Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Zobrazování detailů pro velké grafy

Plzeň 2016 Jakub Žáček

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. června 2016

Jakub Žáček

Poděkování

Chtěl bych poděkovat svému vedoucímu diplomové práce Ing. Richardu Lipkovi Ph.D. za odborné vedení, věcné připomínky a vstřícnost při konzultacích a vypracování diplomové práce.

Dále bych také rád poděkoval všem testerům za ochotu a čas, který věnovali testování ukázkové aplikace.

Abstract

Big graphs zooming

This diploma thesis deals with methods and problems of visualisation of large graphs. You can also find here brief overview of some applications for large graph visualisation. Practical part of this thesis contains implementation of application, which demonstrates five different methods of large graph visualisation (default, manual, node hiding, node clustering and node zooming) and its testing by independent testers. Chosen method was then implemented into Visualisation of power network Project developed by employees and students of Department of Computer Science and Engineering, University of West Bohemia.

Abstrakt

Tato diplomová práce se zabývá způsoby a problematikou vizualizace velkých grafů. Dále zde naleznete stručný přehled několika aplikací určených pro vizualizaci grafů. V praktické části této diplomové práce se nachází implementace aplikace sloužící k demonstraci pěti různých metod vizualizace velkých grafů (výchozí, manuální, skrývání vrcholů, shlukování vrcholů a změny velikosti vrcholů) a její testování pěti nezávislými testery. Vybraná metoda pak byla aplikována v projektu Vizualizace přenosové sítě vyvíjeného zaměstnanci a studenty Katedry informatiky a výpočetní techniky Západočeské univerzity v Plzni.

Obsah

1 Úvod

Vizualizace grafů je hojně využívána pro přehledné znázornění sítí, přenosových soustav, vztahů mezi lidmi či různými objekty a dalších věcí. Tyto grafy jsou pak většinou velmi velké – mohou obsahovat miliony nebo i více vrcholů či hran. I přes jejich velikost je potřeba tyto grafy přehledně zobrazit.

Při vykreslování velkých grafů často vznikají chaotické vizualizace, které nemají žádný přínos pro uživatele. Vykreslený graf je pak nepřehledný a mnohdy se v něm ztrácí důležité informace, protože není možné vykreslit velké množství vrcholů na malou plochu bez toho, aniž by se vrcholy překrývaly a křížilo se velké množství hran.

V teoretické části této diplomové práce jsem prozkoumal výhody a nevýhody různých způsobů a metod vizualizace malých a velkých grafů. Také jsem zde pro přehled uvedl popis několika existujících aplikací pro vizualizaci grafů. V této práci jsem se zvláště zaměřil na grafy se statickým rozložením, které lze snadno vykreslit na mapový podklad.

V praktické části práce jsem vytvořil testovací aplikaci, která implementuje vybrané metody vizualizace grafu. Tato aplikace podporuje přibližování, oddalování a pohyb po grafu. Tuto aplikaci jsem následně poskytl pěti nezávislým testerům, kteří subjektivně zhodnotili přehlednost a chování různých metod u různých grafů.

Z navržených metod jsem pak vybral jednu, kterou jsem implementoval do projektu Vizualizace přenosové sítě vyvíjeného studenty a zaměstnanci katedry informatiky a výpočetní techniky pro katedru kybernetiky Západočeské univerzity v Plzni.

V přílohách jsem uvedl uživatelské příručky k oběma aplikacím.

2 Vizualizace grafů

Tato práce se zabývá především vizualizací grafů a předpokládá již určitou znalost definic a problematiky grafů. Informace o grafech lze nalézt v literatuře (například [1]).

Nejčastějším způsobem vizualizace grafu je zobrazení vrcholů jako bodů (kruhů, obdélníků apod.) a hran jako čar tyto body spojující. Pokud nejsou ohodnocení jednotlivých vrcholů a hran shodná, je možné tato ohodnocení znázornit velikostí bodů (tloušťkou čar), případně jejich barvami, průhledností atp.

Dále je u vizualizace grafů důležitý layout (uspořádání vrcholů) vizualizace. Každý layout má svá využití v různých oblastech. Těch existuje celá řada, níže zmiňuji ty nejpoužívanější (viz obrázky 2.1, 2.2 a 2.3):

- **statický layout** – každý vrchol je ručně umístěn uživatelem. Implementačně velmi jednoduché, ale následně je časově náročné umístění jednotlivých vrcholů.

  Statický layout je vhodný zejména pro vizualizaci grafů na mapových podkladech.

- **kruhový layout** – všechny vrcholy jsou uspořádány do kruhu. Implementačně je poměrně jednoduchý, ale může být problém vrcholy na kruh umístit tak, aby se hrany zbytečně nekřížily. U silně souvislých grafů není tento layout přehledný.

  Případně lze využít různé modifikace kruhového layoutu, například uspořádání do hvězdy, trojúhelníka, krychle atd. Další ukázky podobných layoutů ve 2D i ve 3D lze najít v literatuře[2].

- **lineární layout** – všechny vrcholy jsou umístěny na pomyslné přímce. Hrany jsou pak vedeny mezi jednotlivými vrcholy oblouky.

  Tento layout se například využívá pro vizualizaci Markovských modelů.

- **mřížkový (maticový) layout** – všechny vrcholy jsou umístěny v mřížce. Pokud jsou stupně vrcholů větší než čtyři, bude pravděpodobně tento layout nepřehledný.

- **organický (pružinový) layout** – všechny vrcholy se navzájem odpuzují, u sebe je drží jen hrany. Implementačně asi nejsložitější způsob.

Používá se například pro vizualizaci sociálních (a jiných) sítí. Nevýhodou pak je, že po přidání či odebrání vrcholů se může uspořádání celého grafu úplně změnit. Pak je pro člověka obtížnější se v daném grafu orientovat.

- **stromový a hierarchický layout** – použití především u stromů a u orientovaných grafů, vrcholy předků jsou nad vrcholy potomků. U hierarchického layoutu na rozdíl od stromového může mít vrchol více předků. Stromový layout je na vizualizaci jednoduchý, u hierarchického uspořádání může nastat podobný problém jako u kruhového, tedy mohou se zde zbytečně křížit hrany. Ideální pro tento layout jsou tedy rovinné grafy[1].

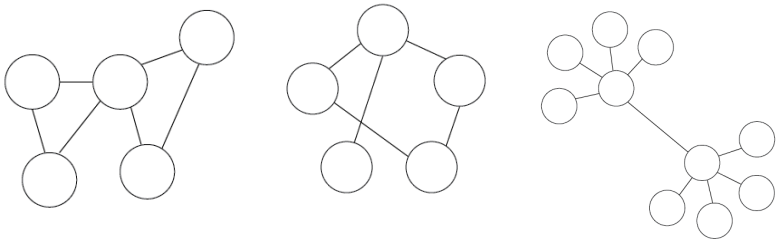
  Typickým příkladem může být vizualizace rodokmenu.

- **náhodný layout** – poloha vrcholů je náhodně určena počítačem. Nemá přílišné praktické použití, ale je nejjednodušší.

  Tento layout je možné využít například při testování.

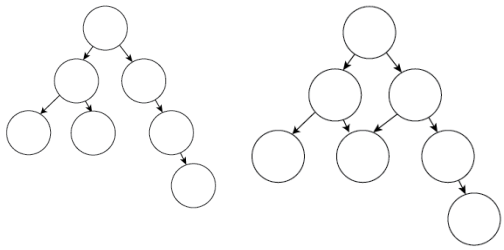
Na obrázcích 2.4 a 2.5 je pro srovnání vizualizace velkých grafů v některých z výše uvedených layoutů.

Pokud je graf zobrazován v počítačové aplikaci (tj. není vytištěný či ve formě obrázku), je možné graf vizualizovat i ve 3D, což může v některých případech zvýšit přehlednost. Pro vizualizaci ve 3D lze využít všechny výše zmíněné layouty (některé po patřičných modifikacích).

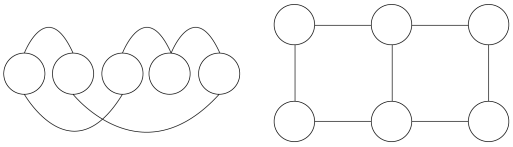


Obrázek 2.1: Náhodný/statický, kruhový a organický layout (zleva)

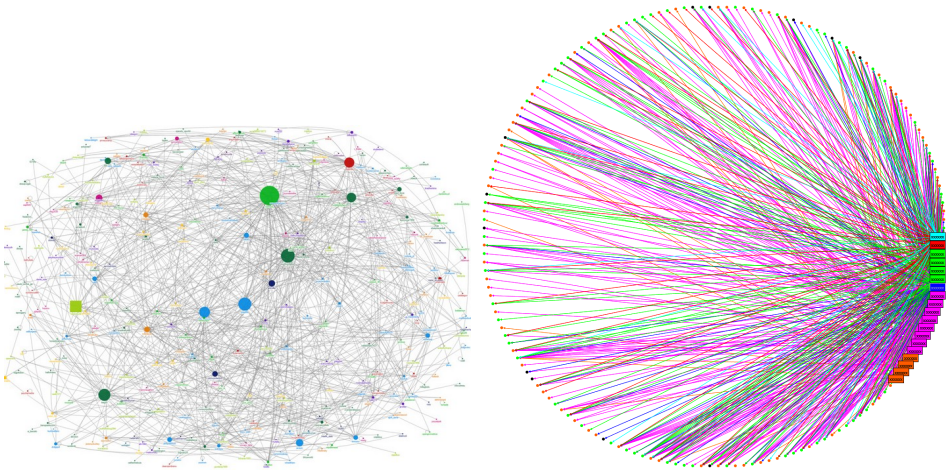
[1]Rovinný graf je graf, který lze v rovině nakreslit tak, aby se žádné hrany nekřížily.



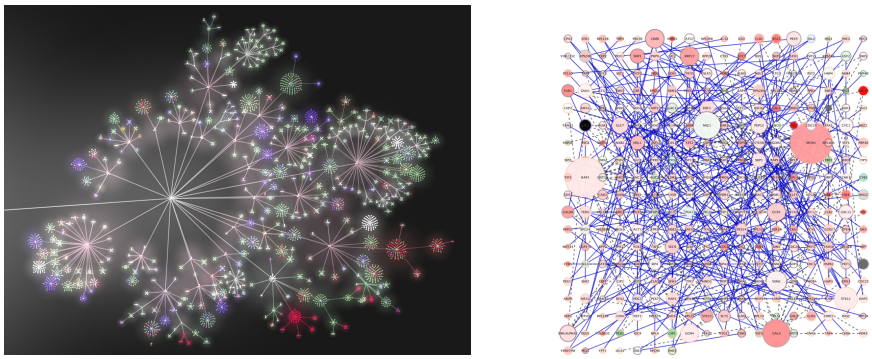
Obrázek 2.2: Stromový a hierarchický layout (zleva)



Obrázek 2.3: Lineární a mřížkový layout (zleva)



Obrázek 2.4: Velké grafy ve statickém[3] a kruhovém[4] layoutu (zleva)



Obrázek 2.5: Velké grafy v organickém[5] a mřížkovém[6] layoutu (zleva)

Pokud uživatele při vizualizaci zajímá topologie grafu, je vhodné, aby vykreslený graf co možná nejlépe dodržoval následující doporučení[7] (ne vždy lze dodržet všechna tato doporučení, například, pokud je graf vykreslen na mapových podkladech a pozice jednotlivých vrcholů jsou staticky určeny, pak většinu z těchto doporučení dodržet nelze.):

- hrany grafu by se měly co nejméně křížit
- hrany by měly být přímé, ideálně bez jakýchkoliv zahnutí
- graf by měl být vykreslen na co nejmenší ploše
- hrany by měly být co nejkratší
- úhel mezi jednotlivými hranami vedoucími z jednoho vrcholu by měl být co největší
- u velkých grafů by vrcholy patřící k sobě měly být vykresleny do shluků poblíž sebe

Konkrétní způsob vizualizace libovolného grafu se vždy liší v závislosti na tom, co daný graf vyjadřuje. Například: Pro vizualizaci silniční sítě je vhodné použít statické uspořádání, protože jednotlivé silnice a křižovatky mají konkrétní polohu, což je pro člověka důležitá informace ve stejné míře jako propojení jednotlivých silnic.

U vizualizace kruhů přátel na sociální síti je naopak vhodné využít organické uspořádání, protože informace o poloze člověka v tomto případě není podstatná a důležité je právě propojení jednotlivých lidí.

Níže uvádím ještě stručný výpis dalších pokročilých vizualizačních metod.

- **OpenOrd Layout**[8] – velmi rychlý 3D organický layout umožňující vizualizaci velkého množství vrcholů, shlukuje vrcholy podobné váhy.
- **Fruchterman-Reingold layout**[9] – 3D organický layout simulující částice ve fyzikálním systému, snažící se minimalizovat energii tohoto systému
- **ForceAtlas 2 layout**[9] – 3D organický layout, u něhož se odpudivé síly jednotlivých vrcholů řídí Barnes-Hutovým výpočtem.
- **Radial Axis layout**[10] – 2D kruhový layout, který vykreslí část grafu na kruh a zbylé vrcholy na oblouky vedoucí ven z tohoto kruhu.
- **GeoLayout**[11] – 2D layout, který je v podstatě synonymem pro statický layout – všechny vrcholy mají pevně danou zeměpisnou délku a šířku.

2.1 Alternativní způsoby vizualizace grafů

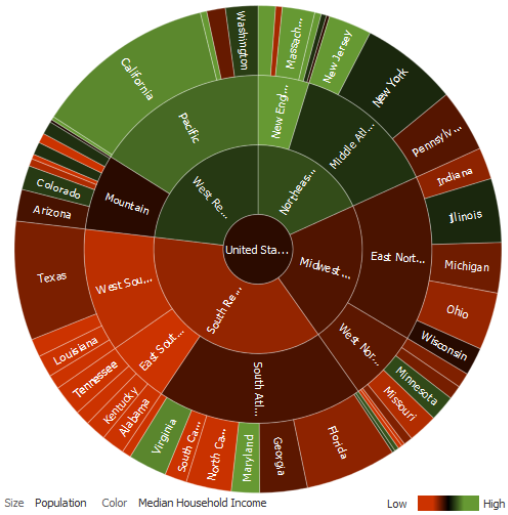
Kromě těchto způsobů vizualizace grafů existují i další způsoby, kterými lze graf reprezentovat a které mají v určitých případech své využití.

2.1.1 Vizualizace stromů

Všechny stromy jsou orientované grafy, které neobsahují smyčky. Kořen stromu je vrchol, do kterého nevedou žádné hrany. Do všech ostatních vrcholů vede právě jedna hrana. List stromu je vrchol z nějž nevede žádná hrana.

Kruhové uspořádání

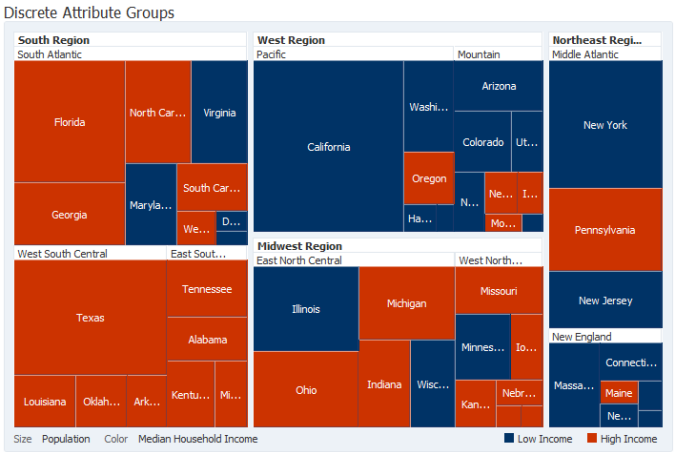
Výše uvedených faktů lze využít pro vykreslení stromu do kruhového uspořádání. Strom je pak vykreslen jako řada soustředných kružnic, jejichž počet odpovídá počtu úrovní stromu. Vznikne tak kruh, který obklopuje řada pásů. Středový kruh představuje kořen, následující pás se rozdělí na tolik částí, kolik má kořen potomků atd (viz obr. 2.6).



Obrázek 2.6: Kruhová vizualizace stromu [12]

Obdélníkové uspořádání

Podobně lze vizualizovat strom i do obdélníku, kdy obdélník samotný představuje kořen a uvnitř je rozdělen na menší obdélníky, které představují přímé potomky atd. (viz obr. 2.7).



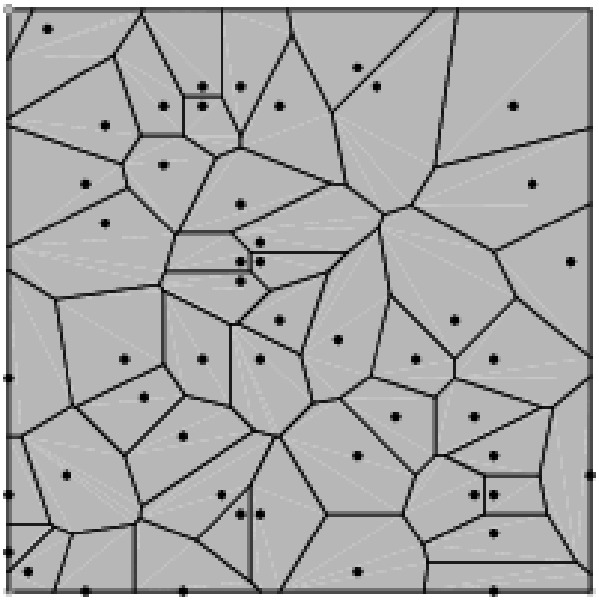
Obrázek 2.7: Obdélníková vizualizace stromu [12]

2.1.2 Voroného diagramy

Voroného diagram[13] rozděluje plochu s body na podoblasti takovým způsobem, že hranice těchto podoblastí leží vždy přesně mezi danými dvěma body (viz obr. 2.8).

Graf pak může propojovat body, které určující jednotlivé hranice, nebo může být reprezentován samotnými hranicemi.

Voroného diagramy lze využít například pro generování přirozeně vypadajícího terénu v počítači (viz [14]).



Obrázek 2.8: Voroného diagram[13]

2.2 Aplikace pro vizualizaci grafů

V této kapitole se nachází stručný přehled několika univerzálních i specializovaných aplikací schopných vizualizace grafů a stromů.

2.2.1 Gephi

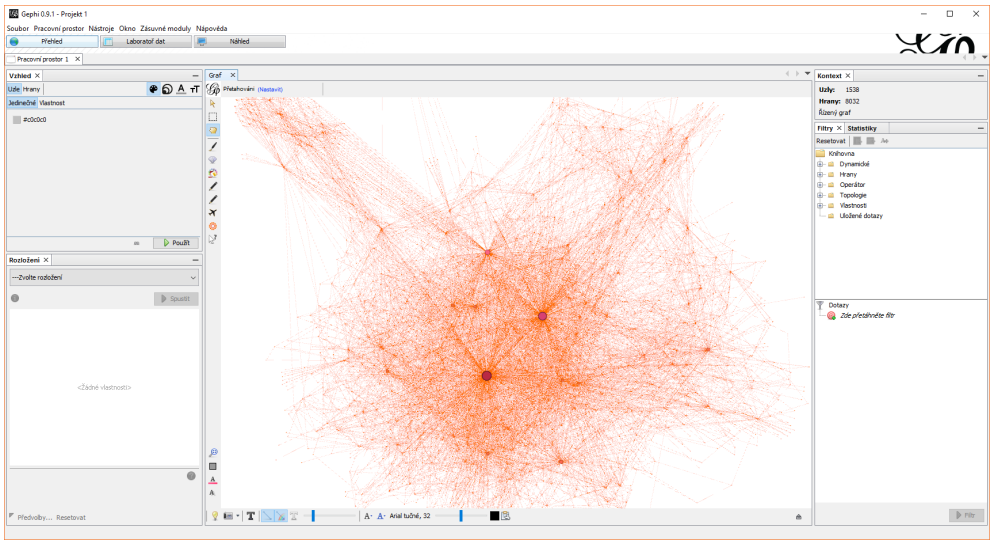
Název	Gephi
Licence	GNU GPLv3, CDDL 1.0
Poslední verze	1. 4. 2016
Podporované layouty	OpenOrd, ForceAtlas 2, Radial Axis, Geo Layout, Fruchternab-Reingold, kruhové, organické a mnoho dalších...
Autor	Gephi konsorcium
Web	`https://gephi.org`

Tabulka 2.1: Gephi

Gephi[15] je pokročilý univerzální nástroj pro vizualizaci a úpravu dat velkých grafů vyvíjený Gephi konsorciem. Podporuje širokou škálu různých funkcí, například editaci dat grafu, vizualizaci různými layouty, vizualizaci grafu v čase, import, export, výpočet nejrůznějších statistik, změnu vzhledu vrcholů a hran a mnoho dalších funkcí. Aplikace je velmi přehledná a intuitivní.

Gephi je přeloženo do několika jazyků včetně češtiny a je vyvíjeno pod open source licencemi GNU GPLv3 a CDDL 1.0.

Tato aplikace používá k zobrazení velkých grafů metodu změny velikosti vrcholů na základě přiblížení.



Obrázek 2.9: Hlavní okno aplikace Gephi

2.2.2 Gource

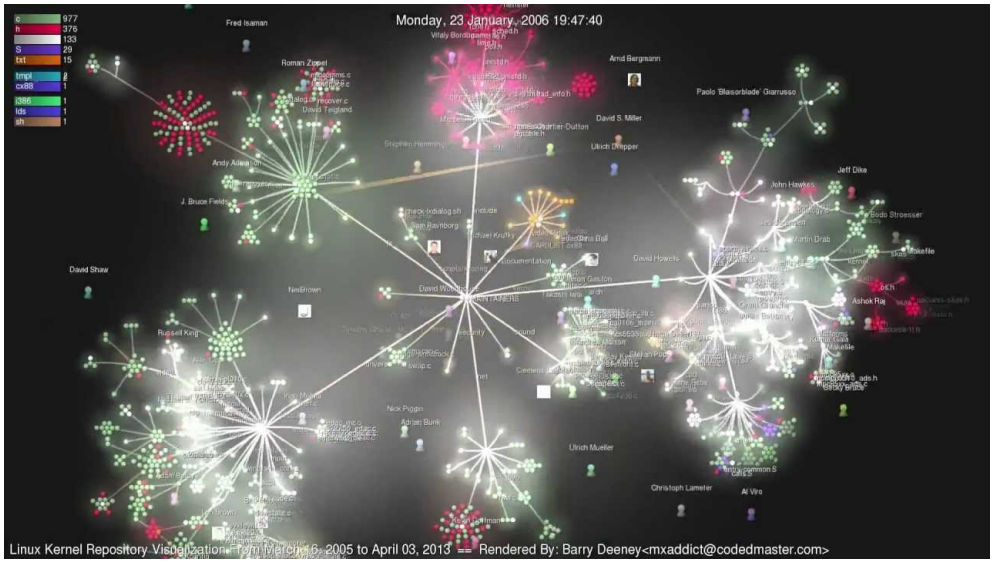
Název	Gource
Licence	GNU GPL
Poslední verze	3. 2. 2016
Podporované layouty	Organické
Autor	Andrew Caudwell
Web	`http://gource.io`

Tabulka 2.2: Gource

Gource[16] je jednoduchá aplikace sloužící k vizualizaci vývoje různých projektů v čase. Gource získává data z libovolného `Git` repozitáře.

Aplikace vykresluje soubory repozitáře jako stromovou strukturu v animaci, která se mění v průběhu času. Gource je vydaný pod licencí GNU GPL a jako takový nemá uživatelské rozhraní. Všechny volby je nutné specifikovat jako argumenty programu a následně se v okně spustí animace. Tu je možné libovolně pozastavovat a měnit v ní přiblížení.

Pro vizualizaci vekých grafů používá Gource metodu shlukování vrcholů představující soubory, které nebyly dlouho změněny, do jednoho vrcholu.



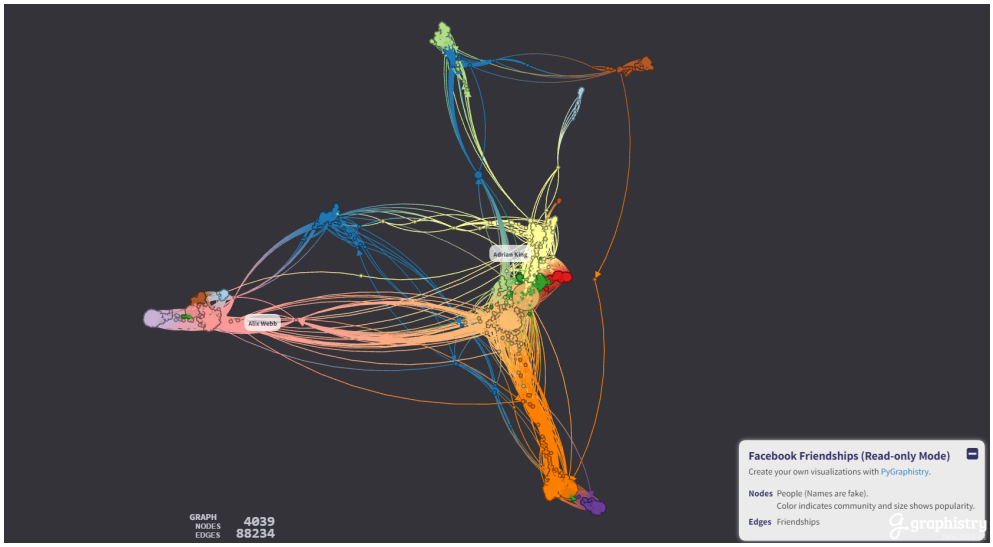
Obrázek 2.10: Vizualizace vývoje Linuxového jádra v aplikaci Gource [17]

2.2.3 PyGraphistry

Název	PyGraphistry
Licence	opensource
Poslední verze	20. 5. 2016
Podporované layouty	ForceAtlas 2
Autor	komunitní projekt
Web	`https://github.com/graphistry/pygraphistry`

Tabulka 2.3: PyGraphistry

Nástroj PyGraphistry[18] napsaný v jazyce Python umožňuje načítat, zpracovávat a následně vykreslit enormně velké grafy představující převážně sociální sítě. Samotná vizualizace probíhá ve 3D v prohlížeči s využitím WebGL. Graf lze libovolně posouvat a přibližovat.



Obrázek 2.11: Vizualizace nástrojem PyGraphistry

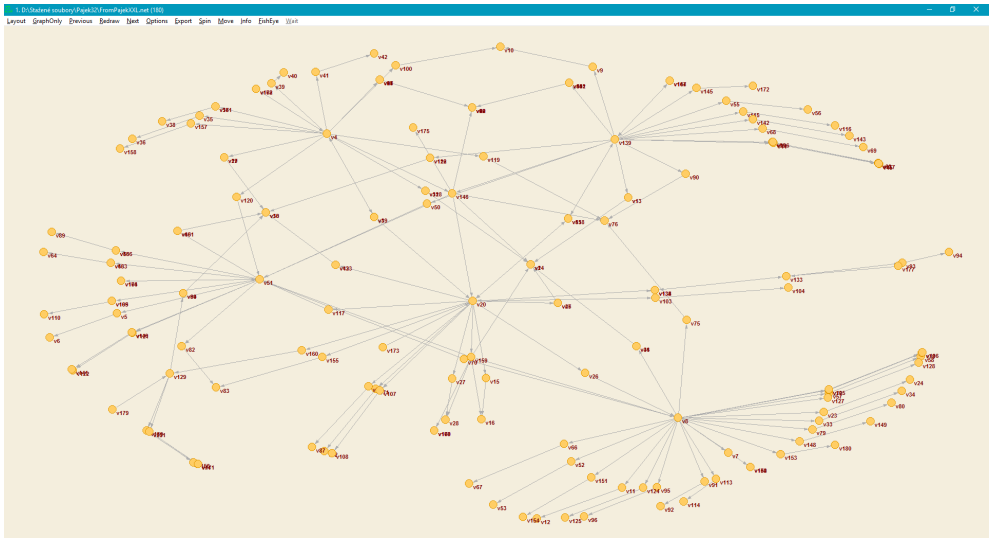
2.2.4 Pajek

Název	Pajek
Licence	zdarma pro nekomerční použití
Poslední verze	20. 5. 2016
Podporované layouty	kruhové, Fruchternab-Reingold, náhodné a další včetně mnoha variant těchto layoutů
Autor	Vladimir Batagejl a Andrej Mrvar
Web	`http://mrvar.fdv.uni-lj.si/pajek/`

Tabulka 2.4: Pajek

Pajek[19] je nástroj na analýzu velkých grafů (podporuje maximálně jednu miliardu vrcholů) a jejich vizualizaci. Tento program je zdarma pro nekomerční použití.

Pajek podporuje velké množství vizualizačních uspořádání. Dále také umožňuje vizualizované grafy upravovat a exportovat je ve 2D i 3D.



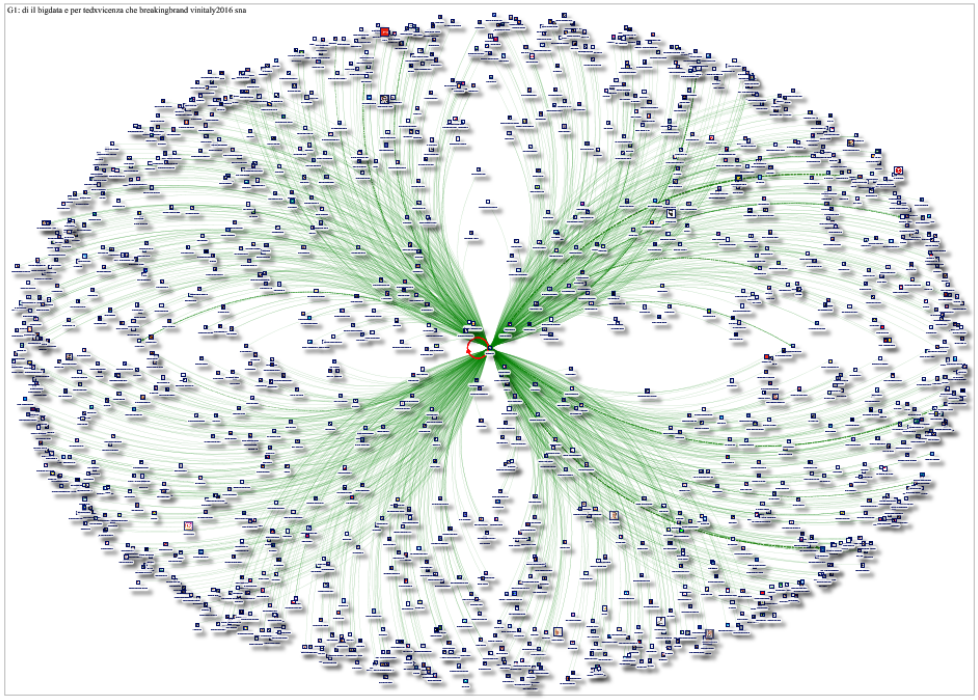
Obrázek 2.12: Vizualizace grafu programem Pajek

2.2.5 NodeXL

Název	NodeXL
Licence	Ms-PL
Poslední verze	26. 1. 2016
Podporované layouty	různé druhy organických layoutů
Autor	Social Media Research Foundation
Web	`http://nodexl.codeplex.com`

Tabulka 2.5: NodeXL

Doplněk do aplikace Microsoft Office Excel s názvem NodeXL[20] umožňující vizualizaci velkých grafů v této aplikaci.



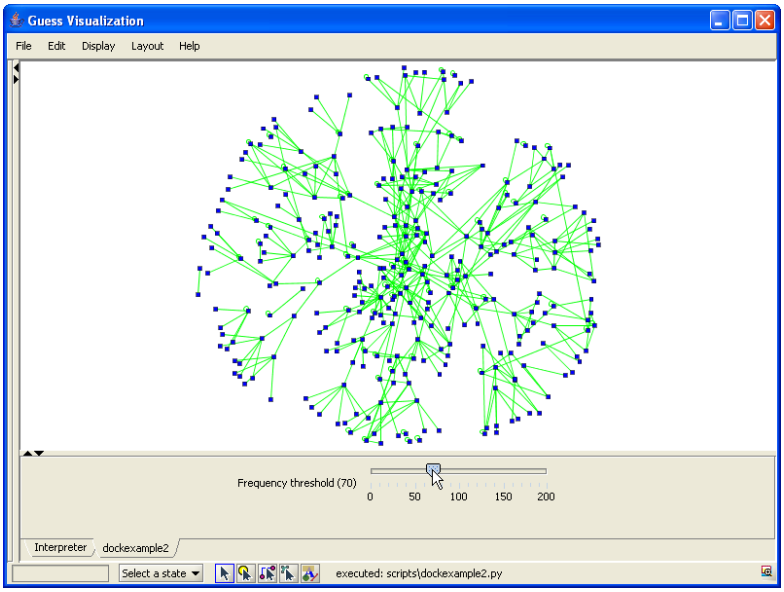
Obrázek 2.13: Vizualizace grafu doplňkem NodeXL[21]

2.2.6 GUESS

Název	GUESS
Licence	GNU GPL
Poslední verze	13. 8. 2007
Podporované layouty	organické a kruhové layouty
Autor	Eytan Adar
Web	`http://graphexploration.cond.org/index.html`

Tabulka 2.6: GUESS

GUESS[22] je nástroj určený pro analýzu a vizualizaci grafů. Obsahuje doménově-orientovaný jazyk Gython založený na Pythonu.



Obrázek 2.14: Vizualizace grafu nástrojem GUESS[22]

2.3 Vizualizace velkých grafů

Velké grafy (řádově stovky a více vrcholů) narozdíl od malých (jednotky až desítky vrcholů) většinou není možné celé vykreslit najednou tak, aby byly vidět všechny vrcholy a hrany a výsledný graf byl stále přehledný. Vrcholy se totiž většinou překrývají, což graf značně znepřehledňuje.

Proto je nutné graf dynamicky měnit tak, aby zůstal stále stejně přehledný.

2.3.1 Tištěný graf

Pokud má být graf vytištěný (nebo má být uložen jako statický obrázek), tak bohužel není většinou možné rozumně graf vizualizovat takovým způsobem, aby byly přehledně vidět všechny vrcholy. Lze využít následující metody:

Graf v celku

Menší grafy je možné vytisknout na dostatečně velký papír. Případně lze vypustit některé vrcholy s nižším ohodnocením (pokud to daná situace umožňuje) tak, aby se ve výsledném zobrazení pokud možno žádné vrcholy nepřekrývaly.

Klad listů

U větších grafů je možné vytvořit klad listů, který bude obsahovat náhled grafu spolu s čísly stránek obsahující detailní pohled.

Klad listů může být původní nezměněný graf, což ale zejména u větších grafů bude působit chaoticky. Nebo je možné klad listů upravit v závislosti na konkrétním případě.

Jednou možnou úpravou je nahrazení podgrafů s nižším ohodnocením jedním vrcholem. Tento způsob je využíván u hierarchických grafů (např. UML model celé aplikace...). Naopak nejde využít u neohodnocených grafů.

Další možností je nahradit podgrafy, jejichž vrcholy jsou blízko u sebe, jedním vrcholem. Tuto metodu lze využít i u neohodnocených grafů, zejména v případech, kdy je pro uživatele klíčová poloha vrcholů.

Omezit počet vizualizovaných vrcholů

Pokud lze některé vrcholy vynechat, je možné vykreslit graf bez těchto vrcholů. Tím se graf může zjednodušit a může být vykreslen se zachovanou přehledností.

2.3.2 Zobrazení grafu v počítači

Narozdíl od tištěného grafu lze v počítači vizualizovat celý graf mnohem přehledněji, protože uživatel může libovolně měnit přiblížení a oblast zobrazeného grafu. Počítačové zobrazení grafu tak navíc uživateli nabízí určitou interaktivitu.

Interakce

Uživatel se může táhnutím myši snadno pohybovat po celém grafu. Kolečkem myši pak může měnit stupeň přiblížení. Dále si může zobrazovat i různé podrobnější informace o daném vrcholu, které není možné znázornit graficky. Stejně tak může jednotlivé vrcholy a hrany upravovat, měnit uspořádání grafu a mazat jednotlivé hrany a vrcholy.

Pokud graf není přiblížený, uživatel vidí celkový náhled grafu. Po přiblížení si uživatel může prohlížet detaily grafu.

Shlukování vrcholů

Celkový pohled na graf lze zjednodušit tím, že se určité podgrafy nahradí jedním vrcholem.

Tato varianta je ve většině případů nejlepší, protože, když má uživatel graf oddálen, vidí důležité vrcholy a zároveň vidí, že zde existují i vrcholy, které jsou méně důležité a zobrazí se až po přiblížení. Po přiblížení jsou pak vidět všechny vrcholy.

Tyto virtuální vrcholy pak musí být propojeny hranami se zbývajícím grafem tak, aby byly zachovány všechny cesty.

Způsobů k určení podgrafů, které mají být nahrazeny, je několik:

Neohodnocený graf – Protože u neohodnoceného grafu není možné určit důležitost jednotlivých vrcholů přímo podle ohodnocení, je nutné pro shlukování využít jejich topologii nebo v případě statického layoutu jejich umístění. Proto je vhodné hledat větší podgrafy, jejichž vrcholy jsou blízko sebe a ty pak nahradit jedním vrcholem.

Tuto metodu není možné použít, pokud je vizualizace grafu nějakým způsobem pravidelná (graf je vykreslen v mřížce či na kružnici), kdy není možné určit shluky tak, aby jejich uspořádání mělo smysl.

Ohodnocený graf [2] – Podobně jako u neohodnoceného grafu je možné určité podgrafy nahradit jediným vrcholem. Ovšem v tomto případě lze využít pro samotné shlukování i ohodnocení vrcholů. Sousedící vrcholy se shodným ohodnocením tak mohou být nahrazeny jedním vrcholem.

Možným vylepšením je znázornění počtu vrcholů, které daný vrchol zastupuje, jeho velikostí, číslem nebo barvou.

Změna velikosti vrcholů

Pokud je u vrcholů znázorněno jejich ohodnocení velikostí, pak je možné velikost jednotlivých vrcholů při oddálnení zmenšovat a při přiblížení naopak zvětšovat. Při minimálním přiblížení pak budou méně významné vrcholy téměř neviditelné, ale zároveň budou vidět veškeré hrany, které uživateli napoví o existenci těchto vrcholů.

Jedná se o implementačně velmi jednoduchou metodu, která zachovává při všech úrovních přiblížení stejnou přehlednost.

[2] Graf nemusí být explicitně ohodnocený, stačí, když jsou jednotlivé vrcholy rozděleny do určitých vrstev, u kterých lze určit jejich významnost.

Přiblížení části grafu

Další variantou je zvětšovat pouze oblast kolem myši uživatele – podobným způsobem jako se chová aplikace Lupa v systému Windows. Na první pohled uživatel vidí celkový přehled o grafu a myší si může prohlížet detail určité části. Kolečkem myši by bylo možné měnit úroveň přiblížení dané oblasti.

Automatické skrývání vrcholů

V této variantě se uživateli zobrazují pouze vrcholy, které odpovídají danému přiblížení a všem menším přiblížením. Toto řešení je přehledné, ale při malém přiblížení se ztrácí informace o vrcholech s nižším ohodnocením.

Manuální vybírání vrstev

Jednodušší variantou tohoto případu je aplikace, kde si uživatel ručně přepíná mezi jednotlivými vrstvami (každá vrstva představuje vrcholy s jedním konkrétním ohodnocením).

Tato metoda je jednodušší na implementaci a pokud mají jednotlivé vrstvy shodné ohodnocení, je tato metoda nejvhodnější, protože narozdíl od ostatních metod nespoléhá na různé ohodnocení všech vrstev.

Tato metoda na druhou stranu není tak uživatelsky přívětivá a přehledná jako výše zmíněné varianty. Navíc podobně jako u předchozího příkladu, při skrytí určitých vrstev se ze souvislého grafu může stát graf nesouvislý a tím se ztratí informace o propojení vzniklých komponent.

2.3.3 Souvislost s vizualizací mapových podkladů

Obdobnou problematiku vizualizace velkých grafů lze pozorovat i u vizualizace mapových podkladů. Částečně protože mapa může být reprezentována jako graf, kde obce jsou vrcholy a silnice mezi nimi jsou hrany.

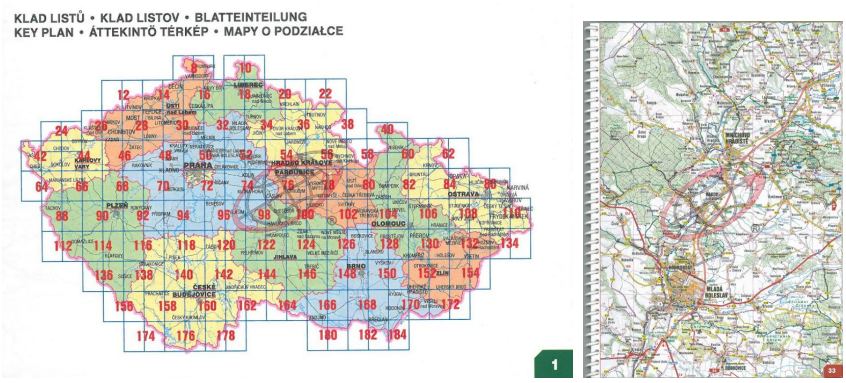
Tištěné mapy

Jak bylo zmíněno výše, ani u map nelze vykreslit např. celou Českou republiku včetně všech vesnic a silnic třetí třídy na rozumně velký papír. Proto je také na začátku všech podrobnějších atlasů klad listů, na němž je vykreslena celá Česká republika pouze s největšími městy a dálnicemi, na které je mřížka s čísly označujícími konkrétní stránky atlasu obsahující detailní pohled na danou oblast (viz obr. 2.15).

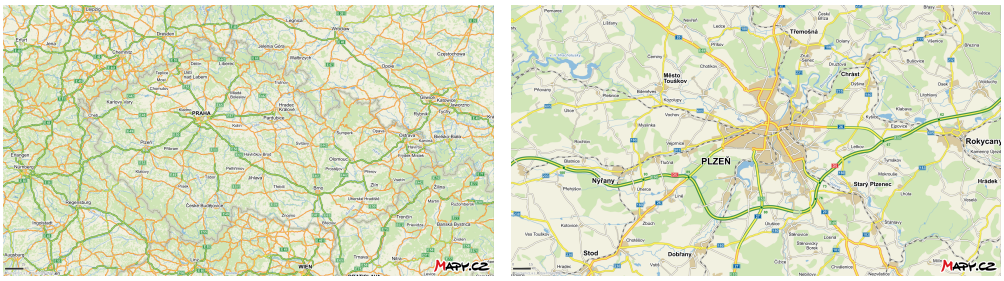
Mapy v počítači

Mapy v počítači nejprve zobrazí např. celou Českou republiku podobně jako klad listů a uživatel následně může přibližovat na konkrétní oblasti a volně se mezi nimi pohybovat. (viz obr. 2.16)

Výhodou oproti tištěným mapám je libovolná přesnost map. Tištěné mapy mají dané pouze jedno měřítko, kdežto počítačové mapy mohou přizpůsobovat měřítko úrovni přiblížení. Stejně tak poskytují i větší pohodlí, protože dnešní mapové portály mají snadné ovládání a umožňují vyhledávání.



Obrázek 2.15: Kladu listů (vlevo) a detail jednoho listu (vpravo) autoatlasu [23]



Obrázek 2.16: Ukázka pohledu na celou ČR (vlevo) a detailu Plzně (vpravo) [24]

2.4 Návrh řešení vizualizace velkých grafů

V této sekci se zaměřím na statický layout, který má význam při zobrazení grafu na mapovém podkladu. To bude využito při implementaci do projektu Vizualizace přenosové sítě.

2.4.1 Viditelnost vrstev v závislosti na přiblížení

Nejprve je nutné určit, na které úrovni přiblížení budou vidět které vrcholy či vrstvy (případně zda bude daný podgraf sloučen do jediného vrcholu).

Určení viditelnosti matematickou funkcí

Tento postup vyžaduje funkci, která bude přepočítávat úroveň přiblížení na ohodnocení zobrazovaných vrcholů. V případě nejistého rozsahu přiblížení a/nebo ohodnocení vrcholů je nutné upravit program, aby parametry funkce přizpůsoboval daným rozsahům.

Využít lze například tuto jednoduchou funkci pro lineární mapování jednoho rozsahu na jiný:

$$y = w_{max} - \frac{a}{z_{max} - z_{min}} \times (w_{max} - w_{min})$$

kde w_{max} a w_{min} je maximální a minimální ohodnocení vrcholů, proměnné z_{min} a z_{max} představují minimální a maximální hodnoty přiblížení a a je aktuální úroveň přiblížení. Vrchol pak bude zobrazen, pokud jeho ohodnocení bude větší nebo rovno y.

Př: Pro úrovně přiblížení v rozsahu 5 - 20 a ohodnocení vrcholů 3000 - 8000 by rovnice vypadala takto:

$$y = 8000 - \frac{a}{20-5} \times (8000 - 3000)$$

V některých případech nelze využít funkci a je nutné ošetřit správné zobrazování vrcholů programově podle potřeby.

Parametry funkce je vždy třeba přizpůsobit velikostem vrcholů a vzdálenostem mezi nimi.

Výhodou této metody je jednoduchost implementace a možnost pracovat i s grafy, které mají vrcholy s velmi rozmanitým ohodnocením.

Pokud se má řídit zobrazování jednotlivých vrstev pravidly, která nelze vyjádřit funkcí, je nutné určovat viditelnost vrstev programově nebo manuálně.

Manuální určení viditelnosti

V tomto případě je nutné manuálně přiřadit jednotlivé vrstvy k daným úrovním přiblížení.

Nevýhodou tohoto řešení je nutnost ručního přiřazení úrovní přiblížení a jednotlivých vrstev. Naopak výhodou je jednoduchost implementace.

Programové určení viditelnosti

Pokud se zobrazování jednotlivých vrstev řídí složitějšími pravidly a graf obsahuje velké množství vrstev, pak je možné určovat, kdy mají být dané vrstvy zobrazeny, programově. Program narozdíl od předchozích dvou metod může brát v potaz mimo ohodnocení uzlů i další parametry.

2.4.2 Automatické skrývání vrcholů

U metody skrývání vrcholů stačí pak už jen zobrazovat či skrývat vrcholy na základě postupů uvedených výše. Hrana by měla být skryta, pokud je skryt alespoň jeden z vrcholů, které spojuje.

Protože tato metoda může skrýt jednotlivé cesty v grafu, je možné zobrazovat hrany bez vrcholů tak, aby zde byl vidět alespoň náznak, že se zde vyskytují nějaké skryté vrcholy a hrany.

2.4.3 Shlukování vrcholů

U této metody se nahrazují určité podgrafy virtuálními vrcholy a hranami, které v původním grafu nejsou uloženy, ale v případě potřeby jsou vykresleny.

Virtuální vrchol představuje skupinu několika vrcholů či podgraf, jehož ohodnocení není dost vysoké na to, aby mohl být zobrazen.

Virtuální hrany propojují virtuální vrcholy se zbytkem grafu, konkrétně s vrcholy, s nimiž byl spojen původní podgraf. Ve výsledném grafu povede mezi dvěma vrcholy vždy právě jedna hrana – proto v případech, kdy více vrcholů z daného podgrafu bylo propojeno s jedním vrcholem, bude virtuální vrchol s daným vrcholem propojen pouze jednou hranou.

Ručně určené virtuální vrcholy a hrany

Spolu s celým grafem jsou uloženy i tyto virtuální vrcholy a hrany s příslušným příznakem. Pak jsou vykreslovány stejným způsobem jako všechny ostatní vrcholy a hrany s tím rozdílem, že jsou vykresleny pouze tehdy, pokud mají stejné ohodnocení jako zobrazené vrcholy s nejnižším ohodnocením.

Výhodou je jednoduchost implementace a její rychlost – nic se nemusí počítat, graf se pouze vykreslí.

Nevýhodou je, že pro celý graf musí být tyto virtuální vrcholy a hrany určeny předem a uloženy s celým grafem. Což je nevýhodné obzvláště u grafů, které se často mění.

Dynamicky určované virtuální vrcholy a hrany

Toto o dost komplexnější řešení vyžaduje algoritmus, jenž je schopen určit podgrafy, které je nutné nahradit virtuálními vrcholy a hranami. Konkrétní algoritmus se může lišit podle dané situace.

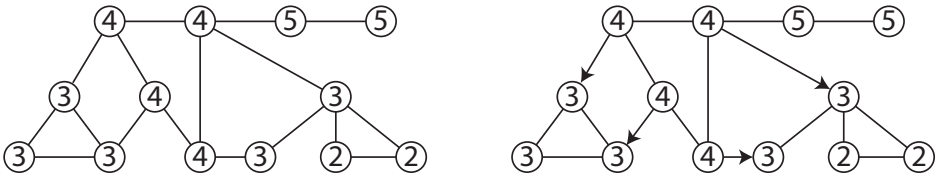
K nalezení virtuálních vrcholů u neorientovaného grafu lze využít například `Tarjanův algoritmus`[25]. Ten pracuje na principu DFS (prohledávání grafu do hloubky). Tarjanův algoritmus se používá pro hledání silně souvislých komponent v orientovaném grafu.

Silně souvislá komponenta je takový podgraf v grafu, v němž se lze dostat z libovolného vrcholu do jiného libovolného vrcholu v daném podgrafu.

Nejprve je nutné určit u grafu různé vrstvy na základě ohodnocení jednotlivých vrcholů (vrcholy se stejným/podobným ohodnocením budou na stejné vrstvě).

Máme-li graf s n vrstvami, pak je nutné algoritmus spustit $n - 1$ krát (pro všechny vrstvy kromě té s nejnižším ohodnocením).

Protože v tomto případě nepracujeme s orientovaným grafem a nehledáme přímo silně souvislé komponenty, je třeba nejprve vstupní graf (který je neorientovaný, viz obr. 2.17 vlevo) upravit na orientovaný graf podle následujících pravidel (k je ohodnocení vrstvy, pro kterou graf aktuálně zpracováváme, viz obr. 2.17 vpravo).



Obrázek 2.17: Původní graf a jeho příprava

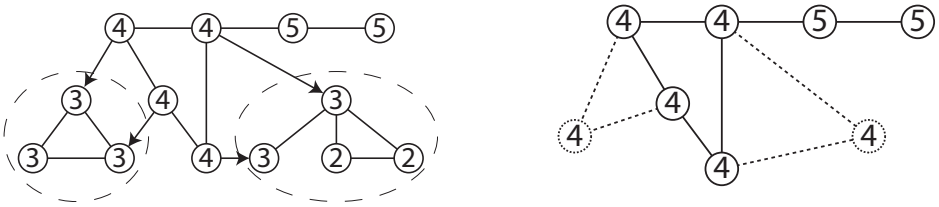
Pro každé dva sousední vrcholy A a B v daném grafu:

- **je-li ohodnocení obou vrcholů menší než** k – pak propojit A a B obousměrně
- **je-li ohodnocení obou vrcholů větší než** k – pak propojit A a B obousměrně
- **je-li ohodnocení obou vrcholů rovné** k – pak propojit A a B obousměrně
- **je-li ohodnocení** $A < k$ **a ohodnocení** $B \geq k$ – pak vést hranu z B do A
- **je-li ohodnocení** $A \leq k$ **a ohodnocení** $B > k$ – pak vést hranu z B do A
- **v ostatních případech** – vést hranu z A do B

Výstupem algoritmu jsou pak vrcholy rozdělené do jednotlivých komponent podle jejich ohodnocení.

Dále je nutné vyloučit ty komponenty, jejichž ohodnocení je větší nebo rovno k (viz obr. 2.18 vlevo). Tyto vrcholy budou na dané vrstvě normálně vykresleny. Pak každá zbývající komponenta představuje jeden virtuální vrchol s ohodnocením k.

Nakonec je potřeba propojit jednotlivé virtuální vrcholy s ostatními vrcholy (viz obr. 2.18 vpravo) a určit správnou pozici virtuálních vrcholů. Tu lze například určit zprůměrováním pozic všech vrcholů v dané komponentě. Pseudokód algortimu viz Zdrojový kód 2.1.



Obrázek 2.18: Nalezení komponent a jejich nahrazení virtuálními vrcholy

Složitost tohoto algoritmu je přibližně $O((l-1) \times (n+2e))$, kde l je počet vrstev, n počet vrcholů a e počet hran daného grafu. Složitost nezahrnuje vyloučení komponent a propojení virtuálních vrcholů se zbytkem grafu.

Výhodou tohoto řešení je, že uživatel není nucen při každé změně grafu ručně doplňovat všechny virtuální vrcholy a hrany.

Nevýhodou je velká výpočetní a časová náročnost algoritmu (především u velkých grafů s velkým počtem vrstev). Není vhodné, aby uživatel čekal na provedení přiblížení několik vteřin, či dokonce minut. Což dělá tuto metodu samu o sobě ve většině případů nepoužitelnou.

Kombinace obou výše uvedených řešení

Pokud se graf nemění příliš často, je nejvhodnější vždy při úpravě grafu programově najít virtuální vrcholy a hrany, uložit je do databáze a následně je normálně vykreslovat.

Tím uživateli odpadá nutnost ručně hledat virtuální vrcholy a hrany a zároveň samotné vykreslování grafu netrvá dlouho. Tato metoda je tedy zřejmě nejvhodnější pro případy, kdy je graf větší a mění se, ale ke změnám nedochází příliš často.

```
graf: findCycles1(graf G, int Vrstva)
  graf tmp = G.clone()
  tmp.removeAllEdges()
  for (X : G.getAllEdges) then
    int vaha1 = X.node1.getWeight
    int vaha2 = X.node2.getWeight
    if (((vaha1 > Vrstva) && (vaha2 > Vrstva)) || ((vaha1 < Vrstva) && (vaha2 < Vrstva)) || ((vaha1 == Vrstva) && (vaha2 == Vrstva))) then
        tmp.addEdge(X.node1.getId(), X.node2.getId())
        tmp.addEdge(X.node2.getId(), X.node1.getId())
      end else if (((vaha1 < Vrstva) && (vaha2 >= Vrstva)) || ((vaha1 <= Vrstva) && (vaha2 > Vrstva))) then
        tmp.addEdge(X.node2.getId(), X.node1.getId())
      end else then
        tmp.addEdge(X.node1.getId(), X.node2.getId())
      end
  end
  TarjanAlg A
  A.run(tmp)
  for (Y : A.getComponents) then
    if (Y.getMaxWeight >= Vrstva) then
      return
    end
    Node N = G.addNode
    N.setVirtual(true)
    for (Z : Y.getNodes) then
      for (E : Z.getEdges) then
        Node other = E.getOther(Z)
        if (other.isVirtual) then
          break
        end
        if (other.getWeight > Z.getWeight && not G.hasNode(N, other)) then
          G.addEdge(N, other)
        end
      end
    end
  end
end
```

Zdrojový kód 2.1: Pseudokód hledání podgrafů s nižším ohodnocením

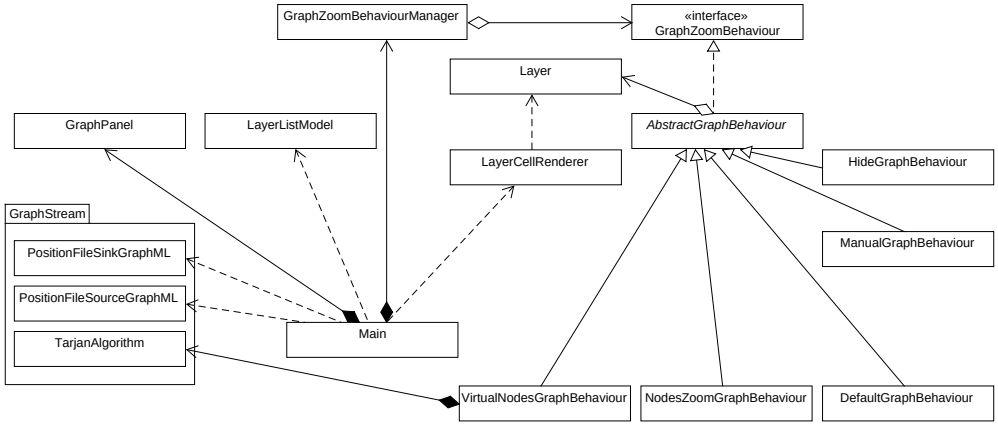
3 Ukázková aplikace

Na ukázku vybraných metod jsem vytvořil jednoduchou aplikaci, která demonstruje jednotlivé způsoby vizualizace velkých grafů. Jeden ze způsobů jsem pak implementoval v projektu Vizualizace přenosové sítě.

Jedná se o jednoduchou aplikaci v jazyce Java, která s využitím knihovny `GraphStream`[26] nabízí možnost vyzkoušet si výše uvedeném metody vizualizace velkých grafů v praxi.

Pro uživatelské rozhraní jsem použil zastaralý `Swing`, protože pro novější `JavaFX` zatím neexistuje žádná vhodná knihovna podporující vizualizaci grafů.

3.1 Popis implementace



Obrázek 3.1: UML diagram programu

3.1.1 Základní třídy

Hlavní třídou celé aplikace je třída `Main` v balíku cz.dawon.java.graph.visualisation, která vykreslí hlavní okno aplikace a stará se celkově o uživatelské rozhraní.

Třída `GraphPanel`, která se nachází v tomtéž balíku, slouží k vykreslování grafu a interakci s ním (posun a přiblížení nebo oddálení).

Třída `GraphZoomBehaviourManager` uchovává různé metody vizualizace grafů a umožňuje mezi nimi přepínat.

V balíku cz.dawon.java.graph.visualisation.ui se nachází třídy `LayerCellRenderer` a `LayerListModel`, které umožňují vykreslit ListBox obsahující vrstvy ve formě zaškrtávacích políček.

V balíku cz.dawon.java.timer se nachází třída `Timer`, která je používána pro měření trvání jednotlivých operací. Protože nijak jinak nezasahuje do běhu aplikace, byla záměrně vynechána z UML diagramu.

Hlavní logika programu se nachází v balíku cz.dawon.java.graph.visualisation.behaviours. Rozhraní `GraphZoomBehaviour` představuje chování grafu podle určité metody.

Toto rozhraní používá třída `AbstractGraphBehaviour`, která implementuje některé metody a tím zjednodušuje kód potomků.

3.1.2 Implementované metody

Následuje popis jednotlivých metod chování grafu:

Výchozí metoda

Výchozí metodu obsluhuje třída `DefaultGraphBehaviour`, která ve všech případech vykresluje celý graf (nebo jeho vybranou oblast) bez jakýchkoliv změn.

Metoda skrývání vrcholů

Tuto metodu obsluhuje třída `HideGraphBehaviour`, která pouze skrývá ty vrcholy, které nemají být vidět podle aktuální úrovně přiblížení.

Metoda změny velikosti vrcholů

O změny vrcholů se stará třída `NodesZoomGraphBehaviour`, která ve stylech grafu mění velikosti vrcholů na základě aktuální úrovně přiblížení a jejich ohodnocení.

Manuální metoda

Ruční metodu ovládá třída `ManualGraphBehaviour`, která zobrazuje a skrývá vrstvy na základě voleb uživatele v panelu na pravé straně okna.

Metoda virtuálních vrcholů

Funkcionalita této metody je ve třídě `VirtualNodesGraphBehaviour`. Ta nejprve pro každou vrstvu v grafu vytvoří kopii daného grafu, následně v ní odstraní všechny hrany a vytvoří nové orientované hrany podle původního grafu a ohodnocení jeho uzlů.

Následně je nad grafem spuštěn Tarjanův algoritmus, který najde všechny silně souvislé komponenty v grafu. Pro výpočet využívám mírně upravenou verzi Tarjanova algoritmu z knihovny `GraphStream` (více viz dále). Ke každé komponentě se vytvoří nový vrchol a ten se propojí s původním grafem. Pozice vrcholu je určena váženým průměrem pozic a ohodnocení všech vrcholů v dané komponentě.

3.1.3 Balík tříd org.graphstream.modified

Tento balík obsahuje mírně modifikované verze tříd přímo z knihovny `GraphStream`.

Třída `TarjanAlgorithm` hledá silně souvislé komponenty v grafu. Její úprava spočívá v tom, že komponenty jsou ukládány do množiny a nikoliv jako ID komponenty v atributu vrcholu. Což předchází zbytečnému procházení celého graf k nalezení vrcholů dané kompoenty.

Třídy `PositionFileSourceGraphML` a `PositionFileSinkGraphML` se používají k načítání a ukládání grafů ve formátu `GraphML`. Tyto třídy bylo nutné upravit, protože původní implementace neuměla ukládat a načítat pozici vrcholů v grafu.

3.2 Zdrojový kód

Veškeré zdrojové kódy a spustitelný soubor ukázkové aplikace včetně všech testovacích grafů jsou uloženy na přiloženém DVD v adresáři `graph-visualisation`.

4 Testování

4.1 Testování ukázkové aplikace testery

4.1.1 Zadání

Níže je citace zadání, které jsem odeslal všem nezávislým testerům spolu s ukázkovou aplikací k otestování. Ti měli za úkol subjektivně otestovat především přehlednost jednotlivých metod vizualizace pro různé grafy. Vyhodnocení pak vyplnili do přiložené tabulky.

Text zadání

Přiložená aplikace slouží jako ukázka vizualizace různých grafů různými způsoby. Dále je přiloženo několik souborů s grafy (soubory s příponou `*.gml`).

V pravé části aplikace lze volit z několika způsobů vizualizace daných grafů:

- ***Default behaviour*** *– výchozí chování grafu, kdy jsou vždy vykresleny všechny vrcholy v zobrazované oblasti grafu.*
- ***Manual Behaviour*** *– ruční výběr zobrazovaných vrstev. Pod rozbalovací nabídkou způsobů vizualizace grafů je možné zobrazovat či skrývat jednotlivé úrovně. V zobrazované oblasti jsou pak vidět pouze vrcholy zobrazených vrstev.*
- ***Hide Behaviour*** *– velmi podobné jako `Manual behaviour` s tím, že vrstvy se zobrazují a skrývají automaticky podle úrovně přiblížení.*
- ***Virtual Node Behaviour*** *– určité podgrafy grafu jsou nahrazeny jedním vrcholem, ten je pak připojen místo tohoto podgrafu.*
- ***Zoom Nodes Graph Behaviour*** *– velikost vrcholů je měněna v závislosti na úrovni přiblížení grafu*

Prosím vyzkoušejte jednotlivé způsoby vizualizace spolu se všemi grafy a vaše subjektivní hodnocení přehlednosti (čísly 1 - 5, kde 1 je nejpřehlednější

a 5 nejméně přehledné) vyjádřete v přiložené tabulce, a to zejména s ohledem na přehlednost a na to, zda se v daném zobrazení mohou ztrácet určité informace o zobrazeném grafu. Zároveň na pár řádcích vyjádřete, který způsob se vám zdál nejhorší a který nejlepší a proč.

Ovládání aplikace je popsáno v uživatelské příručce v příloze.

Hodnotící tabulka

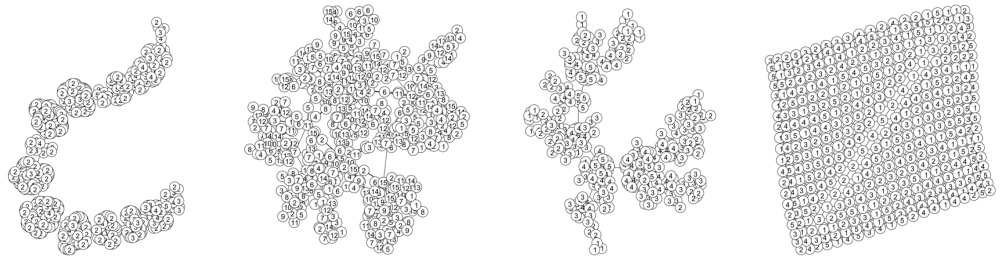
Behaviour \File	graf0.gml	graf1.gml	graf2.gml	graf3.gml
Default				
Manual				
Hide				
Virtual Node				
Zoom Nodes Graph				

Tabulka 4.1: Tabulka rozeslaná testerům

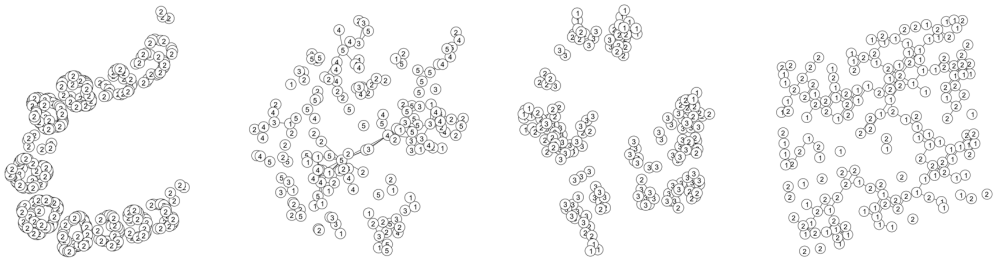
Zadané grafy

Pro testování jsem se rozhodl použít čtyři různé grafy:

- `graf0.gml` – náhodně generovaný graf, kde je jedna linie vrcholů s nejvyšším ohodnocením a na ni jsou napojeny vrcholy s nižším ohodnocením.
- `graf1.gml` – náhodně generovaný graf s náhodně generovaným hodnocením.
- `graf2.gml` – náhodně generovaný graf s hierarchickým ohodnocením vrcholů
- `graf3.gml` – graf vygenerovaný jako mřížka s náhodným ohodnocením vrcholů

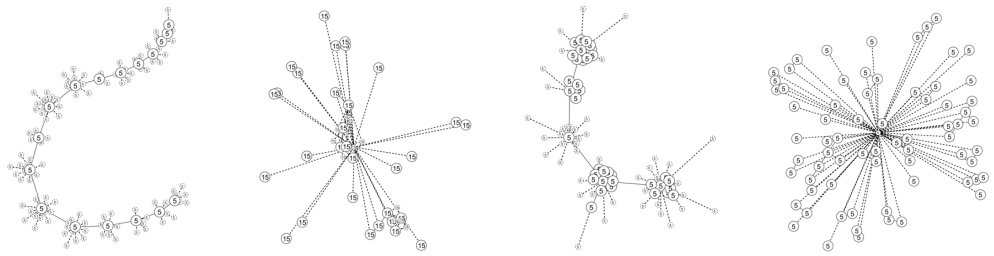


Obrázek 4.1: Zadané grafy graf0.gml až graf3.gml (zleva) zobrazené výchozí metodou

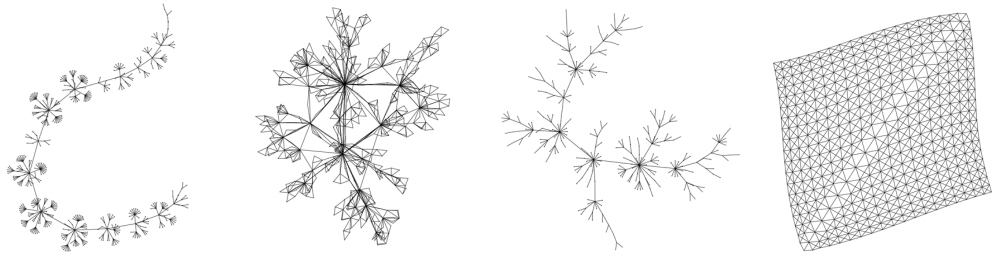


Obrázek 4.2: Zadané grafy graf0.gml až graf3.gml (zleva) zobrazené manuální metodou

Obrázek 4.3: Zadané grafy graf0.gml až graf3.gml (zleva) zobrazené metodou skrývání vrcholů



Obrázek 4.4: Zadané grafy graf0.gml až graf3.gml (zleva) zobrazené metodou virtuálních vrcholů



Obrázek 4.5: Zadané grafy graf0.gml až graf3.gml (zleva) zobrazené metodou změny velikosti vrcholů

4.1.2 Výsledky

Níže jsou citace slovních vyjádření a tabulky s hodnoceními jednotlivých metod a grafů od všech dotazovaných testerů.

Tester 1

Dle mého je nejlepší metoda Zoom Nodes Behaviour. Líbí se mi, že je vidět celá kostra grafu.

Na druhou stranu jako nejhorší se mi jeví Virtual Node Graph Behaviour, protože se mi nahrazování částí grafu zdá zmatené.

Behaviour \File	graf0.gml	graf1.gml	graf2.gml	graf3.gml
Default	5	5	4	2
Manual	3	3	3	2
Hide	2	3	3	3
Virtual Node	4	4	2	5
Zoom Nodes Graph	1	2	1	2

Tabulka 4.2: Hodnocení testera 1

Tester 2

Pokud bych si mezi metodami měla vybrat, volila bych z metod Manual behavour (zde se mi líbí, že graf je zobrazen kompletně a díky možnosti skrývání vrcholů je možné mít vysokou kontrolu nad prohlížením) a Zoom Nodes Graph Behaviour.

Metoda Zoom Nodes Graph Behaviour se hodí zejména pro celkovou představu o rozložení grafu, ale díky možnosti přiblížení funguje dobře i na detaily.

Metodu Virtual Node Behaviour hodnotím jako průměrnou, nahrazované vrcholy by bylo možná vhodnější zobrazit jen jako kroužek či tečku bez číselného ohodnocení a tím naznačit uživateli, že v daném místě jsou nějaké vrcholy, které se zobrazí až po přiblížení.

Jako nejhorší hodnotím metodu Hide Behaviour, která při určitém přiblížení absolutně nevypovídá o grafu jako celku.

Behaviour \File	graf0.gml	graf1.gml	graf2.gml	graf3.gml
Default	3	3	3	2
Manual	2	2	2	1
Hide	3	4	4	5
Virtual Node	3	3	3	4
Zoom Nodes Graph	1	2	1	1

Tabulka 4.3: Hodnocení testera 2

Tester 3

Nejvíce se mi líbí metoda Zoom Behaivour, protože je zdaleka nejpřehlednější a věrně zobrazuje celý graf ve všech úrovních přiblížení.

Default Behaviour se mi naopak zdá nejhorší, protože při oddálení je graf vykreslen tak, že se jednotlivé vrcholy překrývají, celý graf tak působí chaoticky a nepřehledně.

Behaviour \File	**graf0.gml**	**graf1.gml**	**graf2.gml**	**graf3.gml**
Default	4	5	4	5
Manual	3	4	3	4
Hide	1	3	3	2
Virtual Node	3	4	3	3
Zoom Nodes Graph	1	2	1	1

Tabulka 4.4: Hodnocení testera 3

Tester 4

Default je přehledný až při určitém přiblížení, člověk nejprve musí graf více přiblížit, aby byl graf přehledný.

Virtual je při hustším grafu nepřehledný, hlavně díky čárkovaným spojům. Spojuje vrcholy do skupin později, než by bylo vhodné – občas se stane, že je přes sebe hodně vrcholů a pořád nejsou ve skupině.

Zoom nodes graph se zdá být nejlepší, chtělo by to, aby největší vrcholy byly vidět rovnou – ne jen jako bod.

Hide Behaviour mi přijde matoucí – vrcholy a spoje úplně mizí, případně se objevují. Občas je potřeba zbytečně velkého přiblížení pro zobrazení malých vrcholů.

Behaviour \File	**graf0.gml**	**graf1.gml**	**graf2.gml**	**graf3.gml**
Default	3	4	3	2
Manual	2	3	2	2
Hide	2	3	2	3
Virtual Node	2	2	2	4
Zoom Nodes Graph	1	3	1	5

Tabulka 4.5: Hodnocení testera 4

Tester 5

Objektivně je při zohlednění přehlednosti grafů a míře poskytované (a ztracené) informace nejvhodnější Virtual Node Behavior. Jeho hlavní výhodou oproti ostatním metodám je zachování více informací a přitom udržení přehlednosti díky slučování podstromů. Za hlavní nevýhodu považuji nemožnost zobrazení celého grafu v nezměněné podobě a zjevnou výpočetní složitost projevující se u rozsáhlejších grafů.

Nejhůře při testování dopadlo jednoznačně defaultní zobrazení, které sice neztrácí informaci (pokud by dva body grafu nebyly na jednom místě, pak by nepomohlo žádné přiblížení), ale nenabízí žádnou možnost zpřehlednění grafu.

Mou osobní preferencí je Manual Behaviour, který neztrácí žádnou informaci a jako jediný mi nabízí možnost zobrazit to, co zrovna potřebuji.

Behaviour \File	**graf0.gml**	**graf1.gml**	**graf2.gml**	**graf3.gml**
Default	5	5	5	3
Manual	3	2	4	2
Hide	4	3	3	5
Virtual Node	1	1	1	4
Zoom Nodes Graph	2	4	2	1

Tabulka 4.6: Hodnocení testera 5

4.1.3 Zhodnocení výsledků

	graf0.gml	**graf1.gml**	**graf2.gml**	**graf3.gml**	**Prm.**
Default	4.0	4.4	3.8	2.8	**3.8**
Manual	2.6	2.8	2.8	2.2	**2.6**
Hide	2.4	3.2	3.0	3.6	**3.0**
Virtual Nodes	2.6	2.8	2.2	4.0	**2.9**
Zoom Nodes	1.2	2.6	1.2	2.0	**1.8**

Tabulka 4.7: Průměry hodnocení jednotlivých testerů

Z číselných hodnocení (viz tabulka 4.7) je zřejmé, že každá metoda je vhodná pro jiný druh grafu. Nejlépe hodnocenou metodou byla metoda zvětšování vrcholů na základě přiblížení, naopak nejhůře dopadla výchozí metoda.

Výchozí metoda tak nalezne využití především u menších grafů nebo u grafů s většími rozestupy jednotlivých vrcholů. Podle dvou testerů je zcela nevhodná pro vizualizaci grafů.

Manuální metoda je podle testerů přibližně stejně vhodná pro všechny zkoumané grafy. Dva testeři ji dokonce označili za preferovaný způsob vykreslování grafů, protože umožňuje největší kontrolu nad zobrazovanými daty.

Metoda skrývání vrcholů byla testery vyhodnocena jako druhá nejhorší metoda. Testerům nevyhovalo hlavně to, že oddálený graf vizualizovaný tímto způsobem téměř vůbec neodpovídal původnímu grafu.

Metoda virtuálních vrcholů byla testery vyhodnocena jako nejvíce rozporuplná. Někteří ji označili za nejlepší a někteří naopak za nejhorší z testovaných metod. Za nevýhody označili testeři nepřehlednost a pomalejší odezvy při přípravě a vykreslování grafu. Jiní testeři naopak metodu označili jako přehlednou a vhodnou pro vykreslování velkých grafů.

Ze všech metod byla nejlépe hodnocena metoda zvětšování vrcholů na základě přiblížení grafu. Většina testerů ji označila jako preferovaný způsob vizualizace grafu hlavně díky věrnosti zobrazení grafu při všech úrovních přiblížení.

4.1.4 Možná vylepšení na základě připomínek testerů

Změnou, která by velmi prospěla všem metodám, je lépe zkalibrovat úrovně přiblížení a doby, kdy vrcholy mizí či se shlukují. Bohužel u této aplikace toto nejde příliš aplikovat, protože kalibraci je potřeba udělat na míru konkrétním grafům. Tato aplikace je univerzální, tudíž kalibrace pro jeden druh grafu nebude fungovat správně pro jiný.

U metody s virtuálními vrcholy by bylo vhodné u virtuálních vrcholů nezobrazovat jejich ohodnocení, nebo zobrazovat například průměr ohodnocení vrcholů, které představuje.

U metody změn velikosti vrcholů by pomohlo při nejmenším přiblížení zobrazovat vrcholy nejvyšší úrovně alespoň tak velké, aby se do nich vešel popisek či jejich ohodnocení.

4.2 Měření časové výkonnosti aplikace

V ukázkové aplikaci jsem také testoval, jak dlouho při práci s grafem, trvají následující operace:

- `draw` – trvání vykreslení grafu
- `init` – doba přípravy grafu pro zobrazení danou metodou
- `zoom` – doba změny přiblížení grafu, počítal jsem jen první událost tohoto druhu
- `clean` – doba úklidu po dané metodě

Všechna měření byla provedena třikrát a v tabulkách níže jsou výsledky všech tří měření zprůměrovány. Měření probíhala na počítači s Microsoft Windows 10 Pro, Intel Core i7-3632Q2 2.2GHz a 8 GB RAM.

Aplikace byla spuštěna následujícím příkazem ve svém adresáři:

```
mvn exec:java -Djava.compiler=NONE
```

Měření času prováděla pomocná třída `Timer`, která měřila čas metodou `System.nanotime()`. Výsledné doby trvání se zobrazovaly v konzoli.

První sloupec v tabulkách obsahuje číslo testované metody:

1. výchozí metoda
2. manuální metoda
3. metoda skrývání vrcholů
4. metoda virtuálních vrcholů
5. metoda změn velikostí vrcholů

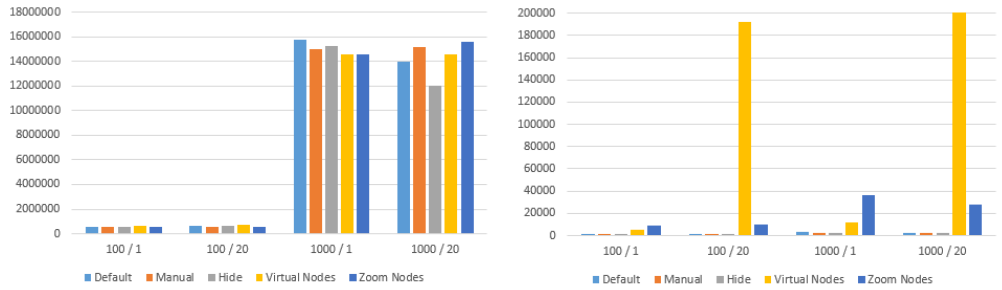
První řádek tabulek označuje `počet vrcholů / počet vrstev` náhodně generovaného grafu, na kterém byly časy měřeny. Při měření časů jsem deaktivoval Java JIT a pro měření každého grafu a metody jsem aplikaci pouštěl vždy znovu, aby nedocházelo ke zkreslení naměřených časů způsobeným optimalizacemi.

	100 / 1				**100 / 20**			
	draw	init	zoom	clean	draw	init	zoom	clean
1	600 368	1 408	1 169	6	607 157	1 537	1 382	6
2	560 953	1 433	1 321	6	582 552	1 416	1 374	6
3	578 443	1 375	3 902	6	636 078	1 508	10 170	7
4	655 306	5 462	3 970	152	727 283	192 290	9 420	14 911
5	567 882	8 595	4 304	2 684	543 131	10 336	7 227	2 644

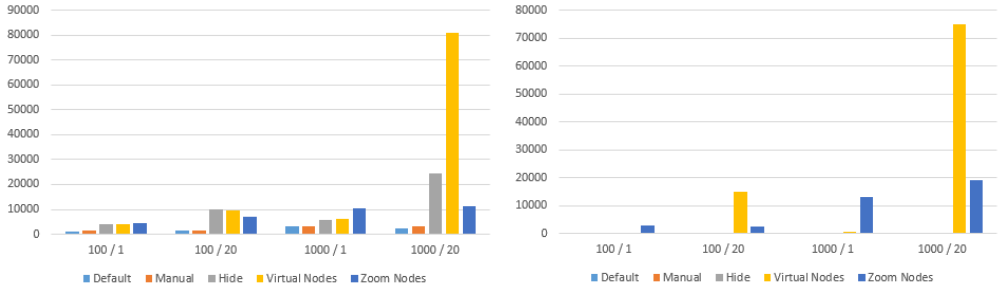
Tabulka 4.8: Trvání jednotlivých operací u jednotlivých metod v ukázkové aplikaci pro malé počty vrcholů; jednotky μs

	1000 / 1				**1000 / 20**			
	draw	init	zoom	clean	draw	init	zoom	clean
1	15 734 157	3 499	3 306	6	14 003 230	2 549	2 525	6
2	15 039 324	2 533	3 113	6	15 167 050	2 515	3 145	9
3	15 240 480	2 491	5 800	5	12 032 639	2 599	24 451	7
4	14 539 843	12 273	6 135	680	14 552 691	822 416	80 727	75 102
5	14 584 355	36 751	10 536	13 036	15 572 930	28 124	11 194	19 119

Tabulka 4.9: Trvání jednotlivých operací u jednotlivých metod v ukázkové aplikaci pro velké počty vrcholů; jednotky μs



Obrázek 4.6: Diagramy znázorňující rychlost operace `draw` (vlevo) a rychlost operace `init` (vpravo) pro jednotlivé grafy; jednotky osy y: μs



Obrázek 4.7: Diagramy znázorňující rychlost operace `zoom` (vlevo) a rychlost operace `clean` (vpravo) pro jednotlivé grafy; jednotky osy y: μs

Níže vysvětlím všechny anomálie, které se vyskytují v tabulkách s časy.

Doby vykreslení jednotlivých grafů o stejném počtu vrcholů jsou většinou podobné, což odpovídá tomu, že téměř ve všech případech se vykresluje stejné množství vrcholů a hran. Výjimkou je graf o 1000 vrcholech a 20 vrstvách vykreslovaných metodou skrývání vrcholů. Zde je zjevné, že na nejnižší úrovni přiblížení tato metoda vykresluje znatelně méně vrcholů a hran než ostatní metody.

Inicializační fáze trvala u prvních tří metod podobně dlouho, protože tyto metody zde provádí stejné operace. U metody virtuálních vrcholů je pak hodnota několikanásobně vyšší, což je způsobeno tím, že se provádí algoritmus hledání virtuálních vrcholů. U metody změny velikosti vrcholu pak trvá většinu času samotná změna velikostí vrcholů.

Důvodem proč u grafů s jednou vrstvou všechny operace trvají déle u poslední metody než u metody virtuálních vrcholů a u grafu s 20 vrstvami tomu je naopak je, že se algoritmus pro hledání virtuálních vrcholů spouští pro $n - 1$ vrstev a v tomto případě je $n = 1$. Tudíž se algoritmus vůbec nespustí.

Doba přiblížení či oddálení se liší u posledních tří metod. U všech těchto metod probíhá zobrazování či skrývání pro větší počet vrcholů než u ostatních metod.

U prvních tří metod je doba úklidu zanedbatelná, protože tyto metody nepotřebují uklízet téměř žádná data. Metoda virtuálních vrcholů musí naopak odstranit z grafu virtuální vrcholy a metoda změny velikosti vrcholu musí nastavit původní velikost vrcholů. Proto tato operace trvá i tisícinásobně déle než u ostatních způsobů.

5 Vizualizace přenosové sítě

Vizualizace přenosové sítě je projekt vyvíjený studenty a zaměstnanci z Katedry informatiky a výpočetní techniky pro Katedru kybernetiky Západočeské univerzity v Plzni. Jedná se o komplexní webovou aplikaci, která umožňuje propočítávat různé situace v elektrické síti a výsledky těchto výpočtů následně vizualizovat na mapě.

Aplikace umožňuje práci s různými modely a druhy výpočtů. Volbu modelu a druh výpočtu je možné provést na úvodní stránce. Po výběru obou hodnot se zobrazí hlavní stránka aplikace.

Hlavní stránka zobrazuje mapový podklad, na kterém je vykreslen graf přenosové sítě vybraného modelu. Jednotlivé vrcholy grafu představují sběrnice a hrany představují elektrické vedení. K jednotlivým sběrnicím mohou být připojeny ještě generátory, zatížení, vypínače a transformátory. Ty sice nejsou na mapě zobrazeny, ale některé výpočetní metody s nimi počítají. Kliknutím na jednotlivé prvky na mapě lze v panelu vpravo zobrazit detaily daného prvku. V tomto panelu je tak vidět typ zvolené položky, její ID a další parametry (jako napěťová úroveň, napětí, výkon atd.). Prvky lze také označovat stisknutím klávesy `control` a kliknutím na daný vrchol. V dolní části se pak zobrazují posuvníky patřící k jednotlivým vybraným prvkům. Posuvník umožňuje nastavit hodnoty určitých parametrů u vybraných prvků a kritické limity dané hodnoty.

Obrázek 5.1: Aplikace Vizualizace přenosové sítě

Když je uživatel s nastavenými hodnotami spokojený může vpravo nahoře tlačítkem `Recalculate` spustit výpočet. Běh výpočtu trvá několik minut. Po úspěšném skončení výpočtu jsou výsledky zobrazeny v grafu. Pokud jsou některé hodnoty po výpočtu mimo vyznačené meze, je jejich posuvník zobrazen v pravé dolní části obrazovky.

V levé horní části obrazovky lze zvolit napěťové úrovně, které se mají na mapě zobrazit a případně mapový podklad, který má být vykreslen pod grafem.

5.1 Použité technologie

Celý projekt se dá rozdělit na dvě základní části – backend a frontend.

5.1.1 Backend

Běh celé aplikace zastřešuje `Java Enterprise Edition` s použitím `AppFuse` `Web Services` a `Jersey`.

O výpočty se stará `MatLab` spojený přes Javovou aplikaci. Veškerá data jsou ukládána do `MySQL` databáze.

5.1.2 Frontend

Veškerou interakci s uživatelem zajišťuje `JavaScript` s využitím knihovny `AngularJS`, která především usnadňuje implementaci `MVC` modelu.

Vykreslení mapových podkladů a grafů nad nimi provádí knihovna `OpenLayers 3`. O vykreslení některých ovládacích prvků se starají knihovny `jQuery` a `jQuery UI`.

5.2 Výběr vhodné metody

V původní implementaci vizualizace přenosové sítě si uživatel manuálně volí napěťové úrovně, které chce vidět. Pak jsou vykresleny ty sběrnice (vrcholy), které se nacházejí na vybraných vrstvách. Toto řešení je sice jednoduché, ale pro uživatele nepohodlné. Navíc při zobrazení pouze některých úrovní může vzniknout nesouvislý graf, který ve skutečnosti není nesouvislý (chybějící hrany jsou pouze skryté). Může tak docházet ke ztrátě informací.

Metoda automatického skrývání méně významných vrcholů není vhodná, protože podobně jako u manuální volby vrstev, by mohlo docházet ke ztrátě informací.

Metoda změn velikostí vrcholů na základě přiblížení a ohodnocení by pro tento účel byla sice vhodná, ale v budoucnu by velikost grafů mohla znázorňovat i jiné informace a pak by význam velikosti vrcholů nebyl jednoznačný. V aplikaci je také nutné vybírání vrcholů, což by mohlo být problematické, pokud by vrcholy byly příliš malé.

Proto se zdá být nejvhodnější poslední metoda – shlukování podgrafů do vrcholů.

5.3 Popis implementace

Zdrojový kód projektu Vizualizace přenosové sítě je výsledkem práce více lidí. Všechny mé významnější úpravy v rámci této diplomové práce jsou explicitně uvedeny v této kapitole.

5.3.1 Databáze

Ačkoliv vizuálně u grafu vrcholy představují sběrnice a hrany elektrické vedení, interně (v databázi) je graf reprezentován jiným způsobem.

Celá databáze je z logického hlediska rozdělena na dvě části. Jedna slouží k reprezentaci grafu představujícího přenosovou síť, druhá uchovává všechny vstupní a výstupní hodnoty výpočtů. Níže popíši nejdůležitější tabulky sloužící k reprezentaci grafu.

Každý objekt (sběrnice, elektrická vedení, transformátory, zátěže...) musí být v databázi definován v tabulkách `Object`, `IdentifiedObject` a v konkrétní tabulce reprezentující daný objekt (například `PowerLine`, `Bus`, `Load` atd.).

Hrany v grafu představují položky tabulky `ConnectivityNode` (musí být také definované v tabulkách `Object` a `IdentifiedObject`).

Každá položka z tabulky `Object` je navázána na jednu z napěťových úrovní, které jsou uloženy v tabulce `VoltageLevel`.

Hrany a vrcholy jsou pak propojeny přes tabulku `Connectivity`.

Příslušnost prvků k danému modelu je určena přes tabulku `ModelConnectivity`.

Vzhledem k tomu, že se graf nebude často měnit, nebude součástí implementace algoritmus k určení virtuálních vrcholů a hran. Ty budou určeny ručně v databázi.

Podrobný ERA model databáze se nachází v příloze B.

5.3.2 Úpravy databáze

Pro implementaci shlukování vrcholů bylo nejprve nutné upravit strukturu databáze:

V tabulkách `Bus` a `Connectivity` jsem přidal sloupec `virtual`. Hodnoty v tomto sloupci označují virtuální sběrnice a virtuální „spojení“ s ostatními objekty.

Následně jsem také do tabulky `VoltageLevel` přidal sloupec `zoomLevel`, který představuje číselnou hodnotu přiblížení mapy, při kterém má být daná napěťová úroveň zobrazena.

5.3.3 Úpravy backendu

V backendu aplikace jsem pouze upravil všechny příslušné DTO objekty a doménové třídy podle změn v databázi, aby bylo možné tato data z frontendu získat.

5.3.4 Frontend

Zde stručně popíši organizaci zdrojového kódu a účel důležitých souborů ve frontentdu.

Nejdůležitějším souborem celého frontendu je soubor `model.js` což je controller pro stránku zobrazující model. Odtud se volají všechny ostatní služby a aktualizuje se uživatelské rozhraní.

Služba `httpService.js` se stará na nejnižší přístupné úrovni o odesílání dat na server a jejich stahování zpět.

V souboru `graphFactory.js` je zpracováván výstup z backendu a vytvářena mapa obsahující všechny informace potřebné pro vykreslení výsledného grafu.

O vykreslení samotné mapy a skrývání a zobrazování jednotlivých vrstev se stará soubor `mapFactory.js`.

Soubor `voltageLevelService.js` získává informace a položky jednotlivých napěťových úrovní.

V pomocném souboru `utilsService.js` se nachází různé funkce, které z logického hlediska nepatří do žádného jiného souboru.

Soubor `model.html` obsahuje uspořádání ovládacích prvků na webové stránce s modelem.

5.3.5 Úpravy frontendu

Ve výchozím stavu je automatické shlukování vrcholů vypnuto. To může uživatel zapnout novým přepínačem `auto` v horní části obrazovky. Když

je automatické shlukování vrcholů aktivováno, nelze zobrazovat a skrývat jednotlivé napěťové úrovně ručně.

Do souboru `mapFactory.js` jsem přidal metody pro skrytí a znovuzobrazení napěťových úrovní a metodu pro odstranění jedné napěťové úrovně.

Soubor `graphFactory.js` jsem upravil tak, aby podporoval virtuální sběrnice a elektrická vedení. U virtuálních objektů bude zakázán jejich výběr.

V souboru `voltageLevelService.js` jsem pozměnil chování metody získávající jednotlivé sběrnice a elektrická vedení tak, aby se v grafu ve výchozím stavu nevykreslovaly virtuální objekty.

Do souboru `olStyles.js` jsem přidal styly pro virtuální sběrnice a elektrická vedení.

Do souboru `model.js` jsem přidal volání do `zoomService.js` a příslušně přizpůsobil chování uživatelského rozhraní.

Do HTML souboru `model.html` jsem přidal přepínač pro aktivaci a deaktivaci shlukování vrcholů.

Do pomocného souboru `utilsService.js` jsem přidal metodu na klonování objektů.

Snažil jsem se uložit co nejvíce funkcionality shlukování vrcholů do souboru `zoomService.js`. V něm se nachází následující metody:

- `activate` – tato metoda je zavolána ve chvíli, kdy uživatel aktivuje automatické shlukování vrcholů.

  Nejprve skryje všechny napěťové úrovně a přidá jednu novou, na kterou budou přidány všechny aktuálně zobrazené vrcholy. Následně stáhne všechna doposud nestažená data napěťových úrovní do paměti prohlížeče (když je automatické shlukování vypnuto, stahují se data jednotlivých úrovní až když si uživatel danou vrstvu zobrazí). Tato stažená data už pak nebudou stahována.

- `deactivate` – tato metoda smaže virtuální napěťovou úroveň a znovu zobrazí všechny vrstvy, které byly před aktivací zobrazené.

- `onZoomChange` – tato metoda je volána, když uživatel přiblíží či oddálí graf s mapou. Pouze ověřuje, zda je shlukování vrcholů aktivní a zavolá funkci `prepareFeatures`.
- `prepareFeatures` – tato metoda prochází všechny vrcholy[3] grafu a určuje, zda mají být vykresleny nebo ne.

  Pokud je daný vrchol na vyšší úrovni než má být zobrazena a není virtuální, pak je rovnou přidán mezi vykreslované vrcholy.

  Pokud patří vrchol na nižší úroveň než má být zobrazena a jedná se o virtuální vrchol, pak je potřeba ověřit, zda bude sousední nevirtuální vrchol vykreslen. Pokud ano, vykreslí se i tento vrchol.

  Ve všech ostatních případech nebude daný vrchol vykreslen.

5.4 Možná vylepšení

Virtuální vrcholy momentálně nelze vybírat, což by se dalo do budoucna změnit, protože původní projekt podporuje výběr různých skupin vrcholů. Tak by bylo možné vybrat jedním klepnutím celý podgraf, který je reprezentovnán virtuálním vrcholem.

V případě, že by se do aplikace přidávalo více modelů by se vyplatilo implementovat i automatický algoritmus pro určování virtuálních vrcholů, jako samostatný program, který by se mohl spouštět vždy při přidání nebo úpravě modelu.

5.5 Zdrojový kód

Zdrojový kód Vizualizace přenosové sítě včetně všech úprav je uložen na přiloženém DVD v adresáři `vps`. Data jednotlivých modelů a výpočtů nemohu přiložit, protože tato data patří cizí společnosti, která si je nepřeje zveřejnit.

[3] vrchol zde představuje sběrnici nebo elektrické vedení

6 Závěr

V první části této diplomové práce jsem popsal různé základní i alternativní metody vizualizace grafů. Dále jsem zde uvedl přehled několika aplikací určených k vizualizaci grafů včetně přehledných tabulek se základními informacemi o těchto aplikacích.

V druhé části jsem implementoval jednoduchou ukázkouvou aplikaci. Tato aplikace umožňuje vykreslení různých grafů pěti různými metodami. Zobrazený graf je možné libovolně přibližovat, oddalovat a volně měnit zobrazovanou oblast. Následně jsem tuto aplikaci dal k otestování pěti nezávislým testerům, kteří subjektivně zhodnotili přehlednost všech testovaných metod na několika grafech.

Zároveň jsem aplikaci otestoval z hlediska časové analýzy pro čtyři různé velikosti grafů všemi vizualizačními metodami a výsledné časy vyhodnotil a zdůvodnil.

Ze zkoumaných metod jsem pro implementaci do projektu Vizualizace přenosové sítě zvolil metodu nahrazování podgrafů virtuálními vrcholy, která se pro tuto situaci ukázala jako nejvhodnější.

Metodu jsem implemenetoval do projektu a otestoval její chování na několika testovacích grafech. Metoda shlukování podgrafů zjednodušila používání aplikace a vykreslený graf je při používání této metody mnohem přehlednější u všech úrovní přiblížení než tomu bylo u manuálního vybírání napěťových úrovní.

Literatura

[1] BONDY, John Adrian a Uppaluri Siva Ramachandra MURTY. *Graph theory with applications*. 5. vyd. New York: Elsevier Science Publishing Co. Inc., 1982, 264 s. ISBN 04-441-9451-7.

[2] COLEMAN, Michael K. a D. Stott PARKER. Aesthetics-based Graph Layout for Human Consumption. *Software: Practice and Experience* [online]. 1996, **1996**(26), 1415 - 1438 [cit. 2016-05-21]. DOI: 10.1002/(SICI)1097-024X(199612)26:12<1415::AID-SPE69>3.0.CO;2-P. Dostupné z: http://onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-024X(199612)26:12%3C1415::AID-SPE69%3E3.0.CO;2-P/abstract

[3] Why graph analyzes will rule PLM in the future? *Beyond PLM* [online]. Newton: O. Shilovitsky, 2013 [cit. 2016-05-20]. Dostupné z: http://beyondplm.com/2013/08/23/why-graph-analyzes-will-rule-plm-in-the-future

[4] Circular Layouts, Binning and Edge Bundling. *Robert Gimeno's Adventures in Data Science* [online]. Barcelona: R. Gimeno, 2013 [cit. 2016-05-21]. Dostupné z: http://www.gimeno.co.uk/circular-layouts-binning-and-edge-bundling

[5] Gource. *AlternativeTo: Crowdsourced software recommendations* [online]. Göteborg: O. Johansson, 2016 [cit. 2016-05-23]. Dostupné z: http://alternativeto.net/software/gource

[6] Tutorial:Introduction to Cytoscape. *University of California San Francisco: Open Tutorials* [online]. San Francisco: University of California San Francisco, 2013 [cit. 2016-05-21]. Dostupné z: http://opentutorials.cgl.ucsf.edu/index.php/Tutorial:Introduction_to_Cytoscape

[7] DIEHL, Stephan. *Software visualization: visualizing the structure, behaviour, and evolution of software*. New York: Springer, 2007. ISBN 978-354-0465-041.

[8] OpenOrd Layout. *Gephi: Marketplace* [online]. Paříž: Gephi Consortium, 2012 [cit. 2016-05-23]. Dostupné z: `https://marketplace.gephi.org/plugin/openord-layout`

[9] Tutorial Layouts. *Gephi* [online]. Paříž: Gephi Consortium, 2016 [cit. 2016-05-23]. Dostupné z: `https://gephi.org/users/tutorial-layouts`

[10] Circular Layout. *Gephi: Marketplace* [online]. Paříž: Gephi Consortium, 2012 [cit. 2016-05-23]. Dostupné z: `https://marketplace.gephi.org/plugin/circular-layout`

[11] GeoLayout. *Gephi: Marketplace* [online]. Paříž: Gephi Consortium, 2012 [cit. 2016-05-23]. Dostupné z: `https://marketplace.gephi.org/plugin/geolayout`

[12] Using ADF Treemap and Sunburst Components. *Oracle Help Center* [online]. Kalifornie: Oracle Corporation, 2016 [cit. 2016-05-20]. Dostupné z: `https://docs.oracle.com/cd/E28280_01/web.1111/b31973/dv_treemap.htm#ADFUI12936`

[13] Voronoi Diagram. *WolframMathWorld* [online]. Oxfordshire: Wolfram Research, 2016 [cit. 2016-05-23]. Dostupné z: `http://mathworld.wolfram.com/VoronoiDiagram.html`

[14] Polygonal Map Generation for Games. *Amit Patel's Home Page* [online]. Kalifornie: A. Patel, 2016 [cit. 2016-05-23]. Dostupné z: `http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/`

[15] *Gephi* [online]. Paříž: Gephi Consortium, 2016 [cit. 2016-05-20]. Dostupné z: `https://www.gephi.org`

[16] *Gource: 2016* [online]. Nový Zéland: Andrew Caudwell [cit. 2016-05-20]. Dostupné z: `http://gource.io`

[17] DEENEY, Barry. Linux Kernel Repository Visualization with Gource (47 Minute Version). Youtube [online]. Zveřejněno 06. 04. 2013 [vid. 2016-05-20]. Dostupné z: `https://www.youtube.com/watch?v=AhDiYPLo3p4`

[18] PyGraphistry: Explore Relationships. *GitHub* [online]. San Francisco: GitHub, 2016 [cit. 2016-05-20]. Dostupné z: `https://github.com/graphistry/pygraphistry`

[19] *Pajek: analysis and visualization of large networks* [online]. Lublaň: Mrvar, 2016 [cit. 2016-05-20]. Dostupné z: `http://mrvar.fdv.uni-lj.si/pajek`

[20] NodeXL: Network Overview, Discovery and Exploration for Excel. *CodePlex: Project Hosting for Open Source Software* [online]. Washington: Microsoft, 2016 [cit. 2016-05-22]. Dostupné z: `http://nodexl.codeplex.com/`

[21] Twitter Users Usernames. *NodeXL GraphGallery* [online]. Kalifornie: Social Media Research Foundation, 2016 [cit. 2016-05-22]. Dostupné z: `http://www.nodexlgraphgallery.org/Pages/Graph.aspx?graphID=71362`

[22] *GUESS: The Graph Exploration System* [online]. Michigan: E. Adyr, 2007 [cit. 2016-05-22]. Dostupné z: `http://graphexploration.cond.org/index.html`

[23] *Turistický autoatlas Česko: 1:100 000 : podrobná automapa, GPS, rejstřík sídel s PSČ, centra měst s rejstříkem : cyklotrasy a turistické trasy, památky UNESCO, hrady, zámky aj., chráněná území přírody.* Vizovice: Freytag & Berndt, 2014. ISBN 9788072243082.

[24] *Mapy.cz* [online]. Praha: Seznam.cz, 2016 [cit. 2016-05-15]. Dostupné z: `http://www.mapy.cz`

[25] TARJAN, Robert. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing* [online]. 1972, **1**(2), 146-160 [cit. 2016-05-17]. DOI: 10.1137/0201010. ISSN 0097-5397. Dostupné z: `http://epubs.siam.org/doi/abs/10.1137/0201010`

[26] BALEV, Stefan, Julien BAUDRY, Antoine DUTOT, Yoann PIGNÉ a Guilhelm SAVIN. *GraphStream: A Dynamic Graph Library* [online]. 2015 [cit. 2016-06-15]. Dostupné z: `http://graphstream-project.org`

[27] *The Apache Maven Project* [online]. Maryland: Apache, 2016 [cit. 2016-05-21]. Dostupné z: `https://maven.apache.org`

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

2D ..... Two-Dimensional/Two Dimensions

3D ..... Three-Dimensional/Three Dimensions

BSD ..... Berkley Software Distribution

CDDL ..... Common Development and Distribution License

ČR ..... Česká republika

DFS ..... Depth-First Search

DTO ..... Data Transfer Object

DVD ..... Digital Versatile Disc

ERA ..... Entity – Relationship – Attribute

GML ..... Graph Markup Language

GNU ..... anglicky pakůň

GPL ..... General Public License

ID ..... Identifier

IDE ..... Integrated Development Environment

IP ..... Internet Protocol

JAR ..... Java Archive

JIT ..... Just-in-time

Ms-PL ..... Microsoft Public License

MVC ..... Model – View – Controller

UML ..... Unified Modeling Language

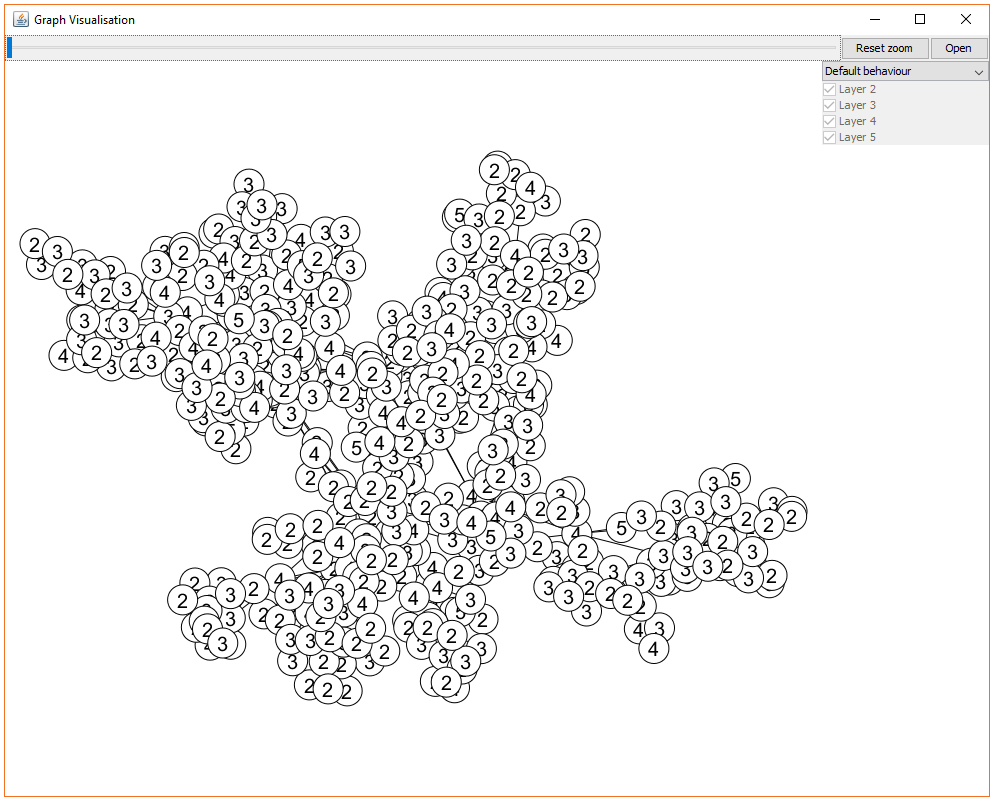
URL ..... Uniform Resource Locator

VPS ..... Vizualizace Přenosové Sítě

A Uživatelské příručky

A.1 Ukázková aplikace

Po spuštění spustitelného `*.jar` souboru se otevře hlavní okno aplikace spolu s náhodně vygenerovaným grafem.



Obrázek A.1: Hlavní okno aplikace

Zobrazený graf lze přibližovat a oddalovat kolečkem myši nebo posuvníkem v horní části okna. Zobrazenou oblast grafu lze měnit stisknutím levého tlačítka myši a táhnutím. Do původního zobrazení se lze vrátit stisknutím tlačíka `Reset Zoom`.

Graf uložený ve formátu `GraphML` (soubor s příponou `*.gml`) lze načíst stisknutím tlačítka `Open`.

V pravé horní části okna se nachází panel pro výběr vizualizační metody a seznam vrstev v grafu. Po výběru metody vizualizace grafu je graf okamžitě překreslen podle zvolené metody. Pokud to metoda umožňuje lze následně měnit zobrazené vrstvy zaškrtávátky v seznamu vrstev.

A.1.1 Kompilace a spuštění aplikace

Aplikace byla vytovřená jako `Maven`[27] projekt, což zajišťuje automatické stažení všech potřebných knihoven. Pro kompilaci aplikace je tedy nutné mít nainstalovaný `Maven`.

Aplikaci lze zkompilovat zavoláním `mvn compile` v adresáři projektu.

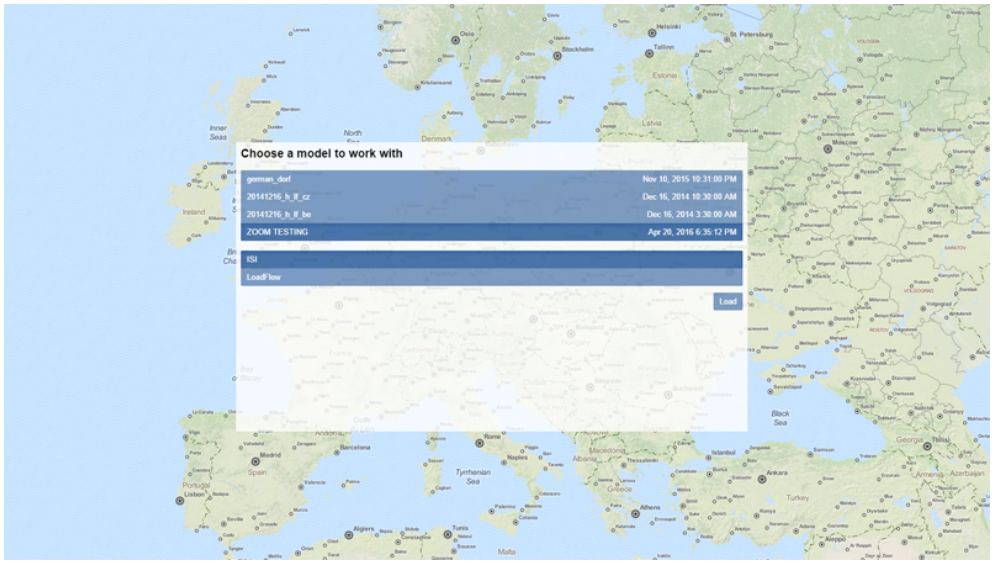
Následně spustitelný soubor `*.jar` lze vytvořit příkazem `mvn package`. V podadresáři `target` vzniknou následně dva `*.jar` soubory. Soubor s názvem `graph-visualisation-1.0-SNAPSHOT.jar`, který obsahuje pouze zkompilovaný kód aplikace bez závislostí a je spustitelný pouze tehdy, pokud je distribuovaný s celým adresářem `target`.

Druhý spustitelný soubor s názvem `graph-visualisation-1.0-SNAPSHOT-jar-with-dependencies.jar` pak obsahuje i veškeré knihovny, na kterých projekt závisí a je možné jej distribuovat samostatně.

Protože byla aplikace vyvíjena v IDE `NetBeans`, je také možno ji přímo otevřít jako projekt, kompilovat a spouštět přímo v tomto IDE.

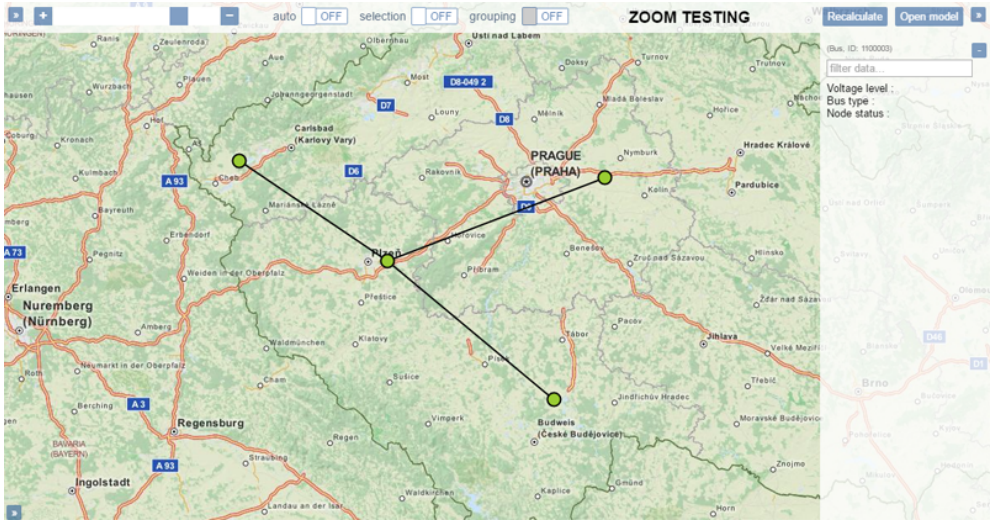
A.2 Vizualizace přenosové sítě

Po spuštění serverového backendu lze zobrazit aplikaci ve webovém prohlížeči na adrese `http://localhost:8080/vizualizace-prenosove-site` (pokud aplikace běží na jiném serveru je nutné nahradit localhost za IP adresu či doménové jméno daného serveru). Nejprve se zobrazí stránka s výběrem modelu a druhu prováděného výpočtu.



Obrázek A.2: Výběr modelu a druhu výpočtu

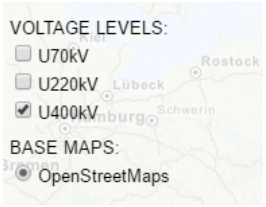
Následně po klepnutí na tlačítko `Load` se zobrazí pohled na samotný model.



Obrázek A.3: Pohled na model

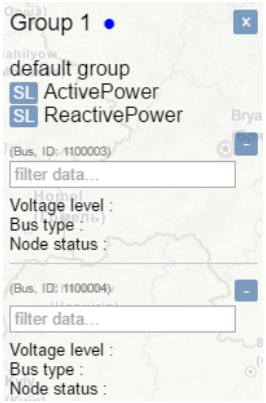
V levém horním rohu okna lze zobrazit panel pro správu zobrazených napěťových úrovní a mapových podkladů (viz obr. A.4). Dále je zde posuvník pro nastavení přiblížení, které lze měnit i kolečkem myši, následně přepínače `auto`, `selection` a `grouping`. Přepínač `auto` umožňuje volit zda uživatel bude

vrstvy přepínat ručně, nebo zda se budou zobrazovat automaticky. Přepínačem `selection` (případně stisknutím klávesy `control`) lze zapnout mód vybírání sběrnic – ve výchozím stavu se neprovádí výběr, jen se zobrazí informace o dané sběrnici. Pokud je zapnutý režim výběru lze přepínačem `grouping` (nebo stisknutím klávesy `shift`) zapnout režim výběru skupin sběrnic. Tím se vybírané sběrnice zároveň přidávají do skupin a je možné je spravovat hromadně. V pravé části horního panelu jsou pak tlačítka `Recalculate` pro spuštění výpočtu a `Open model` pro otevření jiného modelu.



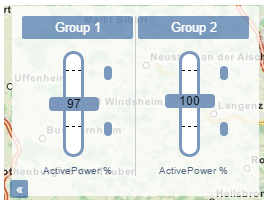
Obrázek A.4: Panel pro správu zobrazených vrstev

Panel na pravé straně pak zobrazuje podrobnosti o vybraných sběrnicích (viz obr. A.5).



Obrázek A.5: Panel informacemi o vybraných sběrnicích

V dolní části obrazovky se pak nachází posuvníky (viz obr. A.6) pro správu vybraných hodnot a jejich kritických mezí. Vlevo se nachází posuvníky pro vybrané položky (či skupiny položek) a vpravo posuvníky pro položky, u nichž výpočet vyhodnotil některé hodnoty jako kritické.



Obrázek A.6: Posuvníky vybraných prvků či skupin prvků

A.2.1 Kompilace a spuštění aplikace

I tato aplikace je `Maven` projektem, což umožňuje snadno kompilaci celého projektu.

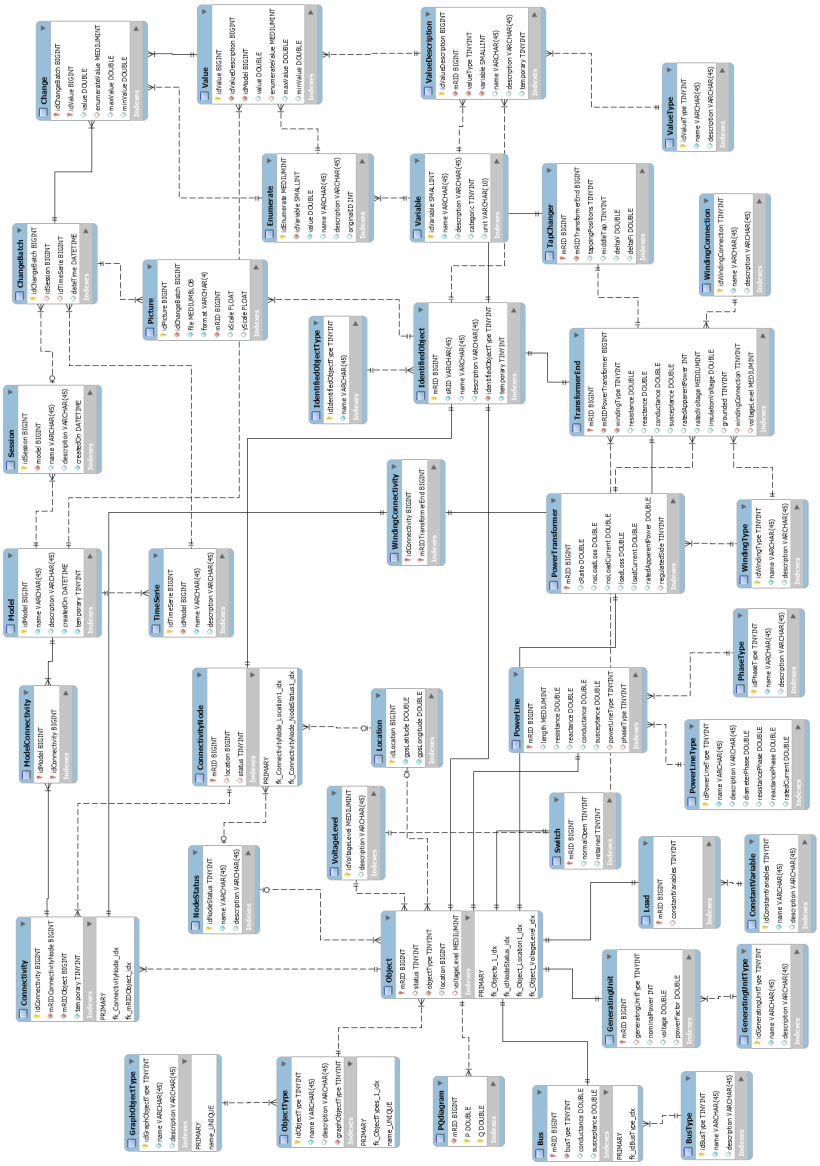
Backend aplikace lze zkompilovat a spustit příkazem `mvn jetty:war`. Je možné přidat také parametr `-P <profil-spuštění>`, kde `<profil-spuštění>` je nutné nahradit případným profilem spuštění, který určuje, k jaké databázi se aplikace bude připojovat. Přidáním parametru `-Dmaven.test.skip` lze vypnout spouštění testů, což značně urychlí start aplikace.

Poté co se spustí backend aplikace, je možné si zobrazit její frontend a používat jej v libovolném webovém prohlížeči na URL adrese:

`http://localhost:8080/vizualizace-prenosove-site`

Pokud backend běží na jiném počítači, je nutné nahradit v URL adrese `localhost` za doménové jméno či IP adresu daného počítače.

B ERA model VPS



Obrázek B.1: ERA model projektu vizualizace přenosové sítě vznikl v rámci tohoto projektu a nejsem jeho autorem.