6: Obrazovka tabuľkového náhľadu v Gephi

Na podobné úpravy je možné použiť aj nástroje na zafarbenie vrcholov alebo hrán podľa ich vlastností. Úpravu rozloženia grafu je možné realizovať príslušnými algoritmami.

Vrcholy alebo hrany grafu je možné filtrovať podľa rôznych funkcií. Okrem toho sa prostredie umožňuje spustiť aj ľubovoľný algoritmus, ktorý pre graf môže napríklad hľadať zhlukovanie, odstraňovať niektoré prvky alebo vypočítať nejaké štatistické vlastnosti.

V tabuľkovom náhľade [viď obr. 1.6] je možné priamo pracovať s dátami, ktoré graf reprezentuje.

Nakoniec Gephi umožňuje aranžovať graf pre lepšiu výstupnú vizualizáciu [viď obr. 1.7]. V tomto náhľade je možné zvýrazňovať prvky grafu, ich popisy, nastaviť, aká časť grafu má byť zahrnutá a podobne.

Obr. 1.7: Obrazovka závernej vizualizácie v Gephi

Kapitola 2

Cieľ práce

Cieľom práce je implementovať algoritmus na hľadanie komunít v grafe, založené na princípe absorbujúcich markovovských reťazcov. Tento algoritmus navrhol vo svojej práci A. Kumar [11]. Algoritmus bol navrhnutý v snahe dosiahnuť dobré výsledné rozdelenie komunít, ale s polynomiálnou výpočtovou zložitosťou.

Tento algoritmus má byť implementovaný ako rozšírenie pluginu Community Detection pre platformu Gephi.

Implementácii algoritmu do pluginu CommunityDetection by zjednodušila jeho migrácia na aktuálnu verziu Gephi, pretože by sa tak zlepšila kvalita výsledku.

Platforma Gephi pri novších verziách vylepšila účinnosť hlavnej grafovej implementácie, využíva pokročilejšiu verziu jazyka Java (Java 1.11 oproti Jave 1.7) a opravila viacero chýb a nedostatkov.

Preto je čiastkovým cieľom migrácia pluginu CommunityDetection na novšiu verziu Gephi (0.9.5).

Po implementácii, má byť algoritmus otestovaný a porovnaný (externá evaluácia) s ostatnými algoritmami v plugine Community Detection:

- SCAN (Structural Clustering Algorithm for Networks)
- Link communities
- SLPA (Speaker-Listener label Propagation Algorithm)

Na priame hodnotenie (internú evaluáciu) sa využije hodnotenie z pluginu Community Detection podľa parametrov:

- Pomer počtu vnútro-komunitných hrán ku počtu hrán medzi rôznymi komunitami.
- Davies-Bouldin index

Algoritmy sa budú spúšťať aj testovať na dátových množinách použitých v diplomovej práci M. Šmitalu [23]. Výsledky tak bude možné jednoducho porovnať.

Kapitola 3

Metodika práce

3.1 Špecifikácia algoritmu

Navrhnutý algoritmus rozširuje pôvodný model pridaním r nových stavov, ktoré sú absorbujúce. Z pôvodných stavov pridáva prechody do týchto nových. Stavy rozdeľuje do zhlukov podľa toho, ktorý stav má najvyššiu pravdepodobnosť, že prechod (začínajúci v danom vrchole) absorbuje.

Rozšírená prechodová matica P je potom:

$$P = \begin{bmatrix} Q & R \\ 0 & I_r \end{bmatrix} \tag{3.1}$$

Kde Q je matica rozmeru $t \times t$ pre t pôvodných stavov, popisujúca prechody medzi pôvodnými stavmi modelu. R je matica rozmeru $t \times r$, ktorá popisuje prechody z pôvodných stavov do novo-pridaných absorbujúcich stavov. 0 je nulová matica $r \times t$ popisujúca prechody z nových absorbujúcich stavov do pôvodných (absorbujúce stavy nemajú prechody do iných stavov, čo vyplýva z ich absorbujúcej vlastnosti). I_r je jednotková matica $r \times r$, ktorá popisuje prechody z absorbujúcich stavov do iných absorbujúcich stavov (absorbujúci stav má jediný prechod - sám na seba).

Pre tento model je možné vyjadriť maticu N počtov očakávaných návštev pred absorbovaním. Prvok N_{ij} reprezentuje počet očakávaných návštev vrcholu j pri prechodoch s počiatočným stavom i pred tým, než dôjde k absorbovaniu do niektorého z absorbujúcich stavov.

Pravdepodobnosť prechodu z vrcholu i do j po práve k krokoch je rovná Q^k_{ij}. Preto:

$$N = \sum_{k=0}^{\infty} Q^k = (I_t - Q)^{-1} \tag{3.2}$$

Kde I_t je jednotková matica $t \times t$.

Pravdepodobnosť, že prechod, začínajúci v stave i bude absorbovaný práve v absorbujúcom stave j je potom vyjadrená prvkom B_{ij} matice B:

$$B = NR \tag{3.3}$$

Algoritmus v prvom kroku inicializuje rozšírený model a každému z pôvodných vrcholov pridá prechody do absorbujúcich stavov s iniciálnou váhou.

Potom opakovane prebiehajú kroky maximalizácie strednej hodnoty (z angl. Expectation Maximization), pokým nie je dosiahnuté toto lokálne maximum. Vyhodnocovacia funkcia pre algoritmus je daná ako:

$$\frac{\sum_{i=0}^{r} P(\text{absorbcia v } c_i|\text{začiatok v zhluku } i)}{r} \tag{3.4}$$

Kde c_i je absorbujúci stav pre zhluk i.

Počas každej iterácie algoritmu sa vyhodnocuje aktuálne rozdelenie do zhlukov a následne sa upravia prechody do absorbujúcich stavov tak, aby sa maximalizovala vyhodnocovacia funkcia.

3.2 Návrh implementácie algoritmu

Pri spúštaní algoritmu sa zadá počet hladaných zhlukov v grafe. Algoritmus načíta maticu susednosti Q z grafu.

Matica Q sa normalizuje - pre každý riadok ostane pomer hodnôt zachovaný, ale hodnoty sú vydelené sučtom hodnôt tohoto riadku.

Vypočíta sa matica N (hodnota N_{ab} reprezentuje očakávaný počet návštev vrcholu b pred absorbovaním, ak prechod začína vrcholom a). Podľa rovnice 3.2 sa dá N vypočítať pomocou inverzie matice $(I_t - Q)^{-1}$. Táto matica však môže byť singulárna - neexistuje pre ňu inverzná matica. Ak táto inverzia teda zlyhá, hodnota nekonečnej sumy sa môže aproximovať sumou po nejaké vysoké číslo. Matica N sa potom normalizuje.

Do grafu sa pridá r nových absorbujúcich stavov (podľa hodnoty nastavenej argumentom). V pôvodnej implementácií [11] autor skúšal viacero inicializačných prístupov a ako najvhodnejší vyhodnotil (pre jeho konzistentnosť) K-Means nad maticou N (3.2). Graf pri tom interpretoval tak, že vrcholy v grafe považoval za ležiace na vektore podľa matice N a vzdialenosť vrcholov A a B za počet návštev B pred absorbovaním prechodu začínajúcom v A (teda N_{AB}). Každý riadok matice tak zodpovedá jednému vektoru v t-rozmernom priestore, kde t je počet stĺpcov matice N.

Inicializácia teda bude využívať na určenie počiatočného rozloženia metódu K-Means. V pôvodnej práci sa nerozoberá, či sa používa naivný algoritmus K-Means alebo nejaká jeho varianta. Preto v práci implementujem K-Means++. K-Means++ oproti základnému algoritmu vylepšuje prvotný výber ťažísk komunít. Sú na to vyberané pozície vrcholov grafu s uprednostňovaním tých vzdialenejších. Tie sa vyberajú

postupne a náhodne, s pravdepodobnosťou úmernou druhej mocnine ich vzdialenosti od najbližšieho iného ťažiska.

Výsledkom inicializácie algoritmom K-Means++ bude matica R, ktorá pridáva prechody do nových absorbujúcich stavov.

Potom nasleduje samotný algoritmus maximalizácie strednej hodnoty (Expectation-Maximization algorithm).

To sa vykoná opakovaným:

- vylepšovaním riešenia

  Hodnoty v matici R sa menia tak, že sa originálnym vrcholom nastaví prechod do toho z absorbujúcich stavov, v ktorom má najväčšiu pravdepodobnosť byť absorbovaný prechod v ňom začínajúci.

  To sa vykoná tak, že sa zostrojí aktuálna matica pravdepodobností absorbovania B (3.3).

  Jej transponovaná matica $V = B^T$, kde hodnota V_{ab} reprezentuje pravdepodobnosť, že prechod začínajúci vrcholom b je absorbovaný v stave a.

  Potom platí:

  $$VB = VNR = K \tag{3.5}$$

  K je matica $r \times r$ ktorej prvok K_{ab} je pravdepodobnosť, že prechod začínajúci v náhodnom vrchole patriacom zhluku a absorbuje zhluk b.

  Za najlepšie riešenie vyhodnocovacia funkcia považuje maximálnu hodnotu na diagonále K, teda pravdepodobností, že vrchol patriaci zhluku v tomto zhluku bude aj absorbovaný.

  Vylepšovanie potom prebieha tak, že sa R_{ji} nastaví ako jediný prechod R_j na váhu 1 pre najvyššiu hodnotu v riadku matice $(VN)^T_{ji}$ a na váhu 0 pre ostatné.

- vyhodnocovaním aktuálneho riešenia

  Zostrojenie matice K (3.5) a odčítanie sumy jej diagonály.

Tieto EM kroky sa opakujú, pokým neprestane stúpať hodnota evaluačnej funkcie.

Nakoniec sa každému vrcholu priradí príslušnosť do zhluku podľa toho, v ktorom absorbujúcom stave končia jeho prechody (podľa aktuálnej matice B 3.3).

Toto finálne rozdelenie je možné interpretovať aj ako mäkké (fuzzy) zhlukovanie, teda s príslušnosťou do viacerých zhlukov, nie iba do zhluku s najvyššou hodnotou. Pre jednoduchosť sa príslušnosť určí na najpravdepodobnejší zhluk. Modifikácia by bola triviálna (pretože matica B sa beztak vypočítava).

3.3 Plán migrácie pluginu Community Detection

Plugin Community Detection od M.Šmitalu [23] bol vytvorený pre prostredie Gephi vo verzií 0.8.2 beta.

Prostredie využívalo na automatizáciu zostavovania softvéru (automating software build processes) nástroj Apache Ant.

Bolo naprogramované v jazyku Java 1.7 a pre jeho spustenie sa vyžadovalo, aby bola nainštalovaná práve táto verzia Javy.

Gephi bolo medzičasom aktualizované a prešlo viacerými zmenami:

- Nástroj na automatizáciu zostavovania softvéru bol nahradený za novší nástroj Apache Maven.

  To vyžadovalo od všetkých pluginov vytvorených pred Gephi 0.9 prechod zo štruktúry projektu pre Ant na projektovú štruktúru pre Maven.

  Pre tento prechod neskôr vznikol nástroj na automatickú migráciu pluginov.

- Časť softvéru ktorá implementovala grafovú štruktúru bola kompletne pretvorená.

  GraphAPI bolo prepísané a osamostatnené do projektu GraphStore. V novej implementácií sa častejšie využívajú jednoduchšie dátové štruktúry a riešenia uľahčujúce využívanie vyrovnávacej pamäte.

- Viacero zmien v aplikačno-programovom rozhraní (Application programming interface - API).

- Novšie verzie Gephi používaju Javu 1.11

  Inštalácia Gephi (od verzie 0.9.3) už Java Runtime Enviroment aj zahŕňa a nie je potrebné ju inštalovať a konfigurovať manuálne.

3.4 Implementácia migrácie pluginu Community Detection

Pre migráciu pluginu Community Detection sa v prvom kroku použil nástroj Gephi Maven Plugin na transformáciu z projektovej štruktúry Apache Antu do štruktúry Apache Mavenu.

Tu sa ako komplikácia vyskytla korupcia v jednom z konfiguračných súborov projektu. Po opravení Gephi Maven Plugin úspešne pretransformoval projektovú štruktúru na Maven.

Tento projekt sa potom pridával do repozitára gephi-plugins, určenom na vývoj pluginov (modulov) pre platformu Gephi.

Pre transformovaný plugin bolo potrebné správne nakonfigurovať objektový model projektu (project object model), pretože rôzne závislosti na knižniciach z iných projektov neboli správne uvedené v konfiguračných súboroch pre Maven, alebo sa menili v rámci platformy Gephi.

Ďalšou zmenou musel kvôli rozdielom v aplikačno-programovom rozhraní prejsť priamo kód pluginu.

Zmeny v rozhraní boli dvoch druhov:

- Presuny funkcionality

  Prístupy k atribútom pre rôzne prvky (vrcholy, hrany, model grafu, ...) už nie sú samostatne - boli zlúčené priamo s týmito prvkami. Pri úpravách zväčsa bolo potrebné len vymeniť použitú metódu alebo triedu. Väčšina takýchto zmien sa dala po vyriešení jednoducho zopakovať na ostatných miestach v projekte.

- Zmeny funkcionality Pôvodne sa ako identifikátory vrcholov a hrán používali číselné hodnoty, s možnosťou priradenia aj textového označenia medzi atribúty. V novších verziách však bola podpora pre číselné identifikátory odstránená. Rozhrania ktoré tieto metódy poskytovali boli odstránené. Ako identifikátor vracala až jednoduchšia trieda Element identifikátor typu Object.

  Pretože sa v dokumentácií nič ďalšie nevysvetľovalo prepísal som najprv všetku prácu s identifikátormi tak, že sa vždy pretypovali int, ktorý býval v pôvodnej verzií návratovou hodnotou pre identifikátory. Toto riešenie bolo jednoduchšie, pretože nemuselo zasahovať do ďalších častí kódu - tie by mali pracovať ako dovtedy. Všetky typy objektov sa ale na hodnoty int jednoznačne pretypovať nedali.

  Druhým podobným riešením bolo obdobne pretypovať všetky identifikátory na reťazcový typ String - do toho sa dala jednoducho vložit textová reprezentácia ľubolnej hodnoty, ktorú metóda ako Object vráti. Logika ostatného kódu ale musela byť upravená, pretože nemohla pracovať s číselnými typmi, ako doteraz.

  Všetko narábanie s identifikátormi vrcholov a hrán muselo byť upravené na prácu s reťazcami. Navyše, všetky porovnávania, alebo výpočty, ktoré sa na čiselné identifikátory spoliehali bolo potrebné zmeniť. Pri úpravách bolo potrebné v projekte vyhľadávať akúkoľvek prácu s metódami typu $getId()$ a manuálne nahradiť všetku logiku tak, aby fungovala rovnako aj pri práci s reťazcami.

  V priebehu implementácie (koncom apríla 2022) platforma Gephi prešla z verzie 0.9.2 až na 0.9.5 a bola aktualizovaná aj dokumentácia. V tej už bolo aj jasne vysvetlené, že podpora pre číselné identifikátory bola odstránená a už sú povolené jedine typy String. Táto aktualizácia zmiernila obšírnosť pretypúvania každého identifikátoru a potvrdila, že s identifikátormi bude nutné narábať reťazcovo.

3.5 Implementácia algoritmu MarkovAlgo

Algoritmus nebol pomenovaný v pôvodnej práci [11] a v tejto implementácii je použitý pracovný názov MarkovAlgo.

Na prácu s maticami sa využíva knižnica apache.commons a jej triedy MatrixUtils, RealMatrix a jej implementácie.

Algoritmus pracuje na dátach načítaných v aktuálnom projekte prostredia Gephi. Pre funkčnosť potrebuje, aby bol graf orientovaný a neobsahoval absorbujúce stavy. Jeho argumentom je počet hľadaných zhlukov r.

Pri načítaní grafu do matice Q algoritmus preskakuje izolované vrcholy, ktoré rovno označuje ako neprislúchajúce do zhlukov. Tieto vrcholy by pri pokračovaní nemohli ovplyvniť prechody (ktoré cez ne nesmerujú) a jedine by pridávali pamäťové a výpočtové nároky. Je to užitočná optimalizácia, pretože sa tým zmenšia rozmery matice Q (ku ktorej sa neskôr má rátať inverzná matica). Okrem toho mohli aj priamo brániť výpočtu inverznej matice - tá sa nedá vypočítať pre maticu, ktorá obsahuje nulový riadok alebo stĺpec.

Podobne sa ukladajú bokom aj vrcholy, ktoré majú rovnaké prechody a susedov ako niektorý z už načítaných vrcholov. Tieto by mali dosiahnuť rovnaké výsledné zadelenie ale tiež by mohli brániť výpočtu inverznej matice - pretože rovnaké riadky matice sa dajú odčítať, čo spôsobí, že matica má nulový riadok a teda je singulárna. Týmto rovnakým vrcholom sa na konci pridelí rovnaké zhlukovanie, aké sa vypočíta pre ten z nich, ktorý je v matici ponechaný.

Matica Q sa normalizuje, aby predstavovala pravdepodobnosti prechodov a nie iba ich váhy.

Ďalej sa vypočíta matica N podľa (3.2), ak je matica singulárna, nahradí sa výpočet sumov mocnín Q po hodnotu N_POW_APPROX. Matica N sa potom normalizuje.

Inicializácia matice R prebehne použitím metódy $kMeans$ nad bodmi v priestore reprezentovanými maticou Q. K-Means++ náhodne vyberá r ťažísk zhlukov tak, aby boli od seba čo najvzdialenejšie. Šanca na výber každého zo zhlukov je priamo úmerná jeho kvadratickej vzdialenosti od iného ťažiska. Vzdialenosť dvoch vrcholov je tu daná ako euklidovská vzdialenosť ich koncov.

Po tomto výbere je K-Means++ zhodné s K-Means a opakuje pridelenie každého vrcholu do najbližšieho ťažiska. Potom sa poloha ťažísk prepočíta a cyklus sa opakuje. Prerozdelenie a posúvanie sa preruší, ak sa už zhlukovanie vrcholov nemení.

Z výsledného zhlukovania K-Means++ sa zostrojí matica R ktorá obsahuje v každom riadku 1 pre zhluk, kam bol vrchol pridelený a 0 pre ostatné prechody.

Po prvotnom nastavení matice R pomocou K-Means sa algoritmus presunie ku krokom maximalizácie strednej hodnoty (Expectation Maximization steps). To preruší ak sa prestane zlepšovať hodnota evaluačnej funkcie (3.4).

Výsledné zhlukovanie je dané maticou B (3.3) a určuje sa ako príslušnosť ku najpravdepodobnejšiemu zhluku (absorbujúcemu stavu).

3.6 Použité dáta

V práci sa na vyhodnotenie kvality zhlukovacieho algoritmu používajú dátove množiny, ktoré boli na testovanie použité aj v pôvodnej implementačnej práci [11]. Sú to:

- American football games [7]

  Graf obsahuje futbalové kluby a ich vzájomné zápasy pre americké univerzity počas jesennej sezóny 2000.

- Zachary's Karate Club [22]

  Typická dátová množina so známym správnym zhlukovaním. Graf reprezentuje priateľstvá v rámci klubu. Počas zaznamenávania dát došlo k rozdeleniu na 2 skupiny.

- Books about US politics [10]

  Graf obsahuje knihy o politike Spojených Štátov predávané na Amazon.com. Prepojené sú tie knihy, ktoré sa často kupovali zároveň.

3.7 Hodnotenie a interpretácia výsledkov

Algoritmus je vyhodnocovaný použitím nástrojov v plugine Community Detection. Tie pokrývajú internú aj externú evaluáciu (priame hodnotenie, alebo porovnávanie s iným výsledkom). Algoritmus sa bude testovať na dátových množinách použitých aj priamo v práci A.Kumara 3.6.

Kapitola 4

Výsledky práce

Výsledkom práce je aktualizovaný plugin Community Detection pre platformu Gephi. Tento plugin sa tak opäť stal prakticky použiteľným pre aktuálnu verziu Gephi. Môže slúžiť na hľadanie zhlukovaní pomocou implementovaných algoritmov, vyhodnocovanie výsledných zhlukovaní alebo iné podporné funkcie, ktoré plugin poskytuje (napr. generovanie LFR modelov). Modul Community Detection môže tiež slúžiť ako základ pre pridávanie nových algoritmov - podobne ako v tejto práci.

Okrem toho je v práci implementovaný a otestovaný algoritmus na hľadanie komunít v grafe, navrhnutý A. Kumarom [11].

Hodnotenie výstupov algoritmu: Pri všetkých použitých dátových množinách bolo správne rozdelenie vopred známe. Preto bolo jednoduché nastaviť tento hľadaný počet zhlukov algoritmu MarkovAlgo.

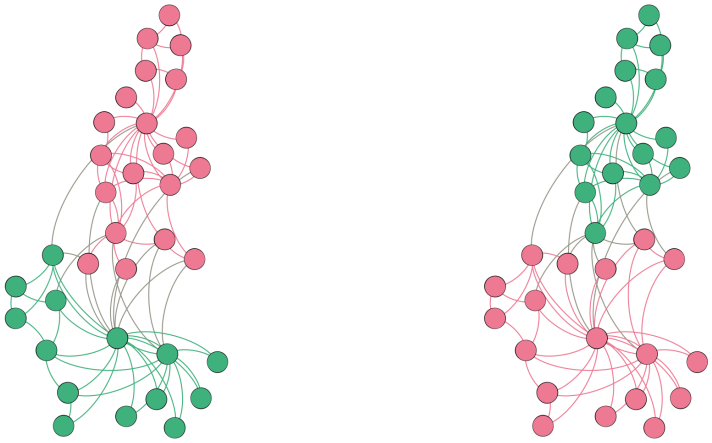
Zápasy v americkom futbale

Graf obsahuje 115 vrcholov družstiev a 613 hrán spoločných zápasov. Správne rozdelenie priraďuje každé družstvo do jeho konferencie. Tých je 11 a jedna skupina je nezávislá.

MarkovAlgo tu pri hodnotení zaostáva za zvyšnými algoritmami. Najviac nesprávne pridelených bolo nezávislých družstiev, čo zodpovedá správaniu uvádzanom A. Kumarom pri evaluácii. Ak pod pojmom 92% presnosť autor myslel hodnotu Rand indexu,

	American football	Zachary's Club	Books ab. US politics
SLPA community	0.960	1.263	1.137
SCAN community	0.910	NaN	0.922
MarkovAlgo Cluster	1.206	1.341	1.242
Link community	0.922	1.098	0.767

Tabuľka 4.1: Hodnoty Davies-Bouldin indexu



Obr. 4.1: Grafy nájdeného rozdelenia (vľavo) a správneho rozdelenia (ground-truth) na Zachary's Karate Club pre MarkovAlgo

tak sa meranie podobá s hodnotou 90.5%. Pri metrikách F-measure, NMI aj DBI ale algoritmus dosahuje citelnejšie zhoršenie oproti ostatnám vyhodnocovaným algoritmom.

Knihy o politike

Tento graf obsahuje 105 vrcholov kníh a 441 hrán spoločných nákupov na Amazon.com. Správne rozdelenie priraďuje knihy do skupín konzervatívne, liberálne a neutrálne.

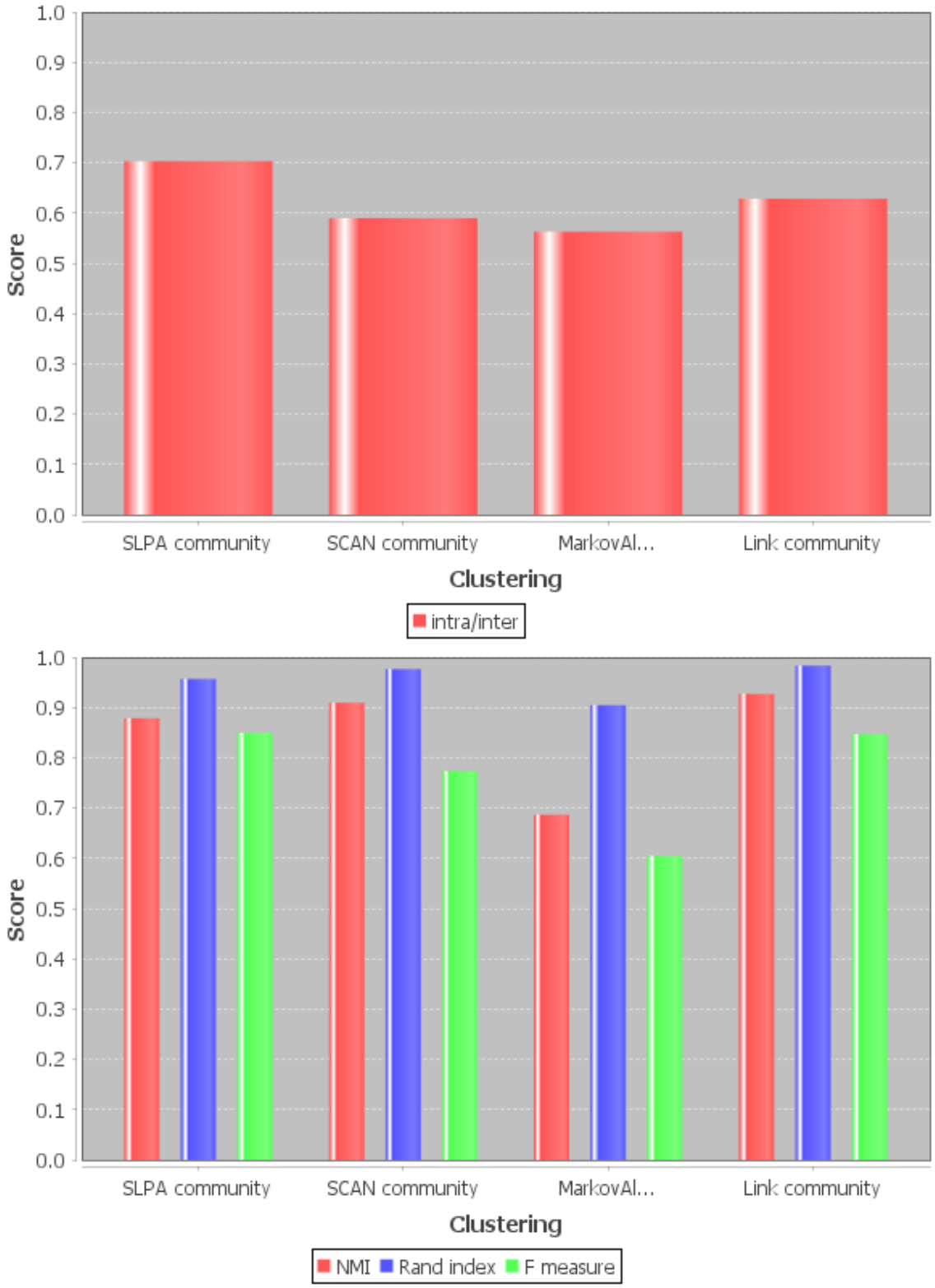
Tu dosahuje MarkovAlgo spolu s SLPA a SCAN algoritmami dobré výsledky zatiaľčo LINK nachádza veľké množstvo malých zhlukov, čo nezodpovedá trom určeným skupinám. Všetky algoritmy majú nižšiu úspešnosť podľa NMI - teda nimi nájdené zhluky majú nízku vzájomnú informáciu. Výsledok Rand indexu 82.6% sa opäť podobá nájdeným 80% v práci A.Kumara.

Zacharyho karate klub

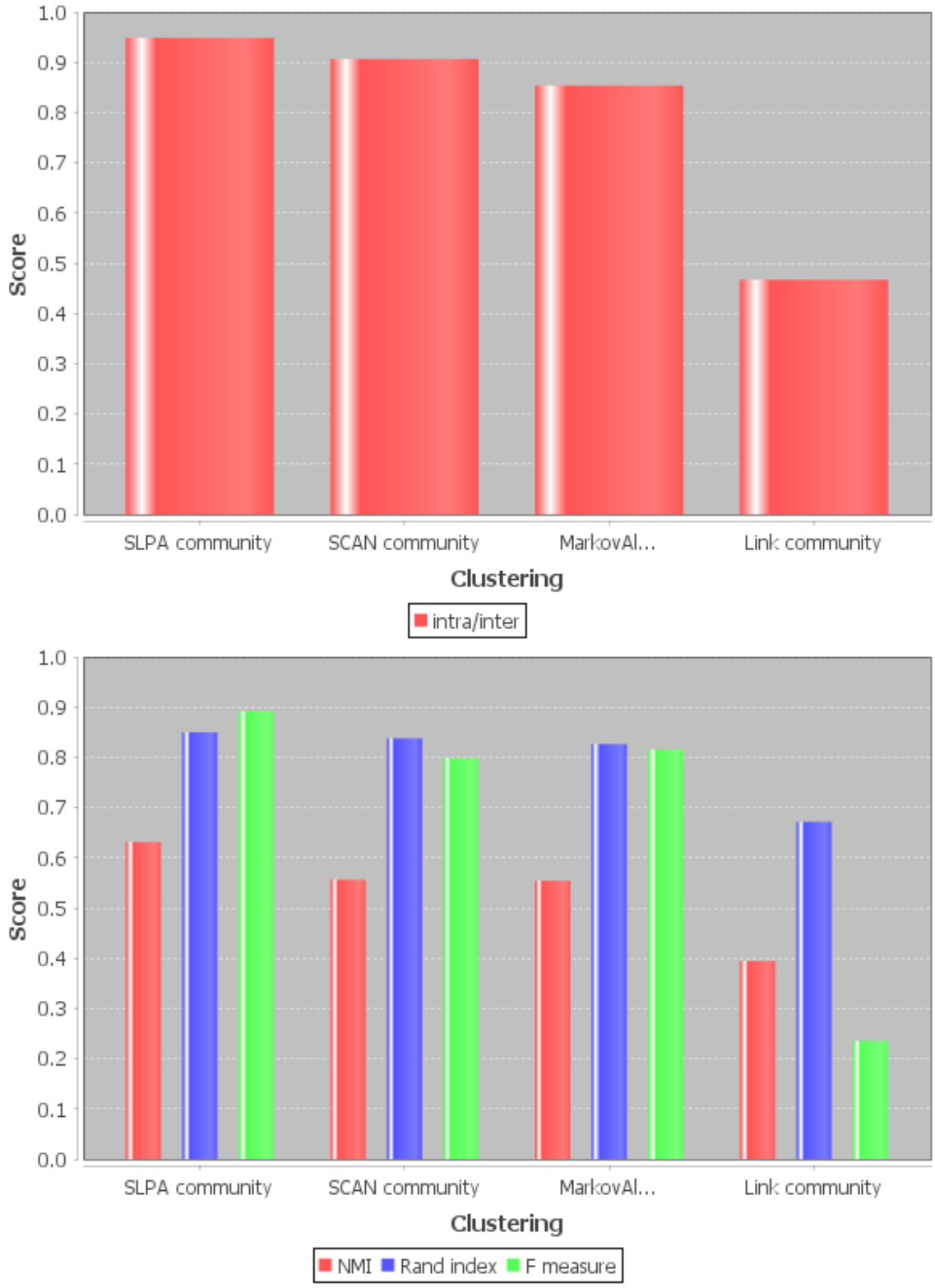
Graf sa skladá z 34 vrcholov - členov klubu a 78 hrán - siete priateľstiev členov. Známe rozdelenie tvorí dve skupiny okolo pohádaných učiteľov.

Algoritmus SLPA tu dosahuje najlepšie rozdelenie s 1 chybným vrcholom na rozhraní skupín. LINK nachádza väčšie množstvo drobnejších skupín. SCAN síce dosahuje mieru intra/inter rovnú 1.0, je to ale preto, že celý graf označil za zhluk.

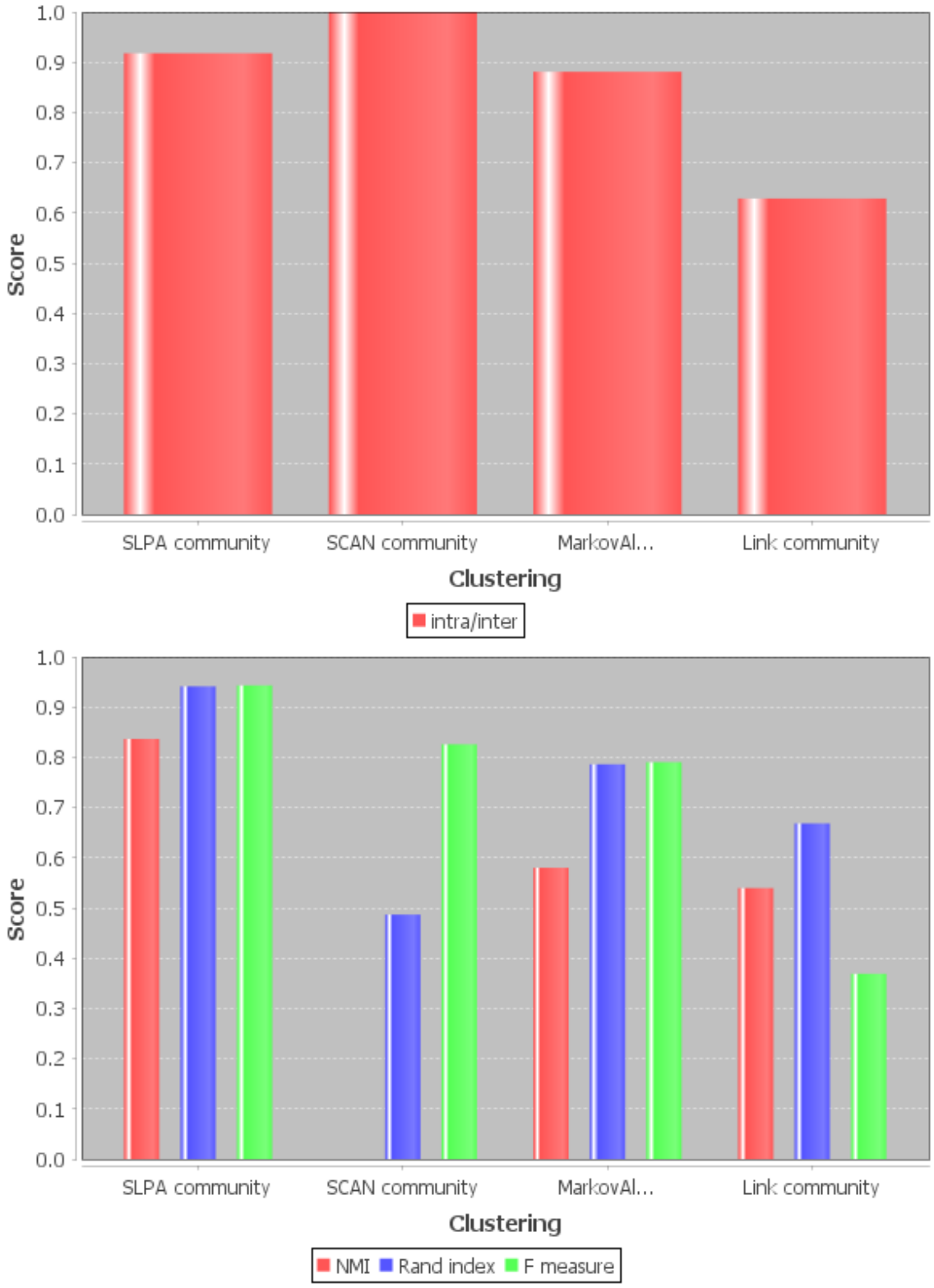
MarkovAlgo dosiahol pomerne dobré rozdelenie s chybou pri 4 vrcholoch na rozhraní skupín 4. Pri viacerých behoch bolo dosiahnuté aj 100% riešenie, spomenuté u A.Kumara.



Obr. 4.2: Grafy evaluácie algoritmov na dátovej množine American football games



Obr. 4.3: Grafy evaluácie algoritmov na dátovej množine Books about US politics



Obr. 4.4: Grafy evaluácie algoritmov na dátovej množine Zachary's Karate Club

4.1 Hodnotenie algoritmu MarkovAlgo

Algoritmus dosahuje dobré zhlukovania pre známy počet skupín. Známy počet skupín je však iba pri dátach, ktoré už sú ohodnotené. Pri nevyriešených dátach by bolo nutné počet zhlukov nejakou heuristikou odhadnúť, alebo skúšať viacero možností.

Algoritmus je citlivý na cykly alebo absorbujúce stavy v grafe. Vtedy vytvára priveľké zhluky, ktoré môžu zahŕňať aj celý graf. Podobne je problémom aj ak graf nie je orientovaný - ak sa vtedy hrana interpretuje ako obojsmerná, znamená to cyklus pri každej hrane. To skončí opäť zahrnutím celého grafu do zhluku a nenájdením nejakého užitočnejšieho zhlukovania.

Algoritmus je založený na princípe absorbujúcich markovovských reťazcov. Preto je rozumná požiadavka na orientovaný graf bez absorbujúcich stavov. Znamená to ale, že algoritmus sa nebude dať použiť pre univerzálnejšie problémy.

Algoritmus potrebuje na začiatku vyrátať inverznú maticu ku matici grafu. Táto operácia prebehne iba raz, ale pre veľké grafy má vysokú časovú náročnosť.

Záver

Práca chcela dosiahnuť implementáciu a otestovanie algoritmu na hľadanie komunít s využitím absorbujúcich markovovských reťazcov. Tento algoritmus bol navrhnutý v práci A.Kumara [11] a mal byť implementovaný ako súčasť pluginu Community Detection, vytvorenom v diplomovej práci M.Šmitalu [23].

V práci sa úspešne aktualizoval plugin Community Detection pre najnovšiu verziu platformy Gephi. Táto migrácia vyžadovala oboznámiť sa s používaním a konfiguráciou nástroja Maven. Do aktualizovaného pluginu bol implementovaný spomenutý zhlukovací algoritmus. Ten bol porovnávaný s ostatnými zhlukovacími algoritmami pluginu Community Detection. Testovanie prebiehalo na dátových množinách použitých v pôvodnej práci, pre umožnenie priameho porovnania výsledkov.

Algoritmus bol v porovnaní s ostatnými podobne presný. Jeho nevýhodami bola potreba určenia počtu hľadaných komunít. Ale aj jeho pomalšie vykonávanie pri väčších grafoch. Okrem toho je jeho využitie limitované na orientované grafy bez absorbujúcich stavov. Na iné univerzálnejšie problémy sa teda nedá použiť.

Ďalší smer vývoja

- Algoritmus by sa mohol zbaviť zákazu absorbujúcich stavov tak, že by ich používal spolu s tými novo-inicializovanými, ktoré sa po načítaní grafu ku nemu vytvárajú.
- Počas behu sú prakticky všetky operácie vykonávané na maticiach. To dáva priestor na účinnú paralelizáciu, ktorou by sa mohli zmierniť dôsledky $O(N^3)$ inverzie grafovej matice.
- Algoritmus sa môže vo viacerých behoch dosť líšiť vo výsledkoch, pretože sa pri inicializácii pomocou K-Means pridáva k riešeniu prvok náhody a potom sa iba hľadá lokálne maximum. Takýchto lokálnych maxím môže byť v priestore riešení veľké množstvo a môže sa opakovať uviaznutie vo výrazne horších výsledkoch než je globálne maximum. Proti uviaznutiu v nízkych lokálnych maximách by sa dal využiť prístup simulovaného žíhania a zlepšiť tak šancu na nájdenie lepšieho lokálneho maxima.

Literatúra

[1] S. Agarwal. Data mining: Data mining concepts and techniques. In *2013 International Conference on Machine Intelligence and Research Advancement*, pages 203–207, 2013.

[2] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[3] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, pages 226–231, 1996.

[5] Wei Fan and Albert Bifet. Mining big data: Current status, and forecast to the future. *SIGKDD Explor. Newsl.*, 14(2):1–5, 2013.

[6] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[7] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[8] Crina Grosan and Ajith Abraham. *Intelligent systems*, volume 17. Springer, 2011.

[9] A. K. Jain, A. Topchy, M. H. C. Law, and J. M. Buhmann. Landscape of clustering algorithms. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pages 260–263 Vol.1, 2004.

[10] Valdis Krebs. Books about us politics. 2004.

[11] Ankit Kumar. Detecting communities using absorbing markov chains. 2013.

[12] O. Kurasova, V. Marcinkevicius, V. Medvedev, A. Rapecka, and P. Stefanovic. Strategies for big data clustering. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pages 740–747, 2014.

[13] P. Louridas and C. Ebert. Machine learning. *IEEE Software*, 33(5):110–115, 2016.

[14] K. K. Pandey and D. shukla. Challenges of big data to big data mining with their processing framework. In *2018 8th International Conference on Communication Systems and Network Technologies (CSNT)*, pages 89–94, 2018.

[15] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005.

[16] V. Rawte and G. Anuradha. Fraud detection in health insurance using data mining techniques. In *2015 International Conference on Communication, Information Computing Technology (ICCICT)*, pages 1–5, 2015.

[17] Jean-Michel Réveillac. 2 - elements of graph theory. In *Optimization Tools for Logistics*, pages 11–29. Elsevier, 2015.

[18] Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, and Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017.

[19] S. van Dongen. A new cluster algorithm for graphs. *Nucleic Acids Research*, 2000.

[20] J. Xie, B. K. Szymanski, and X. Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 344–349, 2011.

[21] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833, 2007.

[22] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

[23] Matej Šmitala. Vyhľadávanie klastrov v grafoch a vizualizácia získaných dát. Master's thesis, Univerzita Komenského v Bratislave FMFI FMFI.KAI, 2013.