Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Detekce komunit v sociálních sítích

Bakalářská práce

Autor: Jaromír Homolka
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Jiří Haviger, Ph.D.

Hradec Králové duben 2015

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 22.4.2015 Jaromír Homolka

Poděkování:

Děkuji vedoucímu bakalářské práce Mgr. Jiřímu Havigerovi, Ph.D. za metodické vedení práce, za pomoc a cenné rady při zpracování této práce.

Anotace

Bakalářská práce je zaměřena na způsoby detekování komunit v sociálních sítích. Tato detekce je velmi důležitá pro další rozbor vlastností sítě. Existuje mnoho algoritmů řešící strukturu sítě, ale mnoho z nich bohužel nestačí na dnešní velikost sociálních sítí. Práce je rozdělena do tří hlavních částí. V první části jsou popsány základní pojmy, kde hlavní doménou této části je sama modularita, která se využívá jako měřítko síly rozdělení komunit. V druhé části bylo vybráno pět algoritmů z balíčku iGraph, které byly následně rozebrány a názorně vysvětleny na jednoduchém grafu. V třetí (praktické) části byly algoritmy aplikovány na různě veliké sítě, kde největší síť měla téměř 4 miliony vrcholů. Pro jednu síť je detailněji popsána výsledná struktura sítě.

Annotation

Title: Community detection in social networks

The bachelor thesis is focused on ways to detect communities in social networks. This detection is very important for further analysis of the network. There are many algorithms solving the structure of the network, but many of them are not sufficient on current size of social networks. The work is divided into three main parts. First chapter describes the basic concepts and the modularity, which is used as a measure of the strength distribution of communities. In the second section were selected five algorithms from the iGraph package, which have been described and clearly explained on a simple graph. In the third (practical) part algorithms have been applied on different sized networks, where the largest network had nearly 4 million vertices. The structure of the resulting network is more described for one network.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Detekce struktury sítě se v posledních letech těší velké pozornosti. Jedna z příčin je určitě velký vzestup sociálních sítí a následné zkoumání specifických skupin těchto sítí pro porozumění chování a vlastností sítě. Hlavním tématem bakalářské práce bude podat srozumitelné informace o jednotlivých pojmech, které se v této problematice využívají, o algoritmech a následné detekci komunit v sociálních sítích.

Cílem práce bude aplikování jednotlivých algoritmů pro detekci komunit v sítích a následné porovnání výsledků těchto algoritmů.

Práce je rozdělena do tří větších částí. První část bakalářské práce je spíše teoretického rázu. Zde bude vysvětlen pojem komunita, která se v této práci bude objevovat mnohokrát. Dále bude popsáno několik pojmů a teorií, které je třeba znát. Blíže bude popsána ukázková síť Karate Club Zachary, jež se stala základem určování správnosti algoritmů, ale hlavní doménou této části bude modularita. Modularita bude podrobněji popsána, bude ukázáno její vyjádření v rovnici či maticové vyjádření, které je více využíváno, včetně negativních a omezujících vlastností této modularity.

V další části bakalářské práce budou rozebrány jednotlivé algoritmy pro detekci komunit v sítích. Všechny algoritmy jsou implementovány v jazyce R využívající knihovnu iGraph, tudíž není nutné složité vysvětlování syntaxe. Prvním algoritmem bude Edge-betweenness. Pracuje na základě hierarchické dekompozice a to postupným odebíráním hran. Dalším algoritmem bude Fast-greedy. Používá hierarchický přístup postupující ze zdola nahoru, kde každý vrchol začíná samostatně. Třetím představeným bude Multi-level algoritmus. Tento algoritmus je velmi zajímavý díky své jednoduché implementaci a hlavní výhodou je jeho rychlost. Předposledním algoritmem je Leading-eigenvector. Rozděluje síť na dvě části na základě vedoucího vlastního vektoru (Leading-eigenvector). Posledním algoritmem, který bude představen, je Label-propagation. Funguje na jednoduchém přiřazování štítků k vrcholům a následném spojování sousedů. Bohužel má jednu velkou nevýhodu, která bude vysvětlena níže.

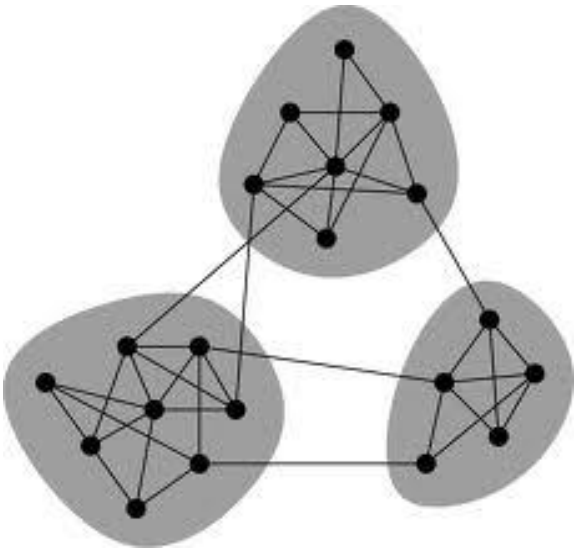
Ve třetí části budou předcházející algoritmy aplikovány na různě veliké sítě. Nejmenší představená síť má 34 vrcholů (Zachary), která je pouze cvičná, další v pořadí je pouze část Facebooku čítající čtyři tisíce vrcholů a největší LiveJournal má téměř čtyři miliony vrcholů. Následně budou blíže popsány komunity na vybrané síti.

V závěru této práce budou shrnuty představené informace. Poukážeme na klady a zápory použití modularity v detekci komunit. Budou zde také popsány výhody a nevýhody jednotlivých algoritmů, a zda mají své opodstatnění i v blízké budoucnosti.

2 Teoretická východiska

2.1 Reprezentace sítě a komunita

Časté reprezentování sítě a v ní několik komunit je pomocí jednoduchých grafů spojených hranami. Jsou hledány pouze důležitá spojení mezi vrcholy. Ve většině případů lze vidět, že v jednotlivých komunitách je velké množství spojení, ale mezi komunitami je jen málo spojení (viz obr. č. 1).



Obr. č. 1 - Ukázka spojení v sítích, zdroj: [1]

V celé práci bude hojně využíván termín komunity. Co to vlastně znamená komunita a jaká je definice pojmu komunity jako takové? Existuje mnoho definic pro komunitu. Mnoho autorů se pokoušelo tento termín vysvětlit, ale v kontextu této bakalářské práce je asi nejpřesnější: *"Definition of a community as an indivisible subgraph."* - M. E. J. Newman [1]. (Definice komunity jako nedělitelný podgraf). V bakalářské práci zazní mnoho termínů o nedělitelnosti komunit.

2.2 Null model

Null (nulový) model se používá v matematice ve studiu statistických vlastností grafu. Null graf je graf, který odpovídá jednomu specifickému grafu, jež je považován za náhodný. V této práci bude důležitý Null model, navržený Newmanem a Girvanem [2], skládající se z náhodné verze původního grafu, kde hrany jsou propojeny náhodně, na základě jednoho omezení, že očekávaný stupeň vrcholu odpovídá stupni vrcholu v původním grafu. Null model je základní myšlenkou definice modularity.

Bohužel v poslední době se používá trochu jiný koncept a to od Remco van der Hofstada [3]. Tento koncept je nahodilá konfigurace hran v konkrétní síti.

Mějme síť s n uzly a každý uzel v má stupeň K_v. Každá hrana je rozdělena na polovinu, tyto poloviny se nazývají stub ("pahýly") a každou polovinu připojíme k jiné polovině. Stupně vrcholů tedy zůstávají stejné, ale struktura sítě je kompletně náhodná. Z jednoduchého výpočtu je $2m$ pahýlů. Tento poznatek bude velmi důležitý v další kapitole.

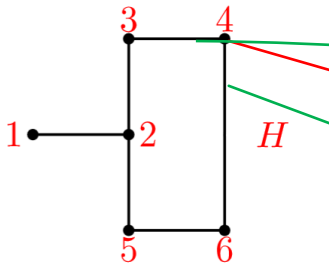
2.3 Laplacianova matice

Laplacianova matice grafu se používá pro maticovou reprezentaci grafu. Někdy je nazývána také Kirchhoffova matice nebo diskrétní Laplacianův graf [11]. Díky Laplacianově matici se dá najít mnoho dalších vlastností grafu. Tato matice úzce souvisí se spektrálním rozdělením grafu, které na základě spektrální teorie grafu bude popsáno dále.

Pro vysvětlení definice matice si vytvoříme graf G s N vrcholy jeho matice $L := (l_{i,j})_{n*n}$ bude definována jako $L = D - A$. Kdy D je diagonální matice stupňů vrcholů grafu a A je matice sousednosti grafu. Z následujícího vzorce je patrno, že $deg(v_i)$ je stupeň vrcholu i [4].

$$l_{i,j} := \begin{cases} \deg(v_i) & i = j \\ -1 & i \neq j \; a \; v_i \; je \; sousedem \; v_j \\ 0 & jinak \end{cases} \quad \textit{Rovnice č. 1}$$

Definice Laplacianovy matice je složitá, proto bude vysvětlena na následujícím příkladu. (Viz obr. č. 2).



	1	2	3	4	5	6
1	1	-1	0	0	0	0
2	-1	3	-1	0	-1	0
3	0	-1	2	-1	0	0
4	0	0	-1	2	0	-1
5	0	-1	0	0	2	-1
6	0	0	0	-1	-1	2

Obr. č. 2 - Názorná ukázka - na diagonále jsou stupně vrcholů (červená), sousedství je označeno - 1 (zelená) ze vztahu L = D - A, zdroj: autor

2.4 Spektrální teorie grafu

V matematice spektrální teorie grafu je studium vlastností grafu ke vztahu k charakteristickému polynomu pomocí vlastních vektorů (v anglicky mluvících zemích používaný termín „eigenvectors“) a vlastních hodnot („eigenvalues“) matice, které jsou přirozeně spojeny s těmito grafy. Typickým příkladem je matice sousednosti a již zmíněná matice Laplaciana.

V neorientovaných grafech je matice sousednosti symetrická, a proto má reálné vlastní hodnoty (množina nazývaná spektrum grafu) a úplnou množinu ortonormálních vlastních vektorů [7]. Význam slova ortonormalita znamená, že vektory v prostoru s definovaným skalárním součinem jsou ortogonální (navzájem kolmé) a mají jednotkovou délku, tedy platí:

$\langle v \,|w\, \rangle = 0$ a zároveň $\|v\| = \|w\| = 1$ *Rovnice č. 2*

V orientovaných grafech je matice sousednosti závislá na pojmenování vektoru (1, 2, 3…), pak je spektrum tohoto grafu neměnné.

Spektrální teorie grafu se zabývá parametry, které jsou definovány pomocí násobku vlastních hodnot matice asociované do grafu. Příkladem může být Colin de Verdiére číslo. Toto číslo je neměnné a je to parametr existující v každém grafu a znázorňuje maximální multiplicitu druhé vlastní hodnoty. Pro zajímavost tento parametr byl zaveden roku 1990 Yves Colin de Verdièrem[9].

Spektrální teorie grafů poskytuje mnoho možností pro vytvoření užitečných algoritmů, které vylepšují modularitu sítě.

2.5 Modularita

Modularita je hodnota, která se osvědčila v řadě vědeckých oblastí. Tento návrh se zabývá rozsáhlými komplexními systémy např. sociální sítě, počítačové sítě, neuronové sítě, umělá inteligence nebo také průmyslové sítě. Mnoho těchto sítí lze reprezentovat pomocí grafu (viz výše). Tímto způsobem lze odhalit neočekávané změny ve struktuře sítě. Většina sítí má určitou strukturu komunit, která má velký význam pro porozumění dynamiky vývoje sítě. Logicky vyplývá, že komunity hustě propojené budou mít rychlejší přenos informací než komunity propojené málo.

Komunita je definována jako skupina hustě propojených uzlů mezi sebou, které jsou řídce spojené se zbytkem. Jednotlivé komunity v náhodných sítích mohou mít zcela odlišné vlastnosti, mezi které patří stupeň uzlu, koeficient clusteru (shluku) a mnoho dalších. Modularita je jedním z měřítek, které při maximalizaci vede k detekci komunit v dané síti.

Autoři Carliss Young Baldwin a Kim B. Clark dospěli k několika zajímavým idejím o modularitě. Prosazovali, že existují dvě hlavní myšlenky v obecném pojetí modularity. První myšlenka je o vzájemné závislosti a nezávislosti v rámci modulů (komunit) [5]. Druhá myšlenka je od McClellanda a Rumelharta: "A module is a unit whose structural elements are powerfully connected among themselves and relatively weakly connected to elements in other units. Clearly there are degrees of connection, thus there are gradation of modularity" [6]. Jinými slovy - moduly jsou jednotky ve větším systému, jsou konstrukčně nezávislé na sobě, ale pracují společně. Systém jako celek musí proto poskytnout rámec či architekturu, umožňující nezávislost struktury, ale také integraci funkcí.

Modularita byla vytvořena jako měřítko síly rozdělení komunit. Sítě s vysokou modularitou, např. biologické sítě (mozek), mají tedy hustou síť spojení jednotlivých uzlů, ale řídké spojení mezi ostatními komunitami. Modularita je základním kamenem pro mnoho optimalizačních metod v hledání komunit v sítích.

Nicméně i modularita má své úskalí a to v podobě limitu rozlišení, projevující se neschopností vyhledat malé komunity. Tento problém bude přesněji popsán na konci této kapitoly.

Dalším velkým problémem je, že v teoriích grafů je většině případů známa velikost celé sítě, ale v reálných problémech není předem známa velikost sítě. Pokud síť bude obrovská, skoro neomezená, tak budou všechny uzly v jedné komunitě a žádný v druhé, tím pádem nebude existovat žádná hrana mezi komunitami. V jistém smyslu je tento výsledek optimální, ale tato situace nám nic neříká o struktuře sítě. Tento jev můžeme uměle zakázat. Potom rozdělení, kde jeden uzel patří do jedné skupiny a zbytek do druhé skupiny bez mezikomunitních hran, bude také zakázán, i když by tento výsledek mohl být optimální. Problémem je, že počítání hran není správnou cestou zjištění struktury komunit. Dobré rozdělení není pouze to, ve kterém je málo mezikomunitních hran, ale nýbrž takové rozdělení, kde počet hran mezi komunitami je menší, než očekávané hrany v Null modelu (viz Null model v předchozí kapitole). Když je počet reálných hran podstatně nižší nebo vyšší, než by se dalo očekávat, snadno se dojde k závěru, že se děje něco zajímavého v síti.

Modularita je tedy podílem hran (spojení), které patří do dané skupiny mínus očekávané hrany v rámci náhodně rozděleních hran. Modularita nabývá hodnot od -1/2 do 1. Pozitivní modularita znamená, že počet hran v komunitách převyšuje počet hran v náhodném rozdělení. A odráží koncentraci hran uvnitř komunit se srovnáním s náhodným rozdělením hran v Null modelu.

2.5.1 Optimalizace modularity

Existuje mnoho různých výpočtů modularity. Vysvětlení výpočtů bude ukázáno na nejběžnějším provedení, zabývající se nahodilostí hran tak, že se zachová stupeň vrcholu. Graf s *n* uzly a *m* hranami bude jednotlivé uzly rozdělovat pouze do dvou skupin. Patří-li uzel *v* do první komunity S_i, bude 1, patří-li do druhé, bude S_i -1. Pak matice sousednosti pro síť bude reprezentována pomocí A, kde A_{ij} = 0 znamená, že spojení mezi *i* a *j* neexistuje a kde A_{ij} = 1 znamená, že spojení *i* a *j* existuje. Samozřejmě, že matice sousednosti může nabývat i jiných hodnot díky takzvaným „multihranám“. Pro jednoduchost vysvětlení se zvolí nejjednodušší způsob bez existence multihran. Z toho vyplývá, že $A_{ij} = A_{ji}$. Modularita je tedy nakonec definována jako podíl hran, které spadají do jedné či druhé skupiny a odečtení hran ve skupinách v náhodném rozdělení, kde se

zachovávají stupně uzlů. Na očekávané hrany použijeme koncept Remco van der Hofstada, který byl vysvětlen v předchozí kapitole. Celkový počet očekávaných celých hran mezi *i* a *j* se bude rovnat k*i*k*j* / 2m. Pro připomenutí - 2m je počet polovin (pahýlů). Proto skutečný počet hran mezi *i* a *j* mínus očekávaný počet hran je A_{ij} - (k_i*k_j) / 2m. Zároveň platí, že 1/2 (s_is_j+1) bude 1, pokud *i* a *j* budou ve stejné komunitě. Naopak bude výsledek nula [10].

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) (s_i s_j + 1) = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j \quad \textit{Rovnice č. 3}$$

Je důležité mít stále na paměti, že tato rovnice má dobré výsledky při rozdělení pouze do dvou skupin. Hierarchické rozdělování pro maximalizaci modularity (postupné rozdělování komunit na další dvě dílčí komunity, které jsou následně rozděleny do dalších dvou menších), je jeden z možných přístupů, jak identifikovat i další menší komunity v síti.

2.5.2 Maticové vyjádření

Alternativou k rovnicovému vyjádření je formulace modularity pomocí matice. Formulace modularity pomocí matice je velmi užitečná při spektrálních optimalizačních algoritmech. Pro sítě, které jsou rozděleny pouze do dvou komunit, lze alternativně definovat S_i = +- 1, kde se následně ukáže, který uzel náleží k dané skupině. Formulace matice modularity je následující:

$$Q = \frac{1}{4m} s^t B s \quad \textit{Rovnice č. 4}$$

kde *s* je sloupcový vektor s prvky s_i a *B* se nazývá maticí modularity s prvky:

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m} \quad \textit{Rovnice č. 5}$$

Všechny řádky a sloupce matice modularity mají součet nula, což znamená, že modularita nerozdělené sítě je vždy nula. Z toho je patrné, že má vždy vlastní vektor (eigenvector) hodnoty (1, 1, 1…) s vlastní hodnotou (eigenvalue) rovnající se nule. Je to velmi podobné s maticí Laplaciana (viz kapitola Graf Laplaciana) a má také stejné vlastnosti.

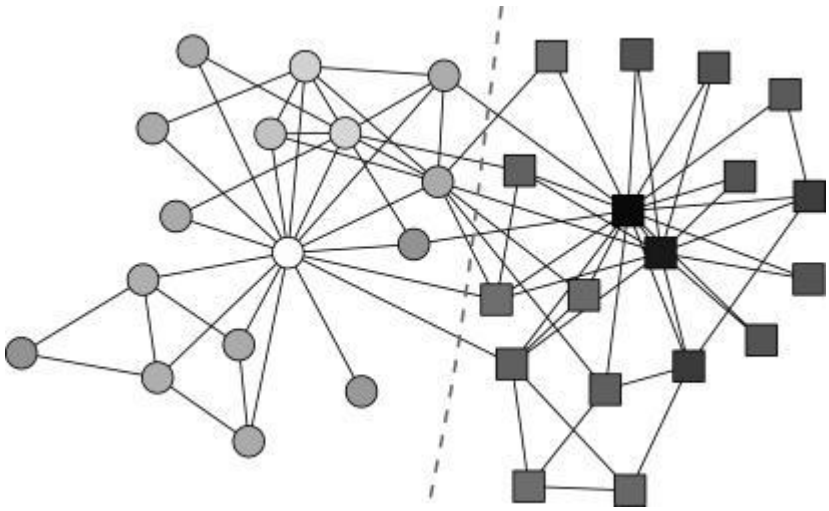
2.5.3 Omezení rozlišení modularity

Z předchozích vztahů je zřejmé, že neschopnost vyhledat malé komunity ve velkých sítích, je hlavním nedostatkem obyčejné optimalizace modularity. Proč

tento jev nastává? Modularita porovnává počet hran v komunitách s počtem hran náhodně přidaných, kde jsou stupně vrcholů totožné, ale hrany jsou náhodně připojené. Tento náhodný Null model, předpokládá, že každý uzel může být připojen k jinému uzlu sítě. Předpoklad je chybný jen tehdy, když je síť rozsáhlá a uzel se nemůže propojit s jakýmkoliv uzlem v síti. S přibývající velikostí sítě se snižuje počet hran mezi dvěma skupinami uzlů. Jestliže síť bude dostatečně velká, potom očekávaný počet hran mezi dvěma komunitami v nulovém modelu bude menší než jedna. Pokud tato situace nastane, už pouhá jedna hrana mezi komunitami může mít za následek sloučení těchto komunit v jednu. Dokonce i řídce propojené kompletní komunity, které mají hustou síť vnitřních hran, budou spojeny v jednu komunitu, což není ideální stav. Ale protože síť je dostatečně rozsáhlá, budou dvě ideální komunity následně spojeny do jedné [8]. Optimalizace modularity nedokáže vyřešit malé komunity, přestože jsou dobře definovány. S touto nedostatečností jsou spojeny všechny optimalizace modularity, které jsou založeny na Null modelu.

2.6 Karate klub Zachary

Karate klub byl založen v roce 1970 a stal se jakýmsi standardem určování kvality detekčních algoritmů. Tato síť je grafickým znázorněním přátelství mezi členy v klubu. Zajímavostí je, že krátce po pozorování se tento klub dostal do rozepře a rozpadl se na dva. Pomocí optimalizace modularity a jejího využití v algoritmech, se síť rozdělí stejně tak, jak to bylo ve skutečnosti. Můžeme si povšimnout, že vrcholy jsou zastíněné podle síly modularity. Nejtmavší a nejsvětlejší uzly odpovídají leaderům těchto dvou frakcí.



Obr. č. 3 - Karate klub Zachary - přátelství mezi členy, zdroj: [1]

3 Popis metod pro detekci komunit v jazyku R (iGraph)

3.1 Edge-betweenness algoritmus

Prvním algoritmem, který bude představen, je algoritmus Edge-etweenness. V kontextu bakalářské práce je algoritmus implementačně zajímavý, ale díky své složitosti pro výpočet není moc vhodný. Pro velké sítě se tento algoritmus stává nepoužitelný. Na druhou stranu přichází s relativně přesnými výsledky pro menší sítě. Tento algoritmus bude popsán v následující kapitole a vychází ze článku od M. Girvana a M. E. J. Newmana v roce 2001 [12].

Algoritmus funguje na principu hierarchické dekompozice. Zaměřuje se na hrany, které jsou uvnitř komunit nejméně, tedy na ty hrany, které jsou "nejvíce" mezikomunitní. Vytváření komunit zde vzniká pomocí postupného odstraňování hran z původního grafu. Na rozdíl od kompozice, kde je snaha najít "nejvíce" vnitrokomunitní hranu a postupným přidáváním nejsilnějších hran v prázdném grafu.

Jak tyto “nejvíce“ mezikomunitní hrany a vrcholy najít? Tento problém byl zkoumán již v minulosti Freemanem [13]. Hodnota mezikomunitního vrcholu *i* je definována jako počet nejkratších cest mezi dvojicemi dalších vrcholů, které procházejí přes daný vrchol *i*. Je to míra vlivu daného uzlu na tok informací mezi jinými uzly. Aby bylo možné najít “nejvíce“ mezikomunitních hran, je zapotřebí zobecnit Freemanovu definici a definovat hranu mezikomunitosti z hran jako počet nejkratších cest mezi dvojicemi vrcholů, které probíhají vedle ní. Pokud je více těchto cest, dostávají stejnou hodnotu "mezi". Když síť bude obsahovat komunity, které jsou zřídka spojené hranami mezi sebou, nejkratší cesta musí jít po těchto hranách. Takže hrany spojující tyto komunity mají vysokou hodnotu "mezi". Odstraněním hran oddělíme komunity od sebe, tím pádem lze odhalit strukturu komunit v dotyčném grafu.

Všechny nejkratší cesty jsou následně spočítány a převedeny na váhy těchto hran. Relativně dlouhé cesty od vzdálených vrcholů mají malou váhu a na druhou stranu kratší cesty mají větší váhu.

Tento algoritmus by se dal ve zkratce popsat takto:

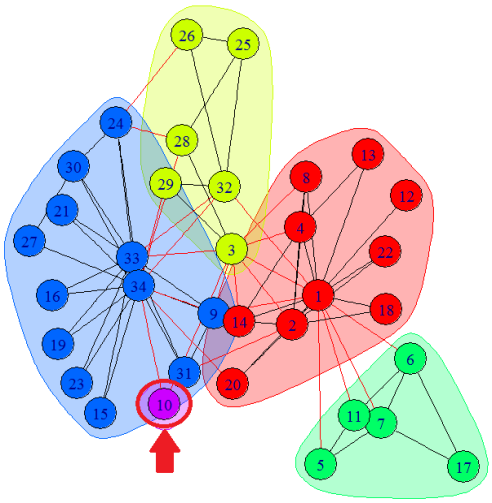
1. Vypočítá hodnotu "mezi" pro všechny hrany v síti.
2. Odstraní hranu s největší hodnotou.
3. Přepočítá se hodnota "mezi" pro zbylé hrany.
4. Opakuje se od 2., dokud budou hrany.

Náročnost výpočtu hodnot m hran v grafu s n vrcholy je v čase O (mn). Výpočet se opakuje při každém odstranění hrany. V nejhorším případě může být O (m^2n) [14]. Naštěstí se tento jev často nevyskytuje, protože ve většině případů počítáme pouze ty hrany, které jsou odstraněním postiženy. Bohužel celkový čas algoritmu je O (n^3) při řídkém grafu a je pro velké sítě kritické. Výsledkem dekompozice je dendrogram, který se vytváří ze shora dolů. Listy dendogramu jsou jednotlivé uzly.

Toto je největší nevýhoda Edge-betweenness algoritmu pro účely detekce komunit ve velkých sítích, ve kterých je hlavní prioritou rychlost výpočtu. Algoritmus při každém odebrání hrany přepočítává hodnoty "mezi" u hran, které jsou tímto odebráním změněny. Ve velkých sítích, na které se budou algoritmy aplikovat, je tento výpočet jednoduše moc pomalý.

Na druhou stranu tento algoritmus dokáže fungovat i na orientovaných grafech. Algoritmy, které by umožňovaly detekovat komunity v těchto grafech, jsou zastoupeny ve velmi malé míře. V balíčku iGraph existují pouze dva algoritmy, které to zvládají. Jedním z nich je Edge-betweenness a druhý InfoMAP. (InfoMAP nebyl použit v této bakalářské práci).

Algoritmus má menší problém při detekci komunit v klubu Zachary. Chybně určí jeden vrchol, ale jinak perfektně určí zbytek vrcholů a rozdělí je správně do komunit (viz obr. 4).

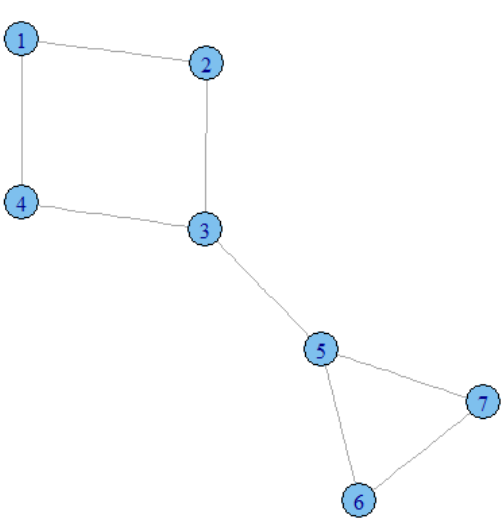


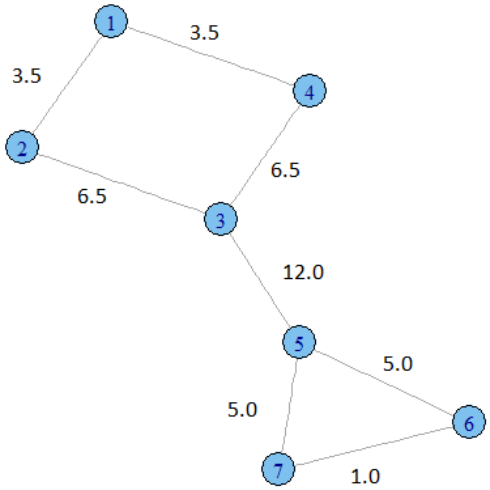
Obr. č. 4 - Chyba algoritmu Edge-betweenness. Algoritmus špatně určí vrchol 10, který by měl ležet v červené komunitě, zdroj: autor

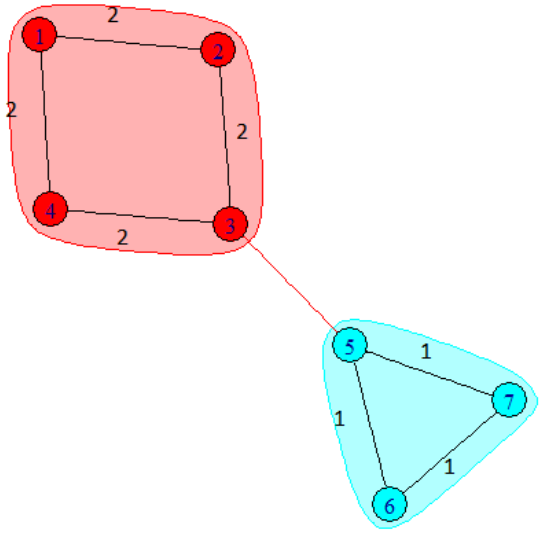
3.1.1 Ukázka

Každý algoritmus bude vysvětlen na jednoduchém grafu o 7 vrcholech spojené 8 hranami. Graf je reprezentován na obrázku č. 5a, kde je na první pohled patrné, jak by měly všechny algoritmy skončit. Výsledkem by měly být dvě komunity. První komunita by byl čtverec s vrcholy číslem 1 2 3 4 a druhá komunita trojúhelník s vrcholy s číslem 5 6 7. Komunity by měly být spojené jednou mezikomunitní hranou, která bude na následujících obrázcích reprezentována červenou barvou.

Algoritmus Edge-betweeness je založen na hierarchické dekompozici, ve které se jednotlivé hrany odebírají na základě své hodnoty "mezi". Na obrázku 5b jsou znázorněny jednotlivé hodnoty "mezi" pro danou hranu. Je patrné, že největší hodnotu "mezi" má hrana, která spojuje obě komunity. Hodnota "mezi" je v tomto případě rovna 12. Po odstranění hrany se hodnoty "mezi" u komunity čtverce změní na hodnoty 2 a u komunity trojúhelník na hodnoty 1 (Obr. č. 5c). Jednotlivé hodnoty vah byly jednoduše vypočítány pomocí příkazu *Edge.betweeness*, který vyhodnocuje jednotlivé nejkratší cesty přes tyto hrany.







Obr. č. 5a - reprezentace jednoduchého grafu o 7 vrcholech a 8 hranách,
Obr. č.5b - Graf s hodnotami "mezi" jednotlivých hran
Obr. č. 5c - Struktura grafu s hodnotami "mezi" pro každou hranu, zdroj: autor

3.2 Fast-greedy algoritmus

Dalším algoritmem pro detekci komunit je Fast-greedy (rychlý hladový algoritmus). Algoritmus vychází z klasického hladového algoritmu, liší se pouze v některých ohledech. Převážně optimalizovanějším způsobem výpočtu matice sousednosti. Vše bude vysvětleno v následujících řádcích bakalářské práce. Algoritmus bude dále popsán na základě článku od A. Clauseta, M. E. J. Newmana a C. Moorea v roce 2004 [15].

Tento algoritmus je založen na hierarchickém přístupu, ale postupuje od zdola nahoru. Algoritmus využívá hladovou optimalizaci modularity, kde se začíná se samostatnými vrcholy, kde každý vrchol je celá jedna komunita. Tyto vrcholy (komunity) se následně spojují do větších komunit na základě největšího nárůstu modularity Q. Jedna z nejjednodušších realizací zahrnuje ukládání matice sousednosti jako pole celých čísel, kde po sloučení komunit, se sloučí řádky a sloupce matice. Pro zjednodušení je třeba definovat dvě veličiny. Jedna e_{ij} je podíl hran, které spojují vrcholy v komunitě i na vrcholy v komunitě j. V této práci je velmi využívána Kroneckerova delta. Tato matematická funkce je obvykle funkce celých čísel. Kroneckerova δ se rovná 1, když se proměnné rovnají, a nula, když se nerovnají. V tomto kontextu jsou-li vrcholy spojené nebo nejsou.

$$e_{ij} = \frac{1}{2m}\sum_{vw} A_{vw}\,\delta(c_v, i)\delta(c_w, j) \qquad \textit{Rovnice č. 6}$$

Druhá je a_i, což je podíl konců hran, které jsou připojeny na vrcholy v komunitě i.

$$a_i = \frac{1}{2m}\sum_{v} k_v \delta(c_v, i) \qquad \textit{Rovnice č. 7}$$

Poté se celá rovnice modularity dá zkrátit na:

$$Q = \frac{1}{2m}\sum_{vw}\left(A_{vw} - \frac{k_v k_w}{2m}\right)\delta(c_v, c_w) = \sum_{i}(e_{ii} - a_i^2) \qquad \textit{Rovnice č. 8}$$

Tato metoda je využívána v klasickém hladovém algoritmu. Nevýhodou je, že v matici jsou ukládány i prvky s hodnotou nula značící vrcholy, které nejsou spojeny a více méně jenom drží paměť a zvyšují náročnost pro výpočet. Ve většině případů je mnoho nul v matici sousednosti. Výpočet největší změny modularity ΔQ a hledání párů pro sloučení v celé matici sousednosti se stává velmi časově náročné. Tento nešvar je eliminován právě v algoritmu Fast-greedy.

V algoritmu se zaměříme na udržení a aktualizaci matice hodnot ΔQ než na matici sousednosti s výpočtem ΔQ. Když dvě komunity nemají společnou hranu, tak není možné, aby spojení zvýšilo modularitu, proto je nadbytečné jej ukládat. Algoritmus si pouze ukládá ty dvojice komunit, které mají jednu nebo více hran mezi sebou. Využívají se pro ukládání největší ΔQ různých datových struktur (halda) [16]. Následkem jednodušší matice a správného výběru datových struktur pro uložení největší ΔQ je časová a paměťová úspora. Je zapotřebí mít tři datové struktury:

1. Matici obsahující ΔQ pro každý pár komunit s nejméně jednou hranou mezi sebou. Každý řádek matice jako binární strom (vkládání a nalézání prvku trvá *O* (log n)) a také jako max-haldu (halda, kde lze nalézt největší prvek v konstantním čase).
2. Max-haldu, která obsahuje největší prvek každého řádku předchozí matice ΔQ s popisem, o který pár komunit se jedná.
3. Obyčejné pole vektorů s prvky a_i.

Komunity začínají jako samostatné vrcholy, kde $e_{ij} = 1/2m$, jsou-li i a j spojeny, anebo kde $a_i = k_i/2m$, nejsou-li spojeny

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & \text{pokud } i, j \text{ jsou spojeny} \\ 0 & \text{jinak} \end{cases} \qquad \text{a pro každé } i\text{:} \quad a_i = \frac{k_i}{2m}$$

Rovnice č. 9

Fast-greedy algoritmus funguje ve zkratce takto:

1. Vypočítají se počáteční hodnoty ΔQ_{ij} a a_i podle předchozích rovnic a naplní se Max-halda s největším prvkem každého řádku v matici ΔQ.
2. Vybere se největší prvek ΔQ_{ij} z haldy, sloučí se tyto komunity, aktualizuje se matice ΔQ a zvětší se modularita ΔQ o danou změnu modularity ΔQ_{ij}.
3. Krok 2 se opakuje, je-li víc než jedna komunita.

Díky haldám se druhý krok rychle vypočítá, protože je potřeba upravit jen některé prvky z matice ΔQ. Komunity i a j se shlukují sloučením i s j tak, že se aktualizuje pouze j-tý řádek a j-tý sloupec a odstraní se i-tý řádek i se sloupcem.

Existují tři pravidla této změny:

1. Je-li komunita *k* připojena k oběma komunitám *i* a *j*, potom platí:
$$\Delta Q'_{jk} = \Delta Q_{ik} + \Delta Q_{jk}$$
2. Je-li komunita *k* připojena k *i*, ale ne k *j*, potom platí:
$$\Delta Q'_{jk} = \Delta Q_{ik} - 2a_j a_k$$
3. Je-li komunita *k* připojena k *j*, ale ne k *i*, potom platí:
$$\Delta Q'_{jk} = \Delta Q_{jk} - 2a_i a_k$$

Tento algoritmus je rychlejší než předchozí díky dobré optimalizaci jednotlivých datových struktur (hald). Algoritmus běží v čase *O (md log n),* kde *d* je hloubka dendrogramu. Mnoho reálných sítí jsou řídké sítě. V řídkých sítích může algoritmus dosahovat až téměř lineární dobu $O(n \log^2 n)$. Tuto skutečnost bude možné sledovat v další části práce.

V reálných situacích není nutné udržovat pro každý řádek Max-haldu. Halda nám umožňuje rychlé vyhledávání největšího prvku v ΔQ, ale na druhou stranu nás stojí velké úsilí ji udržovat. Úsilí je zbytečné, když se největší prvek v řádku nezmění a když se ostatní dva řádky sloučí, což je často vidět v praxi. Je zřejmé, že zmíněná zjednodušená implementace bude mít za následek jiné výsledky než při úplném algoritmu s haldami při každém řádku. Nicméně rozdíly výsledků obou algoritmů jsou v reálných problémech skoro nepodstatné a změny nezpůsobují významné rozdíly v modularitě.

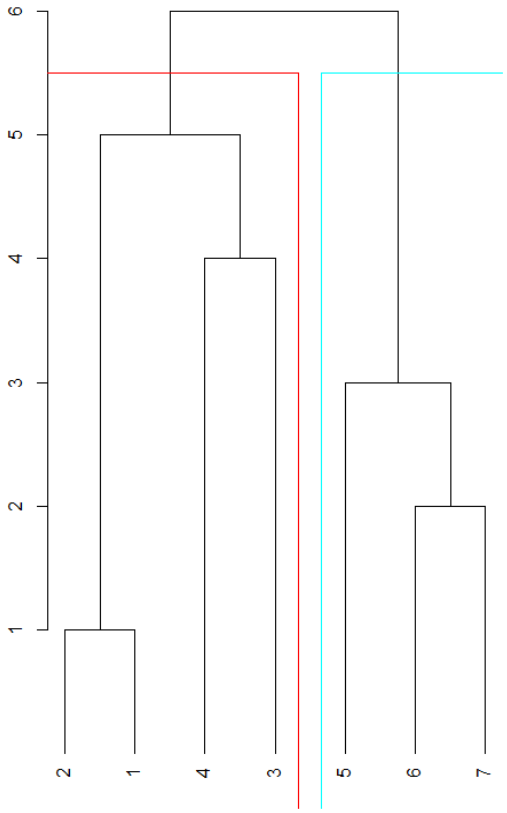
3.2.1 Ukázka

Algoritmus začíná tak, že každý vrchol leží samostatně v komunitě. V každém kroku se spojují tyto vrcholy (komunity) na základě největšího přírůstku modularity.

Na obrázku č. 6a lze vidět počáteční matici s hodnotami modularity pro každý vrchol. Jako první se spojí 1 s 2, následuje spojení se řádky a přepočítá se matice. Pokračují vrcholy 6 a 7. Následuje spojení 5 s předchozí komunitou (6,7).

Dále pak 3 se 4 a komunity 1,2 s 3,4. Postup je zaznamenán na obrázku č. 6b, kde se nachází dendrogram tohoto postupu.

```
> mod.matrix(g,membership(fast))
       [,1]   [,2]   [,3]    [,4]    [,5]   [,6]   [,7]
[1,] -0.250  0.750  0.750 -0.3750 -0.3750 -0.250 -0.250
[2,]  0.750 -0.250 -0.250  0.6250 -0.3750 -0.250 -0.250
[3,]  0.750 -0.250 -0.250  0.6250 -0.3750 -0.250 -0.250
[4,] -0.375  0.625  0.625 -0.5625  0.4375 -0.375 -0.375
[5,] -0.375 -0.375 -0.375  0.4375 -0.5625  0.625  0.625
[6,] -0.250 -0.250 -0.250 -0.3750  0.6250 -0.250  0.750
[7,] -0.250 -0.250 -0.250 -0.3750  0.6250  0.750 -0.250
```



Obr. č. 6a - Počáteční hodnoty přírůstků modularity v matici

Obr. č. 6b - Dendrogram znázorňující postup slučování vrcholů, zdroj: autor

3.3 Multi-level algoritmus

Tento algoritmus je jeden z nejvhodnějších algoritmů pro detekci komunit v sociálních sítích. V relativně krátkém čase dokáže najít největší přínos modularity a následně rozdělit síť na komunity. Algoritmus je náročný na paměťovou stránku než na čas výpočtu. Následující kapitola je popsána na základě článku od Vincent D. Blondela, Jean-Loup Guillaumea, Renaud Lambiottea a Etienne Lefebvrea v roce 2008 [17].

Algoritmus je rozdělen do dvou fází, které se stále opakují. V prvním kroku je každý vrchol sítě samostatnou komunitou. Pro každý vrchol *i* vyhledáme sousedy *j* a vypočítáme změnu modularity na základě sloučení *i* s komunitou *j*. Vrchol *i* je umístěn do té komunity, kde je největší pozitivní přírůstek modularity. Opakování platí pro všechny vrcholy, je-li místo pro zlepšování. Vrchol může být použit několikrát. První fáze končí, když je dosaženo lokálního maxima modularity a nelze ji již zvětšovat. Algoritmus je velmi citlivý na pořadí vrcholů. Uspořádání vrcholu nemá významný vliv na modularitu, ale může mít vliv na čas výpočtu. Tento problém by mohl být v budoucnu vyřešen a algoritmus by byl ještě o něco rychlejší.

Výpočet přírůstku modularity ΔQ při sloučení izolovaného vrcholu *i* do komunity C je následující:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

Rovnice č. 10

Kde $\sum_{in}$ je součet vah jednotlivých spojení v komunitě *C*, $\sum_{tot}$ je součet vah spojení příhod v *C*, k_i je součet vah spojení příhod k vrcholu *i*, $k_{i,in}$ je součet vah spojení z *i* do vrcholů v *C* a *m* je součet vah všech spojení v síti.

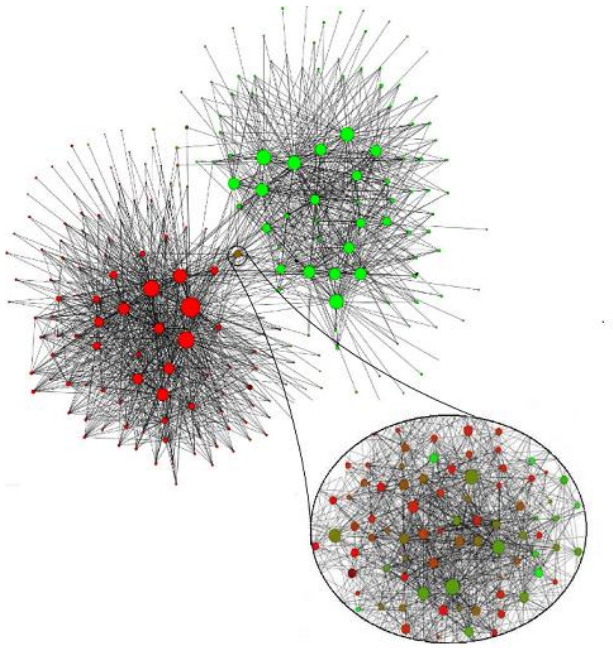
V praxi to funguje tak, že se vyhodnocuje modularita odstraněním *i* z vlastní komunity a přesune se do sousedící komunity.

Druhá fáze Multi-level algoritmu je založena na tvorbě nové sítě. Vrcholy komunity jsou známy z první fáze. Pokud je graf vážený, tak se nové váhy jednotlivých hran vypočítají na základě předcházejících vah mezi komunitami v první fázi algoritmu. Vazby mezi vrcholy stejné komunity jsou v nové síti

reprezentovány jako self-smyčky. Jakmile je nová síť vytvořena, přistupuje se zase na první fázi algoritmu. Počet komunit při každém průchodu algoritmu klesá a většinu času pro výpočet si bere první fáze. Průchody probíhají opakovaně, je-li alespoň nějaký přínos k maximalizaci modularity. Při průchodech se dá zjistit struktura jednotlivých komunit. Pro mnoho sítí je počet průchodů poměrně malý.

Algoritmus má mnoho výhod. Implementace tohoto algoritmu je intuitivní a jednoduchá. Největší výhodou je, že je velmi rychlý díky tomu, jak se snadno vypočítávají přírůstky modularity. Rychlost také ovlivní fakt, že se po několika průchodech algoritmu drasticky snižuje čas výpočtu. Algoritmus je ovlivněn omezením rozlišení modularity, která je popsána v předchozí kapitole. Tento problém se může obejít pomocí více-úrovňového charakteru algoritmu. Omezení je částečně relevantní pro Multi-level algoritmus, protože se shlukuje pouze jedna komunita s druhou v jednom průchodu. Případně se tyto komunity (sloučené vrcholy) sloučí v dalším průchodu. Algoritmus shlukuje komunity po průchodech pro různé úrovně organizace. Někdy jsou výsledky průchodů smysluplné pro detekci hierarchické struktury dané komunity a díky tomu se dokáže pozorovat struktura sítě s požadovaným rozlišením (viz obr. č. 7).

Algoritmus je téměř lineárně časově náročný. Je to velmi velká výhoda při pozdějším testování na sociálních sítích. Při úplném grafu může nabývat až kvadratické náročnosti, ale ve většině případů malých sítí zůstává lineární náročnost.

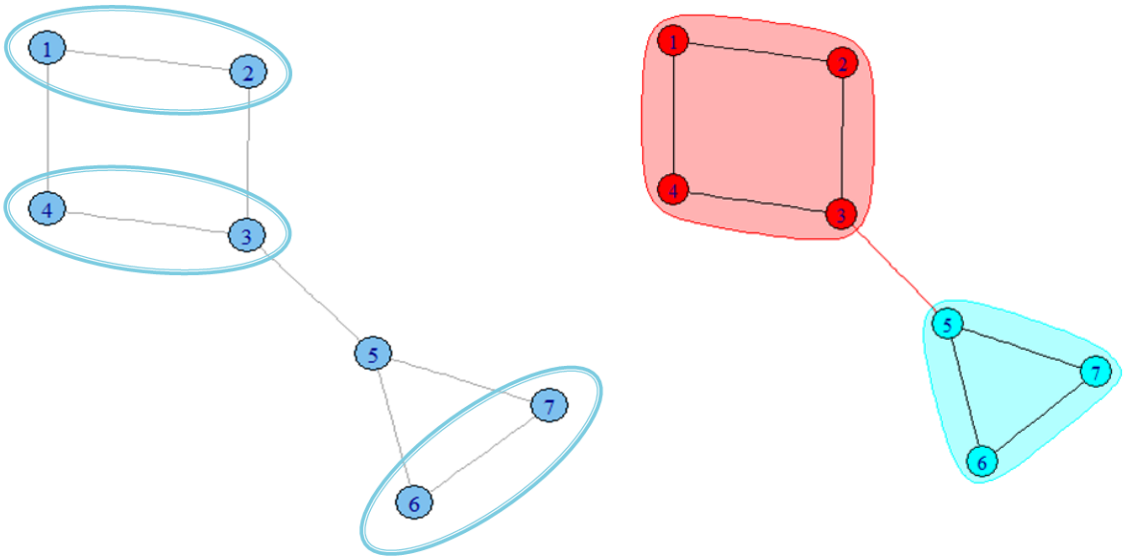


Obr. č. 7 - Ukázka požadovaného rozlišení - Síť znázorňuje jazyky, které jsou používány při telefonování v Belgii (Červená - francouzština, Zelená - holandština). Přiblížení na střední komunitu při vyšším rozlišení ukazuje, že se skládá z několika dalších komunit, kde už je méně patrná separace jazyků. Zdroj: [17]

3.3.1 Ukázka

Multi-level algoritmus vyhledává komunity v síti pomocí jednoduchého sdružování vrcholů. V každém kroku se vrchol spojí s vrcholem s největším přírůstkem modularity. Následně je vytvořena nová síť, na které se znovu opakuje sdružování komunit (vrcholů).

Na prvním obrázku č. 8a lze vidět první fáze algoritmu, kde se spojují vrcholy do dvojic. Na obrázku č. 8b je už ukončený algoritmus.



Obr. č. 8a - První fáze algoritmu - spojení vrcholů do dvojic

Obr. č. 8b - Ukončený algoritmus s dvěma komunitami, zdroj: autor

3.4 Leading-eigenvector algoritmus

Tento algoritmus je také implementován v iGraphu. Rozděluje síť vždy na dvě komunity na základě vedoucího vlastního vektoru (leading-eigenvector). Může být rozšířen o doladěnou část, která není součástí této implementace. Jeho hlavní myšlenka je čerpána z článku od M. E. J. Newmana v roce 2006 [18].

Algoritmus je velmi podobný se spektrálním rozdělením. Jsou zde zásadní rozdíly, které budou popsány níže. Bude zde ukázána hlavní rovnici rozdělení. Nejprve se musí vektor *s* napsat jako lineární kombinace vlastních vektorů u_i z B tak, že $s = \sum_{i=1}^{n} a_i u_i$ s $a_i = u_i^T * s$, potom lze modularita vypočítat takto:

$$Q = \frac{1}{4m} \sum_i a_i u_i^T B \sum_j a_j u_j = \frac{1}{4m} \sum_{i=1}^{n} (u_i^T * s)^2 \beta_i$$

Rovnice č. 11

kde β_i je vlastní hodnota B odpovídající vlastnímu vektoru u_i (popsáno v předchozí části).

Tyto vlastní hodnoty (eigenvalue) jsou označeny sestupně, $\beta_1 \geq \beta_2 \geq \beta_3 \geq ... \geq \beta_n$. Při snaze maximalizovat modularitu využíváme výběr vhodného rozdělení na základě hodnoty vektoru *s*, proto hledáme *s* s největší váhou přínosu modularity samozřejmě pozitivní jako v předchozích případech. Vlastní vektory (eigenvectors) jsou na sebe kolmé. Pokud by neexistovaly žádné další podmínky, tak by se *s* vybíralo snadno na základě u_1. Bohužel je tu omezení, že prvky *s* nabývají hodnot ±1, tudíž se nedá zjistit *s* na základě u_1. Z rovnice je patrné u_1^T*s, že největší přírůstek je při s_i = +1 a nejmenší na s_i = -1. Jinými slovy, všechny vrcholy pozitivní hodnoty jsou v jedné komunitě a s negativní hodnotou v druhé. Takto funguje algoritmus Leading-eigenvector. Vypočítá se vedoucí vlastní vektor matice modularity a rozděluje se na vrcholy podle znaménka prvku vedoucího vektoru.

Existuje také řešení, kde nebudou žádné pozitivní vlastní hodnoty v matici modularity. V tomto případě vedoucí vlastní vektor je vektor (1,1,1…), který odpovídá, že je pouze jedna komunita, obsahující všechny vrcholy. Tento jev je zcela správný a algoritmus nám říká, že neexistuje dobré rozdělení sítě, které by mělo za následek zvýšení modularity. To je velmi významná vlastnost algoritmu. Algoritmus dokáže nejen rozdělit síť do komunit, ale také odmítnout toto

rozdělení, pokud to nemá pozitivní přínos na modularitu. Jednoduše řečeno, síť je nedělitelná, jestliže matice modularity nemá pozitivní vlastní hodnoty.

Algoritmus vychází především ze znaménka prvků ve vedoucím vektoru. Znaménko je nejdůležitější, ale důležitá je i hodnota prvku. Velké prvky budou velkými přírůstky přispívat na modularitu, malé zase malými. Je-li již dokonale rozdělená síť, hodnota prvku vyjadřuje, o kolik se sníží modularita, když se daný vrchol přesune. To znamená, že hodnoty prvků vedoucího vlastního vektoru určuje, jak pevně patří vrchol k dané komunitě. Prvky s velkou hodnotou jsou ústředními vrcholy komunity. Na druhou stranu prvky s menšími hodnotami jsou více ambivalentní a nedrží tak dobře v dané komunitě.

Z předchozích odstavců je zřejmé, že se síť rozděluje jenom do dvou komunit pomocí znaménka prvku ve vedoucím vlastním vektoru. Většina sítí je rozdělována do více než dvou komunit. Jak docílit rozdělení do více komunit? Je to jednoduché - po rozdělení sítě na dvě, použijeme na každou komunitu znova algoritmus a po jeho dokončení znova. To je umožněno pomocí vlastností algoritmu, který je popsán výše, nerozdělovat síť, pokud to nepřináší zvýšení modularity. Bohužel tak přesně by algoritmus nefungoval správně. Po rozdělení sítě na dvě, se musí odstranit mezikomunitní hrany mezi nimi. Pokud by se odstranily bez rozmyslu hned, tak se následně budou počítat špatné přírůstky modularity pro každou komunitu. Je tedy nutné vytvořit další přírůstek ΔQ na modularitu pro další rozdělení skupiny G o velikosti n_g do dvou:

$$Q = \frac{1}{2m}\left[\frac{1}{2}\sum_{i,j\in g} B_{ij}\left(s_i s_j + 1\right) - \sum_{i,j\in g} B_{ij}\right] = \frac{1}{4m}\left[\sum_{i,j\in g} B_{ij} s_i s_j - \sum_{i,j\in g} B_{ij}\right] =$$

$$\frac{1}{4m}\sum_{i,j\in g}[B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}]s_i s_j = \frac{1}{4m} s^T B^{(g)} s \qquad \textit{Rovnice č. 12}$$

Kde δ je Kroneckerova delta (viz Fast-greedy), v této rovnici bylo využito $s_i^2 = 1$ a B je matice n_g *n_g prvků s hodnotami, které se vypočítají takto: $B_{ij}^{(g)} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}$

Ukončení algoritmu je velmi jednoduché. Algoritmus se ukončí, jestliže neexistují žádná další rozdělení, která by přinesla kladný přírůstek ΔQ. To znamená, vedoucí vlastní vektor nemá žádnou pozitivní hodnotu.

Jako v předchozích případech daný algoritmus ve zkratce funguje takto:

1. Vytvoří se matice modularity pro síť.
2. Najde se vedoucí vlastní hodnota (Leading eigenvalue), nejvíce pozitivní hodnota a odpovídající vlastní vektor (eigenvector).
3. Síť se rozdělí na dvě komunity podle znaménka prvků vektoru a celý proces se opakuje. Algoritmus skončí v době, kdy jsou všechny části dále nedělitelné.

V tomto algoritmu jsou všechny komunity v síti reprezentovány jako nedělitelné podgrafy. Doba chodu celého procesu je závislá na hloubce dendrogramu. Průměrná délka dendrogramu je *log n*. Průměrná délka výpočtu je tedy O ($n^2 \log n$), to je relativně náročné v kontextu této práce.

3.4.1 Doladění

Algoritmus může být rozšířen o jemné doladění, které s kombinací vedoucího vlastního vektoru je velmi efektivní.

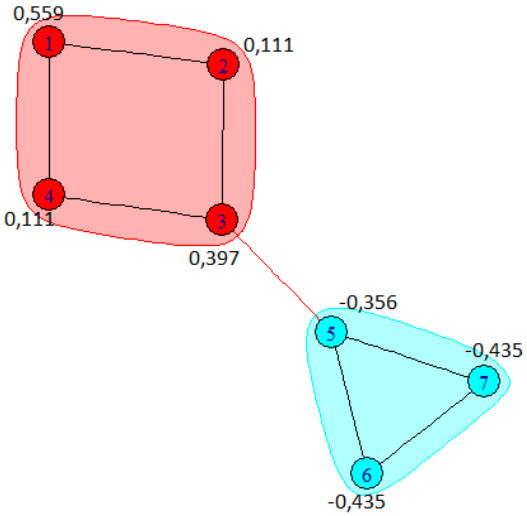
Tato třešnička na dortu je založena na Kerninghan-Lin algoritmu [19]. Algoritmus předpokládá, že síť je už rozdělena do dvou komunit. To nám zajistí předchozí algoritmus. Potom se najdou vrcholy, které při přesunu do jiné komunity mají největší přírůstek modularity v celé síti nebo nejmenší úbytek, až není možné modularitu zvýšit. Přesunout vrchol se může v každém kole pouze jednou. Když už by byly všechny vrcholy přesunuté, tak se hledá stav, který má největší modularitu. V dalším kole se začíná z tohoto stavu a celý proces se opakuje do té doby, pokud je ještě co zlepšovat.

Ve zkratce - algoritmus založený na Kerninghan-Lin pouze zpřesňuje algoritmus vedoucích vlastních vektorů. Toto doladění přidává pouze několik procent na konečné hodnotě modularity. Na klubu Zachary se bez doladění dostane na Q=0,393, ale s doladěním na Q=0,418. Nicméně i tento přírůstek dělá z obyčejného rozdělení komunit vynikající. Negativní je fakt, že toto doladění přidává algoritmu větší časovou náročnost.

3.4.2 Ukázka

Leading-eigenvector algoritmus funguje na základě rozdělení sítě na dvě komunity. V každém kroku algoritmu je graf rozdělen na dvě části na základě vedoucího vlastního vektoru matice. V jednoduchém grafu jako je tento, se hned při prvním kroku graf rozdělí správně. Jednotlivé hodnoty vedoucího vlastního vektoru jsou vypočítány pomocí příkazu v *iGraphu - graph.eigen(g)[c("values", "vectors")]*, kde je vidět jednoduchá syntaxe. Byl využit graf G s požadavkem hodnot (values) a s vektorem (vectors). Tento příkaz lze využít přesně na tento případ algoritmu. Ve výchozím nastavení se počítá na základě největší velikosti vlastní hodnoty tomu odpovídající vlastní vektor.

Samotné hodnoty vektoru jsou tedy V (0.3965481, -0.4352219, -0.4352219, 0.5587870, -0.3561229, 0.1114607, 0.1114607). Na základě znaménka se rozdělí graf na dvě komunity. Pořadí těchto hodnot jsou počítána na základě největší vlastní hodnoty, proto je druhá, třetí (vrchol 6,7) a pátá (vrchol 5) hodnota záporná. Pořadí neodpovídá vrcholům. Lze si povšimnout, že vektor obsahuje tři záporné hodnoty, komunita trojúhelníku, a čtyři kladné hodnoty, komunita čtverce. Na obrázku č. 9 je možno vidět toto shrnutí a u každého vrcholu jeho hodnotu ve vektoru.



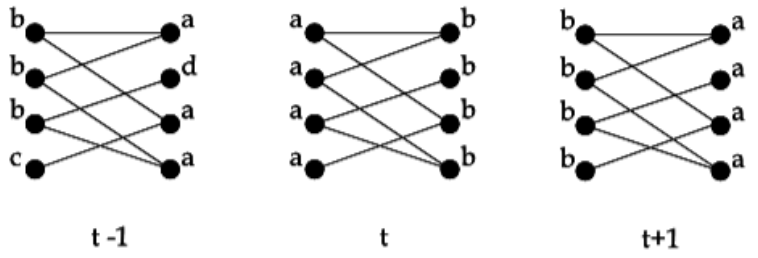
Obr č. 9 - Ukázka algoritmu - u každého vrcholu jsou hodnoty odpovídající vedoucímu vlastnímu vektoru, zdroj: autor

3.5 Label-propagation algoritmus

Label-propagation algoritmus je velmi vhodný pro detekci komunit v sociálních sítích. Využívá jednoduchý přístup přiřazování vrcholů do komunit, díky čemuž je rychlý a časově nenáročný. Bohužel při detekci komunit v síti jeho výsledky závisí na počáteční konfiguraci. V této kapitole je vycházeno z článku od Usha Nandini Raghavana, R´eka Alberta a Soundar Kumara v roce 2007 [20].

Hlavní myšlenkou tohoto algoritmu je přiřadit každému vrcholu označení, štítek (label). Předpoklad je takový, že každý vrchol *X* má své sousedy x_1, x_2 .. x_k , kteří nesou štítek označující příslušnost k dané komunitě. Potom určení vrcholu k *X* komunitě je na základě jeho sousedů. Takto bude každý vrchol přiřazen k určité komunitě. Problém nastává, má-li daný vrchol stejný počet sousedů s více komunitami. Algoritmus přiřadí vrchol náhodně k jedné komunitě. Štítky se šíří rychle a v hustě propojených komunitách je shoda opravdu velmi rychlá. Na konci procesu jsou vrcholy, které mají stejný štítek přiřazeny do stejné komunity.

Proces se provádí opakovaně a při každém kroku se aktualizují štítky na každém vrcholu na základě jeho sousedů. Tato aktualizace může probíhat synchronně či asynchronně. Při synchronní aktualizaci si vrchol *x* při T iteraci aktualizuje štítek na základě štítků při T-1 iteraci. Tedy $C_x(t) = f(C_{x1}(t-1), ..., C_{xk}(t-1))$, kde $C_x(t)$ je označení vrcholu *x* v čase *t*. Problém nastává při bipartitních podgrafech, kde štítky střídají své pozice (viz obr. č. 10). Zde se využívá asynchronní aktualizace štítků, kde $C_x(t) = f(C_{xi1}(t), ..., C_{xim}(t), C_{xi(m+1)}(t-1), ..., C_{xik}(t-1))$, kde $x_{i1}, ..., x_{im}$ jsou sousedy *x*, které již byly aktualizovány v dané iteraci, zatímco $x_{i(m+1)}, ..., x_{ik}$ jsou sousedé, které ještě nejsou aktualizovány v aktuální iteraci. Pořadí, ve kterém jsou vrcholy v síti aktualizovány, je náhodné při každé iteraci. Logicky vyplývá, že na začátku algoritmu máme tolik štítků, kolik je vrcholů, ale na konci procesu máme tolik štítků, kolik je komunit.



Obr č. 10 - Příklad bipartitního grafu, kde při každé iteraci si štítky prohazují místa, zdroj: [20]

V ideálním případě se algoritmus zastaví, když už žádný vrchol nemění svůj štítek. Bohužel existují vrcholy, které mají stejný počet sousedů se dvěma různými štítky. Tyto vrcholy budou při každé iteraci náhodně měnit štítek z jednoho na druhého, i přestože sousedé zůstávají stejní. Z tohoto důvodu se zavedl štítek s maximálním počtem svých sousedů v dané komunitě. Potom je celé rozdělení sítě do disjunktních komunit a každý uzel má alespoň tolik sousedů v rámci své komunity, kolik je sousedů v jiné komunitě. Pokud $C_1, ..., C_p$ jsou štítky, které jsou aktivní v dané době a $d_i^{C_j}$ je počet sousedů vrcholu i se štítkem C_j, tak se algoritmus zastaví, když platí, že vrchol i má štítek C_m potom $d_i^{C_m} \geq d_i^{C_j} \; \forall \, j$.
Fungování tohoto algoritmu by se dalo shrnout takto:

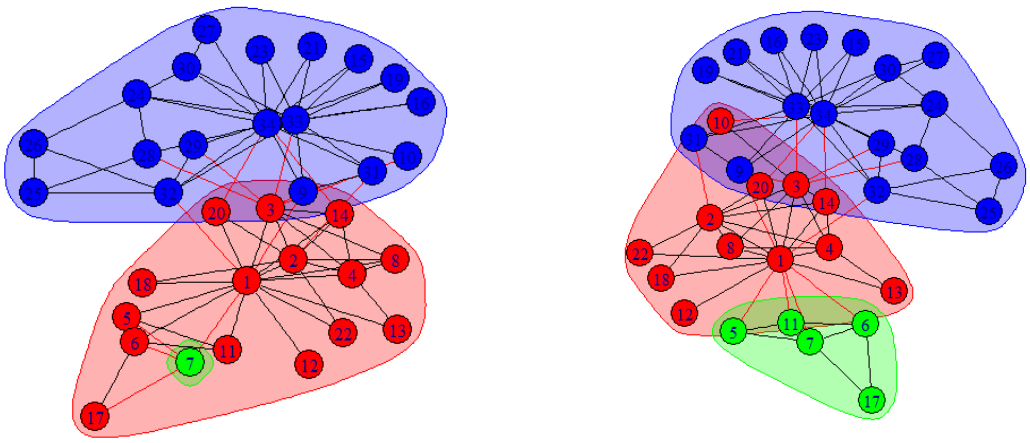
1. Inicializace štítků na všechny vrcholy v síti pro daný vrchol x, $C_x(0) = x$.
2. Nastavit čas $t = 1$
3. Upořádání vrcholů v náhodném pořadí a nastavení na X
4. Pro každý x z X je zvoleno v konkrétním pořadí, ať $C_x(t) = f(C_{xi1}(t), ..., C_{xim}(t), C_{xi(m+1)}(t-1), ..., C_{xik}(t-1))$, kde f vrátí štítek s nejvyšší koncentraci u sousedů, a je –li stejný počet, vybere se náhodně.
5. Jestliže má každý vrchol štítek na základě maximálního počtu štítků sousedů, algoritmus končí, jinak se nastaví t = $t+1$ a vrací se na bod č. 3.

Časová náročnost Label propagation algoritmu je velmi dobrá. Algoritmus potřebuje na své dokončení téměř lineárně dlouhý čas. Pro každou iteraci je celková doba O (m). Bylo zjištěno, že po páté iteraci je většinou 95% vrcholů přiřazeno správně do své komunity [20].

3.4.3 Nevýhoda

Algoritmus je velmi závislý na počáteční konfiguraci a na náhodném přiřazení štítků. Tento jev nastává díky mnoha náhodným situacím v tomto algoritmu. Výsledek ovlivňuje pořadí synchronní a asynchronní aktualizace a náhodné přiřazení štítků při rovnosti počtu sousedů. Balíček iGraph umožňuje nastavení počáteční konfigurace na základě vektoru, který by byl do atributu metody přidán. Kladné hodnoty vektoru by znamenaly oštítkování vrcholu. Na

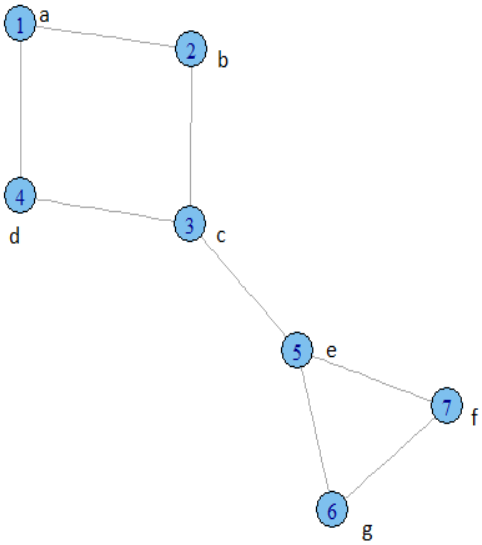
druhou stranu záporné hodnoty by znamenaly nepojmenování vrcholu. Je umožněno i nastavení vrcholů, které bude v grafu pevně dané a nebude se moci změnit. Tento nešvar nahodilosti je ukázán na síti Zachary (obr. č. 11). Modularita se při pěti měření pohybovala od 0,343 (a) do 0,402 (b).

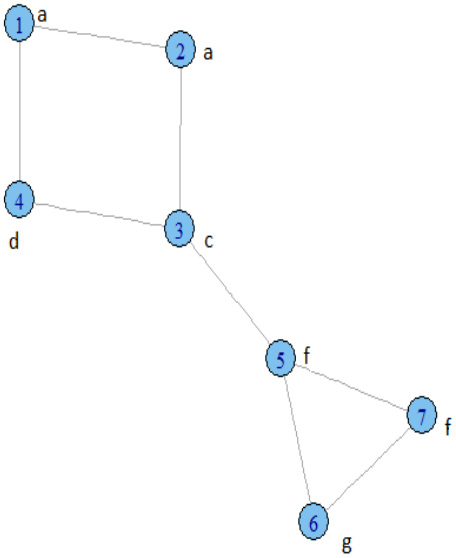


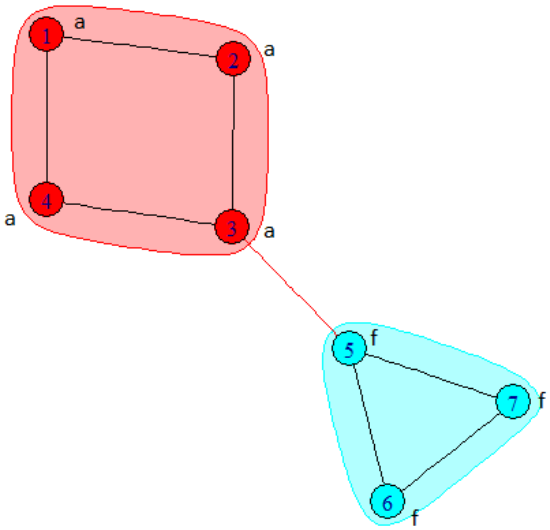
Obr č. 11 -Ukázka různých výsledků algoritmu. Na prvním (a) je Q = 0,343 na druhém (b) je Q = 0,402, zdroj: autor

3.4.4 Ukázka

Label-propagation algoritmus tedy funguje na jednoduchém oštítkováním vrcholů v síti. Díky své jednoduchosti je velmi rychlý. V každém kroku se mění jednotlivé štítky vrcholů na základě jejich sousedů. Algoritmus končí, jsou-li všechny štítky neměnné. Bohužel u složitějších grafů se jeho výsledky velmi různí. To je zapříčiněno díky počáteční konfiguraci, která je náhodně generována. Na prvním obrázku č. 12a je znázorněno počáteční náhodné přiřazení štítků k vrcholům. Na následujícím obrázku je další náhodné přiřazení vrcholů k dané komunitě. Na posledním je znázorněn již ukončený algoritmus s výslednou strukturou grafu.







Obr. č. 12a - Počáteční inicializace štítků na vrcholy

Obr. č.12b - Náhodné přiřazení štítků

Obr. č. 12c - Ukončený algoritmus s konečným přiřazením štítků, zdroj: autor

4 Testování algoritmů na různě veliké sítě

V předchozích částech bylo představeno pět algoritmů pro detekci komunit v sociálních sítích na velmi jednoduchém grafu o 7 vrcholech a 8 hranách a také jejich základní charakteristika. Následující část bakalářské práce bude věnována aplikaci těchto algoritmu v různě velikých sítí.

4.1 Představení testovacích sítí

Všechny následující sítě byly převzaty z oficiálních stránek Standfordské univerzity. Za touto obrovskou kolekcí dat sítí jsou zodpovědní Jure Leskovec a Andrej Krevl [21].

První síť a zároveň nejmenší síť, na kterou budou algoritmy aplikovány, bude dobře známá ukázková síť Karate klub Zachary (viz bod 2.6)

Druhá nejmenší síť je část sociální sítě Facebook. Tato síť se skládá z tzv. kruhů přátel (seznamu přátel) na Facebooku. Vrchol znázorňuje lidi (profily) a hrany znázorňují přátelství mezi lidmi. Data byly shromážděny podle údajů účastníků v průzkumu pomocí mobilní aplikace Facebook. Všechny data byly anonymní. Síť obsahuje 4039 vrcholů s 88 234 hrany.

Další síť je Twitter. Data jsou čerpány z veřejně dostupných zdrojů. Síť obsahuje 81 306 vrcholů a 1 768 149 hran.

Síť Google+ reprezentuje kruhy přátel v sociální síti Google+. Údaje byly shromážděny od uživatelů, kteří se podělili o své kruhy přátel pomocí funkce "sdílení kruhu". Obsahuje 107 614 vrcholů a 13 673 453 hrany.

Dále byla použita síť Amazon. Síť byla vytvořena díky procházení webové stránky Amazon. Je založena na doporučených produktech. Je-li výrobek *i* často koupený s výrobkem *j*, tak neorientovaná hrana vede od *i* k *j*. Síť je složena z 334 863 vrcholů s 925 872 hranami. V této síti je znám i přesný počet komunit – 151037.

Předposlední síť je síť Youtube. Youtube je webová stránka pro sdílení videí a zahrnuje také sociální síť. Uživatelé mohou vytvářet přátelství a sdružovat se do skupin. Tyto uživatelsky definované skupiny jsou brány jako jednotlivé komunity. Obsahuje 1 134 890 vrcholů, 2 987 624 hrany a 8 385 komunit.

Poslední a největší je síť LiveJournal. LiveJournal je bezplatná blogující komunita, kde si uživatelé mohou sdílet přátelství mezi sebou. LiveJournal umožňuje vytvářet skupiny, do kterých se následně sdružují uživatelé využívající tuto síť. Dané skupiny jsou považovány za komunity. LiveJournal se skládá z 3 997 962 vrcholů, 3 4681 189 hran a síť má 287 512 komunit.

Největší síť, která byla k dispozici, byla síť Friendster. Friendster je on-line herní a sociální síť. Obsahuje přes 65 milionů vrcholů a přes 1,8 miliardy hran. Bohužel na tuto síť nebyl dostatečný hardware. Zde se už projevoval nedostatek operační paměti daleko více než náročnost pro výpočet některých algoritmů.

4.1.1 Předpříprava sítí

Data o těchto sítí byla většinou reprezentována pomocí Edgelist. Edgelist je dvousloupcová matice, kde v každém řádku první sloupec reprezentuje počáteční vrchol a druhý sloupec koncový vrchol. Většinou v textové podobě s koncovkou.txt, což iGraph nerad vidí. Bohužel tyto sloupce byly ještě od sebe různě odděleny, nejčastěji tabulátorem nebo čárkou. Často obsahoval i několik řádků textu, než se začal vlastní popis struktury dané sítě.

Kvůli této skutečnosti se musel soubor.txt trošku složitěji importovat do vývojového prostředí. Používala se metoda od Daizaburo Shizuka z Univerzity Nebraska-Lincoln [22].

Následující kód (obr. č. 13) se používal v různých modifikacích na každou síť. V prvním řádku se projížděl celý vybraný soubor řádek po řádku, hledaly se characters (znaky). Pomocí atributu skip bylo přeskočeno několik řádků informací v případě jejich existence. V druhém řádku se nahrazovaly tabulátory (\t) nebo čárky za klasickou mezeru. V následujícím řádku se odstraňovaly čárky a násobné mezery. Další krok rozděloval čísla vrcholů na listy s odlišným vektorem pro každý řádek. Potom nastalo vytvoření prvního sloupce podle prvních prvků vektorů. Následně se vytvořil druhý sloupec pomocí zbytku v každém vektoru, tedy druhý prvek v každém řádku. Dále se vytvořil Edgelist kombinací těchto dvou sloupců a v poslední řadě se tento Edgelist převedl do grafického objektu iGraph, se kterým se mohlo následně pracovat.

```
> radky=scan(file.choose(),what="character",sep="\n",skip=4)
Read 88230 items
> radky=gsub("\t"," ",radky)
> radky=gsub("[ ]+$","",gsub("[ ]+"," ",radky))
> list=strsplit(radky," ")
> sloupec1=unlist(lapply(list,function(x) rep(x[1],length(x)-1)))
> sloupec2=unlist(lapply(list,"[",-1))
> edgelist=cbind(sloupec1,sloupec2)
> g=graph.edgelist(edgelist,directed=FALSE)
```

Obr. č. 13 - Zdrojový kód pro import jednotlivých sítí, zdroj: [22]

4.2 Předpoklady

V první řadě je nutno vědět, na jakém hardwaru se testování algoritmů provádělo. Na silnějším hardwaru by samozřejmě byly výsledky odlišné od naměřených hodnot v této bakalářské práci. Testování probíhalo na klasickém starším notebooku z roku 2012 s parametry:

Tabulka č.1 : Testovací sestava

CPU	Intel core i7-3610QM - 2.3 GHz
RAM	8GB DDR3
Operační systém	Windows 7 - 64bit

zdroj: autor

Musel být nastaven určitý časový limit, do kterého musely algoritmy ukončit svůj výpočet. Pokud algoritmus nesplnil tuto podmínku, bylo automaticky rozhodnuto, že daný algoritmus neuspěl a jeho výpočet byl ukončen. Limit se nastavil na jednu hodinu. Tato hodina byla dostatečně dlouhá, aby se ukázalo, jestli má algoritmus vůbec šanci na rozumné ukončení svého výpočtu. Důkazem jsou výsledky z tabulky č. 2, kde nejdelší úspěšný výpočet trval okolo 16 minut.

Další předpoklad vychází z neschopnosti většiny algoritmů detekovat komunity v orientovaných grafech. Síť Twitter je typickým představitelem orientovaného grafu. Tento nedostatek byl vyřešen jednoduchou konverzí v prostředí iGraphu s použití konverze *as.undirected*. Tento příkaz jednoduše převede graf s orientovanými hranami na graf bez nich. Výsledky bohužel nebudou odpovídat skutečné síti, ale změněné síti bez těchto hran.

Poslední předpoklad byl, že u Label-propagation se provádělo 5 měření na každou síť. Jak bylo popsáno výše, v tomto algoritmu hodně záleží na počáteční náhodné konfiguraci. Bral se výsledek, který odpovídal mediánu těchto měření.

4.3 Aplikace

V této části bude na konci kapitoly ukázána tabulka s výsledky jednotlivých algoritmů z předcházejících sítí.

Pro cvičnou síť Zachary jsou všechny algoritmy úspěšné. Výsledky nad touto sítí nejsou časově měřeny z důvodu svého téměř okamžitého výsledku. Počet komunit se pohybovalo v rozmezí 3 až 5 komunit, kde u Edge-betweenness se tato pátá komunita skládala pouze z jednoho vrcholu (viz Edge-betweenness). Nejlepším algoritmem se na této síti ukázal Multi-level algoritmus s hodnotou modularity Q = 0,418.

Další aplikace byla na části sítě Facebook. Tato síť bude v další kapitole rozebrána podrobněji, proto v této části bude jenom nastíněna. Zde už je první odpadlík Edge-beetweeness, který díky své velké náročnosti na výpočet, tuto síť nezvládá. Čas na výpočet u dalších algoritmů se pohybuje v řádu několika vteřin. Na této síti se již začíná objevovat relativně velký rozptyl počtu komunit. Multi-level algoritmus se zde znovu ukázal jako nejlepší.

Následuje aplikování algoritmů na síti Twitter. Nejpomalejší algoritmus se zde projevil jako Fast-greedy. Jeho čas výpočtu byl přes 16 minut. Výpočetně nejrychlejší čas byl překvapivě algoritmus Label-propagation. To je zapříčiněno tím, že algoritmus v grafu s vyšší hustotou hran lépe zaplavuje síť. Tuto skutečnost lze také vidět na následující síti, kde je rozdíl ještě markantnější. Je zajímavé zde sledovat velký rozptyl komunit. Multi-level s 90 komunitami s hodnou modularity Q=0,803, Fast-greedy s 146 komunitami s hodnotou Q=0,726 a Label-propagation s 1104 komunitami s hodnotou Q=0,775. Na Twitteru se už objevuje velké omezení rozlišení (viz Modularita) pro algoritmus Leading-eigenvector, kde je celá síť rozdělena pouze na 4 komunity. Díky tomuto omezení algoritmus ztrácí hodnotu modularity s relativně velkou nepřesností na ostatní tři algoritmy. Je možné předpokládat, že algoritmus nebude moci další sítě kvalitně rozdělit.

Podle charakteristiky sítě Google+ s velkým počtem hran, je zřejmé, že nejrychlejší algoritmus zde bude opět Label-propagation. Multi-level zde ztrácí přes 45 vteřin. Na této síti se objevil další odpadlík a tím byl Fast-greedy. V časovém limitu hodiny nedokázal ukončit svůj výpočet. Na této síti je rozptyl

komunit malý. Nejlepší výsledek s ohledem na modularitu má Multi-level s počtem 48 komunit a s hodnotou modularity Q= 0,441. Na Google+ se projevuje předešlý nedostatek algoritmu Leading-eigenvektor. Omezení rozlišení je zde už tak velké, že algoritmus označí celou síť jako jednu velkou komunitu. Modularita je samozřejmě v tomto případě nula. Je zřejmé, že algoritmus Leading-eigenvektor v této síti bude vždy končit pouze s jednou komunitou a modularitou Q=0. Pro účel bakalářské práce je algoritmus dále nepoužitelný. Na této síti jsou celkem dva odpadlíci.

V další síti Amazon jsou překvapivě tři úspěšně dokončené algoritmy. V Amazonu byl ve všech atributech nejlepší algoritmus Multi-level. Strukturu sítě našel kolem 5 vteřin, počet komunit byl 249 a s hodnotou modularity Q= 0,926. Fast-greedy, který v předchozí síti Google+ selhal, trval téměř 15 minut s 1525 komunitami a hodnotou modularity Q=0,876. Tady se projevila lineárnější časová náročnost v menších grafech na rozdíl od předcházející sítě. Label-propagation našel dokonce 22 389 komunit s hodnotou modularity Q=0,784. Výpočet trval okolo 30 vteřin.

Při porovnávání jednotlivých počtů komunit se známým počtem komunit jsou výsledky velmi nedostačující. Síť obsahovala přes 150 tisíc komunit, ale testující algoritmy najdou o mnoho méně. Relativně nejlepší Label-propagation našel pouze přes 22 tisíc těchto komunit. U všech algoritmů, které jsou založeny na optimalizaci modularity, se projevuje omezení rozlišení modularity. Algoritmy nezvládají určovat malé komunity v těchto obrovských sítích. Na této síti je také krásně poznat časová náročnost jednotlivých algoritmů. Nejrychlejší algoritmy jsou lineárně časově náročné. V relativně krátkém čase jsou schopny najít strukturu jednotlivých sítí. Algoritmy exponencionálně náročné jsou o mnoho pomalejší a nezvládají fungovat s těmito velkými sítěmi.

Následující síť reprezentující síť s malým počtem komunit, kde algoritmy mají bezpochyby lepší podmínky pro hledání struktury a tudíž i lepší výsledky, se nazývá Youtube. Bohužel pro většinu algoritmů je tato síť už příliš velká. Zde se stal nejlepším algoritmem Multi-level po 15 vteřinách svého výpočtu. Jeho hodnota modularity byla Q=0,685. S počtem komunit 9630 se přiblížil ke skutečnému počtu komunit v síti (8 385). Label-propagation zde trval lehce přes minutu a našel

dokonce téměř 35 tisíc komunit s hodnotou modularity Q=0,537. Zde je na místě porovnat i jednotlivé komunity. Největší komunita v Multi-level algoritmu obsahuje 198 088 vrcholů propojené s 614 056 hranami. Další měla přes 134 tisíc vrcholů a přes 355 tisíc hran. U Label-propagation to byla komunita s 578 540 vrcholy a 1 594 252 hranami, další s téměř 95 tisíci vrcholy a přes 350 tisíci hranami. Multi-level má větší počet velkých komunit. U algoritmu je několik komunit se sto tisícovou hranicí a s desítky tisícovou hranicí. Na druhou stranu Label-propagation má vždy jednu obrovskou komunitu. Další komunity jsou už o mnoho menší. Algoritmus vyhledá mnoho malých komunit. Díky této skutečnosti má Multi-level větší hodnotu modularity, přestože Label-propagation najde více komunit.

Poslední a největší síť je LiveJournal. Na této síti byl čas výpočtu už v řádech minut a dokončily ho opět pouze dva algoritmy. Multi-level algoritmus s přehledem poráží Label-propagation. Multi-level trval přes 4 minuty, našel pouze 7322 komunit s modularitou Q=0,717. Druhý algoritmus trval přes 11 minut, našel 29 123 komunit s modularitou Q=0,463.

Oba algoritmy našly samozřejmě o mnoho méně komunit než je ve skutečnosti v síti. Multi-level má daleko lepší výsledek v oblasti modularity. Znovu se projevuje velikost jednotlivých komunit jako v předchozím případě. Multi-level a jeho největší komunita má 830 785 vrcholů spojené s 6 847 587 hranami. Oproti algoritmu Label-propagation, který přiřadil do své největší komunity 3 091 025 vrcholů spojené s 23 462 969 hranami. Více než 60% všech vrcholů a hran bylo přiřazeno pouze do jedné komunity.

Tabulka č.2: Porovnání algoritmů na různě veliké sítě

	Multi-level	Label-propagation	Fast-greedy	Leading-eigenvector	Edge-betweenness
Zachary-Q Počet	Q=0,418 n=4	Q=0,387 n=3	Q=0,381 n=3	Q=0,393 n=4	Q=0,401 n=5
Facebook	Q=0,835 n=16 t=00:01	Q=0,817 n=55 t=00:01	Q=0,775 n=13 t=00:02	Q=0,799 n=18 t=00:02	X
Twitter	Q=0,809 n=90 t=00:06	Q=0,775 n=1107 t=00:04	Q=0,726 n=146 t=16:15	Q=0,410 n=4 t=00:30	X
Amazon	Q=0,926 n=249 t=00:05	Q=0,784 n=22389 t=00:30	Q=0,876 n=1525 t=14:45	Q=0 n=1 t=00:13	X
Google+	Q=0,441 n=48 t=00:56	Q=0,403 n=152 t=00:11	X	Q=0 n=1 t=00:17	X
Youtube	Q=0,685 n=9630 t=00:15	Q=0,537 n=34867 t=01:03	X	Q=0 n=1 t=00:22	X
LiveJournal	Q=0,717 n=7322 t=04:17	Q=0,463 n=29123 t=11:13	X	Q=0 n=1 t=03:55	X

zdroj: autor

4.4 Porovnání komunit v sítí Facebook

V této části práce bude blíže nahlédnuto do komunit v síti Facebook. Budou porovnány jednotlivé komunity, které mají více než 10% vrcholů. To dělá v síti Facebook více než 400 vrcholů v komunitě.

První algoritmus, který bude zkoumán, je Fast-greedy. Z celkových 13 komunit, které algoritmus detekoval, se našly čtyři komunity s více než 400 vrcholy. Další komunita měla 372 vrcholy a také zde byly 3 komunity s více než 200 vrcholy. Dále se komunity pohybovaly v menších desítkách vrcholů a nejmenší komunita měla 6 vrcholů. Průměrná velikost komunit byla 310 vrcholů, což není vůbec vypovídající údaj. V tomto případě je lepší poukázat na medián počtu vrcholů, který je 205 vrcholů.

Největší komunita měla 982 vrcholy, které byly spojeny 25 443 hranami. Tato komunita je reprezentována na obrázku č. 14a oranžovou barvou. Další komunita s 816 vrcholy a spojená s 13 454 hranami, je reprezentovaná červenou barvou. Následující dvě měly 548 vrcholů s 5356 hranami modré barvy a 546 vrcholů s 13 818 hranami zelené barvy.

Dalším algoritmem je Multi-level. Algoritmus detekoval 16 komunit v síti. Pět komunit splnilo podmínky, že mají přes 400 vrcholů. Mnoho komunit má řádově stovky vrcholů a nejmenší komunita obsahuje 19 vrcholů. Zde je průměr 252 vrcholy a medián má hodnotu 226. Zde už není takový rozptyl od mediánu k průměru.

Největší komunita měla 548 vrcholů spojené s 5356 hranami (oranžová). V těsném závěsu je komunita s 543 vrcholy a 8715 hranami (červená). Následující tři komunity měly 435 vrcholů s 16 694 hranami (modrá), 431 vrchol a 6066 hranami (zelená) a 423 vrcholy a 11 422 hranami (fialová). I když to na obrázku č. 14b nevypadá, tyto tři sítě byly hustěji spojené než největší dvě sítě.

Dalším algoritmem, který bude rozebrán, je Leading-eigenvector. Z celkových 18 komunit splnily podmínku pouze dvě respektive tři komunity. Průměr velikosti komunit je 224 vrcholy a medián je zde 207.

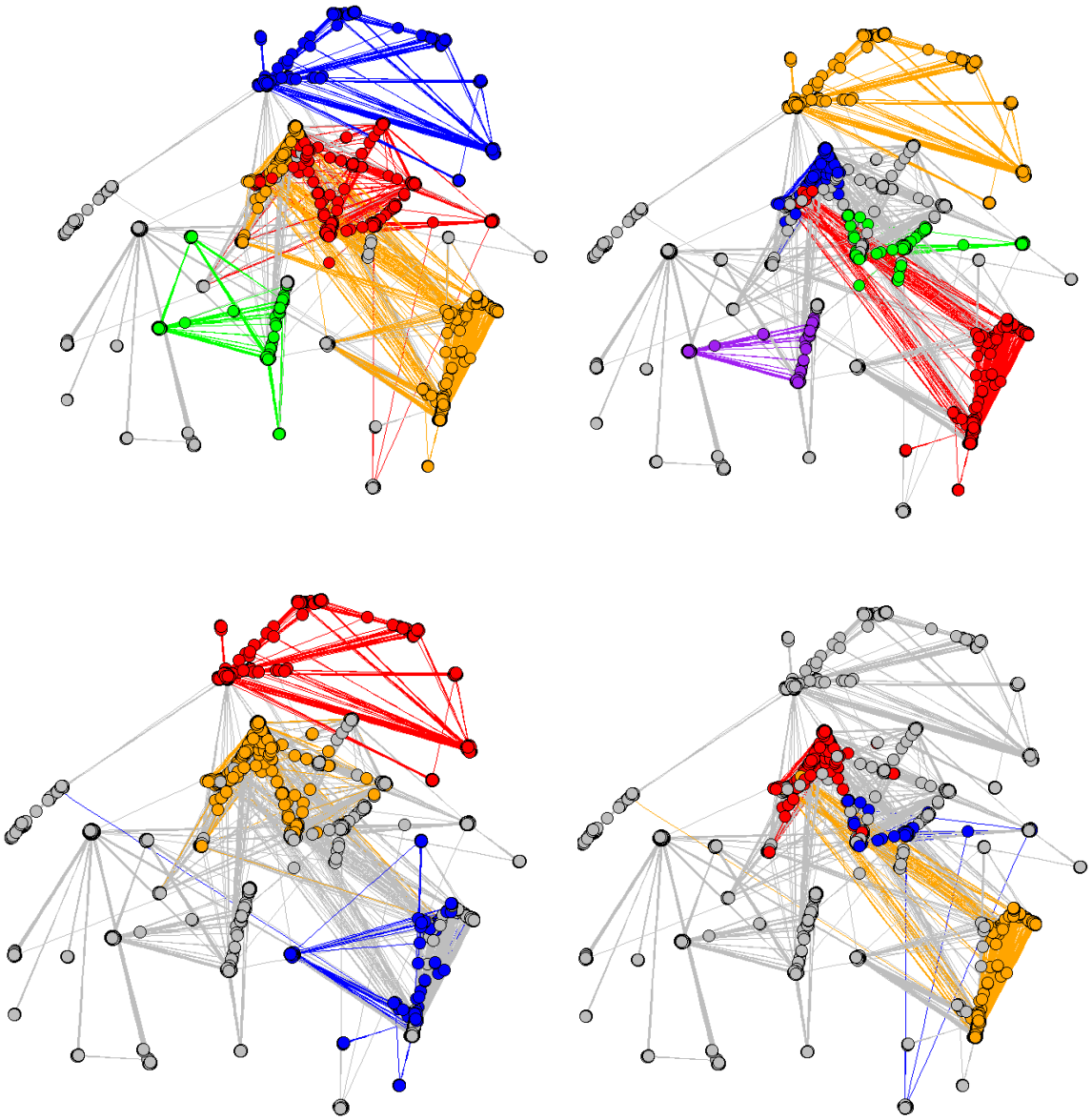
Komunita s největším počtem vrcholů měla 599 vrcholů spojené s 17 664 hranami (oranžová). Druhá v pořadí 544 vrcholy s 5312 hranami (červená).

Poslední, která nesplnila podmínku pouze o jeden vrchol, měla tedy 399 vrcholů s 6471 hranou (modrá).

Posledním algoritmem, který zvládl úspěšně najít strukturu sítě, je Label-propagation. Z celkových 55 nalezených komunit mají pouze dvě přes 400 vrcholů. Potom je zde okolo deseti komunit, které mají rozptyl vrcholů od 116 do 330 a mnoho komunit s desítkami vrcholů. Algoritmus našel deset malých komunit s maximálně s deseti vrcholy. Nejmenší komunita zde má 4 vrcholy. Průměr je 73 vrcholů na komunitu a medián je 21.

Jako největší komunitu algoritmus našel komunitu s 488 vrcholy spojené s 8430 hranami (oranžová). Následující komunita má 465 vrcholů spojené s 16 900 hranami (červená). A pro dokreslení třetí má 330 vrcholů s 5177 hranami (modrá).

V následujícím odstavci budou rozebrány postřehy z těchto komunit. U algoritmů Fast-greedy a Multi-level je komunita, která je naprosto shodná a velmi podobná s výsledkem algoritmu Leading-eigenvector. Tato komunita se nachází nahoře na obrázcích 14a, 14b a 14c. Dále je zajímavý celkový počet vrcholů v největších komunitách. U Fast-greedy to bylo přes 70% všech vrcholů ve čtyřech komunitách. U Multi-level to bylo okolo 60% v pěti komunitách. Leading-eigenvector zde má pouze 40% ve třech komunitách a nejméně Label-propagation lehce přes 30% v také ve třech komunitách.



Obr. č. 14a - Fast-greedy algoritmus - 4 komunity obsahující přes 70% všech vrchol, největší komunity byly reprezentovány oranžovou barvou, druhé v pořadí červenou, třetí modrou a dále zelenou a fialovou

Obr. č. 14b - Multi-level algoritmus - 5 komunity obsahující okolo 60% všech vrcholů

Obr. č. 14c - Leading-eigenvector algoritmus - 3 komunity obsahující okolo 40% všech vrcholů

Obr. č. 14d - Label-propagation algoritmus - 3 komunity obsahující pouze přes 30% všech vrcholů, zdroj: autor

5 Shrnutí výsledků

Cílem bakalářské práce bylo podat srozumitelné informace o základních termínech, jež se používají v problematice detekování komunit, popsání jednotlivých algoritmů, které tyto komunity hledají a v poslední řadě praktická ukázka algoritmů na různě velké sítě.

Bylo vysvětleno, že doposud nejlepším měřítkem rozdělení jednotlivých komunit je pomocí modularity, i přestože má velkou nevýhodu v podobě slučování malých jasně definovaných komunit ve velkých sítích.

Dále byly názorně vysvětleny jednotlivé algoritmy a jejich fungování. Proběhla názorná ukázka těchto algoritmů na jednoduchém grafu, při kterém už bylo možno pozorovat jednotlivé výhody a nevýhody algoritmů.

Následně bylo demonstrováno, že pro mnoho algoritmů jsou současné obrovské sociální sítě téměř nespočitatelné. Tento fakt je zapříčiněný náročností na výpočetní čas těchto algoritmů. Na druhou stranu nejrychlejší algoritmy dokázaly největší síť spočítat v řádech několika minut. Bohužel bylo jasně prokázáno, že při vyšším počtu definovaných komunit ve větších sítí se žádný algoritmus nedokázal přiblížit ke skutečnému počtu komunit v síti. Tato skutečnost je zapříčiněna nevýhodou použití optimalizace modularity a u Label-propagation algoritmu náhodným přiřazováním štítků.

6 Závěry a doporučení

Bakalářská práce měla mnoho cílů. Jedním z nich bylo přiblížení pojmu modularita. Byla předvedena reprezentace modularity v rovnici a její častější využívání maticového vyjádření. Bylo poukázáno na nevýhodu modularity v podobě jejího rozlišení, kde vznikají velké komunity. Modularita je využívána jako měřítko, jak dobře clustering (shlukování) komunit je.

Dalším cílem práce bylo popsat jednotlivé algoritmy pro detekci komunit v síti v jazyku R s použitím knihovny iGraph. Každý algoritmus byl důkladně rozebrán. Vždy byl popsán způsob, jak se dotyčný algoritmus chová v síti a jak se komunity vyhledávají. Názorně bylo ukázáno toto chování na jednoduchém grafu.

Hlavním cílem bakalářské práce bylo aplikování a následné porovnání algoritmů na různě veliké sítě. Porovnání probíhalo na základě výpočetního času, hodnoty výsledné modularity a počtu nalezených komunit. Tři z pěti algoritmů nedokázaly v časovém limitu najít struktury sítí. U jedné sítě byla podrobněji ukázána struktura této sítě s výslednými komunitami. V sítích s velkým počtem dopředu známých komunit všechny algoritmy selhaly. U největší sítě byl rozdíl o více než 250 tisíc komunit v nejlepším případě.

Algoritmy usnadňují společnosti detekovat struktury v síti. Tato detekce komunit je podstatná pro pochopení vztahů v síti a následném sledování dynamiky sítě. Tento fakt je velmi důležitý pro další odvětví např. ve výzkumech, v neurobiologii, v marketingu, v geografii atd. Jsou schopny relativně rychle najít strukturu sítě ve velkém měřítku.

Detekce komunit v sociálních sítích je v současné době velmi výpočetně náročná. Algoritmy jako takové mají určitě své místo v blízké budoucnosti. Samozřejmě mají své negativní stránky a budou se muset neustále zlepšovat a zrychlovat. Mnoho algoritmů představené v této práci jsou již nyní pomalé. I sítě se budou neustále zvětšovat, např. Facebook se svými 1,4 miliardy uživatelů nebo Google, který již nyní indexuje přes miliardy stránek. Je jen otázkou času, kdy nebudou stačit současné algoritmy a bude zapotřebí přijít s novými a rychlejšími algoritmy pro detekci komunit.

7 Seznam použité literatury

[1] NEWMAN, M. E. J. Modularity and community structure in networks. Proceedings of the National Academy of Sciences [online]. 2006-06-06, vol. 103, issue 23, s. 8577-8582 [cit. 2014-06-19]. DOI: 10.1073/pnas.0601602103. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.0601602103

[2] M.E.J, Newman; M.Girvan (2004). "Finding and evaluating community structure in networks". Phys. Rev. E 69 (2).Fjällström P.-O. Linköping Electronic Articles in Computer and Information Science. 1998. Dostupné z: www.ep.liu.se/ea/cis/1998/006.

[3] Remco van der Hofstad (2013). "7". Random Graphs and Complex Networks.

[4] Weisstein, Eric W., "Laplacian Matrix", MathWorld.

[5] BALDWIN, Carliss Y a Kim B CLARK. Design rules. Cambridge, Mass.: MIT Press, c2000-, v. <1 >. ISBN 02-620-2466-7.

[6] Rumelhart, D.E., J.L. McClelland and the PDP Research Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, Cambridge, MA: MIT PressGuimerà R., Amaral L. A. N. Nature. 2005;433:895–900.

[7] SPIELMAN, Daniel,Spectral Graph Theory [online]. s. 30 [cit. 2014-08-28] Dostupné z: http://www.cs.yale.edu/homes/spielman/PAPERS/SGTChapter.pdf

[8] Santo Fortunato and Marc Barthelemy (2007). "Resolution limit in community detection". Proceedings of the National Academy of Sciences of the United States of America 104 (1): 36–41. arXiv:physics/0607100. Bibcode:2007PNAS..104...36F. doi:10.1073/pnas.0605965104. PMC 1765466. PMID 17190818

[9] Colin de Verdière graph invariant. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-08-28]. Dostupné z: http://en.wikipedia.org/wiki/Colin_de_Verdi%C3%A8re_graph_invariant

[10] Girvan M., Newman M. E. J. Proc. Natl. Acad. Sci. USA. 2002;99:7821–7826

[11] PATI, Sukanta. Laplacian matrix of a graph. Matrix Information Geometry, 30.Dostupne z: http://www.lix.polytechnique.fr/~schwander/resources/mig/slides/pati.pdf

[12] GIRVAN, M. a M. E. J. NEWMAN. Community structure in social and biological networks. Proceedings of the National Academy of Sciences [online]. 2002-06-11, vol. 99, issue 12, s. 7821-7826 [cit. 2015-01-09]. DOI: 10.1073/pnas.122653799. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.122653799.

[13] FREEMAN, Linton C. a M. E. J. NEWMAN. A Set of Measures of Centrality Based on Betweenness. Sociometry [online]. 1977, vol. 40, issue 1, s. 35- [cit. 2015-01-09]. DOI: 10.2307/3033543. Dostupné z: http://www.jstor.org/stable/3033543?origin=crossref Weisstein, Eric W., "Laplacian Matrix", MathWorld.

[14] M. E. J. Newman, Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality. Phys. Rev. E 64, 016132 (2001)

[15] CLAUSET, Aaron, M. NEWMAN a Cristopher MOORE. Finding community structure in very large networks. Physical Review E [online]. 2004, vol. 70, issue 6, s. - [cit. 2015-01-09]. DOI: 10.1103/PhysRevE.70.066111. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.70.066111

[16] NEWMAN, M. Fast algorithm for detecting community structure in networks. Physical Review E [online]. 2004, vol. 69, issue 6, s. - [cit. 2015-01-09]. DOI: 10.1103/PhysRevE.69.066133. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.69.066133

[17] BLONDEL, Vincent D, Jean-Loup GUILLAUME, Renaud LAMBIOTTE a Etienne LEFEBVRE. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment [online]. 2008-10-01, vol. 2008, issue 10, P10008- [cit. 2015-01-09]. DOI: 10.1088/1742-5468/2008/10/P10008. Dostupné z: http://stacks.iop.org/1742-5468/2008/i=10/a=P10008?key=crossref.46968f6ec61eb8f907a760be1c5ace52

[18] NEWMAN, M. E. J., Jean-Loup GUILLAUME, Renaud LAMBIOTTE a Etienne LEFEBVRE. Modularity and community structure in networks. Proceedings of the National Academy of Sciences [online]. 2006-06-06, vol. 103, issue 23, s. 8577-8582 [cit. 2015-01-09]. DOI: 10.1073/pnas.0601602103. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.0601602103

[19] Kernighan–Lin algorithm. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-01-09]. Dostupné z: http://en.wikipedia.org/wiki/Kernighan%E2%80%93Lin_algorithm

[20] RAGHAVAN, Usha Nandini, Réka ALBERT a Soundar KUMARA. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* [online]. 2007, vol. 76, issue 3 [cit. 2015-04-04]. DOI: 10.1103/physreve.76.036106.

[21] Stanford Large Network Dataset Collection. LESKOVEC, Jure a Andrej KREVL. [online]. 2014 [cit. 2015-04-04]. Dostupné z: http://snap.stanford.edu/data/

[22] Shizuka lab. SHIZUKA, Daizaburo. [online]. 2. 2. 2013 [cit. 2015-04-04]. Dostupné z: http://www.shizukalab.com/toolkits/sna/sna_data

FIM UHK
UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **Jaromír Homolka**

Obor studia: Aplikovaná informatika
Jméno a příjmení vedoucího práce: **Jiří Haviger**

Název práce:
Detekce komunit v sociálních sítích

Název práce v AJ:
Community detection in social networks

Podtitul práce:

Podtitul práce v AJ:

Cíl práce: Cílem této bakalářské práce je přiblížení metod detekce komunit v sociálních sítích a ukázky jejich použití v jazyce R.

Osnova práce:
1. Úvod
2. Teoretická východiska
3. Popis metod pro detekci komunit v jazyku R
4. Testování algoritmů na různě velkých sítích
5. Závěr

Projednáno dne: 31. 9. 2014

Podpis studenta

Podpis vedoucího práce