Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Detekce komunit v sociálních sítích

Bakalářská práce

Autor: Jakub Minařík

Studijní obor: Aplikovaná informatika

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

Hradec Králové duben 2016

Prohlášení:

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

V Hradci Králové dne 28. 4. 2016 Jakub Minařík

Poděkování:

Děkuji vedoucímu práce Mgr. Jiřímu Havigerovi, Ph.D. za metodické vedení práce, pomoc a cenné rady při zpracování této práce. Dále děkuji své přítelkyni a rodině za vytvoření vhodných podmínek pro práci.

Anotace

Bakalářská práce se zabývá způsoby detekce komunit v sociálních sítích. Detekce komunit v sítích představuje zpravidla výpočetně náročný problém, který závisí na velikosti sítě. V práci jsou vysvětleny základní pojmy, týkající se dané problematiky, včetně pojmu modularita. Následně je vysvětleno 10 algoritmů, které detekují komunitní strukturu sítě. U sedmi z těchto algoritmů je uvedena i názorná ukázka aplikace na jednoduchý graf. V následující části práce jsou představeny sítě, na kterých proběhne testování algoritmů. V závěrečné části je otestováno sedm algoritmů na reálných sociálních sítích. Kvůli zastaralosti knihovny JUNG bylo nutno využít jinou alternativu. Jako tato alternativa byla zvolena knihovna iGraph. V závěru práce jsou shrnuty výsledky a zhodnocena využitelnost a přesnost testovaných algoritmů.

Annotation

Title: Community detection in social networks

The bachelor thesis is focused on community structure detection in social networks. This is usually a computationally demanding task highly dependent on the size of tested network. In the first part, basic concepts are explained, including the modularity. In the next part, 10 community structure detection algorithms are explained. For seven of these algorithms, application on a simple graph is shown. In the next part, tested social networks are introduced. In the final part, seven algorithms are tested on real social network data. Because of obsolescence of JUNG library, an alternative is chosen in the form of iGraph library. The thesis is closed with summary of test results.

Obsah

1. Úvod

Studium sítí má sice dlouhou historii, ale zejména od přelomu tisíciletí si lze povšimnout zvyšujícího se zájmu o toto téma. To je zřejmě zapříčiněno mimo jiné tím, že dnes jsou k dispozici přesné informace o sítích obrovského rozsahu. Jejich analýza pak přináší některé zajímavé poznatky. Mnoho sítí, ať už se jedná o sítě sociální, počítačové, biologické či jiné, se vyznačuje sadou společných rysů, vlastností. Jednou z těchto vlastností je to, že vrcholy v těchto sítích se často shlukují do úzce propojených částí sítě, které jsou řídce propojené se zbytkem sítě.

Takovéto shluky se nazývají komunity a mají zpravidla velký význam pro pochopení celé sítě. Jednotlivé komunity v síti často např. plní určitou funkci. V rámci komunit se zpravidla rychle šíří informace. Detekce komunit se stala cílem výzkumu a bylo představeno velké množství algoritmů, které jsou komunitní strukturu sítě schopny detekovat. Liší se náročností, přesností i principem fungování.

V první části práce jsou představeny základní pojmy z teorie grafů. Dále je popsán pojem komunita a modularita. Jedná se o velice důležité pojmy, které se v práci objevují mnohokrát, a pro dobré pochopení celé práce je nezbytné jim porozumět.

V následující části je popsáno deset algoritmů, které detekují komunitní strukturu. Algoritmy byly vybrány s ohledem na to, aby bylo představeno široké spektrum možných přístupů k řešení zkoumané problematiky. Každý algoritmus je detailně popsán a vysvětlen. U sedmi algoritmů je navíc uvedena názorná ukázka na jednoduchém grafu.

V další části jsou představeny reálné sociální sítě, na kterých jsou algoritmy testovány. Sítě jsou podrobně popsány z hlediska jejich vlastností i významu.

V závěrečné části jsou uvedeny výsledky samotného testování algoritmů na těchto sítích. Knihovna JUNG se neukázala jako vhodný nástroj a místo ní byla zvolena alternativa ve formě knihovny iGraph. S pomocí této knihovny je sedm z deseti dříve popsaných algoritmů testováno na uvedených sociálních sítích. Závěrem jsou zhodnoceny naměřené výsledky.

2. Teoretické pozadí

2.1. Vymezení základních pojmů

Graf je základní prvek v teorii grafů. Jedná se o určitou abstrakci. Síť (nehledě na to, o jakou se jedná) lze převést na graf, který se skládá z vrcholů a hran (viz dále), které je spojují. Je při tom zachována topologie, ale abstrahuje se od ostatních detailů. Formálně lze graf považovat za uspořádanou dvojicí množiny vrcholů V a množiny hran E. Například síť uživatelů sociální sítě Facebook může být reprezentována grafem, jehož vrcholy představují jednotlivé uživatele a hrany přátelství mezi nimi (pokud mezi vrcholy existuje hrana, znamená to, že jsou odpovídající uživatelé přáteli). Značení je následující: graf $G = (V, E)$ s $n = |V|$ vrcholy a $m = |E|$ hranami.

Vrchol neboli uzel, je zpravidla reprezentace určitého objektu. Pokud je vrchol spojený s hranou, říkáme, že s ní inciduje. Vrcholy, které neincidují s žádnou hranou, se nazývají izolované vrcholy.

Hrana zpravidla znázorňuje interakci mezi vrcholy. Nejčastěji proudění informací. Může být orientovaná, to zpravidla znamená, že informace mohou proudit pouze jedním směrem a je v takovém případě nezbytné vrcholy zaznamenávat ve formě uspořádaných dvojic. Hrana, která vede z uzlu do toho samého uzlu, je nazývána smyčkou. Každý ze dvou konců hrany musí vést do nějakého uzlu.

Stupeň vrcholu je počet hran, které s ním incidují. Smyčka zvyšuje stupeň vrcholu o dva.

Sousední vrchol je takový, který je s daným vrcholem spojen hranou. Každý vrchol může mít vícero sousedních vrcholů. Všechny tyto vrcholy pak tvoří sousedství vrcholu.

Ohodnocená hrana je taková hrana, které kromě informace o vrcholech, které spojuje, popřípadě pořadí těchto vrcholů, nese také informaci o své hodnotě. Ve výše uvedeném příkladu by se mohlo jednat například o dobu trvání přátelství v letech. Ohodnocené mohou být i vrcholy, což ale pro účely této práce není tak důležité.

Podgrafu je graf, vzniklý odebráním některých vrcholů nebo hran původního grafu.

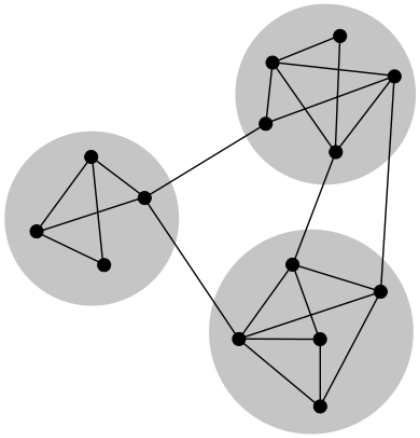
Úplný graf je graf, ve kterém každý vrchol sousedí se všemi ostatními. Značí se K_n, kde n je počet vrcholů.

Multigraf je graf, ve kterém mezi některou dvojicí vrcholů existuje více hran. V takovém grafu se mohou vyskytovat paralelní (rovnoběžné) hrany, což jsou hrany, které spojují stejnou dvojici vrcholů.

Hypergraf je takový graf, jehož hrany mohou spojovat libovolný počet vrcholů.

2.2. Komunita

V práci se velice často setkáte s pojmem komunita, je proto důležité se co nejdříve seznámit s jeho definicí. V rámci jedné komunity jsou vrcholy mezi sebou pospojovány hustěji než mezi jednotlivými komunitami. Komunitu tedy lze definovat jako množinu vrcholů, které jsou hustě propojeny mezi sebou, ale řídce propojeny se zbytkem sítě (viz obr. č. 1). Univerzální definice komunity zatím chybí, což představuje pro jejich detekci značný problém. Různé algoritmy proto pracují s různými definicemi komunit, některé jsou uvedeny v následujících kapitolách práce. Většina velkých sítí projevuje jistou úroveň komunitní struktury, přičemž každá komunita v síti hraje určitou roli, má nějaký význam a určitým způsobem se podílí na funkci celé sítě.



Obrázek 1: Jednoduchá síť rozdělená do tří komunit. Zdroj: [1]

Přesněji lze komunitu popsat (tato definice je obecně uznávaná za prozatím nejpřesnější) tak, že vrcholy se rozdělí do skupin a určí se pravděpodobnosti p_{in}

(pravděpodobnost, že vybraný vrchol bude propojen s vrcholy v dané skupině) a p_{out} (pravděpodobnost, že vrchol bude propojen s vrcholy v jiné skupině). Pokud platí výraz $p_{in} > p_{out}$, pak lze dané skupiny považovat za komunity. S tím úzce souvisí takzvaný GN-benchmark, což je metoda pro testování algoritmů pro detekci komunitní struktury, kterou představili Girvan a Newman [2]. Jedná se o metodu, kdy je uměle vytvořen graf s danými pravděpodobnostmi a na něm jsou pak algoritmy testovány.

Většina sítí se projevuje jistou komunitní strukturou. Detekce komunit může mít značný význam pro pochopení funkce celé sítě a její analýzu. Například hustě propojená sociální skupina (odsud pojem komunita) se bude vyznačovat rychlejším šířením informací.

Jistým problémem pro značnou část algoritmů pro detekci komunitní struktury v sítích je to, že jednotlivé komunity mohou být velice odlišné. Mohou se lišit počtem vrcholů, hustotou hran mezi vrcholy i hustotou hran mezi jednotlivými komunitami. A to zejména v sítích velkého rozsahu. Dalším problémem může být hierarchičnost komunit, jinými slovy výskyt komunit v rámci jiné komunity.

2.3. Základní rozdělení metod pro detekci komunit

Pro detekci komunitní struktury bylo představeno značné množství postupů a algoritmů. Dodnes však není jednoznačně určeno, které postupy jsou nejspolehlivější, protože stále chybí přesně definovaná struktura komunity.

Metody detekce komunit v grafech lze rozdělit na dva základní principiální směry. Jsou jimi dělení grafu (graph partitioning), které nachází využití zejména ve výpočetní technice, a detekce komunitní struktury (community structure detection), která se využívá zejména při analýze sociálních skupin.

2.3.1. Dělení grafu

Metoda často využívaná ve výpočetní technice (např. integrované obvody a paralelní zpracování dat). Typicky řeší problém rozdělení úloh mezi jednotlivé procesory tak, aby komunikace mezi nimi byla omezena na minimum. V takovém případě je znám počet procesorů i přibližný počet jim přidělených úkolů. Známe tedy předem alespoň přibližně počet a velikost komunit, do kterých bude síť rozdělena.

Jedná se o historicky jeden z nejstarších algoritmů pro dělení sítě do skupin a nese anglický název minimum-cut. Slouží k rozdělení sítě do předem známého počtu skupin vrcholů tak, aby počet hran mezi jednotlivými skupinami byl minimální. Dalším požadavkem pro správné fungování tohoto postupu je to, aby jednotlivé skupiny vrcholů měly přibližně stejnou velikost.

Počet komunit musí být předem známý z toho důvodu, že kdyby nebyl implicitně definovaný, došlo by k tomu, že by všechny vrcholy byly přidány pouze do jedné skupiny. Počet hran mezi komunitami by pak sice byl minimální (nula), ale takové řešení nemá žádnou vypovídající hodnotu. To je důvod, proč je nezbytné implicitně specifikovat počet výsledných skupin. Ke stejnému výsledku by se došlo i tehdy, když by nebyla specifikována velikost jednotlivých skupin.

Z principu fungování těchto metod je tedy zřejmé, že nejsou schopny identifikovat komunity v síti, o které předem nemáme téměř žádné informace. Z toho důvodu se stávají pro tuto práci nedůležitými. [1] [2]

2.3.2. Detekce komunitní struktury

Nachází aplikaci zejména v sítích o velkém rozsahu, jako jsou např. sociální, biologické, nebo webové sítě. Předpokládá se, že síť je přirozeně rozdělena do komunit a naším cílem je jejich detekce. Jejich velikost a počet jsou tedy dány samotnou sítí a nám jde pouze o jejich nalezení. Existuje velké množství algoritmů, které se liší výpočetní náročností i přesností. Jednotlivé algoritmy jsou popsány v následujících kapitolách této práce. Mezi základní metody patří blockmodeling, hierarchické shlukování (hierarchical clustering) a metody založené na sousednosti (betweenness-based methods).

2.4. Modularita

Modularita je jedním z historicky nejvýznamnějších pojmů ve studované oblasti. Pojem specifikoval v roce 2003 [3] a dále v roce 2006 Newman [1], na základě této práce zde bude pojem vysvětlen. Obecně se modularita používá ke zjištění, zda nějaká změna v síti (zpravidla přidání nebo odstranění hrany) povede ke zvýšení komunitní struktury. Mezi jeden ze způsobů detekce komunitní struktury patří metoda maximalizace modularity. V rámci této práce se jedná o stěžejní pojem, jeho princip tak bude detailně rozebrán.

Předpokládejme, že známe strukturu nějaké sítě a chceme zjistit, zda je přirozeně rozdělena do jakkoliv velkých komunit, které se navzájem nepřekrývají. Pro správné určení komunitní struktury sítě nestačí nalézt řešení, ve kterém je nejméně hran mezi komunitami, ale ve kterém je těchto hran méně než by se dalo předpokládat. Pokud mezi dvěma skupinami vrcholů existuje právě takové množství hran, jaké by se dalo statisticky předpokládat, těžko zde mohou být vyvozeny závěry o komunitní struktuře. Hledají se naopak případy, kdy tomu tak není. A právě myšlenka, že komunitní struktura sítě koresponduje se statisticky neočekávaným rozmístěním vrcholů, může být vyčíslena pomocí míry zvané modularita.

Modularitu lze vyjádřit jako rozdíl počtu hran, které patří do daných skupin a očekávaného počtu hran v ekvivalentní síti s náhodně rozmístěnými hranami. Nabývá hodnot od -1/2 do 1. Má kladnou hodnotu, když počet hran v daných skupinách je větší než statisticky očekávaný počet hran. Pro dané rozdělení sítě do komunit nám pak modularita udává koncentraci hran v rámci těchto komunit v porovnání s koncentrací hran náhodně rozmístěných nezávisle na komunitách.

Uvažujme graf o n vrcholech takový, že může být rozdělen do dvou komunit pomocí hodnoty proměnné s. Pokud vrchol i patří do komunity 1, pak $S_i = 1$, pokud patří do komunity 2, pak $S_i = -1$. Existence hran mezi vrcholy i a j je zaznamenána v A_{ij} (což odpovídá matici sousednosti). Současně předpokládaný počet hran mezi vrcholy i a j při náhodném rozmístění hran je $k_i k_j/2m$, kde k_i a k_j jsou stupně vrcholů a $m = \frac{1}{2}\sum_i k_i$ je celkový počet hran v grafu. Modularita Q je pak dána jako suma $A_{ij} - k_i k_j/2m$ všech páru vrcholů i a j, které patří do stejné skupiny.

Jelikož hodnota výrazu ½($s_i s_j$+1) se rovná jedné, pokud vrcholy i a j patří do stejné skupiny a nule, když tomu tak není, můžeme modularitu vyjádřit následujícím vztahem.

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

Rovnice 1: modularita, zdroj: [1].

Tuto rovnici můžeme přepsat do maticové podoby následovně.

$$Q = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s},$$

Rovnice 2: maticové vyjádření modularity, zdroj: [1].

Platí, že *s* je sloupcový vektor, jehož hodnoty jsou právě s_i a definovali jsme symetrickou matici *B* následovně.

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

Rovnice 3: matice modularity, zdroj: [1].

Tuto matici nazýváme *matice modularity*. Součet hodnot v každém řádku a sloupci této matice je nulový, takže její vlastní vektor příslušný k vlastnímu číslu nula má tvar (1,1,1,…,1).

S můžeme dále rozepsat jako lineární kombinaci normalizovaných vlastních vektorů $u_{i,}$, takže $s=\sum_i=1^n a_i u_i$, kde $a_i=u_i^T s$. Můžeme psát:

$$Q = \frac{1}{4m} \sum_i a_i \mathbf{u}_i^T \mathbf{B} \sum_j a_j \mathbf{u}_j = \frac{1}{4m} \sum_{i=1}^{n} (\mathbf{u}_i^T \cdot \mathbf{s})^2 \beta_i$$

Rovnice 4: modularita a vlastní čísla, zdroj: [1].

kde β_i je vlastní číslo matice *B* příslušné k vlastnímu vektoru u_i. S těmito hodnotami pracuje zejména Leading-eigenvector algoritmus, který je popsán v následujících kapitolách. Vlastní číslo matice *A* je takové číslo, pro které existuje nenulový vektor

takový, že platí $A \cdot v = \lambda \cdot v$. Vektor *v* je v takovém případě vlastním vektorem matice *A*, který přísluší vlastnímu číslu λ.

2.4.1. Modularita a malé komunity

V sítích o velkém rozsahu ovšem metoda optimalizace modularity selhává z důvodu její neschopnosti rozlišit komunity, které mají výrazně menší velikost, než je velikost celé sítě. Zvětšení počtu komunit totiž nemusí nezbytně znamenat zvětšení celkové modularity.

Důvodem je to, že definice modularity vychází z předpokladu, že je statisticky možné, aby byl každý vrchol propojen s jakýmkoli jiným vrcholem v sítí. Tento předpoklad je ale v sítích o velkém rozsahu mylný. Stejně tak definice modularity pracuje s předpokladem, že očekávaný počet hran mezi komunitami se zmenšuje se zvětšující se sítí. Při dostatečně velké sítí tak předpokládaný počet hran mezi komunitami může být menší než jedna. V takovém případě by i jediná hrana mezi komunitami znamenala, že nehledě na vlastnosti těchto komunit, by jejich spojení celkovou modularitu zvětšilo. Metoda optimalizace modularity, vycházející z obecného null modelu, tak není schopna detekovat ani úplné grafy (jakožto nejsnáze identifikovatelné komunity) v rámci dostatečně rozsáhle sítě.

2.4.2. Null model

Null model je v obecném slova smyslu graf, který si ponechává určité vlastnosti daného grafu, ale současně se jedná o graf náhodný. Využívá se ke zjišťování, zda zkoumaný graf vykazuje určité vlastnosti či nikoliv. V případě modularity se vychází z null modelu navrženého Newmanem a Girvanem [4], který zachovává stupně jednotlivých vrcholů.

3. Algoritmy pro detekci komunit

Existuje velké množství algoritmů pro detekci komunit v sítích, které se liší principem, složitostí i vhodností pro použití v rozsáhlých sítích. V práci budou rozebrány nejdůležitější algoritmy ze všech těchto hledisek s přihlédnutím k jejich vhodnosti použití v sociálních sítích. U algoritmů, které budou v další části práce testovány, je uveden i konkrétní příklad aplikace na grafu karate klubu Zachary. Algoritmy byly vybrány s cílem popsat co nejširší spektrum možných přístupů k řešení dané problematiky.

3.1. Leading-eigenvector algoritmus

Modularita a potřebné proměnné byly popsány v kapitole 2.3. Tento algoritmus byl představen Newmanem v roce 2006 [1]. Na základě této práce bude algoritmus dále popsán. Jedná se o algoritmus, který těží právě z principu modularity a funguje na principu globální optimalizace její hodnoty.

Předpokládejme, že vlastní čísla matice modularity jsou seřazena sestupně, $\beta_1 \geq \beta_2 \geq \ldots \geq \beta_n$. Modularitu lze maximalizovat volbou správného rozdělení sítě nebo ekvivalentně volbou správné hodnoty s. Šlo by tedy o to, využít nejvyšší vlastní čísla. Bez dalších omezení by to znamenalo zvolit s odpovídající vlastnímu vektoru u_1.

Proměnná ale s může nabývat pouze hodnot ±1, takže je nutné maximalizovat celý výraz $u_1^T \cdot s$. Je zřejmé, že maxima je dosaženo volbou $s_i = 1$, pokud odpovídající prvek u_i je kladný a $s_i = -1$ v opačném případě. Jinými slovy všechny vrcholy, jimž odpovídají kladné prvky, jsou zařazeny do jedné skupiny a ostatní do druhé. Z toho plyne algoritmus pro rozdělení sítě: spočítáme vlastní vektor matice modularity a rozdělíme vrcholy do dvou skupin podle znamének jednotlivých prvků tohoto vektoru.

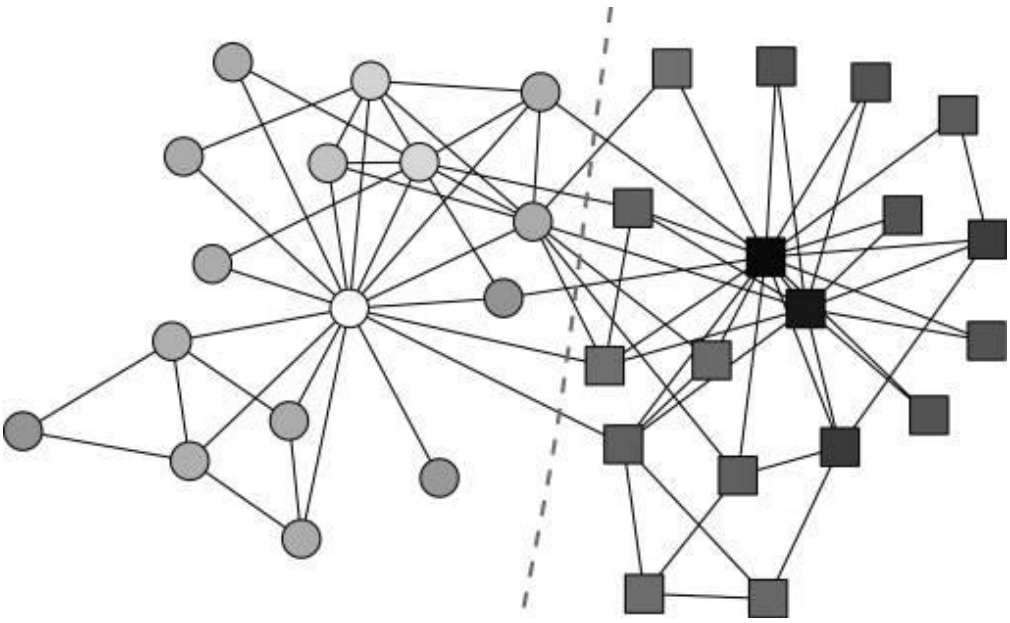
Může se stát, že neexistují žádná kladná vlastní čísla matice modularity. Pak vlastní vektor má tvar (1,1,…,1). To by znamenalo, že všechny vrcholy budou přiřazeny postejné skupiny, což je ale v tomto případě v pořádku. Z takového výsledku totiž plyne, že neexistuje žádné rozdělení vrcholů takové, aby modularita nabývala kladné hodnoty. Modularita takto nerozdělené sítě pak má hodnotu nula. V případě, že

není možné vrcholy efektivně rozdělit do dvou skupin, tedy tento algoritmus žádné rozdělení neprovede, což je žádaný postup.

Roli ale nehrají pouze znaménka prvků vlastního vektoru, ale také jejich velikosti. Vrcholy odpovídající prvkům s velkou hodnotou mají velký vliv na modularitu. Tato hodnota nám v podstatě vyjadřuje, jak velký vliv na modularitu by mělo přesunutí jednotlivých vrcholů z jedné skupiny do druhé. Vrcholy, u kterých je tato hodnota nejvyšší, jsou centrální vrcholy daných skupin.

3.1.1. Klub Zachary

Funkci algoritmu můžeme demonstrovat na asi nejznámějším příkladu, který se stal jakýmsi standardem pro testování algoritmů pro detekci komunit. Jedná se o graf, který znázorňuje přátelství mezi 34 členy karate klubu Zachary, zachycený na americké univerzitě v roce 1970.



Obrázek 2: Klub Zachary. Zdroj:[1]

Krátce po zachycení této sociální struktury se klub rozpadl na dva nezávislé kluby z důvodu vnitřních rozporů. Tvary vrcholů značí zachycená přátelství a tečkovaná čára rozdělení grafu výše zmiňovaným algoritmem. Z grafu je zřejmé, že tato dvě rozdělení jsou v naprosté shodě.

Odstíny vrcholů odpovídají velikostem prvků vlastního vektoru a zde se opět potvrzuje, že algoritmus správně reflektuje realitu. Zejména 3 vrcholy s největší vahou odpovídají členům, kteří se stali vedoucími nově vzniklých klubů.

3.1.2. Rozdělení sítě do více než dvou komunit

Mnoho sítí je přirozeně rozděleno do více než dvou komunit a proto je žádoucí schopnost nalézt rozdělení sítě na vícero částí. Asi nejčastěji používaným postupem v tomto případě je opakované dělení sítě na dvě části.

Není ovšem správné po prvním rozdělení jednoduše vypustit hrany spojující jednotlivé skupiny a aplikovat algoritmus znovu na každý podgraf, protože by tím došlo ke změně stupňů vrcholů. Správný postup zde je výpočet ΔQ, což je změna modularity při dalším dělení sítě, a snaha maximalizovat tento rozdíl. ΔQ nám tedy značí změnu modularity rozdělením skupiny g o velikosti n_g na dvě skupiny pomocí následujících výpočtů.

$$\begin{aligned}\Delta Q &= \frac{1}{2m}\left[\frac{1}{2}\sum_{i,j\in g} B_{ij}(s_i s_j + 1) - \sum_{i,j\in g} B_{ij}\right] \\ &= \frac{1}{4m}\left[\sum_{i,j\in g} B_{ij}s_i s_j - \sum_{i,j\in g} B_{ij}\right] \\ &= \frac{1}{4m}\sum_{i,j\in g}\left[B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}\right]s_i s_j \\ &= \frac{1}{4m}\,\mathbf{s}^T\mathbf{B}^{(g)}\mathbf{s},\end{aligned}$$

Rovnice 5: změna modularity při dělení grafu, zdroj: [1].

kde δ_{ij} je je Kroneckerovo delta, využilo se toho, že $s_1^2 = 1$ a $B_{(g)}$ je matice $ng \times ng$ s prvky o indexech i,j vrcholů patřících do skupiny g o hodnotách

$$B_{ij}^{(g)} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}.$$

Rovnice 6: matice modularity při opakovaném dělení sítě, zdroj: [1].

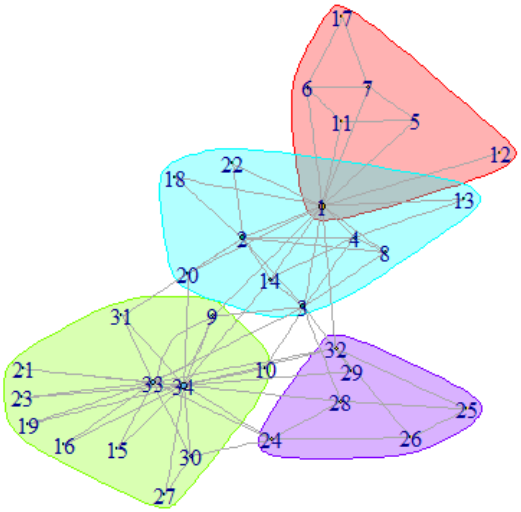
Je také důležité rozhodnout, kdy ukončit dělení na podsítě. Tento postup dokáže toto rozhodnutí učinit díky tomu, že lze poznat, kdy už nedochází ke zvyšování modularity dalším dělením, neboli ΔQ není kladné. Dostačující (nikoliv však nezbytnou) podmínkou také je, když neexistují žádná kladná vlastní čísla matice $B^{(g)}$. Stačí tedy ověřit, zda nejvyšší z těchto vlastních čísel je kladné a podle toho rozhodnout o ukončení procesu.

3.1.3. Shrnutí

1. Sestaví se matice modularity pomocí *rovnice 3* a určí se její největší vlastní číslo a příslušný vlastní vektor.
2. Síť se rozdělí na dvě části podle znamének prvků tohoto vektoru.
3. Body 1 a 2 se opakují pro každý vzniklý podgraf s využitím *rovnice 6*.
4. Pokud je u některého z podgrafů zjištěno, že dalším dělením nemůže dojít ke zvýšení celkové modularity, ponechá se nerozdělen.
5. Algoritmus je ukončen, pokud už žádny z podgrafů nemůže být dál rozdělen.

Z toho plyne, že komunitu je také možno definovat jako dále nedělitelný podgraf. Sestavování matice modularity je poměrně náročné, což se podepisuje na celkové náročnosti tohoto algoritmu.

3.1.4. Aplikace

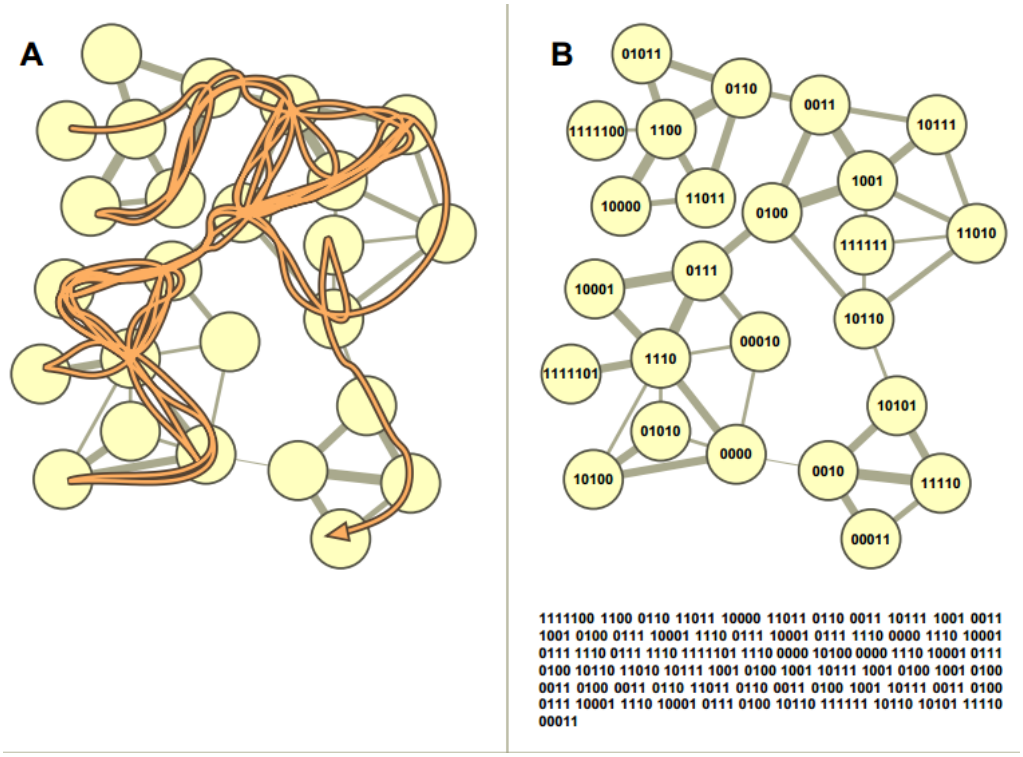


Obrázek 3: Aplikace algoritmu Leading-eigenvector na graf karate klubu Zachary. Počet komunit n=4, modularita Q= 0,391. Zdroj: autor.

3.2. Infomap algoritmus

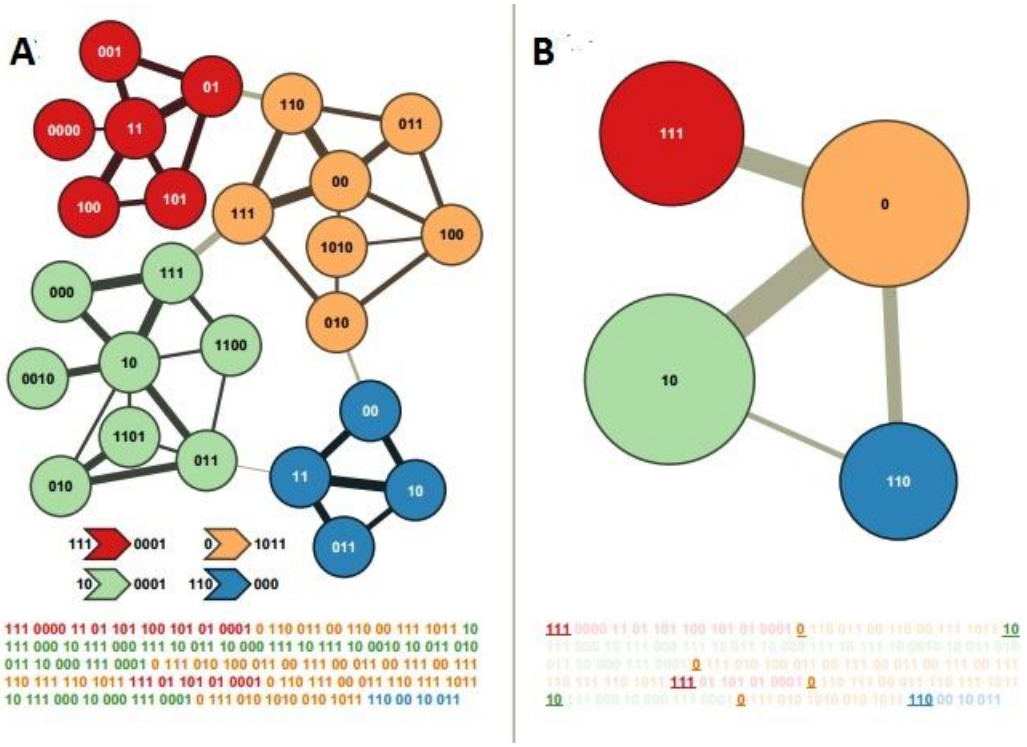
Algoritmus byl představen roku 2008 M. Rosvallem a C. T. Bergstromem [5]. Na základě této práce bude algoritmus dále popsán. Je založen na proudění informací v síti. Vychází z principu komprese informací o náhodné cestě a hodí se na ohodnocené i orientované grafy. K detekci komunit, ze kterých se síť skládá, dochází tím, že se nalezne optimálně zkomprimovaný popis informačních toků v síti. Popis informačního toku je problém z oblasti komprese/kódování. Klíčovou myšlenkou je v tomto případě to, že tok dat lze komprimovat do kódu, ze kterého je následně možné rozeznat, jak byl tento tok generován. Algoritmus využívá k simulaci toku dat náhodnou cestu. Náhodná cesta je taková, kdy je každý další krok této cesty náhodně vybrán ze sousedů aktuálního vrcholu. Algoritmus pracuje s pojmem mapa. Graf je totiž v podstatě mapa sítě; je zde abstrahováno od nepodstatných informací, zatímco ty důležité jsou zachovány a je přesně známo, jaká míra abstrakce byla zvolena (stejné principy se užívají i v kartografii). K tomuto se hodí Huffmanovo kódování, což je algoritmus, využívaný pro bezeztrátovou kompresi dat, který konvertuje znaky vstupního souboru do bitových řetězců různé délky [6]. Tato délka odpovídá četnosti

výskytu řetězců. Znaky, které se vyskytují nejčastěji, jsou reprezentovány nejkratšími řetězci, a znaky, které se vyskytují nejvíce zřídka, jsou naopak reprezentovány nejdelšími řetězci. To lze snadno převést na graf a vrcholy s různými četnostmi výskytu v náhodné cestě.



Obrázek 4a: Příklad náhodné cesty o 71 krocích v grafu s 25 vrcholy, tloušťka linky znázorňuje relativní pravděpodobnost, že náhodná cesta bude vést přes danou hranu. Obrázek 4b: Příklad Huffmanova kódování pro danou cestu. Každý vrchol má unikátní kód, jehož délka závisí na počtu přechodů potenciální nekonečně dlouhé náhodné cesty přes tento vrchol. Zdroj: [5]

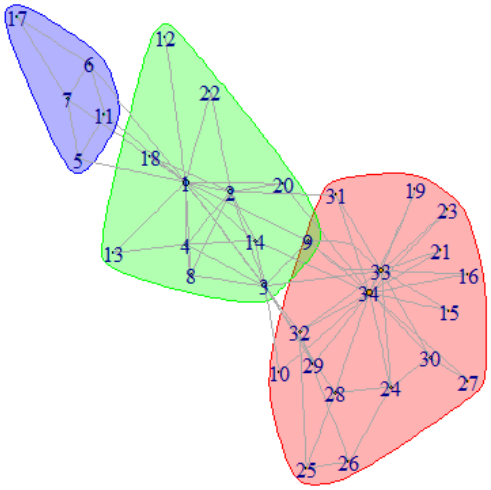
Samotné kódování vrcholů ale k tvorbě mapy - a tím pádem i odhalení struktury sítě - nestačí. K tomu je třeba další úroveň popisu, kdy se zachovají unikátní názvy pro jednotlivé komunity, ale názvy jednotlivých vrcholů lze používat opakovaně. Podobný přístup je volen i na reálných mapách. Města mají unikátní jména, ale názvy ulic se opakují. Využije se tak struktury sítě (zejména toho, že náhodná cesta se statisticky drží v určité komunitě po dlouhé časové úseky) ke snížení datové náročnosti popisujícího kódu.



Obrázek 5a: Další uroveň popisu grafu; názvy jednotlivých komunit jsou unikátní, názvy vrcholů uvnitř těchto komunit se znovupoužívají. Popisný kód je tak ještě kratší. Obrázek 5b: Zjednodušení sítě na pouhé komunity. Zdroj: [5]

Algoritmus tedy nejprve pomocí náhodné cesty určí, jaká je pravděpodobnost, že tato cesta povede přes jednotlivé vrcholy. Na základě této pravděpodobnosti je každému vrcholu přiřazen kód určité délky. Pomocí těchto kódů je možné detekovat jednotlivé komunity a tento popis tak ještě zjednodušit. Algoritmus je vhodný na ohodnocené i orientované grafy, nicméně pro velké sítě je jeho náročnost příliš vysoká.

3.2.1. Aplikace



Obrázek 6: Aplikace algoritmu Infomap na graf karate klubu Zachary.Počet komunit n=3, modularita Q= 0,403. Zdroj: autor.

3.3. Walktrap algoritmus

Algoritmus je, stejně jako předchozí, založen na náhodných cestách v grafu a představili ho roku 2005 Pascal Pons a Matthieu Latapy [7]. Na základě této práce bude algoritmus dále popsán. Vychází ze stejného předpokladu, tedy že náhodná cesta se často v komunitách „zachytí". Na rozdíl od předchozího případu se ale tento algoritmus může pyšnit poměrně nízkou náročností: v řídce propojených grafech $O(n^2 log(n))$.

Náhodná cesta v grafu představuje Markovův řetězec, což znamená, že pravděpodobnosti přechodu do následujícího stavu závisejí pouze na současném stavu, nikoliv na předchozích stavech. [8] V každém kroku náhodné cesty je tedy pravděpodobnost přechodu z vrcholu i na vrchol j dána vztahem $P_{ij}=A_{ij}/d(i)$, kde A_{ij} odpovídá matici sousednosti grafu a $d(i)$ je stupeň vrcholu i. Z toho vyplývá matice pravděpodobností přechodů náhodné cesty o délce t P^t_{ij}. Díky této matici lze o náhodné cestě učinit dvě tvrzení:

1. Pokud se délka náhodné cesty, začínající ve vrcholu *i* blíží nekonečnu, pak pravděpodobnost, že povede přes vrchol *j*, závisí pouze na stupni vrcholu *j*, nikoliv na počátečním vrcholu.
2. Poměr pravděpodobností, že náhodná cesta určité délky povede z vrcholu *i* do *j* a z vrcholu *j* do *i*, závisí pouze na stupních *d(i)* a *d(j)*. Neboli pravděpodobnost přechodu z *i* do *j* se rovná pravděpodobnosti přechodu z *j* do *i*.

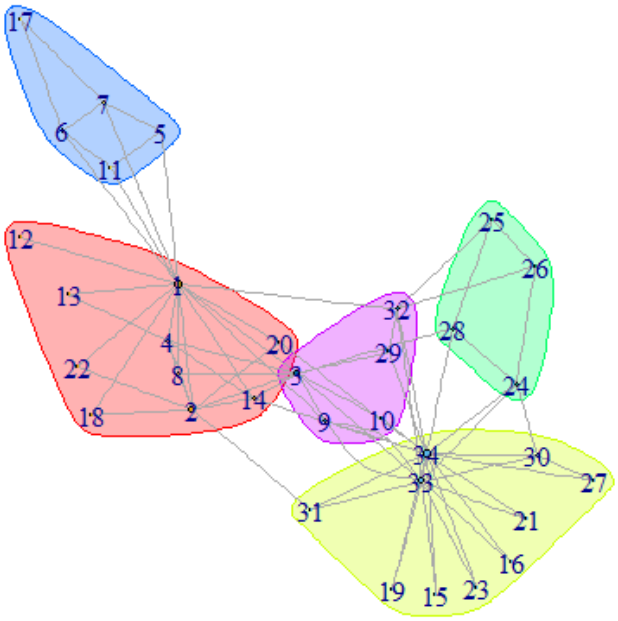
Pokud tedy dva vrcholy leží ve stejné komunitě, pravděpodobnost přechodu mezi nimi bude vysoká. Pravděpodobnosti přechodů mezi vrcholy jsou také ovlivněny jejich stupni, protože náhodná cesta častěji vede přes vrcholy, které mají vysoký stupeň. Na základě matice pravděpodobností přechodů mezi vrcholy lze stanovit způsob výpočtu vzdálenosti mezi nimi. Tato vzdálenost bude velká, pokud vrcholy leží v rozdílných komunitách.

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^{t} - P_{jk}^{t})^2}{d(k)}}$$

Rovnice 7: vzdálenost mezi vrcholy, kterou využívá Walktrap algoritmus, zdroj: [7]

Na základě této vzdálenosti je tak možné učinit závěry o komunitní struktuře sítě. Algoritmus přináší poměrně dobrý poměr mezi přesností a náročností.

3.3.1. Aplikace



Obrázek 7: Aplikace Walktrap algoritmu na graf karate klubu Zachary. Počet komunit n=5, modularita Q= 0,387. Zdroj: autor

3.4. Edge-betweenness algoritmus

Algoritmus byl představen M. Girvanem a M. E. J. Newmanem v roce 2001 [2]. Na základě této práce bude algoritmus dále popsán. Někdy se také nazývá GN algoritmus. Jedná se o vůbec první moderní algoritmus detekce komunitní struktury. Funguje na principu postupného odstraňování hran, které leží mezi komunitami. Navzdory poměrně snadno uchopitelnému principu je však algoritmus výpočetně náročný a sám o sobě nepřináší čitelné výsledky. Z grafu jsou postupně odstraňovány hrany, které leží mezi komunitami, což ve svém důsledku vede k detekci těchto komunit.

Algoritmus souvisí s myšlenkou, že informace v grafu proudí nejkratšími cestami. Vliv vrcholu na takovéto proudění informací v grafu je možné vyčíslit počtem nejkratších cest mezi ostatními vrcholy v grafu, které daným vrcholem procházejí. Problematikou centrality vrcholu se zabýval Linton C. Freeman již v roce 1977 [9]. Právě on ve své práci definoval tuto míru, zvanou edge-betweenness, což lze přeložit jako hodnotu „mezi”, v tomto případě bude ale vhodnější zůstat u anglické

terminologie. Hodnota edge-betweenness vrcholu *i* je definována jako počet nejkratších cest mezi páry ostatních vrcholů v grafu, které procházejí vrcholem *i*. Pokud mezi dvojicí vrcholů existuje více než jedna nejkratší cesta, všechny tyto cesty mají v daném kontextu stejnou váhu. Pokud v grafu existují komunity, které jsou propojeny pouze malým množstvím hran, pak všechny nejkratší cesty mezi vrcholy těchto komunit, musí procházet právě po některé z těchto hran. Z toho vyplývá, že hrany mezi komunitami mají zpravidla vysokou míru edge-betweenness.

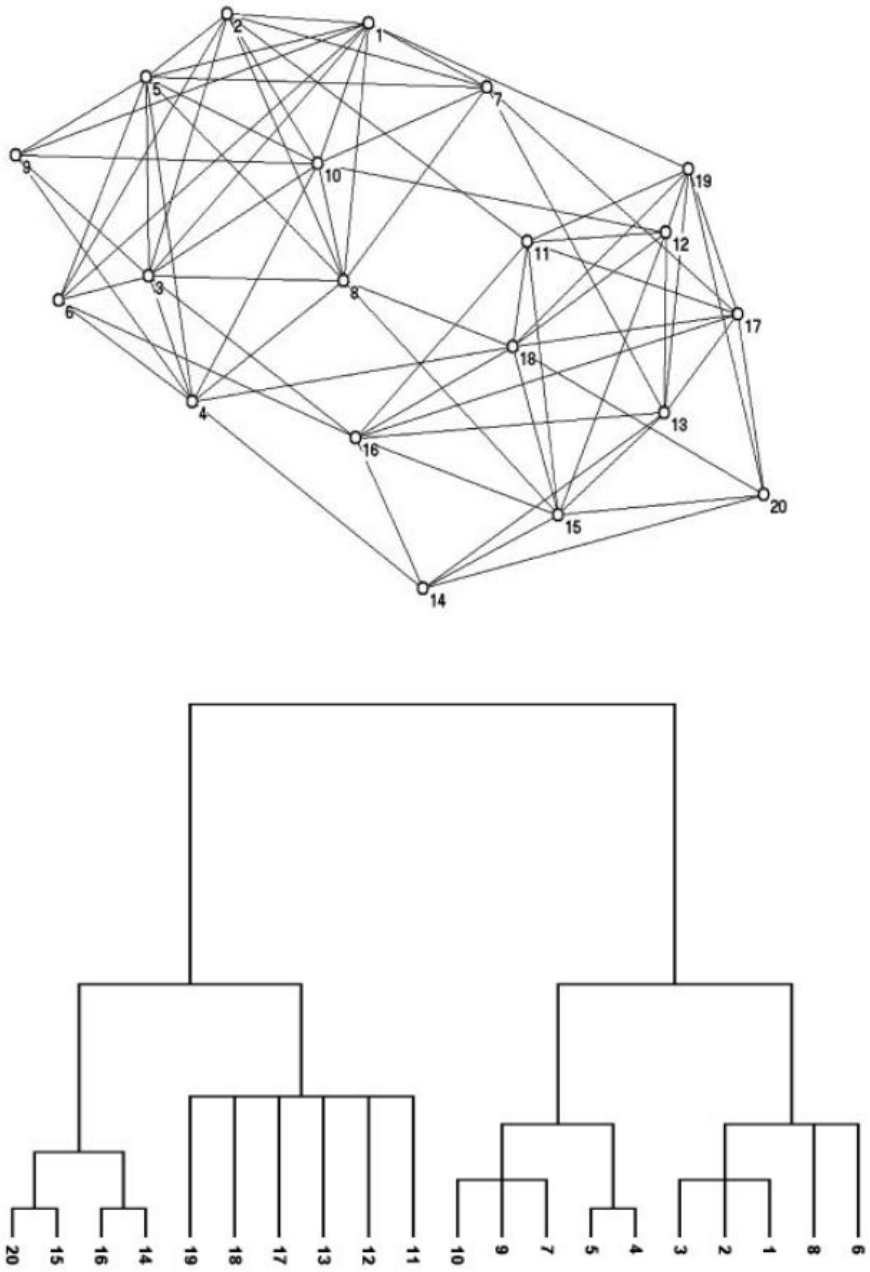
Algoritmus funguje následujícím způsobem:

1. Vypočte se hodnota betweenness pro všechny hrany v síti.
2. Odstraní se hrana s nejvyšší hodnotou.
3. Přepočte se hodnota pro všechny hrany, které byly odstraněním hrany ovlivněny.
4. Body 2 a 3 se opakují, dokud zbývají hrany.

Už z tohoto stručného popisu je zřejmé, že algoritmus bude poměrně náročný pro velké sítě. K výpočtu hodnoty betweenness se využívá Newmanovy metody [10]. Časová náročnost je pro *m* hran v grafu s *n* vrcholy $O(mn)$. Pokud algoritmus končí odstraněním všech hran, náročnost může být v nejhorším případě $O(m^2n)$, protože hodnota betweenness se přepočítává po každém odstranění hrany. Tato hodnota se ale vypočítává jen pro hrany, které byly předchozím odstraněním ovlivněny, což jsou zpravidla pouze hrany ve stejné komunitě. Z toho vyplývá, že v sítích s výraznou komunitní strukturou bude náročnost tohoto algoritmu znatelně nižší. Přepočítávat hodnotu betweenness po každém odstranění hrany je nezbytné, protože tímto odstraněním dojde ke změně celého grafu. Pokud existuje mezi komunitami více než jedna hrana, pak nelze s jistotou říct, že všechny budou mít vysokou hodnotu betweenness, ale to, že minimálně jedna z nich bude mít tuto vlastnost. Přepočítáním této hodnoty tak zajistíme, že po odstranění hrany bude vždy alespoň jedna ze zbývajících hran mít tuto vlastnost.

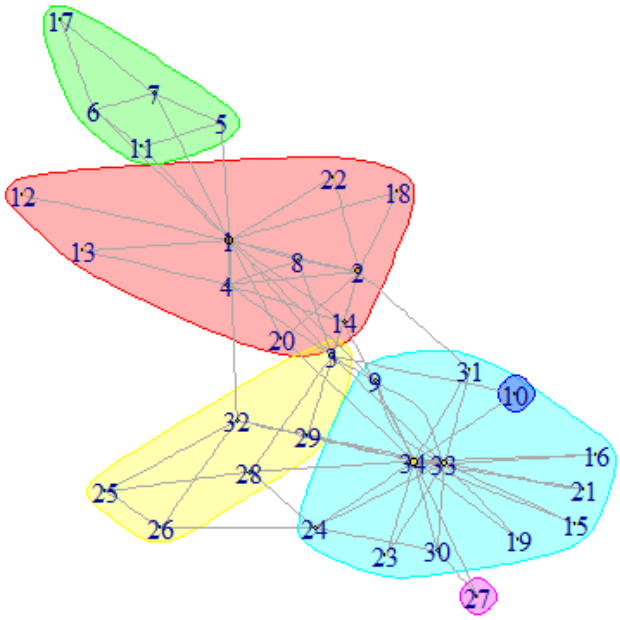
V řídce propojených sítích se složitost algoritmu blíží $O(n^3)$, což pro velké sítě činí algoritmus nepoužitelným. Dalším problémem tohoto algoritmu je jeho výstup. Algoritmus končí odstraněním všech hran, což samo o sobě nemá význam. Výsledek je tak nutné reprezentovat jiným způsobem, zpravidla formou dendogramu (viz obr. 8). Z takto vzniklého dendogramu však není jasné, zda je dané rozdělení sítě spolehlivé. K tomu by bylo vhodné použít například výpočet modularity, čímž by se však náročnost

algoritmu dále zvýšila. Algoritmus se tak stal cílem mnoha vylepšení, z nichž některá budou uvedena v následujících kapitolách.



Obr. 8: Graf o dvaceti vrcholech a jemu odpovídající dendogram, vzniklý aplikací edge-betweenness algoritmu. Z dendogramu je zřejmé, jak byly postupně vrcholy rozřazeny do komunit, nelze však určit, jak je toto rozdělení spolehlivé. Zdroj: [11]

3.4.1. Aplikace



Obrázek 9: Aplikace Edge-betweenness algoritmu na graf karate klubu Zachary. Počet komunit n=6, modularita Q= 0,385.Zdroj: autor.

3.5. Fast-greedy algoritmus

Vylepšení klasického greedy (hladového) algoritmu, představeného Newmanem roku 2003 [3]. Využívá vhodné datové struktury a představili ho roku 2004 Clauset, Newman a Moore [12]. Na základě této práce bude algoritmus dále popsán. Jedná se o jeden z algoritmů, založený na optimalizaci modularity. V původním algoritmu jsou v matici modularity ukládány i prvky s hodnotou nula, které značí, že vrcholy nejsou spojeny. Vyhledávání v takto rozsáhle matici je pak časově náročné. Algoritmus se právě tomuto jevu dokáže vyhnout implementací optimalizované verze výpočtu matice sousednosti. Výpočetní čas je touto implementací snížen v případě řídce propojených sítí na $O(n * log^2 n)$. Algoritmus ke svému fungování potřebuje tři datové struktury:

1. Matici, obsahující ΔQ_{ij}, pro každý pár vrcholů i a j. V případě uložení dat v podobě binárního stromu lze data vyhledávat a vkládat v čase $O(log\ n)$.
2. Max-haldu, obsahující největší prvek každého řádku předchozí matice, společně s odkazy na odpovídající vrcholy.

3. Pole vektorů s prvky a_i . Jedná se o podíl konců hran, které jsou připojené k vrcholům v komunitě i. Spočítá se podle následující rovnice.

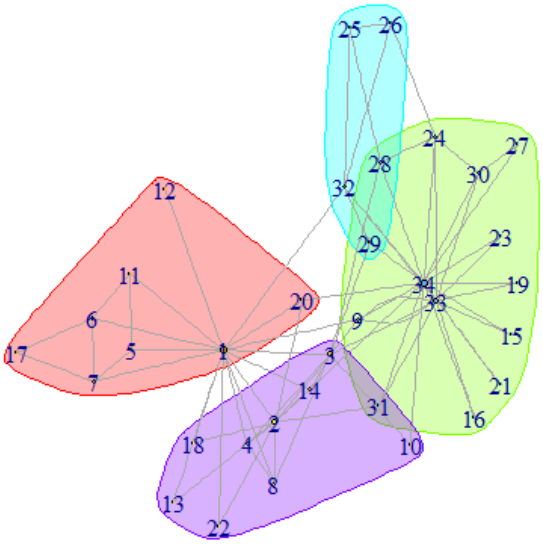
$$a_i = \frac{1}{2m} \sum_v k_v \delta(c_v, i)$$

Rovnice 8: podíl konců hran připojených ke komunitě, zdroj: [12].

Algoritmus funguje odzdola nahoru, to znamená, že se začíná na grafu, ve kterém každý jeden vrchol představuje samostatnou komunitu. Postupně jsou přidávány hrany podle toho, jak toto přidání přispěje k celkové modularitě. Algoritmus pak funguje v následujících krocích:

1. Spočítají se počáteční hodnoty ΔQ_{ij} *a* a_i a max-halda se naplní odpovídajícími daty.
2. Z max-haldy se vybere největší ΔQ_{ij}, odpovídající komunity se spojí, matice ΔQ, max-halda i a_i se odpovídajícím způsobem upraví. K celkové modularitě Q se přičte ΔQ_{ij}.
3. Krok 2 se opakuje, dokud nezbývá poslední komunita.

3.5.1. Aplikace

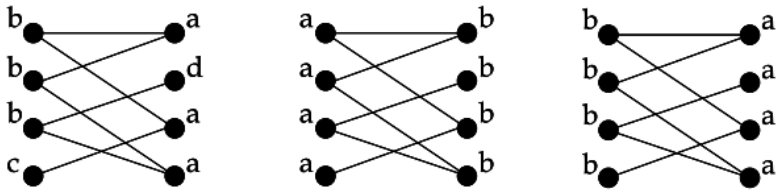


Obrázek 10: Aplikace Fast-greedy algoritmu na graf karate klubu Zachary. Počet nalezených komunit n=4, modularita Q=0,393. Zdroj: autor.

3.6. Label-propagation algoritmus

Algoritmus je velice rychlý a představili ho roku 2007 Usha Nandini Raghavan, R´eka Albert a Soundar Kumara [13]. Na základě této práce bude algoritmus dále popsán. Funguje na základě přidělování štítků jednotlivým vrcholům, respektive komunitám. Předpokládejme, že vrchol x má sousední vrcholy $x_1, x_2,..., x_k$. Každý z těchto sousedů má štítek, který značí, do jaké komunity patří. Pak vrchol x je zařazen do té samé komunity, do jaké patří největší množství jeho sousedů, v případě shody je jeho příslušnost volena náhodně. V počátečním stavu má každý vrchol unikátní štítek a postupně jednotlivé vrcholy mění štítky podle svých sousedů. Hustě propojené vrcholy pak mají stejné štítky a šíří se, dokud je to možné. Na konci procesu jsou všechny vrcholy se stejnými štítky zařazeny do stejných komunit.

Proces se provádí opakovaně a v každém kroku mění každý vrchol svůj štítek podle svých sousedů. To ale přináší dva problémy. V případě, že se v síti nachází bipartitní podgraf, štítky začnou oscilovat. Druhým problémem je to, že pořadí vrcholů, které mění své štítky, je voleno náhodně. Jednotlivé výsledky algoritmu se tak mohou lišit na základě počáteční konfigurace.



Obrázek 11: Ukázka bipartitního grafu, ve kterém štítky začnou oscilovat. Tento problém lze řešit asynchronní aktualizací štítků. Zdroj: [13]

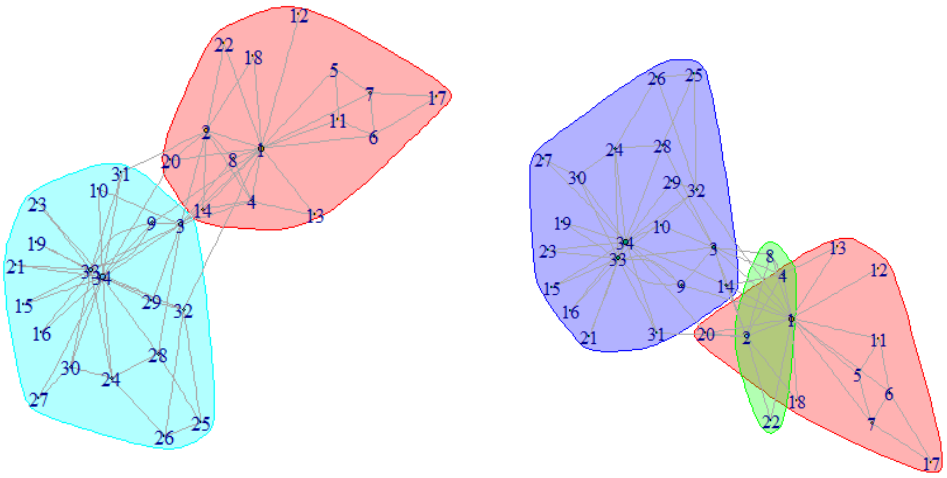
V ideálním případě by byl proces ukončen v případě, že žádný z vrcholů už nezmění svůj štítek. Pokud však má nějaký vrchol stejné množství sousedů ve vícero komunitách, je jeho příslušnost vybrána náhodně, takže by se mohlo stát, že by proces nebyl nikdy ukončen. Je proto nutné ukončovací podmínku specifikovat jinak. Proces je ukončen, když každý vrchol v síti má štítek shodný s největším počtem jeho sousedů. Proces fungování algoritmu pak lze shrnout do následujících kroků:

1. Inicializace štítků pro každý vrchol.

2. Nastavení času *t=1*.
3. Seřazení vrcholů do náhodného pořadí
4. Pro všechny vrcholy v tomto pořadí přenastavit štítek podle největšího počtu štítků jeho sousedů. V případě shody je příslušnost volena náhodně.
5. Pokud má každý vrchol stejný štítek jako největší počet jeho sousedů, algoritmus je ukončen. Jinak je čas *t* navýšen o jedna a pokračuje se od bodu 3.

3.6.1. Aplikace

Z popisu fungování algoritmu je zřejmé, že důležitou roli v něm hraje náhoda. Počáteční seřazení vrcholů značně ovlivňuje celkový výstup algoritmu.



Obrázek 12: Nahodilost Label-propagation algoritmu. Při ponechání náhodného pořadí štítkování vrcholů se v jednotlivých aplikacích algoritmu výsledky značně liší. V prvním případě je počet komunit n=2 a modularita Q=0,371. Ve druhém případě je počet komunit n=3 a modularita Q=0,315.

3.7. Multi-level algoritmus

Algoritmus staví na základu od Newmana a Girvana. Roku 2008 ho představili Blondel a kolektiv [14]. Na základě této práce bude algoritmus dále popsán. Tentokrát se jedná o metodu lokální maximalizace modularity v okolí každého vrcholu. Algoritmus je vhodný pro ohodnocené grafy a není zásadně ovlivněn velikostí sítě. Autoři dokázali v síti se 118 miliony vrcholy identifikovat komunity během 152 minut (v roce 2008).

Algoritmus postupuje odzdola nahoru, takže zpočátku je každý vrchol považován za samostatnou komunitu. Následně se pro každý vrchol *i* a jeho sousední vrcholy *j* vyhodnotí, jaký vliv na modularitu by mělo přesunutí vrcholu *i* do stejné komunity, do které patří vrchol *j*. Vrchol *i* je pak zařazen do komunity, pro kterou je rozdíl modularity největší (avšak pouze v případě, že je tento rozdíl kladný). Pokud kladný přírůstek není nalezen, vrchol *i* je ponechán v původní komunitě. Tento proces se opakuje pro všechny vrcholy, dokud je možné zvyšovat modularitu. Tato první fáze algoritmu tedy končí lokální maximalizací modularity, což se projeví tím, že již žádný vrchol není vhodné přesunout.

Funkce tohoto algoritmu je ovlivněna tím, v jakém pořadí jsou body k testování vybírány. Toto pořadí neovlivňuje výsledek rozřazení, nýbrž čas, za který se dosáhne výsledku. Efektivita tohoto algoritmu pramení zejména z toho, že výpočet změny modularity při přesunutí izolovaného vrcholu do určité komunity není výpočetně náročný. Tato změna při přesunutí vrcholu *i* do komunity *C* se počítá podle následující rovnice.

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

Rovnice 9: lokálně měřená změna modularity, zdroj: [14].

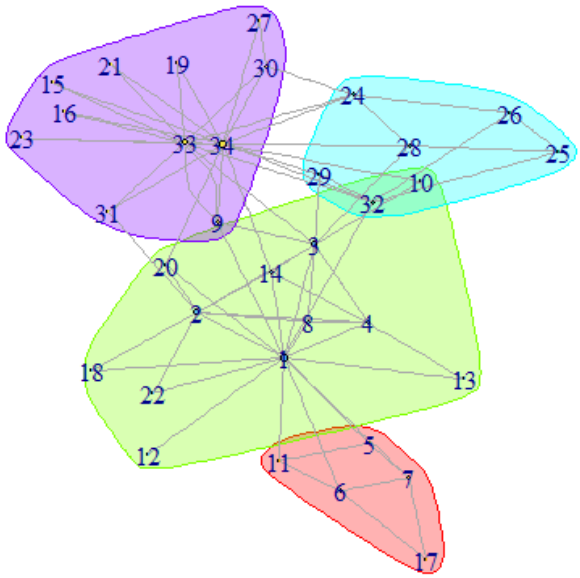
$\sum_{in}$ je součet hodnot hran uvnitř komunity *C*, $\sum_{out}$ je součet hodnot hran incidentních s vrcholy v komunitě *C*, k_i je součet hodnot hran incidentních s vrcholem *i*, $k_{i,in}$ je součet hodnot hran mezi vrcholem *i* a komunitou *C* a *m* je součet hodnot všech hran v celé síti. Podobný výraz lze využít pro výpočet změny modularity v případě odebrání vrcholu *i* z komunity. Změna modularity je pak v praxi počítána jako změna modularity při odebrání vrcholu *i* z komunity a tato změna při jeho přidání do sousední komunity.

V druhé fázi algoritmu je vytvořena nová síť, kde vrcholy představují komunity nalezené v první fázi. Hodnoty hran mezi novými vrcholy jsou dány součtem hodnot hran mezi vrcholy v odpovídajících původních komunitách. Hrany, které spojovaly vrcholy uvnitř těchto komunit, jsou nyní reprezentovány smyčkami o odpovídající hodnotě.

Dokončením této druhé fáze vzniká nová síť, na kterou je možné algoritmus znovu aplikovat a iterovat, dokud je možné zvyšovat modularitu. Nejnáročnější na výpočet je samozřejmě první iterace, protože následně vzniklé sítě jsou již značně redukovány. Počet těchto iterací nebývá vysoký a ovlivňuje, do jaké hloubky hierarchie bude síť rozebrána (vznikají komunity, které se skládají z komunit).

Algoritmus, tak jak byl představen, přináší jisté nesporné výhody. Je intuitivní a poměrně jednoduchý na implementaci. Další výhodou je nízká výpočetní náročnost, protože výše uvedený způsob rozdílu modularity je rychlý a postupným iterováním se drasticky snižuje složitost grafu. Rozlišovací schopnost modularity (neboli neschopnost identifikovat malé komunity) je zde eliminována tím, že v první fázi se zkoumá přesunutí jediného vrcholu z jedné komunity do jiné. Pravděpodobnost, že dvě komunity budou sloučeny v jednu postupným přesouváním vrcholů, je mizivá.

3.7.1. Aplikace



Obrázek 13:Aplikace Multi-level algoritmu na graf karate klubu Zachary. Počet nalezených komunit n=4, modularita Q=0,416. Zdroj: autor.

3.8. Radicchi a spol.

Jednou z vylepšených verzí GN-algoritmu je algoritmus, který představili roku 2003 Radicchi a spol. [11]. Na základě této práce bude algoritmus dále popsán. Místo hodnoty betweenness využívá lokální míru, takže její výpočet není tak náročný. Dalším zásadním rozdílem mezi tímto algoritmem a jeho předchůdcem je podmínka ukončení. O ukončení algoritmu edge-betweenness se nejčastěji rozhoduje na základě modularity, kdežto zde se využívá vlastností samotných komunit. Algoritmus rozlišuje dva typy komunit: silné a slabé. V silné komunitě má každý vrchol více spojení s vrcholy v rámci dané komunity než se zbytkem sítě. Ve slabé komunitě je celkový součet spojení mezi vrcholy v rámci této komunity větší, než součet spojení se zbytkem sítě. Platí, že každá silná komunita je současně slabou komunitou.

Míra, kterou algoritmus používá k výběru hran, které leží mezi komunitami, se nazývá koeficient edge-clustering, což lze přeložit jako koeficient shlukování. Vyplývá z myšlenky, že hrany, které spojují vrcholy ležící v odlišných komunitách, jsou součástí malého množství trojúhelníků, zatímco v rámci komunit existuje velké množství trojúhelníků. Následuje rovnice, podle které lze tento koeficient pro hranu spojující vrcholy *i* a *j* vypočítat.

$$\tilde{C}_{i,j}^{(3)} = \frac{z_{i,j}^{(3)} + 1}{\min[(k_i - 1), (k_j - 1)]}$$

Rovnice 10: koeficient shlukování, zdroj[11].

V čitateli rovnice se nachází počet trojúhelníků, které na dané hraně leží (zvětšený o jedna, aby se předešlo nulovému výsledku), a ve jmenovateli je pak jejich maximální možný počet. Výpočet této lokální hodnoty je úspornější, než výpočet betweenness a vede ke snížení náročnosti algoritmu na $O(m^2)$ pro rozsáhlé sítě.

3.9. Voltage algoritmus

Algoritmus představili roku 2003 Fang Wu a Bernardo A. Hubeman [15]. Na základě této práce bude algoritmus dále popsán. Jeho složitost roste lineárně s velikostí sítě. Vychází z fyzikálního pojetí sítě. Nahlíží na graf jako na elektrický obvod, kterým teče proud, procházející vrcholy. Pomocí Kirchhoffových zákonů je pak

možné spočítat napětí každého vrcholu. Vrcholy jsou v rámci komunit úzce propojeny, takže jejich napětí bude přibližně stejné. K velkému rozdílu v napětí dochází právě mezi komunitami, kde jsou hrany řídké a odpor velký. Příslušnost vrcholu, který leží mezi komunitami, je pak určena podle toho, ke které komunitě se blíží velikost jeho napětí (je buď větší, nebo menší než hraniční hodnota).

Mějme dva vrcholy *A* a *B*, o kterých víme, že každý leží v jiné komunitě (situace, kdy tato informace není předem známá, bude vysvětlena později). Každá hrana má stejný odpor a tyto dva vrcholy mají dané napětí, například 1 a 0. Za těchto předpokladů lze síť považovat za elektrický obvod s proudem, protékajícím každou hranou. Řešením Kirchhoffových rovnic je možné spočítat napětí každého vrcholu a na základě toho rozhodnout, do jaké patří komunity. Pokud je napětí větší, než daný práh (v tomto případě například 0,5), bude vrchol patřit do první komunity a v opačném případě do druhé.

Nechť vrchol *C* má *n* sousedních vrcholů $D_1,\ldots,D_n$. Podle Kirchhoffova zákona součet proudů do uzlu vstupujícího se rovná součtu proudů z uzlu vystupujícího. Jinými slovy součet proudů, protékajících uzlem se rovná nule. Můžeme psát:

$$\sum_{i=1}^{n} I_i = \sum_{i=1}^{n} \frac{V_{D_i} - V_C}{R} = 0,$$

kde I_i je proud tekoucí z D_i do *C*.

$$V_C = \frac{1}{n} \sum_{i=1}^{n} V_{D_i}.$$

Rovnice 11: výpočet napětí vrcholu, zdroj: [15]

Z toho vyplývá, že napětí vrcholu se rovná aritmetickému průměru napětí vrcholů s ním sousedících. Pokud většina vrcholů, sousedících s vrcholem *C*, patří do komunity, která má napětí větší, než daný práh, pak i V_C tento práh pravděpodobně překročí a vrchol *C* tedy bude zařazen do dané komunity. Algoritmus lze snadno aplikovat i na ohodnocené grafy, a to tak, že se stanoví vodivost hrany, odpovídající její hodnotě. Průměr v *rovnici 8* v takovém případě bude počítán jako vážený průměr.

Na základě *rovnice 11* může být Kirchhoffova rovnice pro obvod o *n* vrcholech určena následujícím způsobem. Pro první vrchol se zvolí $V_1 = 0$ a pro druhý vrchol $V_2 = 1$. Tyto dva vrcholy pak lze nazývat póly. Napětí $V_3,...,V_n$ lze pak spočítat lineárními rovnicemi ve tvaru:

$$V_i = \frac{1}{k_i} \sum_{j=3}^{n} V_j \, a_{ij} + \frac{1}{k_i} a_{i1} \quad \text{for } i = 3, \ldots, n.$$

Rovnice 12

Zde k_i je stupeň vrcholu i a a_{ij} je matice sousednosti grafu. Následuje představení matic a přepis Kirchhoffových rovnic do maticové podoby.

$$V = \begin{pmatrix} V_3 \\ \vdots \\ V_n \end{pmatrix}, \quad L = \begin{pmatrix} k_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{43} & k_4 & \cdots & -a_{4n} \\ \cdots & & & \cdots \\ -a_{n3} & -a_{n4} & \cdots & k_n \end{pmatrix}, \quad D = \begin{pmatrix} a_{31} \\ \vdots \\ a_{n1} \end{pmatrix}$$

$$V = L^{-1} D$$

Rovnice 13: maticové vyjádření výpočtu napětí, zdroj: [15]

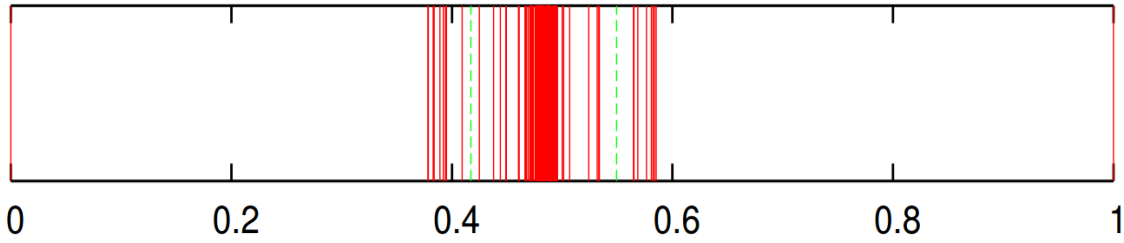
L je Laplaceova matice podgrafu původního grafu, který obsahuje vrcholy 3,...,*n*. Laplaceova matice *Q* je čtvercová matice, kde Q_{ii} se rovná stupni vrcholu *i*, Q_{ij} = -1 pokud je vrchol *i* spojený hranou s vrcholem *j* a $Q_{ij} = 0$ jinak. [16] Na rozdíl od algoritmů, které pracují s modularitou zde nejsou počítány vlastní vektory, čímž je snížena výpočetní náročnost. Nejdříve je nastavena $V_1 = 1$ a $V_2 = ... = V_n = 0$ v čase *O(V)*. Podle *rovnice 11* se dále nastavuje napětí jednotlivých vrcholů podle jejich sousedů, dokud se nedojde k poslednímu vrcholu *n*. Tím je uzavřen jeden celý okruh v čase *O(E)*. Po násobném opakování tohoto procesu je nalezeno řešení, jehož přesnost závisí na počtu těchto opakování, nikoliv na velikosti grafu. Celkový čas tohoto řešení je tedy *O(V + E)*, nehledě na velikost sítě.

Tím je vyřešen výpočet napětí jednotlivých vrcholů, stále je však nutné mít předem informace o síti, aby bylo možné určit dva vrcholy (póly), které leží ve dvou komunitách. Dále je nutné určit práh, podle kterého se bude určovat, do které komunity vrcholy patří. Pokud informace o síti nejsou předem známy, existují možnosti, jak tyto proměnné určit.

Jak již bylo řečeno, jelikož hrany mezi jednotlivými komunitami se vyskytují řídce, místní odpor je větší, než místní odpor v rámci těchto komunit. Z toho vyplývá, že napětí klesá hlavně na spojnicích komunit. Práh by tedy měl být umístěn v místě, kde dochází k největšímu lokálnímu rozdílu v napětí přibližně uprostřed spektra. Stačí tedy vzestupně seřadit hodnoty napětí vrcholů, určit medián hodnot, najít největší rozdíl v napětí poblíž tohoto bodu (s danou tolerancí, která v důsledku určuje povolenou velikost výsledných komunit) a hodnotu napětí v tomto bodě určit jako práh, podle kterého dojde k rozdělení vrcholů do dvou komunit.

Dalším problémem je, že algoritmus musí mít předem určené dva póly (vrcholy, které leží v rozdílných komunitách). K určení těchto pólů je možné využít statistickou metodu. Tato metoda funguje na základě opakovaného náhodného vybrání dvou vrcholů a aplikování algoritmu na rozdělení grafu do dvou komunit. Přibližně polovina výsledků by pak statisticky měla být správná, protože polovina náhodně vybraných vrcholů by skutečně patřila do rozdílných komunit. Pokud však vrcholy nebudou vybírány náhodně, ale z výběru se vyřadí vrcholy, které spolu přímo sousedí, pravděpodobnost nalezení správného výsledku bude více než 0,5. Pravděpodobnost, že dva náhodně vybrané vrcholy patří do stejné komunity, také klesá s rostoucím počtem komunit (přibližně se jedná o nepřímou úměrnost). Postup při dělení sítě do více než dvou komunit je pak následovný:

1. Náhodně se vyberou dva body, jejichž vzdálenost je větší nebo rovna dvěma a na základě tohoto výběru se provede výpočet napětí jednotlivých vrcholů.
2. Velikosti napětí se seřadí podle velikosti.
3. Na obou koncích spektra se s danou tolerancí (ovlivňující velikost výsledných komunit) hledá největší rozdíl v napětí, v těchto dvou bodech jsou umístěny prahy a vznikají dvě skupiny vrcholů (viz obr. 14).
4. Body 1 až 3 se opakují (čím větší počet iterací, tím přesnější výsledek).
5. Vybere se bod, který se v nějaké skupině vyskytuje s nejvyšší četností. K němu se přidají body, které se s nejvyšší četností vyskytují ve stejné skupině jako tento zvolený bod. Tím je detekována komunita.
6. Bod 5 se znovu aplikuje na zbylou část sítě, dokud zbývají vrcholy.

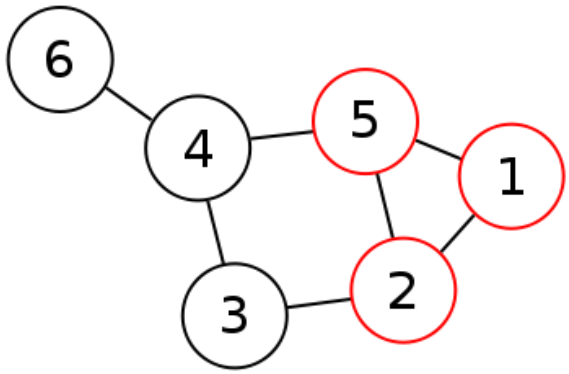


Obrázek 14: Velikost napětí nabývá hodnot mezi nulou a jedničkou (póly). Napětí je seřazeno podle velikosti a na obou koncích spektra se hledá největší rozdíl. V tomto případě je velikost výsledné komunity nastavena na zhruba 4 až 13 vrcholů. Algoritmem zvolené prahy jsou znázorněny zelenou čarou. Výsledkem je detekce dvou skupin, které splňují dané velikostní omezení. Zdroj: [15]

Algoritmus je velice vhodný, pokud jde o určení, do jaké komunity patří jeden vybraný vrchol. Místo výběru dvou náhodných pólů se určí daný bod jako jeden pól a druhý pól je náhodně vybrán z ostatních vrcholů, jejichž vzdálenost od daného bodu je větší nebo rovna dvěma. Ve výsledku tak vždy dojde k detekci komunity, do které vybraný vrchol patří.

3.10. C-finder

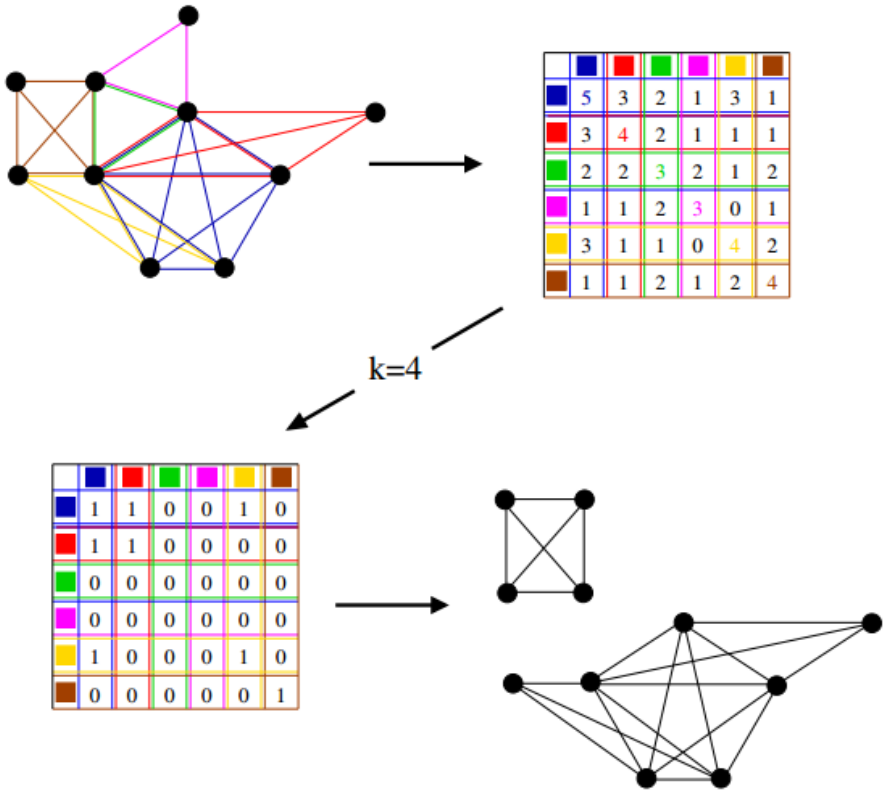
Tento lokální algoritmus představili roku 2005 Palla a kol. [17]. Na základě této práce bude algoritmus dále popsán. Jako jeden z prvních algoritmus řeší i případy, kdy komunity přesahují, tzn. sdílejí určité vrcholy, což je v kontextu sociálních sítí obzvláště důležité. Vychází z předpokladu, že uvnitř komunity bývají vrcholy typicky propojeny s mnoha ostatními - ale ne nezbytně všemi - vrcholy v této komunitě. Jinými slovy komunita může být interpretována jako spojení menších úplných podgrafů, které sdílejí vrcholy. **Klika** je podgraf nějakého grafu, který je úplným grafem (viz obr. 15).



Obrázek 15: Největší klika grafu (1,2,5). Zdroj: [18]

Algoritmus pracuje s pojmem *k*-klika, což je klika o *k* vrcholech, která může být podgrafem větší kliky. Dvě kliky spolu sousedí, pokud mají *k-1* společných vrcholů. *K*-kliková komunita je pak spojení všech sousedících klik o *k* vrcholech. V praxi se v rámci reálných sítí běžně vyskytují úplné podgrafy o desítce až stovce vrcholů. Úplný podgraf K_n obsahuje *n* nad *k* klik o *k* vrcholech, což je v případě takto velkých úplných podgrafů obrovské číslo. Algoritmus, který by hledal všechny kliky a zkoumal jejich sousedství, by tak byl velice náročný. Ze vztahu úplného podgrafu a *k*-kliky je ale zřejmé, že úplný podgraf tyto kliky obsahuje. Je tak výhodnější v síti hledat velké úplné podgrafy, a kliky hledat v místech, kde se tyto podgrafy překrývají. Rozdíl mezi běžnou klikou a *k*-klikou je ten, že *k*-klika může být podgrafem většího úplného grafu.

Algoritmus nejprve vyhledá všechny úplné podgrafy v síti, které nejsou součástí větších úplných podgrafů (kliky). Z nich následně vytvoří matici překrývajících se klik (viz obr. 16). Tato matice je symetrická a každý její řádek (i sloupec) představuje kliku a hodnoty představují počet společných vrcholů těchto klik. Na hlavní diagonále této matice se tedy nacházejí velikosti jednotlivých klik. Vrcholy, které představují překrytí dvou klik, vždy tvoří úplný podgraf. *K*-klikové komunity pro danou hodnotu *k* se nachází v místech, kde počet společných vrcholů v této matici je alespoň *k-1*. Tyto hodnoty se dají nalézt odstraněním prvků v matici, které jsou menší než *k-1* (popřípadě *k* v případě hlavní diagonály). Zbylé prvky jsou nahrazeny jedničkami a výsledné oddělené komponenty jsou *k*-klikové komunity.



Obrázek 16: Znázornění extrakce k-klikových komunit pro k=4. Jednotlivé kliky jsou znázorněny různými barvami. Z matice překrývajících klik jsou odstraněny příslušné hodnoty, zbylé jsou nahrazeny jedničkami. Z této výsledné matice jsou zřejmé k-klikové komunity, které jsou nakonec znázorněny. Zdroj: [17]

Výhoda tohoto postupu spočívá v tom, že jakmile je sestavena matice překrývajících se klik, je možné v krátkém čase najít komunity s libovolnou hodnotou *k*. A jak najít kliky, ze kterých je tato matice sestavena? Hledání se provádí od největších k nejmenším. Velikost největší možné kliky v síti je určena podle stupňů vrcholů. Počínaje touto velikostí algoritmus vždy vybere určitý vrchol, extrahuje všechny kliky dané velikosti, jenž obsahují daný vrchol, a nakonec odstraní tento vrchol a hrany s ním incidentní, čímž je zabráněno opakované detekci stejných klik. Jakmile nezbývá žádný vrchol, velikost hledaných klik se zmenší o jedna a vyhledávání se spustí znovu na původním grafu. Detailnější popis hledání klik není pro tuto práci relevantní. Náročnost tohoto algoritmu zásadně závisí na zkoumané síti, není proto možné uvést konkrétní výraz.

4. Popis knihovny JUNG/iGraph

JUNG, neboli Java Universal Network/Graph Framework je knihovna v programovacím jazyce Java, sloužící k modelování, analýze a vizualizaci dat ve formě grafů. Nabízí široké spektrum prostředků k práci s grafy.

Jedná se o poměrně rozsáhlou knihovnu, která umožňuje reprezentaci entit a jejich vztahů. Umí pracovat s orientovanými i neorientovanými grafy, grafy s paralelními hranami i hypergrafy. Knihovna také umožňuje entitám i vztahům přiřadit metadata. Je licencována BSD, což znamená, že může být volně šířena s uvedením autora.

Knihovna v sobě také zahrnuje řadu algoritmů pro analýzu sítí. Například algoritmy pro dekompozici, optimalizaci, generování náhodných grafů, statistickou analýzu a výpočet vzdáleností. V rámci této práce jsou nejzajímavější algoritmy pro clustering, neboli shlukování. Zde ale narážíme na poměrně zásadní problém, kterým je zastaralost knihovny. Knihovna pochází z roku 2003 a momentálně se nachází ve verzi 2.0. Poslední zásadní update zaznamenala v roce 2009 [19], ten se ale algoritmů pro detekci komunit nedotkl. Je to na jednu stranu pochopitelné, protože se jedná o poměrně dospělou a rozsáhlou knihovnu, na druhou stranu je ale škoda, že jí autoři přestali věnovat pozornost. V knihovně jsou implementovány pouze dva algoritmy pro detekci komunit, z nichž ani jeden není použitelný na rozsáhlé sítě. Navíc grafy sociálních sítí jsou šířeny zpravidla ve formě edgelistů, což je formát, který tato knihovna také nezná. Alternativou v jazyce java by mohla být např. knihovna JGraphT. Ta ovšem algoritmy pro detekci komunit postrádá úplně. Obecně se jazyk java pro matematické operace příliš nehodí.

Existují mnohem efektivnější a vyspělejší alternativy, které se k analýze rozsáhlých sítí hodí daleko lépe. Např. jazyk R je přímo určený pro statistickou analýzu dat a jejich zobrazení. Pro práci s grafy v tomto jazyce jmenujme například open source knihovnu iGraph, která je dostupná také v jazycích Python a C/C++. Na rozdíl od JUNG je tato knihovna mnohem modernější, nabízí implementaci zajímavějších algoritmů, dokáže zpracovávat rozsáhlé sítě a pracovat s edgelisty. Samotné testování algoritmů na reálných datech bude proto uskutečněno pomocí této knihovny v prostředí RStudio.

5. Testované sítě

Data o reálných sítích jsou dostupná na stránkách Stanford Large Network Dataset Collection, jejímiž autory jsou Jure Leskovec a Andrej Krevl [20]. Tato data jsou šířena ve formě edgelistů v souborech txt. V knihovně iGraph se lépe pracuje například se soubory csv, nicméně i s textovými soubory si iGraph dokáže poradit. Edgelist je formát, ve kterém jsou na každém řádku uvedeny vrcholy, které jsou spojeny hranou. Pro tuto práci je zajímavá zejména kategorie sítí, ve kterých mají vrcholy implicitně určenou příslušnost do komunit [21]. U těchto sítí je vždy uveden počet komunit a také pravidlo, podle kterého byly vrcholy do komunit rozděleny.

První a nejmenší z nich je síť DBLP. Jedná se o síť, která zachycuje seznam vědeckých článků z kategorie computer science. Jednotlivé vrcholy značí autory článků a hrany představují spolupráci. Pokud spolu tedy dva autoři spolupracovali na nějakém článku, budou vrcholy, které je reprezentují, spojeny hranou. V této síti je 317 080 vrcholů, 1 049 866 hran a 13 477 komunit. Komunity jsou zde tvořeny formou publikace těchto článků - například vědecké časopisy nebo konference. Pokud tedy dva autoři publikovali svou práci ve stejném časopise, či na stejné konferenci, jsou zařazeni do stejné komunity.

Druhou nejmenší sítí je síť nákupů na Amazonu. Pokud tedy zákazníci, kupující produkt *i*, také často kupovali produkt *j*, graf obsahuje neorientovanou hranu z *i* do *j*. V síti je 334 863 vrcholů, 925 872 hran a 75 149 komunit. Komunity jsou zde tvořeny všemi kategoriemi produktů, které Amazon uvádí.

Další testovanou sítí je sociální síť YouTube, která slouží ke sdílení videí, ale také sociální síť. Na této sociální síti mohou uživatelé uzavírat přátelství a také tvořit skupiny. Právě tyto uživatelsky vytvořené skupiny jsou zde brány jako komunity. Síť se skládá z 1 134 890 vrcholů a 2 987 624 hran a je zde uvedeno 8 385 skupin, tedy komunit.

Poslední a největší testovanou sítí je sociální síť LiveJournal, na které mohou uživatelé přidávat blogové příspěvky. Opět se zde tvoří přátelství a uživatelsky definované skupiny. Síť je tvořena 3 997 962 vrcholy a 34 681 189 hranami. Počet uvedených komunit se v tomto případě vyšplhal na 287 512.

Tabulka č. 1:Přehled testovaných sítí.

Název sítě	**Počet vrcholů |V|**	**Počet hran |E|**	**Počet komunit n**
DBLP	317 080	1 049 866	13 477
Amazon	334 863	925 872	75 149
YouTube	1 134 890	2 987 624	8 385
LiveJournal	3 997 962	34 681 189	287 512

U sítí DBLP a Amazon jsou dokonce uvedeny konkrétní struktury komunit. Opět ve formě textových souborů, kde tentokrát každý řádek představuje jednu komunitu. Na každém řádku jsou tedy uvedeny id vrcholů, které do dané komunity patří. Z těchto souborů je tedy určit informace o komunitách. U zbylých sítí údaje v těchto souborech neodpovídají uváděným počtům komunit, takže nebudou brány v potaz. Následuje ukázka kódu.

```
komunity=scan(file.choose(),what = "character",sep= "\n")
komunity=gsub("\t", " ", komunity)
komunityList=strsplit(komunity," ")
velikostiKomunit=lapply(komunityList, length)
velikostiKomunitVektor=unlist(velikostiKomunit)
minKomunity=min(velikostiKomunitVektor)
maxKomunity=max(velikostiKomunitVektor)
prumerKomunity=mean(velikostiKomunitVektor)
medianKomunity=median(velikostiKomunitVektor)
plot(velikostiKomunitVektor,type = "l")
```

Obrázek 17: Ukázka kódu pro analýzu reálných komunit. Na prvním řádku je načten txt soubor a vytvořen list, ve kterém každý prvek představuje řádek původního souboru (tedy seznam vrcholů v dané komunitě). Na druhém řádku jsou tabulátory nahrazeny mezerami, Na třetím řádku jsou od sebe odděleny jednotlivé vrcholy na každém řádku. Na čtvrtém řádku je vytvořen list, ve kterém jsou uloženy pouze velikosti komunit (tedy počty vrcholů). Na pátém řádku je tento list převeden na vektor, aby se na něm daly provádět matematické operace, které následují na následujících čtyřech řádcích. Na posledním řádku jsou velikosti komunit graficky znázorněny. Zdroj: autor.

6. Testování algoritmů na reálných datech

Jak již bylo řečeno, testování algoritmů proběhne s využitím programovacího jazyka R v prostředí RStudio. Je také nezbytné uvést, na jakém hardwaru bylo testování prováděno. Jedná se o notebook Asus, osazený procesorem Intel Core i5-5200U o taktu 2,2 GHz a 8 GB DDR3 operační paměti. Operační systém je Windows 8.1 64bit. Vytížení procesoru se během testování pohybovalo zpravidla mezi 30-40% a procesy vyžadovaly 1-2 GB operační paměti.

Následuje ukázka kódu, pomocí kterého bylo měření prováděno. V proměnné *g* se nachází objekt s grafem, který byl načten z edgelistu, staženého z dříve uvedených zdrojů.

```
start.time <- Sys.time()

infoamazon <- cluster_infomap(g)

end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

Obrázek 18: Ukázka zdrojového kódu, použitého při testování algoritmů. Je zde pěkně vidět jednoduchost jazyka R jako takového, i jednoduchost knihovny iGraph. Nejprve je zaznamenán čas spuštění algoritmu, následně jsou do proměnné infoamazon uloženy komunity, které byly v grafu g nalezeny použitím příkazu cluster_infomap, který využívá Infomap algoritmu. Následně je zaznamenán čas ukončení běhu algoritmu. Rozdíl zaznamenaných časů pak přináší přesnou informaci o době běhu algoritmu a v proměnné infoamazon je uloženo výsledné rozdělení vrcholů do komunit. Zdroj: autor.

Je také třeba stanovit časovou hranici, po které bude testování zastaveno. Tato hranice byla stanovena na 90 minut. U Label-propagation algoritmu je vždy uveden průměr z pěti měření, kvůli odlišným výsledkům. Následuje aplikace algoritmů na pět reálných sítí. Zkoumané parametry jsou následující:

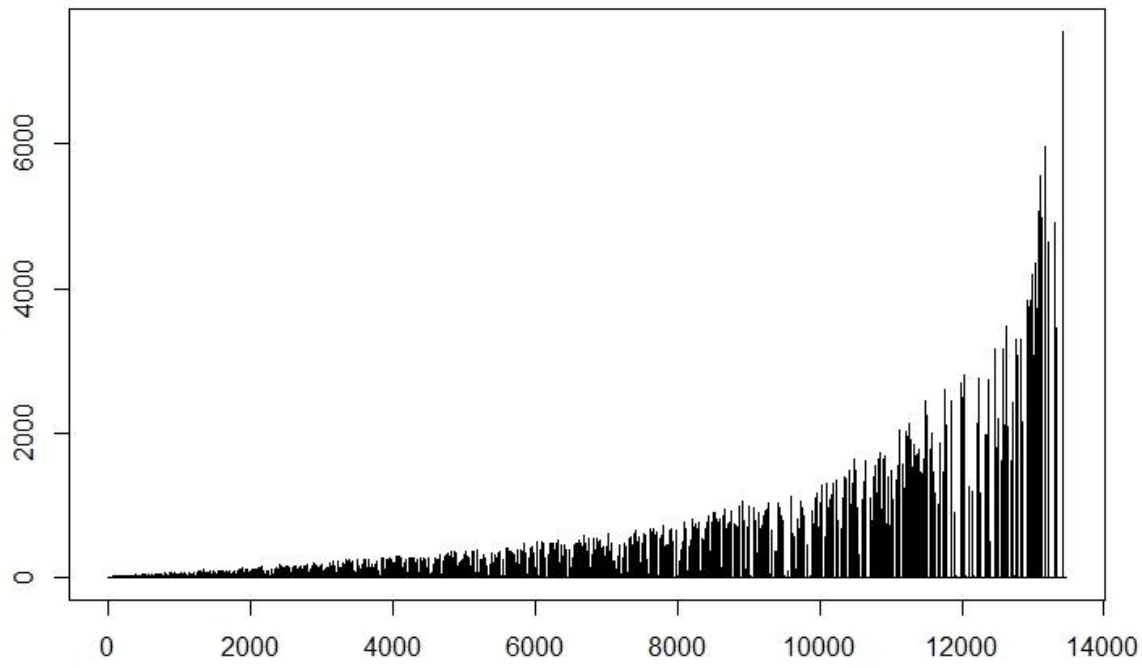
1. n… počet nalezených komunit
2. Q… modularita výsledného rozdělení
3. t… doba výpočtu algoritmu

4. $|V|_{max}$, $|V|_{min}$, $|V|_{avg}$, $|V|_{med}$… velikosti komunit (největší, nejmenší, průměr velikostí, medián velikostí)

6.1. Síť DBLP

Tabulka č. 2: detail sítě DBLP.

Počet vrcholů |V|	**Počet hran |E|**	**Počet komunit n**	**$V\|_{max}$;$\|V\|_{min}$;$\|V\|_{avg}$; $\|V\|_{med}$**
317 080	1 049 866	13 477	7 556; 6; 53,4; 8

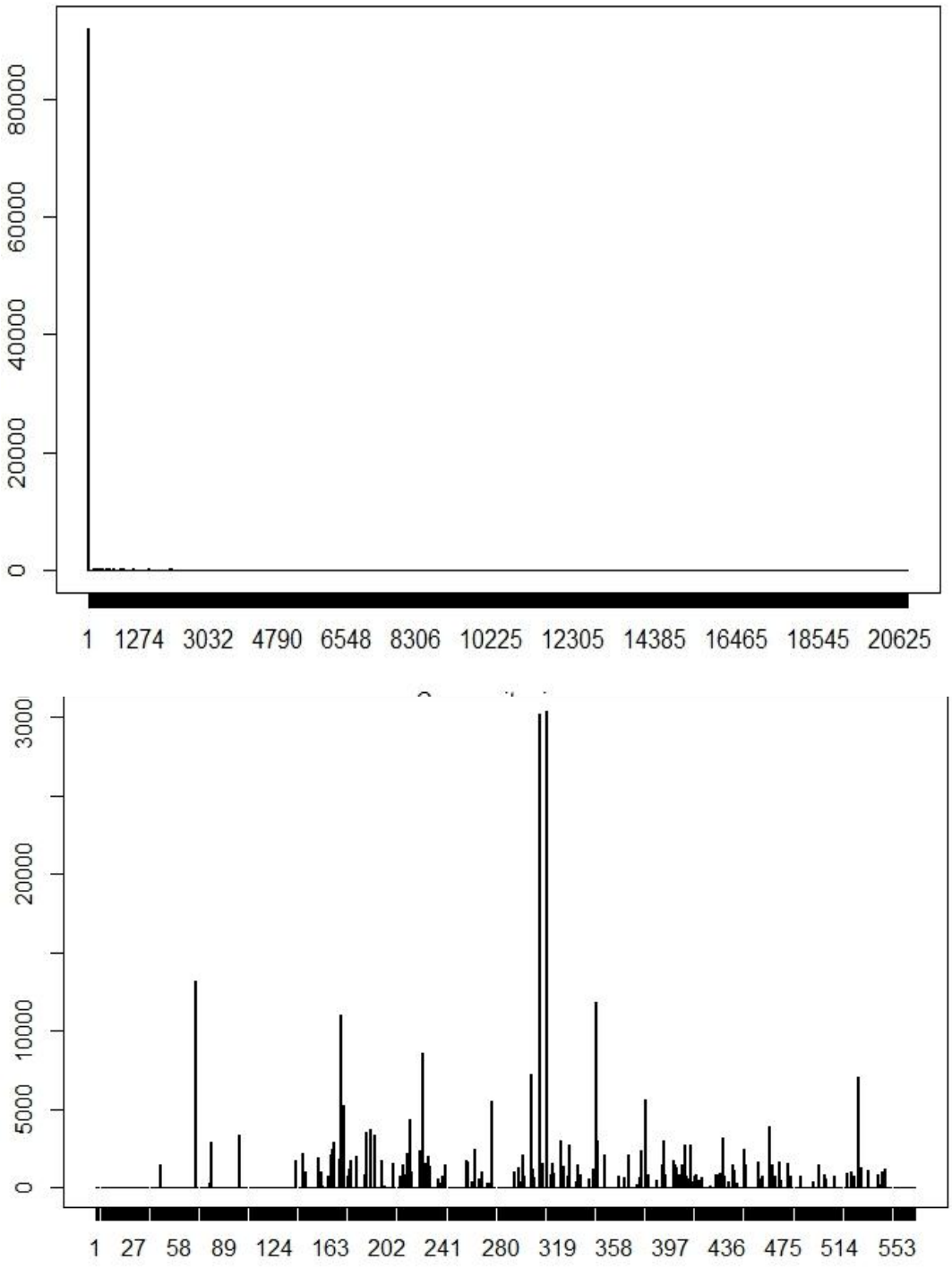


Obrázek 19: Velikosti reálných komunit v síti DBLP. Největší komunita má 7 556 vrcholů, nejmenší 6 vrcholů, průměr velikostí je 53,4 vrcholu a medián 8 vrcholů. Zdroj: autor.

Tabulka č. 3: Výsledky testování na síti DBLP.

Algoritmus	**n**	**Q**	**t**	$\|V\|_{max}$; $\|V\|_{min}$; $\|V\|_{avg}$; $\|V\|_{med}$
Fast-greedy	3 141	0,734	40:46	54 690; 3; 100,9; 7
Walktrap	-	-	-	-
Leading-eigenvector	2	0,024	00:12	-
Infomap	-	-	-	
Edge-betweenness	-	-	-	-
Label-propagation	20 918	0,674	01:23	91 512; 3; 15,2; 7
Multi-level	565	0,809	00:04	30 427; 4; 561,2; 10

Detekci komunit v síti s nejmenším počtem vrcholů zvládly pouze 3 algoritmy. Fast-greedy algoritmus označil 3 141 komunit o průměrné velikosti kolem stovky vrcholů. Label-propagation algoritmus při každém z pěti pokusů zařadil kolem 90 tisíc vrcholů do jedné komunity a zbylé vrcholy rozdělil do více než 20 tisíc mnohem menších komunit. Multi-level algoritmus detekoval dvě komunity s více než 30 tisíci vrcholy, zatímco ostatní vrcholy rozdělil do více než 550 menších komunit. Algoritmus Leading-eigenvector není schopen rozlišit komunity kvůli omezení modularity.

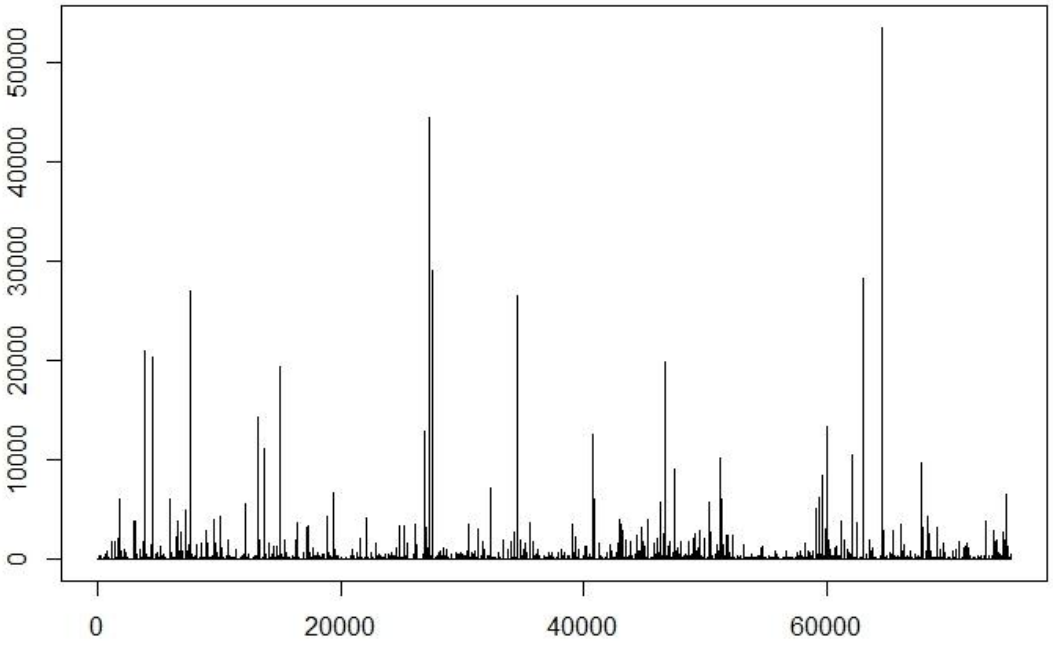


Obrázek č. 20: Porovnání výsledku algoritmu Label-propagation a Multi-level na síti DBLP. Zdroj: autor.

6.2. Síť Amazon

Tabulka č. 4: Detail sítě Amazon.

Počet vrcholů |V|	**Počet hran |E|**	**Počet komunit n**	**$\|V\|_{max}$;$\|V\|_{min}$;$\|V\|_{avg}$; $\|V\|_{med}$**
334 863	925 872	175 149	53 551; 3; 30,2; 5

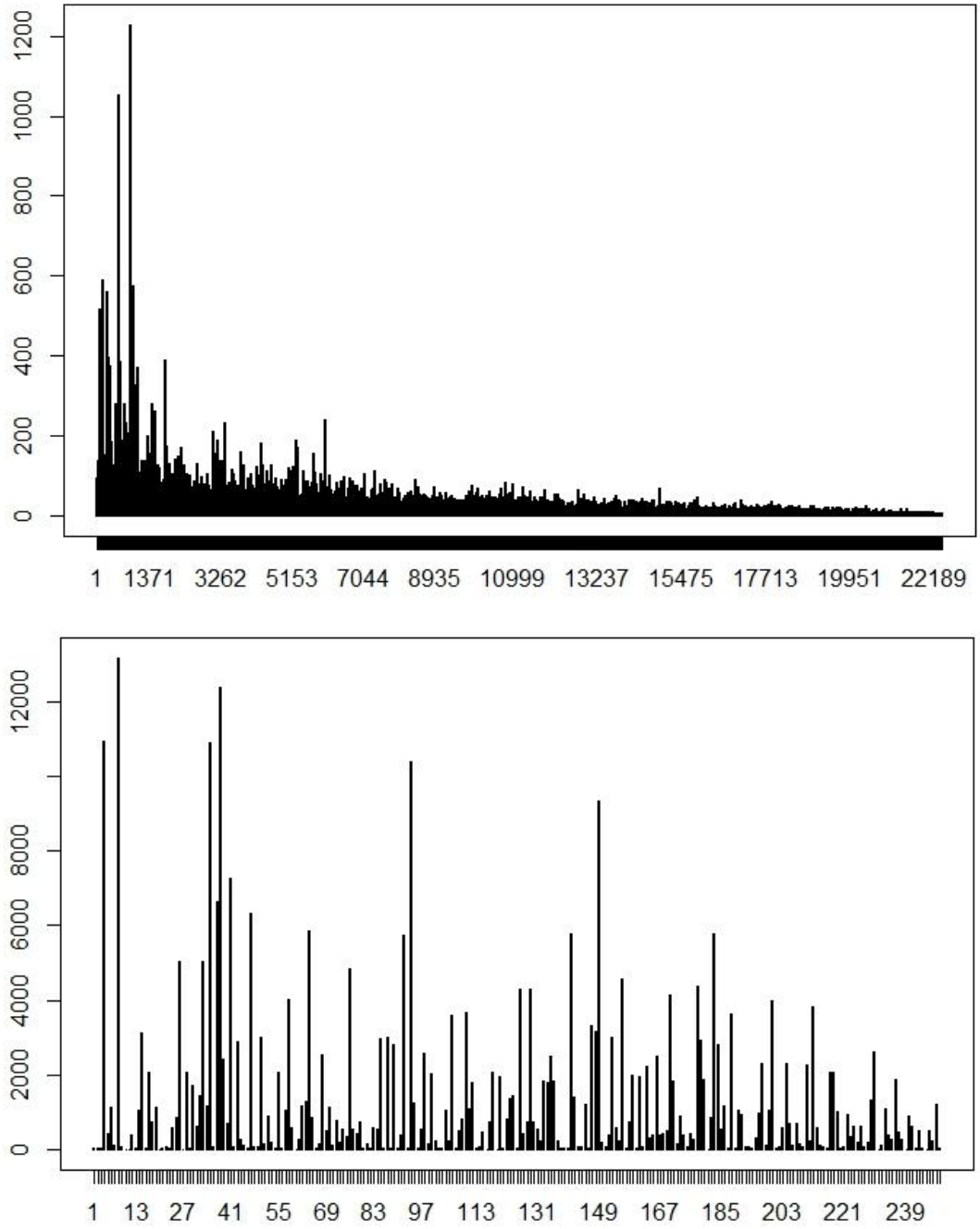


Obrázek 21: Velikosti reálných komunit v síti Amazon. Největší komunita má 53 551 *vrcholů, nejmenší 3 vrcholy, průměr velikostí je 30,2 vrcholu a medián 5 vrcholů. Zdroj: autor.*

Tabulka č. 5: Výsledky testování na síti Amazon.

Algoritmus	**n**	**Q**	**t**	$\|V\|_{max}$; $\|V\|_{min}$; $\|V\|_{avg}$; $\|V\|_{med}$
Fast-greedy	1 525	0,876	16:20	45 757; 3; 219,6; 12
Walktrap	14 905	0,849	19:58	49 880; 2; 22,5; 8
Leading-eigenvector	1	0	00:14	-
Infomap	-	-	-	-
Edge-betweenness	-	-	-	-
Label-propagation	22 223	0,786	00:34	1123; 3; 14,9; 10
Multi-level	249	0,926	00:04	13 167; 11; 1 344,8; 534

Detekci komunit v druhé nejmenší síti zvládly 4 algoritmy. Fast-greedy algoritmus detekoval 1 525 komunit. Walktrap algoritmus Detekoval 14 905 komunit. Label-propagation algoritmus vždy detekoval dvě komunity s více než jedním tisícem vrcholů a zbytek vrcholů rozdělil přibližně do 22 tisíc komunit. Multi-level algoritmus detekoval 5 komunit s více než 10 tisíci vrcholy a zbytek vrcholů rozdělil do 244 komunit.



Obrázek č. 22: Porovnání výsledku algoritmu Label-propagation a Multi-level na síti DBLP. Zdroj: autor.

6.3. Síť YouTube

Tabulka č. 6: Detail sítě YouTube.

Počet vrcholů |V|	**Počet hran |E|**	**Počet komunit n**	**$\|V\|_{max}$;$\|V\|_{min}$;$\|V\|_{avg}$; $\|V\|_{med}$**
1 134 890	2 987 624	8 385	-

Tabulka č. 7: Výsledky testování na síti YouTube.

Algoritmus	**n**	**Q**	**t**	**$\|V\|_{max}$; $\|V\|_{min}$; $\|V\|_{avg}$; $\|V\|_{med}$**
Fast-greedy	-	-	-	-
Walktrap	-	-	-	-
Leading-eigenvector	-	-	-	-
Infomap	-	-	-	-
Edge-betweenness	-	-	-	-
Label-propagation	40 688	0,666	03:41	248 986; 2; 27,6; 5
Multi-level	9 630	0,685	00:17	198 088; 3; 117,8; 5

Dle očekávání detekci komunit ve třetí síti zvládly pouze dva nejméně náročné algoritmy. Label-propagation algoritmus opět detekoval jednu komunitu, která je nepoměrně větší než zbylé komunity. Průměrně zahrnovala 248 986 vrcholů, zatímco ostatní vrcholy byly zařazeny do více než 40 tisíc komunit. Multi-level algoritmus detekoval celkem 6 komunit s velikostí přes 50 tisíc vrcholů. Zbytek vrcholů zařadil do 9 629 komunit,

6.4. Síť LiveJournal

Počet vrcholů |V|	**Počet hran |E|**	**Počet komunit n**	**$\|V\|_{max}$;$\|V\|_{min}$;$\|V\|_{avg}$; $\|V\|_{med}$**
3 997 962	34 681 189	287 512	-

Algoritmus	**n**	**Q**	**t**	$\|V\|_{max}$; $\|V\|_{min}$; $\|V\|_{avg}$; $\|V\|_{med}$
Fast-greedy	-	-	-	-
Walktrap	-	-	-	-
Leading-eigenvector	-	-	-	-
Infomap	-	-	-	-
Edge-betweenness	-	-	-	-
Label-propagation	87 563	0,695	10:51	1 158 239; 2; 43,5; 9
Multi-level	21 532	0,791	00:43	576 487; 35; 177,3; 148

V největší síti LiveJournal opět uspěly pouze dva z testovaných algoritmů. Label-propagation algoritmus zařadil přes milion vrcholů do jedné komunity a ostatní vrcholy rozdělil do mnohem menších komunit. Celkem jich detekoval 87 563. Multi-level algoritmus detekoval dvě komunity s více než půl milionem vrcholů. Celkem detekoval 21 532 komunit.

7. Shrnutí výsledků

Většina testovaných algoritmů nebyla schopna úspěšně detekovat komunity v rozsáhlých sítích. Algoritmy Fast-greedy, Walktrap, Leading-eigenvector, Infomap, ani Edge-bewteenness ve vymezeném čase nepřinesly výsledky. Vyplývá to už z jejich principu - jsou jednoduše příliš náročné pro tak velké množství hran a vrcolů. Algoritmus Leading-eigenvector doplácí na rozlišovací schopnost komunity a pro velké sítě je nepoužitelný.

Jediné dva algoritmy, použitelné na sítě velkého rozsahu, jsou Label-propagation a Multi-level. Oba tyto algoritmy dosahují velice dobrých časů. Multi-level algoritmus zpravidla detekuje menší množství komunit, ale má větší celkovou modularitu. Všechna důležitá data jsou uvedena výše v tabulkách.

Velké rozdíly mezi průměrnou velikostí detekovaných komunit a jejich mediánem značí nerovnoměrné rozdělení. Zejména Label-propagation algoritmus má tendenci zařadit velké množství vrcholů do jediné komunity a zbylé pak rozdělit do nepoměrně menších komunit.

U nejmenších dvou sítí byly k dispozici i detailní informace o reálných komunitách. Při porovnání výsledků aplikovaných algoritmů s těmito komunitami je vidět, že se neshodují. Otázkou však zůstává, do jaké míry tyto implicitně definované komunity opravdu odpovídají komunitní struktuře daných sítí.

Z uvedeného vyplývá, že žádný z uvedených algoritmů nebyl schopen přinést výsledky, které by vypovídajícím způsobem reflektovaly realitu. Velikosti i počty detekovaných komunit se většinou neshodují s opravdovou strukturou sítě.

8. Závěry a doporučení

V práci byl čtenář seznámen s důležitými pojmy z oboru teorie grafů, dále s pojmy komunita a modularita. Detekce komunitní struktury v sociálních sítích byla uvedena do kontextu a byl vysvětlen její význam (nejen v sociálních sítích).

Dále bylo detailně vysvětleno 10 vybraných algoritmů pro detekci komunit. Cílem bylo představit co nejširší spektrum možných přístupů k řešení dané problematiky. U sedmi z těchto algoritmů byl uveden i praktický příklad aplikace na jednoduchý graf.

Následně byly představeny sociální sítě, na kterých bylo prováděno testování algoritmů. Sítě byly detailně popsány z hlediska struktury i významu.

V závěrečné části práce bylo 7 algoritmů testováno na reálných datech s podporou knihovny iGraph v programovacím jazyce R. Výsledky testování byly přehledně sepsány do podrobných tabulek.

Pět ze sedmi testovaných algoritmů se ukázalo pro velké sítě jako nepoužitelné. Zbylé dva algoritmy naopak byly schopny komunity detekovat velice rychle. Detekované komunity však neodpovídají těm reálným.

Z uvedeného vyplývá, že tento obor se pravděpodobně stane terčem dalšího studia, protože sítě se stále rozrůstají a jejich analýza je čím dál důležitější. Zásadním problémem zůstává nejednotná definice komunity a neschopnost algoritmů ve velkých sítích v krátkém čase detekovat komunity, které odpovídají realitě.

9. 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. 2015-05-21]. DOI:10.1073/pnas.0601602103. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.0601602103

[2] 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. 2016-02-21]. DOI: 10.1073/pnas.122653799. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.122653799.

[3] NEWMAN, M. Fast algorithm for detecting community structure in networks. Physical Review E [online]. 2004, vol. 69, issue 6, s. - [cit. 2015-02-21]. DOI: 10.1103/PhysRevE.69.066133. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.69.066133

[4] NEWMAN, M. E. J. a M. GIRVAN. Finding and evaluating community structure in networks. Physical Review E [online]. 2004, 69(2), - [cit. 2016-04-28]. DOI: 10.1103/PhysRevE.69.026113. ISSN 1539-3755. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.69.026113

[5] ROSVALL, M. a C. T. BERGSTROM. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences [online]. 2008, 105(4), 1118-1123 [cit. 2016-04-28]. DOI: 10.1073/pnas.0706851105. ISSN 0027-8424. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.0706851105

[6] SOBOTA, Branislav a Ján MILIÁN. Grafické formáty. 1. vyd. České Budějovice: Kopp, 1996. ISBN 80-85828-58-8.

[7] PONS, Pascal a Matthieu LATAPY. Computing Communities in Large Networks Using Random Walks [online]. s. 284 [cit. 2016-04-28]. DOI: 10.1007/11569596_31. Dostupné z: http://link.springer.com/10.1007/11569596_31

[8] KOŘENÁŘ, Václav. Stochastické procesy. Vyd. 1. Praha: Vysoká škola ekonomická, 2002. ISBN 80-245-0311-5.

[9] FREEMAN, Linton C. A Set of Measures of Centrality Based on Betweenness. DOI: 10.2307/3033543. ISBN 10.2307/3033543. Dostupné z: http://www.jstor.org/stable/3033543?origin=crossref

[10] NEWMAN, M. E. J. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. Physical Review E [online]. 2001, 64(1), - [cit. 2016-04-28]. DOI: 10.1103/PhysRevE.64.016132. ISSN 1063-651x. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.64.016132

[11] RADICCHI, F., C. CASTELLANO, F. CECCONI, V. LORETO a D. PARISI. Defining and identifying communities in networks. Proceedings of the National Academy of Sciences [online]. 2004, 101(9), 2658-2663 [cit. 2016-04-28]. DOI: 10.1073/pnas.0400054101. ISSN 0027-8424. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.0400054101

[12] CLAUSET, Aaron, M. E. J. NEWMAN a Cristopher MOORE. Finding community structure in very large networks. Physical Review E [online]. 2004, 70(6), - [cit. 2016-04-28]. DOI: 10.1103/PhysRevE.70.066111. ISSN 1539-3755. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.70.066111

[13] RAGHAVAN, Usha Nandini, Réka ALBERT a Soundar KUMARA. Near l linear time algorithm to detect community structures in large-scale networks. Physical Review E [online]. 2007, 76(3), - [cit. 2016-04-28]. DOI: 10.1103/PhysRevE.76.036106. ISSN 1539-3755. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.76.036106

[14] 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, 2008(10), P10008- [cit. 2016-04-28]. DOI: 10.1088/1742-5468/2008/10/P10008. ISSN 1742-5468. Dostupné z: http://stacks.iop.org/17425468/2008/i=10/a=P10008?key=crossref.46968f6ec61eb8f907a760be1c5ace52

[15] WU, F. a B. A. HUBERMAN. Finding communities in linear time: a physics approach. The European Physical Journal B - Condensed Matter [online]. 2004-3-1, 38(2), 331-338 [cit. 2016-04-28]. DOI: 10.1140/epjb/e2004-00125-x. ISSN 1434-6028. Dostupné z: http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1140/epjb/e2004-00125-x

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

[17] PALLA, Gergely, Imre DERÉNYI, Illés FARKAS a Tamás VICSEK. Uncovering the overlapping community structure of complex networks in nature and society. Nature [online]. 2005-6-9, 435(7043), 814-818 [cit. 2016-04-28]. DOI: 10.1038/nature03607. ISSN 0028-0836. Dostupné z: http://www.nature.com/doifinder/10.1038/nature03607

[18] Klika (teorie grafů). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-04-28]. Dostupné z: https://cs.wikipedia.org/wiki/Klika_(teorie_graf%C5%AF)

[19] Jung. Java Universal Network/Graph Framework. [online]. 24.1.2010 [cit. 2016-04-18]. Dostupné z: http://jung.sourceforge.net/index.html

[20] LESKOVEC, Jure a Krevl, Andrej. Stanford. Large Network Dataset Collection. [online]. 2015 [cit. 2015-04-04]. Dostupné z: https://snap.stanford.edu/data/

[21] YANG, Jaewon a Jure LESKOVEC. Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics - MDS '12 [online]. New York, New York, USA: ACM Press, 2012, s. 1-8 [cit. 2016-04-28]. DOI: 10.1145/2350190.2350193. ISBN 9781450315463. Dostupné z: http://dl.acm.org/citation.cfm?doid=2350190.2350193

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Minařík Jakub	Jana Dítěte 1567, Přelouč	I1301217

TÉMA ČESKY:

Detekce komunit v sociálních sítích

TÉMA ANGLICKY:

Community detection in social networks

VEDOUCÍ PRÁCE:

Mgr. Jiří Haviger, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce:
Popis algoritmů a ukázka na cvičných datech v jazyce Java s podporou knihovny JUNG pro analýzu grafů.

Osnova:
1. Úvod
2. Popis algoritmů
3. Stručný popis knihovny JUNG
4. Testování na reálných datech
5. Závěr

SEZNAM DOPORUČENÉ 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. 2015-05- 21]. DOI:10.1073/pnas.0601602103. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.0601602103

2. 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. 2016-02-21].DOI: 10.1073/pnas.122653799. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.122653799.

3. NEWMAN, M. Fast algorithm for detecting community structure in networks. Physical Review E [online]. 2004, vol. 69, issue 6, s. - [cit. 2015-02-21]. DOI: 10.1103/PhysRevE.69.066133. Dostupné z: http://link.aps.org/doi/10.1103/PhysRevE.69.066133

Podpis studenta: ..... **Datum:** 29.4.2016

Podpis vedoucího práce: ..... **Datum:** 29.4.2016