

**ŽILINSKÁ UNIVERZITA
V ŽILINE**

Fakulta riadenia a informatiky

2D a 3D vizualizácia s využitím fyzikálnych herných prostredí

Dizertačná práca

28360020243010

Študijný program: Aplikovaná informatika

Študijný odbor: Informatika

Školiace pracovisko: Žilinská univerzita v Žiline, Fakulta riadenia a informatiky

Vedúci dizertačnej práce: doc. Ing. Jozef Kostolný, PhD.

Konzultant: doc. Ing. Michal Zábovský, PhD.

Žilina, 2024

Ing. Alexander Brezáni

Čestné vyhlásenie

Vyhlasujem, že som zadanú dizertačnú prácu vypracoval samostatne, pod odborným vedením vedúceho práce/školiťa a používal som len literatúru uvedenú v práci.

Žilina 2. Mája 2024

podpis 

Pod'akovanie

Ďakujem vedúcim dizertačnej práce, pánovi doc. Ing. Michalovi Zábovskému, PhD. za odborné vedenie, cenné rady a pomoc pri vypracovaní písomnej časti dizertačnej práce, ako aj pomoc pri iných podstatných experimentoch súvisiacich s našim výskumom. Pánovi doc. Ing. Jozefovi Kostolnému, PhD. za odborné vedenie, cenné rady a pomoc s organizačnými náležitosťami dizertačnej práce. Mojej manželke Ing. Laure Brezániovej za neustálu podporu, cenné rady, zaujímavé myšlienky a čerstvý pohľad na danú problematiku. Ďalej by som rád poďakoval Ing. Kataríne Zábovskej, PhD., Ing. Borisovi Bučkovi, PhD., Ing. Martinovi Mazúchovi, za diskusie, ktoré obohatili prácu o nové uhly pohľadu, ich podporu a konštruktívnu kritiku. V neposlednej rade by som rád poďakoval rodičom a priateľom za ich podporu počas celého štúdia a písania práce.

Abstrakt

Práca prezentuje súčasné možnosti vizualizácie informácií s ohľadom na efektívny spôsob ich prezentácie a reprezentácia v dvoj a trojrozmernom priestore. Výsledok výskumu predstavuje vytvorenie nových algoritmov a metodológie pre interaktívne spracovanie dát s využitím metód vizualizácie dát a informácií, špecificky vizualizácie rozsiahlych sieťových štruktúr. Dizertačná práca popisuje výskum problému vizualizácie veľkých množín dát a ich štruktúry (prepojení), analyzuje existujúce a nové techniky viacrozmernej vizualizácie dát a zameriava sa na využitie prostredí s implementovaným fyzikálnym prostredím používaných najmä pri vývoji hier a realistických vizualizácií pre potreby vizualizácie sieťových dát. Výsledkom práce sú nové vizualizačné algoritmy pre 3D prostredie použiteľné pre vizualizáciu sieťových štruktúr – grafov a metodológia vytvárania trojrozmerných vizualizácie s ohľadom na použitie prostredí pre vývoj počítačových hier. Podstatnou výhodou takéhoto prístupu je priame zapojenie používateľa-analytika do procesu získavania znalostí z dát.

Kľúčové slová: vizualizácia dáta, viacrozmerné dáta, vizualizačné algoritmy, interaktívne spracovanie dát, analýza dát

Abstract

The work presents the current possibilities of information visualization with regard to the effective way of their presentation and representation in two and three-dimensional space. The result of the research is the creation of new algorithms and methodology for interactive data processing using data and information visualization methods, specifically visualization of large network structures. The dissertation describes research on the problem of visualization of large data sets and their structure (connections), analyzes existing and new techniques of multidimensional data visualization and focuses on the use of environments with an implemented physical environment used mainly in the development of games and realistic visualizations for the needs of network data visualization. The result of the work are new visualization algorithms for a 3D environment that can be used for the visualization of network structures - graphs and a methodology for creating three-dimensional visualizations with regard to the use of environments for the development of computer games. A significant advantage of such an approach is the direct involvement of the user-analyst in the process of obtaining knowledge from data.

Keywords: data visualization, multidimensional data, visualization algorithms, interactive data processing, data analysis

Obsah

1	ÚVOD.....	1
2	OPIS DIZERTAČNEJ PRÁCE	3
2.1	VEDECKÉ OTÁZKY	3
2.2	ZÁKLADNÉ EXPERIMENTÁLNE OVERENIE.....	4
2.3	TÉZY DIZERTAČNEJ PRÁCE	4
2.4	AKTUÁLNY STAV	5
3	SÚČASNÝ STAV RIEŠENEJ PROBLEMATIKY DOMA A V ZAHRANIČÍ.....	7
3.1	VIZUÁLNA ANALÝZA	7
3.2	PRECHOD OD 2D K 3D VIZUALIZÁCIÁM	8
3.3	TEÓRIA GRAFOV	9
3.4	ALGORITMY	9
3.4.1	<i>Fruchterman-Reingold</i>	10
3.4.2	<i>ForceAtlas</i>	10
3.4.3	<i>YifanHu</i>	11
3.4.4	<i>OpenOrd</i>	11
3.4.5	<i>ForceAtlas2</i>	13
3.4.6	<i>Kruhové rozloženie</i>	13
3.4.7	<i>Vretenové rozloženie</i>	14
3.4.8	<i>Geografické rozmiestňovanie</i>	15
3.5	NÁSTROJE	15
4	CIEĽ PRÁCE.....	17
5	METODIKA PRÁCE A METÓDY SKÚMANIA.....	18
5.1	CHARAKTERISTIKA OBJEKTU SKÚMANIA	18
5.1.1	<i>Reprezentácia grafu</i>	18
5.1.2	<i>Charakteristiky vrcholov</i>	23
5.1.3	<i>Charakteristiky hrán</i>	25
5.1.4	<i>Charakteristiky sietí</i>	25
5.2	PRACOVNÉ POSTUPY	28
5.2.1	<i>Vizualizácia a charakteristika sietí</i>	28
5.2.2	<i>Modelovanie sietí a odvodzovanie</i>	30
5.2.3	<i>Procesy v sieťach</i>	31
5.3	SPÔSOB ZÍSKAVANIA ÚDAJOV A ICH ZDROJE	31
5.3.1	<i>Dataset: Game of Thrones (Hry o tróny)</i>	32
5.3.2	<i>Dataset: Witcher (Zaklínač)</i>	32

5.3.3	<i>Dataset: Lazega</i>	33
5.3.4	<i>Dataset: Network Science</i>	33
5.3.5	<i>Dataset: Pokec</i>	33
6	VÝSLEDKY PRÁCE	34
6.1	VIZUÁLNA ANALÝZA V DVOJROZMERNOM PRIESTORE	34
6.1.1	<i>Výhody vizualizácie v 2D priestore</i>	35
6.1.2	<i>Nevýhody vizualizácie v 2D priestore</i>	35
6.1.3	<i>Metódy a algoritmy vizualizácie v 2D</i>	36
6.1.4	<i>Jednoduchý strunový algoritmus</i>	36
6.1.5	<i>Algoritmus Fruchterman-Reingold</i>	38
6.1.6	<i>Barycentrický algoritmus</i>	40
6.1.7	<i>Algoritmus Kamada-Kawai</i>	41
6.1.8	<i>Algoritmus Force Atlas</i>	43
6.1.9	<i>Porovnanie výsledkov</i>	46
6.2	VIZUÁLNA ANALÝZA V TROJROZMERNOM PRIESTORE	59
6.3	METODIKA PRÍPRAVY DÁT PRE VIZUALIZÁCIE V 3D	60
6.3.1	<i>Predspracovanie siete</i>	61
6.3.2	<i>Príprava dát</i>	61
6.3.3	<i>Škádovanie hodnôt hrán</i>	61
6.3.4	<i>Normalizácia parametrov pre hodnoty hrán</i>	62
6.4	ALGORITMUS 1: PRUŽINOVÝ ALGORITMUS (SPRING LAYOUT ALGORITHM)	62
6.4.1	<i>Modifikácia 1</i>	63
6.5	ALGORITMUS 2: SILOU RIADENÝ ALGORITMUS (FORCE DRIVEN ALGORITHM)	64
6.6	ALGORITMUS 3: PRUŽINOVO-SILOVÝ ALGORITMUS	68
6.7	ALGORITMUS 4: PRUŽINOVO-SILOVÝ ALGORITMUS S KONTROLOVANÝM ROZMIESTŇOVANÍM	70
6.7.1	<i>Ukotvenie</i>	73
6.7.2	<i>Modifikácia 1. Zmena veľkosti sférického obmedzenia</i>	73
6.7.3	<i>Modifikácia 2 Ohraničenie priestoru v okolí vrcholov</i>	74
6.7.4	<i>Modifikácia zobrazenia hrán</i>	75
6.7.5	<i>Postup vizualizácie</i>	76
6.8	MODIFIKÁCIE 3D ZOBRAZENIA – INTERAKTÍVNE FUNKCIE	77
6.8.1	<i>Techniky zamerania a kontextu (Focus + Context)</i>	78
6.8.2	<i>Priblíženie a posunutie (Zooming and Panning)</i>	79
7	DISKUSIA	81
7.1	POROVNANIE ČASOVEJ ZLOŽITOSTI	86
7.2	VIZUÁLNE POROVNANIE	87
7.3	POROVNANIE ZHLUKOV A PRIESTOROVEJ VZDIALENOSTI	90

7.3.1	<i>Pružinový algoritmus</i>	91
7.3.2	<i>Silou riadený algoritmus</i>	93
7.3.3	<i>Pružinovo-silový algoritmus s kontrolovaným rozmiestňovaním.....</i>	93
ZÁVER		96
ZOZNAM PRÍLOH.....		II

Zoznam obrázkov

Obrázok 1: Rozmiestňovanie algoritmom Fruchterman-Reingold (zdroj: gephi.org)	10
Obrázok 2: Rozmiestňovanie algoritmom ForceAtlas (zdroj: gephi.org).....	11
Obrázok 3: Rozmiestňovanie algoritmom YifanHu (zdroj: gephi.org)	12
Obrázok 4: Rozmiestňovanie algoritmom OpenOrd (zdroj: gephi.org)	12
Obrázok 5: Rozmiestňovanie algoritmom ForceAtlas2 (zdroj: gephi.org).....	13
Obrázok 6: Rozmiestňovanie algoritmom Circular layout (zdroj: gephi.org).....	14
Obrázok 7: Rozmiestňovanie algoritmom Radial Axis layout (zdroj: gephi.org)	14
Obrázok 8: Rozmiestňovanie algoritmom GeoLayout (zdroj: gephi.org)	15
Obrázok 9: Dataset siete vytvorenej zo sociálnej siete Pokec.....	19
Obrázok 10: Reprezentácia grafu maticou príľahlosti [50]	20
Obrázok 11: Reprezentácia grafu pomocou zoznamu príľahlosti [50]	20
Obrázok 12: Zobrazenie grafu definované pomocou knižnice <i>igraph</i>	22
Obrázok 13: Zobrazenie grafu v nástroji <i>Gephi</i>	23
Obrázok 14: Vizualizácia siete osôb.	29
Obrázok 15: Vizualizácia grafu algoritmom Fruchterman-Reingold.....	38
Obrázok 16: Pseudokód algoritmu Fruchterman-Reingold [34].....	39
Obrázok 17: Pseudokód pre Tutteho barycentrický algoritmus [44]	41
Obrázok 18: Pseudokód algoritmu Kamada-Kawai [63]	43
Obrázok 19: Vizualizácia algoritmom Force Atlas 2	44
Obrázok 20: Porovnanie algoritmov a) Fruchterman-Reingold, b) Force Atlas, c) Force Atlas 2	45
Obrázok 21: Dataset Game of Thrones – základný algoritmus	47
Obrázok 22: Dataset Game of Thrones – Fruchterman-Reingold	47
Obrázok 23: Dataset Game of Thrones – Kamada-Kawai.....	48
Obrázok 24: Dataset Witcher - základný algoritmus.....	49
Obrázok 25: Dataset Witcher – Fruchterman-Reingold	49
Obrázok 26: Dataset Witcher – Kamada-Kawai	50
Obrázok 27: Dataset Lazega - základný algoritmus	51
Obrázok 28: Dataset Lazega - Fruchterman-Reingold	51
Obrázok 29: Dataset Lazega - Kamada-Kawai.....	52
Obrázok 30: Dataset Network Science - základný algoritmus	53
Obrázok 31: Dataset Network Science - Fruchterman-Reingold	53
Obrázok 32: Dataset Network Science – Kamada-Kawai.....	54
Obrázok 33: Dataset Pokec5000 – základný algoritmus	55

Obrázok 34: Dataset Pokec5000 – Fruchterman-Reingold	55
Obrázok 35: Dataset Pokec5000 – Kamada-Kawai.....	56
Obrázok 36: Dataset Pokec1000 – základný algoritmus	57
Obrázok 37: Dataset Pokec1000 – Fruchterman-Reingold	57
Obrázok 38: Dataset Pokec1000 – Kamada-Kawai.....	58
Obrázok 39: Zobrazenie záťažového experimentu	63
Obrázok 40: Vizualizácia dát v nástroji Gephi	65
Obrázok 41: Vrcholy odstrčené silou za okraj vizualizácie	66
Obrázok 42 Ideálny stav síl.....	67
Obrázok 43 Experimentálna vizualizácia dát v Gephi3D	68
Obrázok 44 Výsledná vizualizácia dát v aplikácií	69
Obrázok 45 Vytvorený zhuk spôsobený aplikovanými silami.....	70
Obrázok 46 Sférické obmedzenie	71
Obrázok 47 Vizualizácia dát so sférickým obmedzením.....	72
Obrázok 48 Rôzna veľkosť sférického obmedzenia	73
Obrázok 49 Malé sférické obmedzenie Obrázok 50 Veľké sférické obmedzenie..	74
Obrázok 51 Diagram vizualizácie	77
Obrázok 52 Upriamanie pohľadu na jeden uzol.....	79
Obrázok 53 Pokec1000 - ForceAtlas2 a Kawada – Kawai	88
Obrázok 54 Pokec1000 - Pružinovo-silový algoritmus so skrytými hranami	88
Obrázok 55 Pokec1000 - Pružinovo-silový algoritmus	89
Obrázok 56 Network Science - základný algoritmus a Fruchtreingold algoritmus.....	89
Obrázok 57 Network Science - Pružinovo-silový algoritmus	89
Obrázok 58 Pokec5000 - Fruchterman-Reingold a Force Atlas2	90
Obrázok 59 Pokec5000 Pružinovo-silový Algoritmus	90

Zoznam tabuliek

Tabuľka 1: Zoznam skúmaných datasetov	32
Tabuľka 2: Čas výpočtu pre dataset Game of Thrones v 2D	48
Tabuľka 3: Čas výpočtu pre dataset Witcher v 2D	50
Tabuľka 4: Čas výpočtu pre dataset Lazega v 2D	52
Tabuľka 5: Čas výpočtu pre dataset Network Science v 2D	54
Tabuľka 6: Čas výpočtu pre dataset Pokec5000 a Pokec1000 v 2D	58
Tabuľka 7 Časové porovnanie existujúcich algoritmov	58
Tabuľka 8 Časové porovnanie algoritmov	86
Tabuľka 9 Časové porovnanie pružinovo-silového algoritmu s kontrolovaným rozmiestňovaním	87
Tabuľka 10 - Výsledky experimentov s jednoduchým pružinovým algoritmom (v abstraktných jednotkách používaných v Unity Engine)	92
Tabuľka 11 - Výsledky experimentov so silou riadeným algoritmom (v abstraktných jednotkách používaných v Unity Engine).	93
Tabuľka 12 - Výsledky experimentov s pružinovo-silovo riadeným algoritmom s kontrolovaným rozmiestňovaním (v abstraktných jednotkách používaných v Unity Engine)	94
Tabuľka 13 - Prehľad výkonnostných metrík pre jednoduchý pružinový algoritmus, force-directed algoritmus a force-directed algoritmus s pružinami a kontrolovaným rozmiestňovaním	95

1 ÚVOD

Spracovanie a analýza dát je v súčasnej dobe štandardná činnosť aplikovaná v mnohých odvetviach priemyslu, výskumu, školstva, marketingu a iných. Porozumenie analyzovaným dátam a ich efektívne použitie môže byť v mnohých prípadoch náročný proces, pri ktorom dochádza k mnohým chybám spôsobeným prezentáciou (vizualizáciou) dát, ako aj samotnou interakciou používateľa-analytika s dátami. Zložitosť a štruktúra analyzovaných dát sa neustále zvyšuje, pričom použitie štandardných vizualizačných metód sa v mnohých prípadoch ukazuje ako neefektívne, keďže neposkytujú dostatočnú silu vizualizácie na zachytenie všetkých vlastností analyzovaných dát [1] [2].

Pre získanie informácie o vzťahoch medzi dátami je nutné poznať ich vzájomné prepojenia vychádzajúce z modelovaného konceptu. Okrem štandardných metrík používaných v štatistickej analýze, korelácie a kovariancie, je v prípade tzv. Prepojených dát (angl. *linked data*) možné použiť reprezentáciu dát v podobe grafu a príslušné dobre definované grafové metriky.

Samotná grafová reprezentácia dát má okrem toho viacero výhod. Po prvé, umožňuje vizuálny a intuitívny spôsob pochopenia vzťahov a spojení medzi dátovými bodmi. Po druhé, umožňuje efektívne a flexibilné dopytovanie, pretože konkrétne reprezentácie grafov sú navrhnuté tak, aby sa ľahko prechádzali a prehľadávali. Okrem toho grafová reprezentácia údajov podporuje zložité a dynamické dátové štruktúry, vďaka čomu je vhodná pre scenáre s vyvíjajúcimi sa a vzájomne prepojenými údajmi. Vo výsledku teda grafová reprezentácia dát uľahčuje efektívnu analýzu a modelovanie sietí, ako sú sociálne siete, systémy odporúčaní a dodávateľské reťazce.

Vizualizáciu dát je možné realizovať s využitím viacerých dimenzií priestoru. Každý n-rozmerný priestor má svoje špecifické metódy a algoritmy. Pre človekom spracovateľné výsledky je možné využiť jedno, dvoj, a trojrozmerný priestor napriek skutočnosti, že existujú aj postupy pre vizualizáciu n-rozmerných dát. Základné predpoklady vizualizácie dát v priestore sú [3]:

- **Jednorozmerné (1D) dáta:** Údajové položky môžu byť vizualizované s využitím napr. histogramov. Dáta však majú len jednu premennú a každá dátová položka má jednu hodnotu.
- **Dvojrozmerné (2D) dáta:** Na vizualizáciu údajov s dvoma premennými je možné použiť dvojrozmerné grafy. Tieto dve premenné možno použiť na porovnanie

údajov ich vizualizáciou pomocou čiarového grafu. Na vizualizáciu 2D údajov možno použiť aj bodové grafy, koláčové grafy, stĺpcové grafy a mapy.

- **Trojrozmerné (3D) dáta:** Trojrozmerné grafy, ako napríklad 3D stĺpcové grafy a realistické zobrazenia, môžu zobraziť viac informácií ako 1D a 2D grafy. Na vizualizáciu údajov v 3D priestore sa používajú tri premenné. Väčšina softvérových nástrojov je vhodná na vizualizáciu údajov malých rozmerov, ako sú 2D a 3D dáta.
- **N-rozmerné dáta:** N-rozmerné dáta sú dáta n rozmerov, t.j. viacrozmerné dáta s n rozmermi. V niektorých prípadoch je na vizualizáciu potrebných viac premenných, pričom v takýchto situáciách sa pre zrozumiteľnosť výslednej vizualizácie používa technika viacrozmerného grafu. Príkladmi metód sú rozptylové matice, mapy a glyfy¹.

Viacrozmerný priestor a moderné technológie obohatenej a virtuálnej reality, umožňujú nový pohľad na analýzu dát a následnú interakciu. Využitie viacrozmerného priestoru prináša dostatočnú flexibilitu pre manipuláciu s dátami, čo môže výrazne ovplyvniť porozumenie používateľa-analytika zobrazeným dátam. Za tohto predpokladu môžeme uvažovať o využití moderných technológií, ako platformy pre interakciu s vizualizovanými dátami [4].

Trojrozmerný priestor umožňuje zakomponovať reálne prostredie do procesu analýzy dát, pričom je možné využitie fyzikálnych vlastností a zákonitostí pre zložitejšie vizualizácie. Fyzikálne herné prostredia ako napríklad Unity² alebo Unreal Engine³, poskytujú možnosť zakomponovania fyzikálnych vlastností pre vizualizované dáta, pri zachovaní vhodnej úrovne optimalizácie samotnej prípravy výslednej vizualizácie. Výhodou práce v trojrozmernom priestore je možnosť implementácie grafov a ich využitie v procese vizualizácie komplexných dátových štruktúr. Efektívnym spojením spomínaných technológií a metód, môžeme predpokladať zlepšenie porozumenia analyzovaným dátam, ako aj ich spracovania a použitia.

¹ Glyfy sú grafické entity - zobrazenia, ktoré reprezentujú jednu alebo viac údajových hodnôt prostredníctvom atribútov, ako je tvar, veľkosť, farba a poloha. Používajú sa pri vizualizácii údajov a informácií a sú špeciálne vhodné na zobrazovanie zložitých, viacrozmerných množín údajov.

² <https://unity.com>

³ <https://www.unrealengine.com/>

2 OPIS DIZERTAČNEJ PRÁCE

Cieľom dizertačnej práce je využitie prirodzených vzťahov medzi dátami s ohľadom na grafové reprezentácie dát v procese vizualizácie. Prepojené dáta sú zvyčajne vizualizované ako zložité grafy alebo siete. Vzhľadom na zložitosť údajov je 2D projekcia založená len na súradniciach x a y komplikovaná a neumožňuje prirodzený interakčný mechanizmus pre ďalšie skúmanie údajov. 3D vizualizácia využíva dodatočný rozmer z , vyžaduje však určitú úroveň interakcie a pohybu kamery na odhalenie prekrytých resp. zakrytých informácií. V dizertačnej práci sa zameriavame na 3D vizualizáciu založenú na fyzickom prostredí poskytovanom nástrojom Unity tak, aby sme rešpektovali požiadavky na interakciu, pohyb kamery a využitie fyzických síl na priestorové rozmiestnenie dát. Ukážeme výhody tohto prístupu spolu s obmedzeniami založenými na prispôsobení herného prostredia pre 3D vizualizáciu grafov.

Trojrozmerný priestor však oproti dvojrozmernému prináša nové problémy, s ktorými je potrebné sa vysporiadať, pretože pridaním tretej dimenzie sa mnohé algoritmy používané v 2D priestore stanú neefektívne až nepoužiteľné. V našej práci sme sa zamerali na silou riadené algoritmy, dobre implementovateľné v prostrediach s fyzikálnym modelom, s cieľom usporiadať graf v podobe siete v 3D priestore čo najzrozumiteľnejšie..

2.1 Vedecké otázky

V rámcovom projekte boli zadefinované nasledovné výskumné otázky (VO):

VO1: Akým spôsobom je možné využiť fyzikálne herné prostredia na efektívnu vizualizáciu veľkých dátových množín?

VO2: Aké informácie o dátach je možné vizualizovať pomocou fyzikálnych zákonitostí?

VO3: Ako efektívne vybrať a spojiť existujúce vizualizačné metódy využívajúce fyzikálne zákonitosti, pre účel interaktívnej analýzy dát?

VO4: Ako zahrnúť používateľa-analytika do procesu získavania znalostí z dát, pri zachovaní efektívneho a rýchleho spracovania dátových množín?

VO5: Akým spôsobom zabezpečiť spoľahlivú interakciu používateľa-analytika s vizualizovanými dátami vo viacrozmernom priestore v prostredí obohatenej reality?

Aby sme mohli získať základné informácie pre zodpovedania definovaných výskumných otázok, bolo potrebné realizovať viacero experimentov, vytvoriť niekoľko funkčných prototypov a algoritmov na overenie výsledkov v herných prostrediach.

2.2 Základné experimentálne overenie

Na základe výsledkov úvodných experimentov boli zistené skutočnosti, ktoré definujú v akej miere je možné zaťažiť herné prostredie za účelom vizualizácie veľkého objemu dát. Odpoveď na otázky VO1 a VO2 sú zásadné s ohľadom na správnu definíciu metodológie a algoritmov pri využití dostupných možností implementovaných v herných prostrediach. Ďalšie experimenty boli zamerané na otázku VO3, kde sa preukázali skutočnosti, ktoré popisujú nedokonalosť a chyby algoritmov vytvorených pre 2D priestor v 3D priestore. Jednoduchý prevod algoritmu z 2D do 3D je veľmi zložitý zrealizovať, pretože pri mnohých postupoch slúži dimenzionálne ohraničenie ako jedna z vlastností algoritmu, ktorá musí byť nahradená inou metódou v 3D. Výsledky realizovaných experimentov slúžia na definíciu téz dizertačnej práce.

2.3 Tézy dizertačnej práce

Podľa výsledkov overenia predpokladov pre získanie odpovedí na formulované výskumné otázky je zrejmé, že samotný prevod algoritmov z 2D do 3D nie je praktický. Vizualizácia veľkého objemu dát sa podľa výsledkov experimentov javí ako realizovateľná, za predpokladu, že dáta sú správne upravené a spracované, a zároveň použité vizualizačné techniky a postupy zodpovedajú štruktúre datasetu.

Pre upresnenie problému **skúmania vizualizácie dát v 3D priestore za použitia herných prostredí s využitím silou riadených algoritmov**, sme zadefinovali nasledovné tézy:

1. **Návrh algoritmov pre využitie fyzikálnych herných prostredí na efektívnu vizualizáciu veľkých dátových množín.** Vzhľadom na špecifiká fyzikálneho prostredia je potrebné vytvoriť nové algoritmy, modifikácia existujúcich 2D algoritmov nie je na základe vlastných úvodných experimentálnych zistení postačujúca.
2. **Vytvorenie metodiky pre prípravu dát na vizualizáciu v 3D fyzikálnom hernom prostredí.** Vychádzame z predpokladu, že vhodná modifikácia vstupných dát prispieva k efektívnemu fungovaniu vizualizačných algoritmov.
3. **Využitie vstupných zariadení poskytujúcich 3D vstupy** (Leap motion, CAVE, rozšírená realita a pod.) pre zabezpečenie efektívnej interakcie s 3D vizualizáciou.

Unifikovaná práca s dátami a ich vizualizáciou v 3D priestore je stále predmetom aktívneho výskumu, pričom mnohé zdroje sú zamerané na úzko špecifikované problémy, ktoré nie sú aplikovateľné v širšom rozsahu. **Tézy dizertačnej práce boli definované**

tak, aby ich výsledok mohol reprezentovať jednotný postup vytvárania vizualizácie, pre rôzne štruktúry dát. Interakcia používateľa predstavuje v celom postupe podstatnú rolu, preto sme sa rozhodli zakomponovať nástroje poskytujúce dostatočnú a intuitívnu manipuláciu s 3D priestorom ako jeden z predmetov výskumu.

2.4 Aktuálny stav

Popis aktuálneho stavu vychádza z aplikácií teórie grafov, teórie vizualizácie dát a informácií a vývoja aplikácií pre 3D. **Graf** je abstraktná reprezentácia vzťahov medzi definovanou množinou dátových entít. Teória grafov sa v oblasti matematiky zaoberá štúdiom grafov, ktoré sú definované *matematické štruktúry používané na párové vzťahy medzi objektmi*. Formálne sa vzťahy nazývajú hrany a dátové entity vrcholy.

V prípade, že sú grafy znázornené vizuálne, vrcholy a hrany majú svoju špecifickú vizuálnu reprezentáciu definovanú bodmi/kružnicami a čiarami. **Zobrazenie grafu** je špeciálna vizuálna reprezentácia samotného grafu, ktorá môže výrazne zlepšiť celkové pochopenie problému, ktorý predstavuje základná štruktúra grafu.

Vizualizácia grafov je užitočná pre lepšie pochopenie štruktúry údajov a podporuje objavovanie znalostí. Existujú nástroje na všeobecnú vizualizáciu grafov alebo nástroje na vizualizáciu grafov pracujúce so vzťahmi špecifickými pre oblasť znalostí, napr. molekulárna biológia. Jedným z najpoužívanejších nástrojov je Gephi⁴, ktorý implementuje rôzne algoritmy na vizualizáciu grafov, funkcie pre vizualizáciu dát v reálnom čase, grafové metriky a ďalšie dátovo orientované funkcie.

Väčšina metód pre vizualizáciu grafov je založená na 2D algoritmoch umožňujúcich umiestňovať vrcholy a hrany do dvojrozmernej roviny (angl. *node-link layout*) s ohľadom na celkovú čitateľnosť štruktúry grafu. Zvyčajne sa na vykreslenie vizualizácie používa ortogonálna projekcia⁵. Kvôli veľkosti a zložitosti dát reprezentovaných grafom (zvyčajne označovaným ako sieť) je v 2.5D prístupe pridaná ďalšia dimenzia reprezentujúca hĺbku. Potom sú uzly umiestnené v rôznych „hĺbkach“ na základe toho, akú majú individuálnu hodnotu hĺbky pre polohu [5].

⁴ <https://gephi.org>

⁵ Ortogonálna projekcia je matematický koncept, ktorý sa používa v geometrii a grafike. Je to projekcia bodov alebo objektov na rovinu tak, že sa zachovávajú uhly a vzájomné vzdialenosti medzi nimi.

3D vizualizácie grafov sa vykresľujú pomocou trojrozmernej projekcie a zvyčajne vyžadujú určitú úroveň pohybu kamery na odhalenie prekrytých údajov [6]. Niektoré pokročilé algoritmy pre 3D vizualizácie navyše využívajú silovo orientovaný prístup na generovanie rozloženia pre celkové priestorové rozmiestnenie grafu [7].

Požiadavky na pohyb kamery, interakciu s užívateľom a fyzikálne podmienené rozmiestňovanie definuje základný rámec pre vizualizáciu 3D grafov. Všetky spomenuté funkcionality zabezpečujú herné enginy spolu s pokročilým fyzikálnym a interakčným modelom, detekciou kolízií a interakciou užívateľa v reálnom čase s pokročilými operáciami založenými na kamere.

3 SÚČASNÝ STAV RIEŠENEJ PROBLEMATIKY DOMA A V ZAHRANIČÍ

Téma dizertačnej práce patrí do vednej oblasti vizualizácie dát a informácií. Táto oblasť je špecifická tým, že z časti presahuje do kreatívnej oblasti a dizajnu. Zameranie na prepojené dáta na druhej strane eliminuje potrebu kognitívnej vedy a kvalitatívnych metód pre posudzovanie výslednej vizualizácie a poskytuje možnosť zamerania sa na grafovú analýzu štruktúry dát.

Problematika analýzy sietí (angl. *network analysis*) je špecifickou oblasťou bádania. Napriek relatívne novej definícii vychádza základ vedy o sieťach z teórie grafov a štatistiky. Napriek skutočnosti, že vizualizácia dát a vizuálna analýza dát je súčasťou veľkého množstva nástrojov, je celkové rozpracovanie danej problematiky zväčša limitované, ako bolo spomenuté, aplikačnou doménou resp. špecificky riešeným problémom. V rámci teoreticko-analytickej časti sme preto postupovali najmä experimentálnym spôsobom s aplikáciou rámcových zistení uvedených nižšie.

Analyzované oblasti výskumu vizualizácie sa zameriavajú na odstránenie limitov konvenčných metód na reprezentáciu údajov (často statických a s úzkym rozsahom) tak, aby bolo možné získať dostupné informácie nachádzajúce sa v analyzovaných údajoch [8] [9].

Optimálny spôsob, ako dosiahnuť takéto pokročilé vylepšenia, je využiť potenciál interaktivity na podporu komplexnej analýzy údajov. Takéto interaktívne systémy obohatené o integrované vizuálne rozhrania (synergia modulov pre simuláciu, analýzu a vizualizáciu) sú navrhnuté s cieľom prekonať obmedzenú ľudskú analytickú kapacitu na syntetizovanie množstva rôznorodých údajov do informačných náhľadov [10].

3.1 Vizuálna analýza

Samotný proces interaktívnej vizuálnej analýzy údajov je známy ako *vizuálna analýza* (angl. *Visual Analytics* -VA) [11]. VA spája postupy, ako je dolovanie údajov a štatistické techniky na vizualizáciu a syntetizáciu informácií intuitívnym spôsobom, pričom ďalej využíva vizuálne vnímanie ľudského oka na ich formovanie do základných vedomostí.

Prepojenie vizualizácie s interaktívnymi funkciami je kľúčovým aspektom pri navrhovaní VA systémov. Ťažisko takéhoto prepojenia sa bežne kladlo na 2D zobrazenia obsahujúce abstraktné údaje. Tieto techniky sú celkom vhodné pri zvažovaní vizualizácie spojitých číselných atribútov s cieľom odhaliť potenciálne vzťahy, vzory a korelácie medzi týmito atribútmi. 2D reprezentácie však nie sú vhodné pre všetky úlohy

a v určitých kontextoch nefungujú vôbec. Napríklad pri analýze geopriestorových údajov dochádza v takýchto vizualizáciách k prekrytiu (angl. *occlusion*), čo vedie k strate informácií [12]. Je to najmä kvôli veľkému počtu dátových bodov, ktoré sa navzájom prekrývajú, čo sťažuje ľudskému analytikovi vidieť jednotlivé dátové body a robiť závery o potenciálnych vzťahoch. Toto môže potenciálne narušiť vizuálne vnímanie a závery, ktoré sa týkajú: objavovania (lokalizácie geometrického znázornenia), prístupu (získavania tvaru, farby, resp. inej vlastnosti), priestorových vzťahov (vzájomný vzťah geometrických zobrazení) [13].

Okrem toho, relatívne určovanie polohy v 2D priestore nie je rovnako vhodné na vykonávanie úlohy absolútneho určovania polohy v 3D priestore, pretože prirodzené fyzické obmedzenia 2D priestoru znemožňujú zohľadniť celé spektrum ďalších viacrozmerných geopriestorových atribútov [14]. Alternatívne vizualizácie využívajúce 3D zobrazenia obohatené o komplexné navigačné funkcie môžu byť preto vhodnejšie pri skúmaní vnútornej štruktúry takýchto údajov. 3D vizualizácie v zásade zachytávajú celé spektrum rozmerov základných údajov a podporných informácií, zatiaľ čo techniky interakcie a navigácie napodobňujú zmysel pre skúmanie prostredia reálneho sveta známym spôsobom.

3.2 Prechod od 2D k 3D vizualizáciám

V súčasnosti však narastá potreba kombinovať 2D a 3D vizualizácie a ich vlastné navigačné a prieskumné techniky. Kombinácia 3D pohľadov s tradičnými 2D vizualizáciami dát prostredníctvom viacerých prepojených modulov ponúka viacero výhod. Keďže sa pri skúmaní údajov nespoliehame iba na jednu rovinu reprezentácie, prieskum údajov sa stáva efektívnejším. Pri konceptualizácii takýchto integrovaných vizuálnych systémov je však potrebná aj určitá úroveň sofistikovanosti.

Ciele konceptualizácie je možné zhrnúť do piatich odlišných cieľov dizajnu, ktoré silne závisia od základných údajov a analyzovanej domény [15]:

1. *Vysoký výkon a kvalita 3D geometrického vykresľovania* - dosiahnutie vysoko detailnej kontextovej geometrie.
2. *Možnosti multivariačnej analýzy* - integrácia rôznych typov vizualizačných diagramov na podporu rôznych analytických metód.
3. *Adekvátny výpočet a kvantifikácia priestorových javov* - napr. realistické mapovanie tieňov a odrazové vlastnosti.
4. *Spoločný priestorový kontext pre geometrické reprezentácie* - napr. použitie jednotného súradnicového systému.

5. *Úzka integrácia medzi geometriou a jej atribútovými aspektmi* - v princípe ide o integráciu cieľov 1 a 2.

Aplikácia vizualizácií sa venujú práce [16] [17] [18], zameriavajú sa však na štandardné 2D vizualizácie informácií prakticky použiteľné v reportingu a business-intelligence aplikáciách. Väčšina publikácií z oblasti 3D vizualizácií sa venuje vedeckým vizualizáciám aplikovaným zväčša ako výstup komplexných modelov z oblasti strojárstva alebo stavebníctva [19]. Pre oblasť sieťovej analýzy a vizualizácie, najmä v nástroji Gephi, sa ako jeden z najkvalitnejších zdrojov informácií javí stránka a publikácie Martina Grandjeana [20].

3.3 Teória grafov

Neoddeliteľnou súčasťou problematiky vizualizácie prepojených dát je ich reprezentácia vo forme grafov. Oblasť teórie grafov, resp. sietí, sa venuje veľké množstvo prác v zahraničí, nezanedbateľná časť výskumu sa však realizuje aj na Slovensku.

Základné práce spájajúce teóriu grafov a sietí pochádzajú z konca 90. rokov, kedy dochádzalo k masívnej aplikácii už existujúcich a rozpracovaných poznatkov z oblasti aplikovanej matematiky do oblasti spracovanie dát a ich analýzy. Základné práce ešte operujú s pojmom graf [21] [22] [23], v ďalších rokoch sa však stále viac objavuje pojem sieť [24] [25]. Z algoritmického pohľadu sa teórii grafov venujú samostatné publikácie [26] [27] [28].

Na Slovensku tvoria silný základ práce z oblasti teórie grafov a diskkrétnej matematiky [29] [30] [31]. Tieto práce tvoria základ riešenia problematiky dizertačnej práce, keďže majú kvalitne rozpracovaný aplikačný charakter a sú prakticky použiteľné pre oblasť analýzy dát.

3.4 Algoritmy

Grafové algoritmy tvoria základ pre prácu s grafovými štruktúrami, samotný proces vizualizácie však vyžaduje ďalšie skupiny algoritmov zamerané na rozmiestňovanie a zobrazovanie informácií získaných z dát [32] [33].

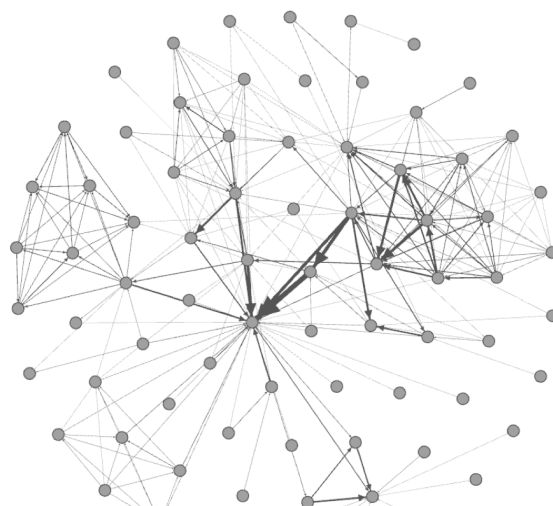
Algoritmy pre rozmiestňovanie vrcholov v 2D tvoria základ pre vizuálnu reprezentáciu grafu. Bez aplikácie vhodného algoritmu nie je možné získať kvalitnú informáciu o štruktúre analyzovaného grafu resp. siete. Dôležitým faktom je, že aplikácia algoritmu na rovnaké údaje môže viesť k rôznym výsledným zobrazeniam.

V nasledujúcej časti uvedieme zásadné algoritmy, ktoré je možné použiť pre rozmiestňovanie, pričom väčšina je k dispozícii v nástroji Gephi a tiež v knižniciach jazyka Python.

3.4.1 Fruchterman-Reingold

Algoritmus Fruchterman-Reingold [34] simuluje graf ako systém hmotných častíc. Uzly sú hmotné častice a hrany sú pružiny medzi časticami. Algoritmy sa snažia minimalizovať energiu tohto fyzikálneho systému. Ide viac-menej o štandardný a veľmi často používaný algoritmus, problémom je jeho pomalosť. Algoritmus je silovo orientovaný, má celkovú zložitosť $O(n^2)$. Je vhodný pre grafy do 1000 vrcholov a nepracuje s ohodnotením hrán.

Author:	Thomas Fruchterman & Edward Reingold ¹
Date:	1991
Kind:	Force-directed
Complexity:	$O(N^2)$
Graph size:	1 to 1 000 nodes
Use edge weight:	No



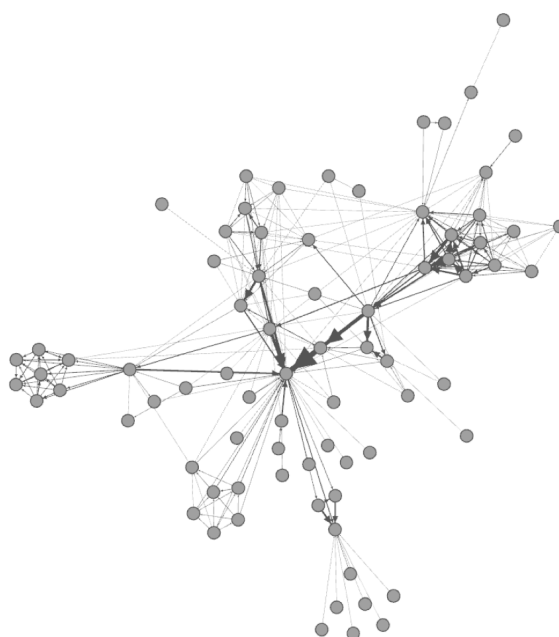
¹ Fruchterman, T. M. J., & Reingold, E. M. (1991). Graph Drawing by Force-Directed Placement. Software: Practice and Experience, 21(11).

Obrázok 1: Rozmiestňovanie algoritmom Fruchterman-Reingold (zdroj: gephi.org)

3.4.2 ForceAtlas

Idie o algoritmus navrhnutý špeciálne pre nástroj Gephi. Zaujímavosťou je, že samotný algoritmus nie je výsledkom matematickej práce, ale inžinierskeho snaženia o dosiahnutie efektívneho zobrazenia grafu (rovnako ako v prípade algoritmu ForceAtlas2). Je navrhnutý tak, aby priestorovo rozmiestnil siete malého sveta (angl. *small-world networks*) resp. bezrozmerné siete (angl. *scale-free networks*). Zameriava sa na kvalitu výsledného zobrazenia a skúmanie reálnych dát tak, aby umožnil rigoróznú interpretáciu grafu s čo najmenším počtom skreslení a dobrú čitateľnosť. Jeho nevýhodou je pomalosť. Algoritmus je silovo orientovaný, má celkovú zložitosť $O(n^2)$. Je vhodný pre grafy do 10 000 vrcholov a pracuje s ohodnotením hrán.

Author:	Mathieu Jacomy
Date:	2007
Kind:	Force-directed
Complexity:	$O(N^2)$
Graph size:	1 to 10 000 nodes
Use edge weight:	Yes



Obrázok 2: Rozmiestňovanie algoritmom ForceAtlas (zdroj: gephi.org)

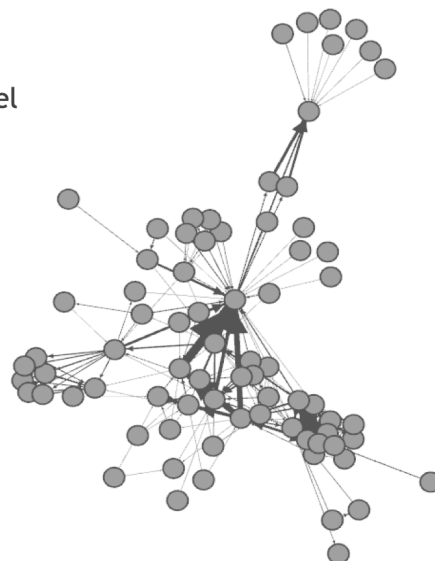
3.4.3 YifanHu

Ide o veľmi rýchly algoritmus s dobrými výsledkami pre veľké grafy [35]. Kombinuje silovo riadený model s technikou zosilňovania grafu (viacúrovňový algoritmus) s cieľom zníženie celkovej zložitosti. Opudivé sily na jeden uzol zo zhľuku vzdialených uzlov sú aproximované pomocou Barnes-Hut aproximácie [36]. Algoritmus podporuje automatické zastavenie. Algoritmus je silovo orientovaný a viacúrovňový, má celkovú zložitosť $O(n * \log(n))$. Je vhodný pre grafy od 100 do 100 000 vrcholov a nepracuje s ohodnotením hrán.

3.4.4 OpenOrd

Pracuje s neorientované ohodnotenými grafmi a cieľi na lepšie rozlíšenie zhľukov [37]. Umožňuje paralelné spúšťanie pre urýchlenie výpočtu a automaticky sa zastaví. Algoritmus je pôvodne založený na algoritme Fruchterman-Reingold a pracuje s pevným počtom iterácií riadených prostredníctvom metódy simulovaného žihania. Vzdialené okraje sú orezané tak, aby sa zhľuky mohli oddeliť. Algoritmus je silovo orientovaný s implementovanou metódou simulovaného žihania, má celkovú zložitosť $O(n * \log(n))$. Je vhodný pre grafy od 100 do 1 000 000 vrcholov a pracuje s ohodnotením hrán.

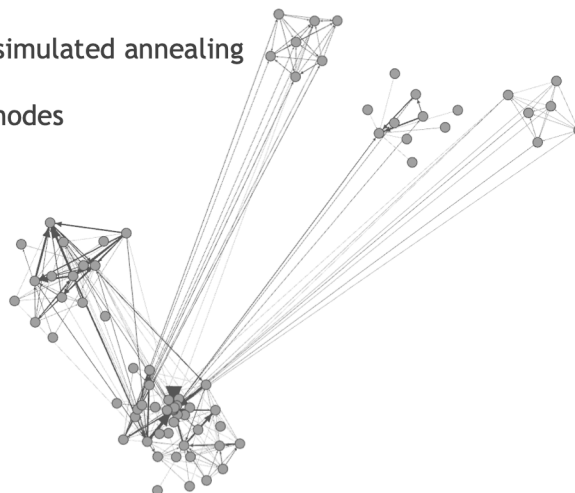
Author: Yifan Hu¹
 Date: 2005
 Kind: Force-directed + multilevel
 Complexity: $O(N \log(N))$
 Graph size: 100 to 100 000 nodes
 Use edge weight: No



¹ Y. F. Hu, Efficient and high quality force-directed graph drawing. The Mathematica Journal, 10 (37-71), 2005.

Obrázok 3: Rozmiestňovanie algoritmom YifanHu (zdroj: gephi.org)

Author: S. Martin, W. M. Brown, R. Klavans, and K. Boyack¹
 Date: 2010 (VxOrd)
 Kind: Force-directed + simulated annealing
 Complexity: $O(N \log(N))$
 Graph size: 100 to 1 000 000 nodes
 Use edge weight: Yes



¹ S. Martin, W. M. Brown, R. Klavans, and K. Boyack, "OpenOrd: An Open-Source Toolbox for Large Graph Layout," SPIE Conference on Visualization and Data Analysis (VDA)., 2011

Obrázok 4: Rozmiestňovanie algoritmom OpenOrd (zdroj: gephi.org)

3.4.5 ForceAtlas2

Ide o vylepšenú verziu algoritmu Force Atlas na zvládnutie veľkých sietí pri zachovaní kvalitného zobrazenia výslednej siete [38]. Odpudzovanie uzlov je aproximované výpočtom Barnes-Hut, čo znižuje zložitosť algoritmu. Príťažlivé a odpudivé sily sú nahradené parametrom škálovania. Algoritmus je silovo orientovaný, má celkovú zložitosť $O(n * \log(n))$. Je vhodný pre grafy do 1 000 000 vrcholov a pracuje s ohodnotením hrán.

Author:	Mathieu Jacomy ¹
Date:	2011
Kind:	Force-directed
Complexity:	$O(N * \log(N))$
Graph size:	1 to 1 000 000 nodes
Use edge weight:	Yes



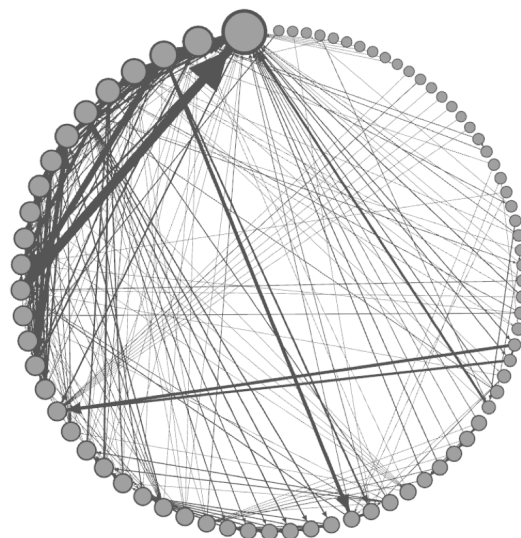
¹ <http://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout/>

Obrázok 5: Rozmiestňovanie algoritmom ForceAtlas2 (zdroj: gephi.org)

3.4.6 Kruhové rozloženie

Ide o jedno zo základných, ale v určitých prípadoch aj veľmi efektívnych rozmiestnení grafu. Algoritmus vykresľuje uzly v kruhu zoradené podľa identifikátora, metriky (stupeň, centralita, medziľahlosť a i.) alebo podľa atribútu. Používa sa na zobrazenie rozloženia uzlov s ich prepojeniami (vzťahmi). Algoritmus nie je silovo orientovaný, má celkovú zložitosť $O(n)$. Je vhodný pre grafy do 1 000 000 vrcholov.

Author: Matt Groeninger¹
Date: 2010
Kind: Circular
Complexity: $O(N)$
Graph size: 1 to 1 000 000 nodes

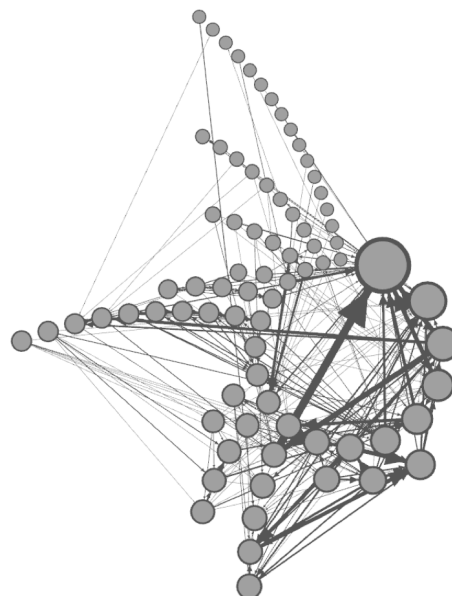


Obrázok 6: Rozmiestňovanie algoritmom Circular layout (zdroj: gephi.org)

3.4.7 Vretenové rozloženie

Algoritmus pre vretenové rozloženie (angl. *Radial Axis Layout*) je doplnkom pre kruhové rozloženie. Zoskupuje uzly a vykresľuje skupiny v osiach (alebo lúčoch) vyžarujúcich von z centrálneho kruhu. Skupiny sa generujú pomocou metriky (stupeň, centralita, medzilahlosť a i.) alebo atribútu. Algoritmus nie je silovo orientovaný, má celkovú zložitosť $O(n)$. Je vhodný pre grafy do 1 000 000 vrcholov.

Author: Matt Groeninger¹
Date: 2011
Kind: Circular
Complexity: $O(N)$
Graph size: 1 to 1 000 000 nodes



¹ <http://gephi.org/plugins/circular-layout/>

Obrázok 7: Rozmiestňovanie algoritmom Radial Axis layout (zdroj: gephi.org)

3.4.8 Geografické rozmiestňovanie

Špecifickým algoritmom vhodným pre priestorovú, geografickú vizualizáciu je algoritmus geografického rozmiestňovania (GeoLayout). Algoritmus pracuje so súradnicami vo forme zemepisnej šírky a dĺžky na nastavenie polohy uzlov v sieti. K dispozícii je niekoľko projekcií vrátane mercatorovej, ktorý používajú Google Maps a ďalšie online služby. Hodnoty atribútov uzla pre súradnice by mali byť v číselnom formáte. Algoritmus nie je silovo orientovaný, má celkovú zložitosť $O(n)$. Je vhodný pre grafy do 1 000 000 vrcholov.

Author:	Alexis Jacomy ¹
Date:	2010
Kind:	Geographic
Complexity:	$O(N)$
Graph size:	1 to 1 000 000 nodes



¹ <http://gephi.org/2010/map-geocoded-data-with-gephi/>

Obrázok 8: Rozmiestňovanie algoritmom GeoLayout (zdroj: gephi.org)

Podrobnejšie informácie je možné nájsť väčšinou v implementáciách algoritmov a metód v špecifických nástrojoch. Kvalitne je spracovaná napríklad v nástroji Gephi [39] resp. na wiki stránke tohoto nástroja [40].

Podrobný prehľad princípov a techník vizualizácie sieťových grafov možno nájsť v upravenom zväzku Kaufmanna a Wagnera [41]. Informácie o vývoji vizualizačných algoritmov je možné nájsť texte di Battistu, Eadesa, Tamassia a Tollisa [42]. Nakoniec, klasické práce venujúce sa vizualizácii údajov vo všeobecnosti sú v knihách Tukeyho [43], Tufteho [44] a Clevelanda [45] [46].

3.5 Nástroje

Praktické informácie pre analýzu dát v 2D pomocou nástrojov pre vizuálnu analytiku je možné nájsť väčšinou vo forme video tutoriálov alebo praktických „receptov“ pre

špecifické problémy a nástroje [47]. Dôležitým zdrojom informácií je konkrétna dokumentácia každého nástroja, pretože implementácia jednotlivých algoritmov a techník sa môže v rôznych nástrojoch zásadne líšiť.

Nástroje pre štandardnú 2D vizualizáciu sú väčšinou známe z oblasti business intelligence a reportingu. V rámci dizertačnej práce spomenieme iba stručne niektoré nástroje, podrobnejšie sa venujeme iba nástrojom reálne použítim pri riešení vedeckého problému.

Základné nástroje pre 2D vizualizáciu, s ktorými je možné sa bežne stretnúť sú Tableau, Google Charts, Visual.ly, IBM Watson, Plotly, Fusioncharts alebo Power BI. Nástroje orientované na vývoj a výskum nových foriem vizualizácie sú D3.js, Grafana. Špecifické nástroje pre vizualizáciu grafov/sietí sú Gephi, Pajek, Cytoscape. Okrem uvedených nástrojov je veľmi často vizualizácia súčasťou nástrojov a jazykov pre analytiku a modelovanie, ako sú napr. MATLAB, R, Python a i.

Nástroje pre 3D vizualizáciu sú väčšinou definované ako nástroje pre vytváranie 3D modelov, či už statických alebo dynamických. V princípe neexistuje natívna podpora pre vizualizáciu dát v takýchto nástrojoch, preto je potrebné si takéto riešenie implementovať v týchto nástrojoch pomocou skriptov resp. knižníc. Medzi 3D nástroje patria Maya, 3D Studio Max, Blender, nástroje pre tvorbu hier sú zastúpené najmä prostrediami Unreal Engine a Unity3D.

V rámci riešenia dizertačnej práce sme použili prostredie Unity3D najmä z dôvodu jeho lepších možností pre vývoj vlastných aplikácií, ktoré nie sú vo svojej podstate počítačovými hrami. V skutočnosti sa vývoj počas riešenia práce nezastavil a v súčasnosti nepovažujeme ani voľbu Unreal Engine vo verzii 5 za nevýhodnú, samotné riešenie však nie je možné vytvárať ako univerzálne pre obe prostredia.

4 CIEĽ PRÁCE

Cieľom práce je vytvorenie metodiky a algoritmov pre trojrozmernú vizualizáciu dát (3D vizualizáciu), ktoré je možné reprezentovať v podobe grafov označovaných aj ako **prepojené dáta** resp. **siete**.

Cieľom 3D vizualizácie je poskytnúť vizuálnu reprezentáciu zložitej vnútornej štruktúry sietí, uľahčiť pochopenie a analýzu sieťových štruktúr a odhaliť vzory a vzťahy v rámci sieťových údajov. 3D vizualizácia umožňuje používateľom skúmať a interagovať so sieťou v trojrozmernom priestore, čím poskytuje komplexnejší a intuitívnejší spôsob práce. Pomocou vizualizácie siete v 3D môžu používatelia získať prehľad o priestorovom usporiadaní uzlov a spojení, identifikovať zhľuky alebo komunity v rámci siete a identifikovať akékoľvek vzory alebo anomálie, ktoré nemusia byť ľahko rozpoznateľné v tradičnom 2D zobrazení. Okrem toho môže 3D vizualizácia siete podporovať prieskum dynamických sietí, kde sa spojenia a vzťahy medzi uzlami časom menia. Celkovo je cieľom 3D vizualizácie siete zlepšiť porozumenie a analýzu zložitých sietí, čo umožní používateľom robiť informované rozhodnutia a získať hlbší prehľad o analyzovaných údajoch.

Predpokladom dizertačnej práce bola hypotéza, že problém analýzy veľkého množstva dát je možné čiastočne zjednodušiť a skvalitniť použitím vizualizačných techník s cieľom vytvorenia lepšej predstavy o charaktere skúmaných dát. V rámci práce sme preto:

- v teoreticko-analytickej časti preskúmali súčasné možnosti vizualizácie informácií s ohľadom na efektívny spôsob ich reprezentácie v 2D a 3D,
- v metodicko-aplikačnej časti vytvorili metodiku a algoritmy na základe výsledkov a zistení a nástrojov pre tvorbu počítačových hier.

Pre implementáciu samotnej vizualizácie je nutné brať do úvahy relatívnu zložitosť vytvárania vizualizácií vo viacrozmerných priestoroch najmä s ohľadom na potenciálny objem spracovávaných dát. Ľudské vnímanie má tiež svoje limity definujúce množstvo informácií, ktoré je človek schopný spracovať zároveň.

5 METODIKA PRÁCE A METÓDY SKÚMANIA

Zo štatistického hľadiska existuje metodologický základ pozostávajúci z postupov a nástrojov, ktoré sú spoločné pre špecifickú netriviálnu podskupinu výskumných oblastí súvisiacich so sieťovou vedou. Okrem toho je možné a celkom užitočné kategorizovať špecifické úlohy analýzy sieťových údajov v rôznych doménach podľa štatistickej taxonómie [48]. V súlade s takouto taxonómiou budeme postupovať od deskriptívnych metód k metódam modelovania a vyvodzovania, pričom tieto sú rozdelené do dvoch podoblastí, ktoré zodpovedajú modelovaniu a definícií samotných sietí a procesom v sieťach.

5.1 Charakteristika objektu skúmania

Graf $G = (V, E)$ je matematická štruktúra pozostávajúca z množiny *vrcholov* V (tiež označovaných *uzly*) a množiny hrán E (nazývaných tiež *linky*), kde prvky E sú neusporiadané dvojice $\{u, v\}$ rôznych vrcholov, kde $u, v \in V$.

Počet vrcholov $N_v = |V|$ a počet hrán $N_e = |E|$ sa nazýva *stupeň* resp. *veľkosť* grafu G . Vrcholy grafu sú často bez straty všeobecnosti jednoducho označované celými číslami $1, \dots, N_v$ a hranami.

Na základe typu hrán môžeme grafy rozdeliť na dva základné typy:

- *neorientovaný graf* – je graf, v ktorom hrany nie sú orientované a teda neexistuje poradie incidentných uzlov (prislúchajúcich hrane).
- *orientovaný graf* – je graf, v ktorom sú hrany orientované (existuje usporiadanie uzlov prislúchajúcich danej hrane). Pre hrany orientovaného grafu platí, že

$$E \subseteq \{ (x, y) \mid (y, x) \in V^2, \text{ kde } x \neq y \}$$

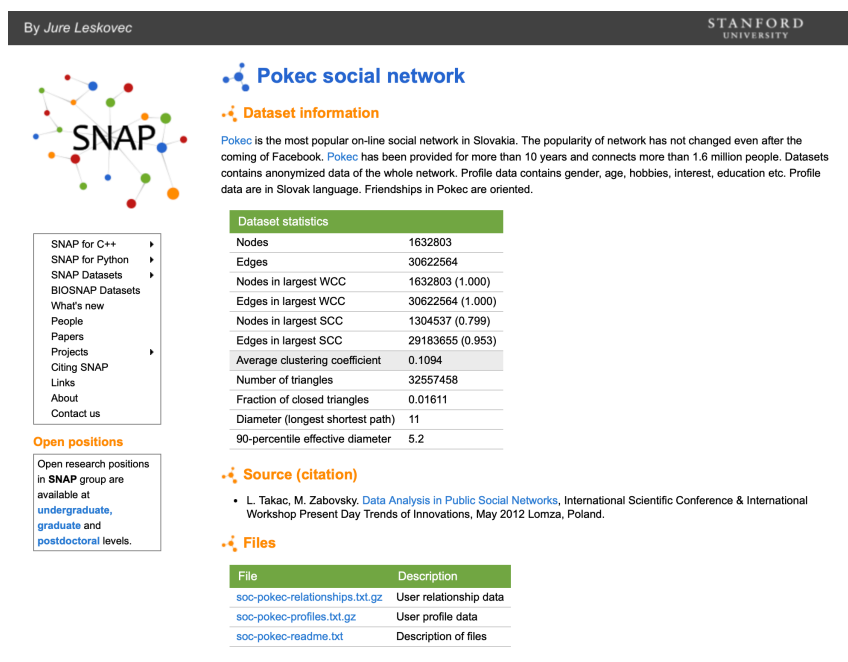
V prípade, že sú hranám priradené špecifické hodnoty (*váhy*), hovoríme o ohodnotenom grafe (resp. konštantnom v prípade nedefinovaných alebo rovnakých váh pre hrany).

5.1.1 Reprezentácia grafu

Informácie na vytvorenie grafu je potrebné definovať v určitej štruktúre. Je potrebné si uvedomiť, že väčšina sietí, ktoré sa v praxi používajú v procese sieťovej analýzy majú niekoľko desiatok až miliónov vrcholov a hrán. Príkladom môže byť dataset *Pokec*⁶, ktorý

⁶ <https://snap.stanford.edu/data/soc-pokec.html>

bol v rámci predchádzajúceho výskumu vytvorený pred niekoľkými rokmi, obsahuje viac ako 1,6 milióna vrcholov a viac ako 30 miliónov hrán (Obrázok 9) [49].



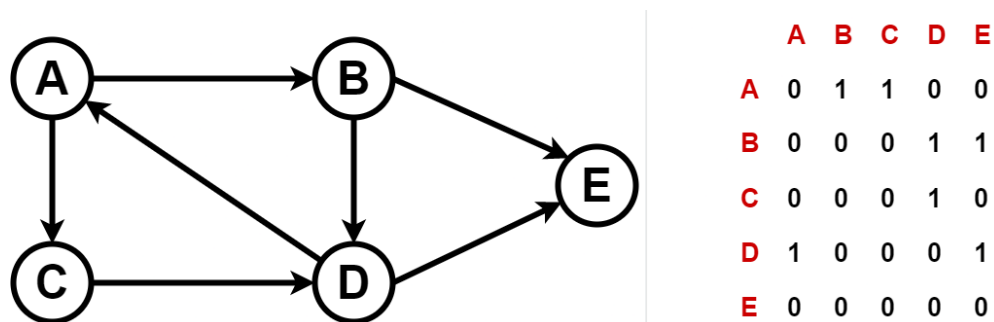
Obrázok 9: Dataset siete vytvorenej zo sociálnej siete Pokec.

Údaje o vrchoch a hranách je preto potrebné mať uložené v dátovom súbore. Základná reprezentácia grafu vyžaduje existenciu troch zoznamy: zoznam susedov, zoznam hrán a matice príľahlosti.

Matica príľahlosti (angl. *adjacency matrix*) reprezentuje štruktúru, ktorá obsahuje informáciu a príslušnosti uzlov hranám. Alternatívnou štruktúrou je **zoznam príľahlosti** (zoznam hrán).

V prípade použitia matice príľahlosti ide o metódu, pri ktorej reprezentujeme graf ako štvorcovú maticu rozmeru $N_v \times N_v$. Ak existuje hrana medzi dvoma vrcholmi, hodnota ich zodpovedajúcej bunky je 1, inak je hodnota 0. Je zrejmé, že s použitím matice príľahlosti je možné implementovať orientované aj neorientované grafy a tiež definovať váhy prislúchajúce hranám.

Príklad reprezentácie grafu pomocou matice príľahlosti je na Obrázok 10. [50]. V prípade ohodnoteného grafu je možné nahradiť jednotky v matici príľahlosti príslušnými hodnotami – váhami.



Obrázok 10: Reprezentácia grafu maticou príľahlosti [50]

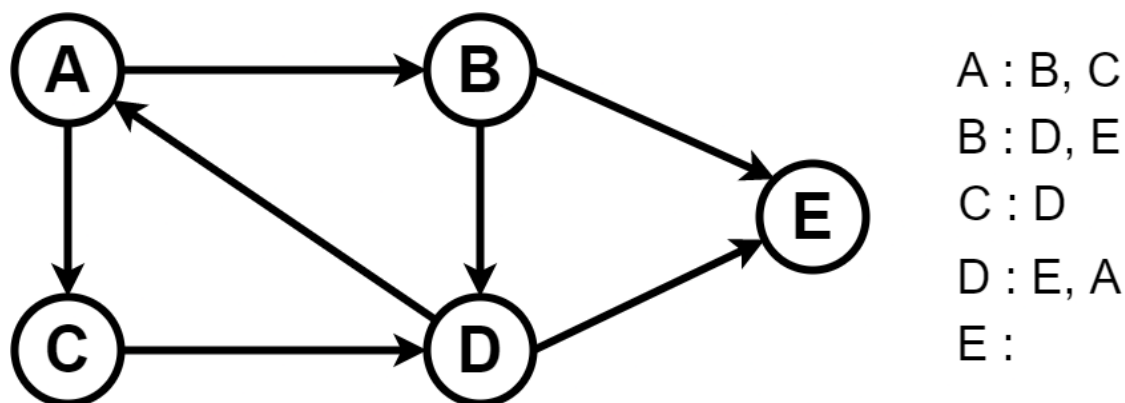
Reprezentácia grafu s použitím matice príľahlosti v jazyku Python⁷, ktorý je používaný v procese analýzy dát, je ukázaná vo výpise nižšie. Je potrebné poznamenať, že pre prácu s grafmi existuje viacero knižníc, ktoré je možné použiť na spracovanie, vizualizáciu a analýzu grafov/sietí.

```

1. graph = [
2.     [0, 5, 9, 0, 0],
3.     [0, 0, 0, 2, 7],
4.     [0, 0, 0, 3, 0],
5.     [1, 0, 0, 0, 13],
6.     [0, 0, 0, 0, 0]
7. ]

```

V prípade použitia reprezentácie pomocou zoznamu susedov je reprezentácia uvedená na Obrázok 11.



Obrázok 11: Reprezentácia grafu pomocou zoznamu príľahlosti [50]

V tejto metóde spájame všetky susedné uzly definovaného uzla dohromady. Z uzla A môžeme prejsť do uzlov B a C. Vytvoríme teda zoznam všetkých susedných uzlov A, ktorý má tvar A: [B, C].

⁷ <https://www.python.org>

V jazyku Python môžeme použiť údajovú štruktúru *dictionary* na implementáciu takéhoto grafu. Kľúče budú konkrétnym uzlom a hodnotou každého kľúča bude zoznam jeho susedných uzlov [50].

```

1. graph = {
2.     'A': ['B', 'C'],
3.     'B': ['D', 'E'],
4.     'C': ['D'],
5.     'D': ['E', 'A'],
6.     'E': [],
7. }
8.
9. for key, val in graph.items():
10.     print(f"{key}-->{val}")

```

V prípade použitia jazyka Python je k dispozícii viacero knižníc pre prácu s grafmi. Keďže v ďalšom texte budeme definovať niektoré základné charakteristiky a metriku pomocou funkcií implementovaných v knižniciach jazyka Python, uvedieme základný prehľad knižníc už v tejto časti práce.

Názvy jednotlivých knižníc sú podobné, preto je v zátvorke uvedený názov inštalácie pomocou *pip* (*dist*) a názov, ktorý možno importovať (*pkg*) [51].

- *python-igraph* (dist: *igraph*, mod: *igraph*) je sada Python rozhraní pre *igraph*, kolekciu nástrojov sieťovej analýzy s dôrazom na efektivitu, prenosnosť a jednoduchosť použitia. Je ľahko inštalovateľná pre širokú škálu platforiem, ťaží z príspevkov používateľov knižnice v jazyku C a väzieb na jazyk R.
- *NetworkX* (dist: *NetworkX*, mod: *networkx*) je balík Pythonu na vytváranie, manipuláciu a analýzu štruktúry, dynamiky a funkcií zložitých sietí. Je implementovaný na základe knižníc NumPy a SciPy, a preto podporuje všetky bežné platformy.
- *graph-tool* (dist: *graph-tool*, mod: *graph_tool*) je efektívny balík na manipuláciu a štatistickú analýzu grafov, založený na C++ Boost Graph Library paralelizovaný pomocou OpenMP. Nie je možné ho nainštalovať pomocou pip-u, ale je k dispozícii cez systém conda.
- *rustworkX* (dist: *rustworkx*, mod: *rustworkx*) Rustworkx je univerzálna grafová knižnica pre Python napísaná v Ruste tak, aby využila výkon a bezpečnosť, ktorú Rust poskytuje. Je navrhnutá tak, aby poskytovala vysoko výkonnú knižnicu pre grafy na všeobecné použitie a pre akúkoľvek aplikáciu Python.

Všetky vyššie uvedené knižnice implementujú funkcie na generovanie grafov, algoritmy, štatistiky a vizualizáciu (do obrázkových súborov, Matplotlib a Cairo). Všetky sú otvoreným softvérom alebo Open Source knižnice.

Pre prezentáciu základných konceptov teórie grafov a algoritmov pre vizualizáciu v 2D budeme používať knižnicu *python-igraph*, ktorej použitie je jednoducho aplikovateľné

nielen v prostrediach s jazykom Python, ale aj v nástrojoch využívajúcich jazyk R resp. C/C++.

Ukážka vytvorenia grafu na Obrázok 10 pomocou knižnice *python-igraph* spolu s grafickým zobrazením výsledného grafu je ukázaná v nasledujúcom kóde.

```

1. import igraph as ig
2. import matplotlib.pyplot as plt
3.
4. # Graph definition
5. n_vertices = 5
6. edges = [(0, 1), (0, 2), (1, 3), (1, 4), (2, 3), (3, 0), (3, 4)]
7. g = ig.Graph(n_vertices, edges, directed=True)
8.
9. # Attributes for vertices
10. g.vs["name"] = ["A", "B", "C", "D", "E"]
11.
12. # Graph plot
13. fig, ax = plt.subplots(figsize=(5,5))
14. ig.plot(g,
15.         target=ax,
16.         vertex_label=g.vs["name"])
17. plt.show()
18.

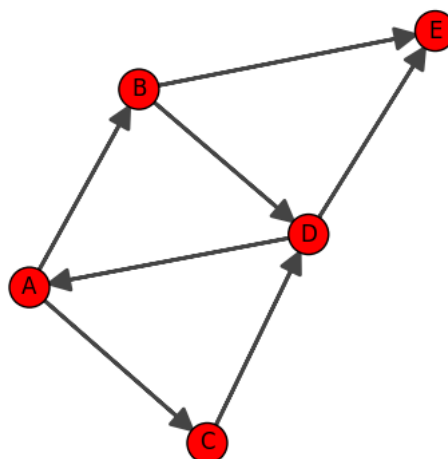
```

Výsledkom skriptu je zobrazenie grafu na Obrázok 12. V ďalšom texte využijeme knižnicu na zobrazenie ďalších vlastností a charakteristík jednotlivých prvkov grafu, ako aj výsledného grafu samotného. Základnú informáciu o grafe môžeme získať pomocou volania *g.summary* s výsledkom zobrazeným v textovej podobe.

```

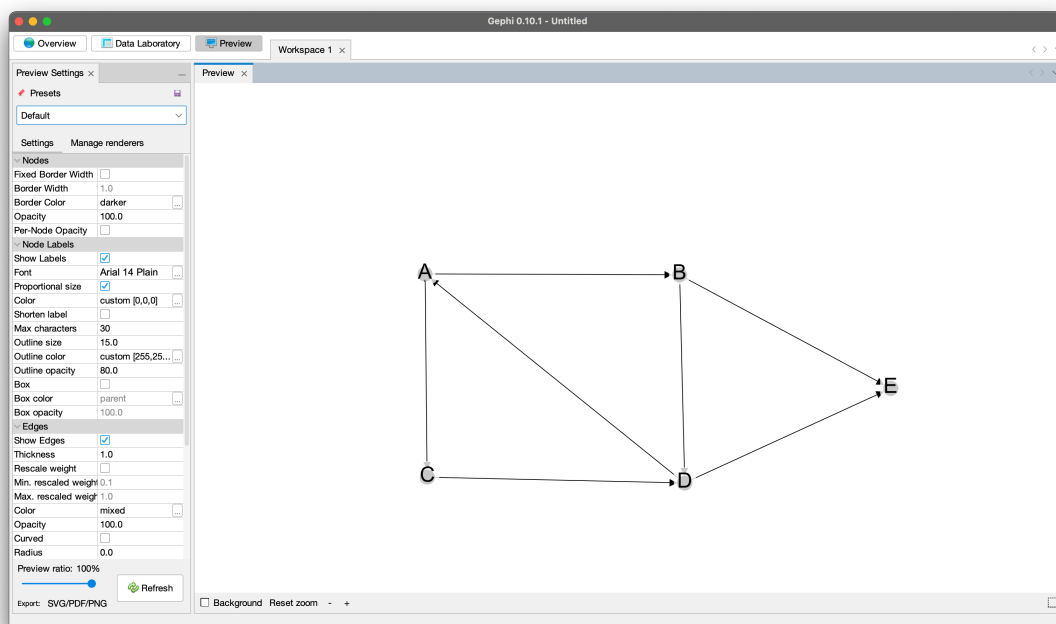
1. g.summary(verbosity=1)
2. # IGRAPH DN-- 5 7 --
3. + attr: name (v)
4. + edges (vertex names):
5. A->B, A->C, B->D, B->E, C->D, D->A, D->E

```



Obrázok 12: Zobrazenie grafu definované pomocou knižnice *igraph*

Rovnaký graf je možné zobraziť aj v nástroji Gephi (Obrázok 13), ktorý umožňuje priamu manipuláciu s celkovou vizuálnou podobou grafu a jednoduchú definíciu pomocou importu dát z rôznych formátov alebo priameho zadanie do tabuľky vrcholov a hrán.



Obrázok 13: Zobrazenie grafu v nástroji Gephi

5.1.2 Charakteristiky vrcholov

Vrchol je základnou štruktúrou, ktorá bude predmetom skúmania. V procese vizualizácie zodpovedá vrchol špecifickej, väčšinou zásadnej skupine charakteristík spracovávaných údajov. Nie je však možné presne definovať, či určitý atribút v údajoch má byť modelovaný ako vrchol alebo hrana.

Základné vlastnosti vrcholov, ktoré je možné skúmať a využiť v procese vizuálnej analýzy a samotnej vizualizácie, vychádzajú z teórie grafov:

- **Stupeň vrcholu** $\deg(v)$ grafu $G = (V, E)$ je rovný počtu hrán incidentných s vrcholom v . Vstupným (resp. výstupným) stupňom sa potom rozumie počet hrán, ktoré do vrchola v vstupujú (resp. z neho vychádzajú).
- **Rozdelenie stupňov vrcholov** - Pre sieťový graf G môžeme definovať f_d ako podiel vrcholov $v \in V$ a ich stupňa $\deg(v) = d$. Množina $\{f_d\} d \geq 0$ sa nazýva rozdelenie stupňov vrcholov grafu G .
- **Sila vrcholu** - V prípade ohodnotených hrán je užitočná ďalšia charakteristika – sila vrcholu. Ide o sumu ohodnotení hrán incidentných s vrcholom v .

Centrálnosť uzla je dôležitá charakteristika s ohľadom na posúdenie dôležitosti uzla v sieti. V priebehu rokov bolo navrhnutých viacero spôsobov určenia centrálnosti uzla. Základným meradlom je vyššie definovaný *stupeň uzla*, existuje však niekoľko ďalších užitočných definícií, ako napríklad *príľahlosť*, *medziľahlosť* resp. *centrálnosť vlastného vektora* [48].

Príľahlosť (blízkosť) (angl. *closeness centrality*) sa pokúšajú zachytiť predstavu, že vrchol je centrálny, ak je blízko k čo najväčšiemu počtu iných vrcholov. Štandardným prístupom, ktorý zaviedol Sabidussi [52], je nechať centrálnosť inverzne meniť s mierou celkovej vzdialenosti vrcholu od všetkých ostatných,

$$c_{Cl}(v) = \frac{1}{\sum_{u \in V} dist(v, u)} \quad (1)$$

kde $dist(v, u)$ je geodetická vzdialenosť medzi vrcholmi $u, v \in V$. Na porovnanie rôznych grafov a iných mier centrálnosti sa táto miera často normalizuje tak, aby ležala v intervale $[0, 1]$, vynásobením faktorom $N_V - 1$.

Medziľahlosť (angl. *betweenness centrality*) sa zameriava na zachytenie pozície, v akej sa vrchol nachádza medzi ostatnými párami vrcholov. Idea je založená na posúdení dôležitosti uzla na základe jeho umiestnenie vzhľadom na cesty v grafe. Uzol, ktorý sa nachádza na väčšom počte ciest je považovaný za dôležitejší ako uzol, ktorý je na menšom počte ciest. Najčastejšie používanú definíciu zaviedol Freeman [53], ktorý ju definuje ako,

$$c_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)} \quad (2)$$

kde $\sigma(s, t|v)$ je celkový počet najkratších ciest medzi s a t , ktoré prechádzajú cez vrchol v , a $\sigma(s, t)$ je celkový počet najkratších ciest medzi s a t bez ohľadu na to, či prechádzajú vrcholom v . V prípade, že najkratšie cesty sú jedinečné, $c_B(v)$ vyjadruje počet najkratších ciest prechádzajúcich cez v . Táto miera centrálnosti môže byť obmedzená na jednotkový interval delením faktorom $(N_V - 1)(N_V - 2)/2$.

Ďalšie miery centrálnosti sú založené na pojmoch „status“, „prestíž“ alebo „rank“. Znamená to, že sa snažia zachytiť myšlienku, že čím centrálnejší sú susedia vrcholu, tým centrálnejší je samotný vrchol. Tieto miery sú implicitné v ich definícií a zvyčajne sa dajú vyjadriť pomocou vyjadrenia vlastných vektorov vhodne definovaných lineárnych systémov rovníc [48], pričom existuje viacero vyjadrení centrálného vlastného vektora.

Napríklad Bonacich [54] vychádzajúci z práce Katza [55] a iných definoval mieru centrálnosti ako

$$c_{Ei}(v) = \alpha \sum_{\{u,v\} \in E} c_{Ei}(u) \quad (3)$$

Vektor $c_{Ei} = (c_{Ei}(1), \dots, c_{Ei}(N_V))^T$ je riešením problému vlastných čísel $A c_{Ei} = \alpha^{-1} c_{Ei}$, kde A je matica príľahlosti pre sieťový graf G . Bonacich [54] tvrdí, že optimálna hodnota $\alpha - 1$ je najväčšia vlastná hodnota A , a teda c_{Ei} je zodpovedajúci vlastný vektor. Keď je G neorientovaný a spojitý, najväčšia vlastná hodnota A bude jednoduchá a jej vlastný vektor bude mať položky, ktoré sú všetky nenulové a zdieľajú rovnaké znamienko. Konvenciou je použitie absolútnych hodnôt, ktoré budú automaticky ležať medzi 0 a 1 podľa ortonormality vlastných vektorov [48].

Rozšírenia týchto meraní centrálnosti z neorientovaných na orientované grafy sú jednoduché. V prípade orientovaných grafov však existujú ďalšie užitočné možnosti. Napríklad algoritmus HITS, založený na koncepte „hubov a autorít“, ako ho zaviedol Kleinberg [56] v kontexte World Wide Web, charakterizuje dôležitosť takzvaných *hubov* (*centier*) podľa počtu autoritatívnych uzlov, na ktoré ukazujú a *autoritatívnych uzlov* podľa počtu hubov, ktoré naň ukazujú. Vzhľadom na maticu príľahlosti A pre orientovaný graf, sú huby určené podľa centrálnosti vlastného vektora matice $M_{hub} = A A^T$ a authority podľa centrálnosti $M_{auth} = A^T A$.

5.1.3 Charakteristiky hrán

Všetky doteraz diskutované súhrnné miery (t. j. stupeň a iné, všeobecnejšie pojmy centrálnosti) sú definované pre vrcholy. V praxi sa najčastejšie modelujú zásadné vlastnosti vrcholmi, čo je dôvodom definície metrík grafu najmä pre vrcholy. Niektoré otázky na vlastnosti grafu sú však prirodzene spojené s hranami. Príkladom môže byť otázka na to, ktoré väzby na sociálnej sieti sú najdôležitejšie pre šírenie informácií alebo fám. Centrálnosť hrany, ktorá analogicky rozširuje centrálnosť vrcholu priradením hodnoty, ktorá odráža počet najkratších ciest prechádzajúcich danou hranou, je potom vhodná metrika, ktorú môžeme použiť [48].

5.1.4 Charakteristiky sietí

Zložitejšia situácia je v prípade charakteristiky grafu resp. siete ako celku. Väčšina otázok v sieťovej analýze sa obmedzuje na otázky týkajúce sa **súdržnosti siete**, čiže do akej miery sú podmnožiny vrcholov súdržné („zlepené“) vzhľadom na vzťah definujúci

hrany v sieťovom grafe. V literatúre sa táto charakteristika označuje anglickým termínom *kohézia* (angl. *network cohesion*).

Existuje mnoho spôsobov, ako môžeme definovať súdržnosť siete v závislosti od kontextu otázky, ktorá sa kladie. Definície sa líšia napríklad v rozsahu, od lokálneho (napr. triády) po globálne (napr. obrovské komponenty), a tiež v rozsahu, v akom sú špecifikované explicitne (napr. kliky) a implicitne (napr. „zhluky“ alebo „spoločenstvá“) [48].

Komponenty grafu resp. podgrafy sú jedným zo spôsobov, ako definovať súdržnosť siete. Ide o špecifikáciu určitého podgrafu (podgrafov), ktorý je predmetom záujmu analytika. Kanonickým príkladom takéhoto podgrafu je *klika*. Pripomeňme, že **kliky** sú úplné podgrafy, a teda podmnožiny vrcholov, ktoré sú plne súdržné v tom zmysle, že všetky vrcholy v podmnožine sú spojené hranami. Ďalšie modifikácie konceptu komponentov, napr. triády, census, motívy je možné získať z literatúry [48] resp. dokumentácie knižníc a nástrojov.

Pojmy hustota a relatívna frekvencia sú spojené práve so špecifickými štruktúrami objavujúcimi sa v grafe. Ide o zisťovanie, či sa špecifická štruktúra v grafe nachádza, prípadne ako často.

Hustota grafu je frekvencia existujúcich hrán vzhľadom na potenciálne hrany. Napríklad v (neorientovanom) grafe G bez slučiek a bez viacnásobných hrán je hustota podgrafu $H = (V_H, E_H)$ definovaná ako

$$den(H) = \frac{|E_H|}{|V_H|(|V_H| - 1)/2} \quad (4)$$

Hodnota $den(H)$ bude ležať medzi nulou a jednotkou a poskytuje mieru toho, ako blízko je H ku klike. V prípade, že G je orientovaný graf, menovateľ v definícii hustoty sa nahradí výrazom $|V_H|(|V_H| - 1)$.

Relatívne jednoduchý koncept hustoty je zaujímavý vďaka voľnosti, ktorú máme pri výbere podgrafu H . Ak napríklad položíme $H = G$, získame hustotu celkového grafu G . Naopak, ak zadefinujem $H = H_v$ ako množinu susedov vrcholu $v \in V$ a hrán medzi nimi, získame mieru hustoty v bezprostrednom susedstve vrcholu v [48].

Relatívna frekvencia sa používa napr. pri definovaní pojmu *zhlukovanie* v grafe. Termínu *zhlukovací koeficient* sa zvyčajne vzťahuje na množstvo

$$c|_T(G) = \frac{3\tau_\Delta(G)}{\tau_3(G)} \quad (5)$$

kde $\tau_\Delta(G)$ je počet trojuholníkov v grafe G a $\tau_3(G)$ počet spojených trojíc (t. j. podgraf troch vrcholov spojených dvoma hranami, niekedy nazývaný aj 2-hviezda). Hodnota $c|_T(G)$ sa alternatívne nazýva *tranzitivita grafu* a je štandardnou veličinou, o ktorú sa zaujíma v literatúra zaoberajúce sa výskumom sociálnych sietí, kde sa označuje aj ako zlomok tranzitívnych trojíc (angl. *fraction of transitive triples*).

Základnou otázkou o štruktúre grafu je, či sa daný graf delí na samostatné podgrafy. Ak nie, môžeme sa pokúsiť kvantifikovať, ako blízko k tomu, aby sme to dokázali, je. S takýmito problémami úzko súvisia otázky spojené s tokom „informácií“ v grafe.

Graf G sa nazýva **k-vrcholovo-spojité**, ak

- a) počet vrcholov $N_V > k$ a
- b) po odstránení ľubovoľnej podmnožiny vrcholov $X \subset V$ mohutnosti $|X| < k$ zostane podgraf spojitý.

Podobne sa G nazýva **k-hranovo-spojité** s hranou, ak

- a) $N_V \geq 2$ a
- b) odstránenie akejkoľvek podmnožiny hrán $Y \subseteq E$ mohutnosti $|Y| < k$ ponecháva podgraf, ktorý je spojený. Vrcholová (hranová) konektivita G je najväčšie celé číslo, takže G je k-vrcholovo- (k-hranovo-) spojitý.

Dá sa ukázať, že vrcholová spojitosť je zhora ohraničená hranovou spojitosťou, ktorá je zase ohraničená zhora minimálnym stupňom d_{min} vrcholov G [48].

Posledná dôležitá charakteristika, **rozklad** resp. **partitioning** grafu, sa vzťahuje na segmentáciu súboru prvkov do prirodzených podmnožín. Formálnejšie ide o rozdelenie $C = \{C_1, \dots, C_K\}$ konečnej množiny S rozkladom S na K disjunktných, neprázdnych podmnožín C_k tak, že $\bigcup_{k=1}^K C_k = S$. Pri analýze sieťových grafov sa je rozklad užitočným nástrojom na hľadanie podmnožín vrcholov, ktoré vytvárajú „súdržnosť“ s ohľadom na definované vzťahy.

Kohézna podmnožina vrcholov sa vo všeobecnosti vzťahuje na podmnožinu vrcholov, ktoré sú medzi sebou dobre prepojené a zároveň sú relatívne dobre oddelené od zostávajúcich vrcholov.

Formálnejšie, algoritmy rozkladu grafov zvyčajne hľadajú rozdelenie $C = \{C_1, \dots, C_K\}$ množiny vrcholov V grafu $G = (V, E)$ takým spôsobom, že množiny $E(C_k, C_{k'})$ hrán

spájajúcich vrcholy v C_k s vrcholmi v C'_k sú relatívne malé v porovnaní s množinami $E(C_k) = E(C_k, C_k)$ hrán spájajúcich vrcholy v C_k . Daný problém rozdelenia grafov sa v literatúre o komplexných sieťach bežne označuje aj ako *detekcia komunit*.

5.2 Pracovné postupy

V tejto kapitole sa budeme zaoberať rôznymi pracovnými postupmi v oblasti analýzy sietí. Sietová analýza je dôležitým nástrojom pre pochopenie vzťahov a štruktúry v rôznych komplexných systémoch.

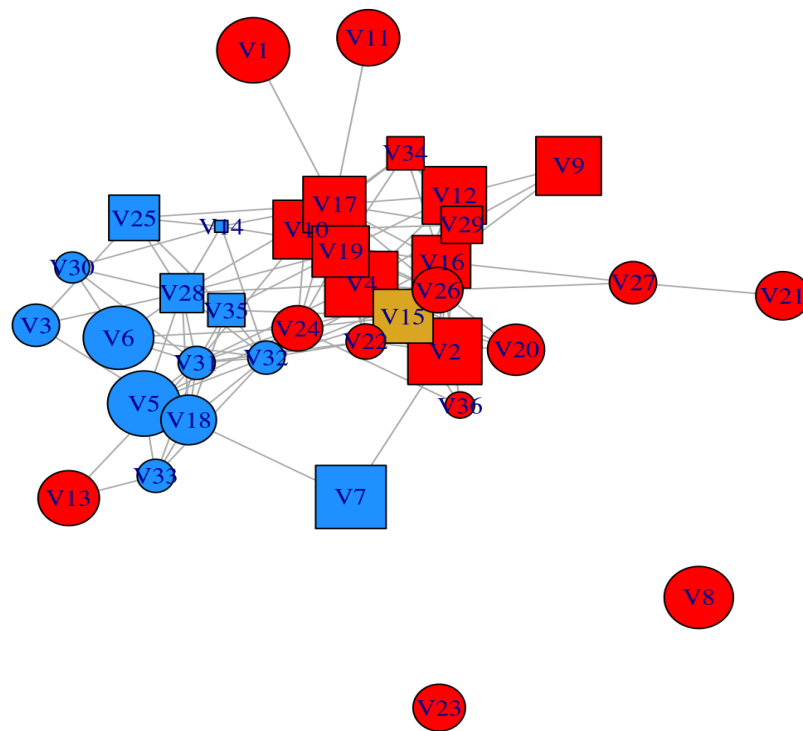
5.2.1 Vizualizácia a charakteristika sietí

Deskriptívna (opisná) analýza údajov je základnou metódou skúmania charakteristík sietí a dát. Spoločne s numerickou charakterizáciou siete sú zvyčajne jedným z prvých krokov v sieťovej analýze. V skutočnosti tvoria deskriptívne analýzy pravdepodobne väčšinu publikovaných sieťových analýz [48].

Príklad vizuálneho zobrazenia osôb (právnikov v právnickej kancelárii) je na Obrázok 14: Vizualizácia siete osôb.. Pôvodné dáta (zobrazené sú vybrané vizuálne významné dáta) obsahujú údaje získané počas výskumu merania vzťahov medzi 71 právnikmi (partnermi a spolupracovníkmi) právnickej firmy, t. j. ich sieť silných spolupracovníkov, poradenskú sieť, sieť priateľstva a siete nepriamej kontroly [57]. Originálne dáta sme spracovali a vizualizovali s použitím jazyka R, celý kód je uvedený v prílohe B.

Vizuálne zobrazenie na Obrázok 14 obsahuje viacero dôležitých aspektov vstupných údajov v jednom diagrame. Na znázornenie siete sa používa graf s vrcholmi zodpovedajúcimi právnikom a hranami pre spoluprácu medzi dvojicami právnikov. Okrem toho sa rozdiely vo farbe vrcholu, tvare, veľkosti a mena používajú na označenie umiestnenia kancelárie, typu praxe, rokov v advokátskej kancelárii a odpracovaných rokov.

Lazega Lawyers

Obrázok 14: Vizualizácia siete osôb.⁸

Charakterizácia sietí (grafov) prostredníctvom numerických hodnôt je ďalším dôležitým aspektom deskriptívnej analýzy sietí. Na rozdiel od štandardných základných štatistických ukazovateľov, ako sú jednoduché určenia stredu (napr. priemer, medián

⁸ Zdrojový súbor údajov pochádza zo sieťovej štúdie partnerstva v oblasti práva obchodných spoločností, ktorá sa uskutočnila v americkej právnickej firme s názvom SG&R, 1988-1991 v Novom Anglicku. Zahŕňa (okrem iného) merania vzťahov medzi 71 právnikmi (partnermi a spolupracovníkmi) tejto firmy, t. j. ich sieť silných spolupracovníkov, poradenskú sieť, sieť priateľstva a siete nepriamej kontroly. Súčasťou súboru údajov sú aj rôzne atribúty členov, vrátane služobného veku, formálneho postavenia, kancelárie, v ktorej pracujú, pohlavia, navštevovanej právnickej školy, individuálnych meraní výkonu (odpracované hodiny, prinesené poplatky), postojov týkajúcich sa rôznych možností politiky riadenia atď. [57]

atď.) a rozptylu (napr. štandardná odchýlka, rozsah atď.) pre množinu reálnych čísel sa sumárne hodnoty pre siete snažia zachytiť charakteristiky grafu. Napríklad pre údaje o právnikovi na obrázku je prirodzené položiť si otázku, do akej miery dvaja právnici, ktorí obaja spolupracujú s tretím právnikom, budú pravdepodobne spolupracovať aj medzi sebou. Tento pojem korešponduje s koncepciou *tranzitivity* na sociálnej sieti a možno ho numericky zachytiť prostredníctvom výpočtu podielu trojíc vrcholov, ktoré tvoria trojuholníky (t. j. všetky tri páry vrcholov sú spojené hranami), zvyčajne zhrnutých do takzvaného zhukového koeficientu. V tomto prípade je príslušná charakteristika explicitným súhrnom štruktúry siete. Na druhej strane, vzhľadom na dva typy právnikov zastúpených v týchto údajoch (t. j. právnici a súdne spory), je tiež prirodzené pýtať sa, do akej miery právnici každého typu spolupracujú s tými istými alebo rôznymi typmi. Tento pojem korešponduje s iným konceptom sociálnej siete – *asortativitou* – a možno ho kvantifikovať pomocou typu korelačnej štatistiky (tzv. *koeficientu assortitivity*⁹), v ktorej sa porovnávajú spojené páry vrcholov. V tomto prípade sa pozornosť sústreďuje na atribút spojený s vrcholmi siete (t. j. právnická prax) a štruktúra siete hrá menej zásadnú úlohu [48].

5.2.2 Modelovanie sietí a odvodzovanie

Samotné sieťovému modelovaniu je potrebné venovať dostatočnú pozornosť. Vo všeobecnosti existujú dve triedy sieťových modelov: matematické a štatistické. *Matematickými modelmi* rozumieme modely špecifikované prostredníctvom (typicky) jednoduchých pravdepodobnostných pravidiel na generovanie siete, kde sú pravidlá často definované v snahe zachytiť konkrétny mechanizmus alebo princíp (napr. „bohatí zbohatnú“). Naproti tomu *štatistickými modelmi* rozumieme modely (často aj pravdepodobnostné) špecifikované aspoň čiastočne so zámerom prispôbiť ich pozorovaným údajom – umožňujúce napríklad posúdenie vplyvu určitých premenných

⁹ Asortativita (pozitívna assortativita) je tendencia uzlov spájať sa s inými uzlami s podobnými vlastnosťami v rámci siete. Disasortativita (negatívna assortativita) je tendencia uzlov spájať sa s inými uzlami s odlišnými vlastnosťami v rámci siete. Vysoký stupeň assortitivity je mierou preferenčnej väzby v organizáciách, kde sú vysoko prepojené vrcholy navzájom prepojené a veľké množstvo vrcholov s nízkym stupňom tvorí zvyšok siete. Asortativita siete meria, ako závisí pravdepodobnosť spojenia medzi dvoma uzlami (jednotlivcami) v sieti od stupňov týchto dvoch uzlov (stupeň je počet spojení, ktoré má každý uzol).

na tvorbu hrán v sieti. Aj keď medzi týmito dvoma triedami modelov určite existuje určitý prienik, literatúra venujúca sa konkrétnej problematike je v zásade samostatná pre každú triedu zvlášť [48].

Zo štatistického hľadiska sú však takéto matematické modely vo všeobecnosti príliš jednoduché na to, aby zodpovedali skutočným sieťovým údajom. Napriek tomu sú užitočné nielen na to, aby umožnili získať formálny pohľad na to, ako môžu špecifické mechanizmy vytvárania hrán ovplyvniť štruktúru siete, ale bežne sa používajú aj pri definovaní nulových tried sietí, voči ktorým sa hodnotí významnosť štrukturálnych prvkov nájdených v pozorovanej sieti.

V rámci praktickej časti dizertačnej práce nám vytvorenie nástroja pre vizualizáciu sietí na základe fyzikálnych zákonitostí umožňuje vytvárať siete priamo bez nutnosti matematického resp. štatistického prístupu. Ide teda o tretí spôsob modelovania sietí, ktorý je možný vďaka využitiu fyzikálnych herných prostredí. Pre overovanie výslednej vizualizácie však použijeme štandardné metódy pre overovanie a popis sietí, ktoré popíšeme v časti venujúcej sa vyhodnocovaniu výsledkov.

5.2.3 Procesy v sieťach

Samostatnou oblasťou pre hodnotenie sietí je pohľad na procesy v sieťach. Ako objekty na reprezentáciu interakcií medzi prvkami komplexného systému sú sieťové grafy často hlavným cieľom sieťovej analýzy. Na druhej strane však v mnohých prípadoch ide o konkrétnu veličinu (alebo atribút) spojenú s každým z prvkov v systéme, ktorá je v konečnom dôsledku najzaujímavejšia. Napriek tomu v prostredí sietí môžeme očakávať, že táto veličina bude významným spôsobom ovplyvnená interakciami medzi prvkami, a preto môže byť sieťový graf stále relevantný pre modelovanie a analýzu. Formálnejšie si môžeme predstaviť stochastický proces ako „žijúci“ v sieti a indexovaný podľa vrcholov v sieti. Rôzne otázky týkajúce sa takýchto procesov možno interpretovať ako problémy predikcie buď statických alebo dynamických sieťových procesov [48].

Väčšina systémov študovaných z perspektívy založenej na sieti sú svojou podstatou dynamické. Nie je preto prekvapením, že mnohé procesy definované v sieťach sú presnejšie chápané ako dynamické, a nie statické procesy.

5.3 Spôsob získavania údajov a ich zdroje

Údaje potrebné pre dosiahnutie cieľov dizertačnej práce je možné získať viacerými spôsobmi. Pre overovanie použitia navrhnutých metód a algoritmov použijeme základné, porovnávacie/benchmarkové datasety verejne dostupné na stránkach niektorých

nástrojov (Gephi), na stránkach Kaggle a pre veľké prepojené dáta použijeme dataset z predchádzajúceho výskumu [49] verejne dostupný na portáli Snap Stanfordskej univerzity. Použitie verejne dostupných datasetov poskytuje možnosť replikácie experimentov a výsledkov uvedených v dizertačnej práci, ako aj možnosť overenia dosiahnutých záverov.

Sumárny prehľad datasetov je uvedený v nasledovnej tabuľke.

Tabuľka 1: Zoznam skúmaných datasetov

Názov	Popis	Platforma	Počet	
			záznamov	atribútov
Game of Thrones	Hry o tróny – vzťahy postáv (5 kníh)	Kaggle	682-1008 ¹⁰	25
Witcher	Zaklínač – fantasy románová séria	Kaggle	2600	12
Lazega	Sieť vzťahov v právnickej firme	R	36	8
Network Science	Sieť spoluautorov	Gephi	1589	3
Pokec	Dáta sociálnej siete	Snap	1632803	7

5.3.1 Dataset: Game of Thrones (Hry o tróny)

Dataset, pomocou ktorého je možné vykonávať hĺbkové analýzy a získať cenné informácie o rôznych aspektoch kníh Hier o tróny. Umožňuje skúmať chronologické poradie kníh, vývoj postáv v celej sérii, analyzovať témy a motívy prítomné v rôznych knihách, skúmať geografické polohy a prostredia a ďalšie aspekty diela pozostávajúceho z piatich kníh (5 samostatných datasetov).

Zdroj: <https://www.kaggle.com/datasets/mmmarchetti/game-of-thrones-dataset>

5.3.2 Dataset: Witcher (Zaklínač)

Zaklínač je fantasy románová séria a je inšpiráciou pre viaceré počítačové hry a seriál. Séria pozostáva z rôznych zápletiiek a časových línií, a preto je zaujímavé analyzovať, ako vyzerá vizualizácia série kníh a ako sa v priebehu série mení dôležitosť postáv.

Zdroj: <https://www.kaggle.com/datasets/avasadasivan/witcher-network>

¹⁰ Dataset obsahuje 5 špecifických datasetov, každý zodpovedajúci príslušnej knihe Hry o tróny.

5.3.3 Dataset: Lazega

Súbor údajov pochádza zo sieťovej štúdie partnerstva v oblasti práva obchodných spoločností, ktorá sa uskutočnila v severovýchodnej americkej právnickej firme s názvom SG&R, 1988-1991 v Novom Anglicku. Zahŕňa (okrem iného) merania sietí medzi 71 právnikmi (partnermi a spolupracovníkmi) tejto firmy, t. j. ich sieť silných spolupracovníkov, poradenskú sieť, sieť priateľstva a siete nepriamej kontroly. Súčasťou súboru údajov sú aj rôzne atribúty členov, vrátane služobného veku, formálneho postavenia, kancelárie, v ktorej pracujú, pohlavia, navštevovanej právnickej školy, individuálnych meraní výkonu (odpracované hodiny, prinesené poplatky), postojov týkajúcich sa rôznych možností politiky riadenia atď. Ide len o podmnožinu pôvodne zhromaždených údajov pre 36 partnerov firmy.

Zdroj: <https://search.r-project.org/CRAN/refmans/sand/html/lazega.html>

5.3.4 Dataset: Network Science

Spoluautorská sieť vedcov pracujúcich na teórii sietí a experimentoch tak, ako ich zostavil M. Newman v máji 2006 [58].

Zdroj: <https://gephi.org/datasets/netscience.gml.zip>

5.3.5 Dataset: Pokec

Pokec je najpopulárnejšia online sociálna sieť na Slovensku, ktorá spája viac ako 1,6 milióna ľudí. Datasets obsahuje anonymizované dáta celej siete. Údaje profilu obsahujú pohlavie, vek, záľuby, záujmy, vzdelanie atď. Údaje profilu sú v slovenskom jazyku. Priateľstvá v Pokeci sú orientované.

Zdroj: <https://snap.stanford.edu/data/soc-pokec.html>

6 VÝSLEDKY PRÁCE

Viacere najlepšie algoritmy na výpočet rozložení jednoduchých neorientovaných grafov patria do triedy známej ako **silovo riadené algoritmy**. Takéto algoritmy, známe tiež ako pružinové algoritmy, **vypočítajú rozloženie grafu iba pomocou informácií obsiahnutých v samotnej štruktúre grafu, namiesto toho, aby sa spoliehali na znalosti špecifické pre danú doménu definované vrcholmi, hranami resp. atribútmi**. Grafy vykreslené pomocou týchto algoritmov sú zo svojej podstaty esteticky príjemné, vykazujú symetriu a majú tendenciu vytvárať rozloženia bez kríženia pre rovinné grafy. Prehľadové práce v tejto oblasti publikoval napr. Di Battista a kol. [59] resp. Brandes [60].

Jednou z prvých silovo orientovaných metód kreslenia (vizualizácie) grafov je algoritmus založený na barycentrických reprezentáciách [61]. Tradičnejšie sa metóda rozloženia pomocou silovej reprezentácie používa podľa Eadesa [62], najčastejšie je však možné pracovať s implementáciou algoritmu podľa Fruchtermana a Reingolda [34]. Obe implementácie sa spoliehajú sily pružiny definované analogicky ako v Hookovom zákone¹¹. Pri týchto metódach existujú odpudivé sily medzi všetkými uzlami a zároveň príťažlivé sily medzi uzlami, ktoré sú vzájomne incidentné [63].

Pri riešení dizertačnej práce sme sa zamerali na zmienené silovo riadené algoritmy z dôvodu využitia fyzikálnych možností herných prostredí pre prácu so štruktúrou grafu bez potreby analýzy grafu pomocou štandardných grafových metrík. **Prvú experimentálnu implementáciu sme vytvorili na základe vyššie definovaných pravidiel pre implementáciu silovo riadených algoritmov s tým, že sme použili implementáciu v 3D priestore definovanom natívne v nástroji Unity3D.**

6.1 Vizualná analýza v dvojrozmernom priestore

Výhody silovo riadených algoritmov v 2D priestore spočívajú v ich jednoduchosti a zrozumiteľnosti. Táto forma vizualizácie poskytuje rýchle a intuitívne prezentácie dát, čo je výhodné pre rôzne aplikácie vizualizácií sociálnych sietí alebo dátových štruktúr. Napriek týmto výhodám však 2D priestor nesie aj nevýhody. Obmedzený priestor môže viesť k prekryvaniu vrcholov a hrán, čo znižuje čitateľnosť grafu. Taktiež, nedostatok

¹¹ Hookov zákon opisuje vzťah medzi deformáciou pevného telesa vyvolanou pôsobením napätia a veľkosťou tohto napätia.

realizmu môže obmedziť schopnosť zachytiť detaily vzťahov a vzdialeností medzi vrcholmi.

Sily medzi uzlami možno vypočítať na základe ich teoretických vzdialeností v grafe, určených dĺžkami najkratších ciest medzi nimi. Napríklad algoritmus Kamada a Kawai [64] využíva sily pružiny úmerné teoretickým vzdialenostiam grafu. Vo všeobecnosti metódy orientované na silu definujú účelovú funkciu, ktorá mapuje každé rozloženie grafu na číslo v R^+ , ktoré predstavuje energiu rozloženia. Táto funkcia je definovaná tak, že nízke energie zodpovedajú usporiadaniám, v ktorých sú susedné uzly blízko od seba na základe určitej vopred špecifikovanej vzdialenosti a v ktorých sú nesusediace uzly dobre rozmiestnené [63].

6.1.1 Výhody vizualizácie v 2D priestore

Tradičný spôsob vizualizácie grafov a sietí, vrátane veľkých prepojených dát má niekoľko výhod. Na základe charakteru algoritmov a obmedzení zobrazovacej plochy sa však s rastúcou zložitou vzťahov resp. veľkosťou siete tieto výhody postupne strácajú. Medzi zásadné pozitíva 2D vizualizácie patrí:

- *Jednoduchosť a zrozumiteľnosť:* V 2D priestore sú silou riadené grafy často jednoduché a ľahko zrozumiteľné. Ide najmä o prípad grafov do veľkosti desiatok vrcholov.
- *Rýchlosť výpočtu:* Algoritmy pre 2D priestor majú tendenciu byť rýchlejšie v porovnaní s ich 3D ekvivalentmi. Existujú aj algoritmy, ktoré je možné jednoducho paralelizovať, čo ďalej zvyšuje rýchlosť vytvorenia výslednej vizualizácie.
- *Široké využitie:* V 2D priestore sú silou riadené algoritmy vhodné pre širokú škálu aplikácií.

6.1.2 Nevýhody vizualizácie v 2D priestore

Pri analýze vizualizácií v 2D priestore sme sa zamerali najmä na identifikáciu nedostatkov dvojrozmernej vizualizácie, ktoré je možné odstrániť pridaním ďalšieho rozmeru a implementáciou silovo orientovaného prístupu s využitím existujúcich možností fyzikálnych prostredí. Medzi základné nedostatky 2D vizualizácie patria:

- *Obmedzený priestor:* V 2D priestore môže dôjsť k prekryvaniu vrcholov a hrán, čo môže zhoršiť čitateľnosť grafu. Tento nedostatok je zásadný v prípade existencie zhlukov, ktorých priemer je malý.

- *Nedostatok realizmu*: 2D vizualizácia nemusí plne zachytiť vzťahy a vzdialenosti medzi vrcholmi. Ide najmä o prípady, kedy je potrebné uvažovať o dĺžke hrán ako o reálnej hodnote vyjadrujúcej napríklad priestorovú dĺžku.
- *Obmedzené možnosti interakcie*: V prípade blízko ležiacich vrcholov nemusí byť zrejماً potreba interakcie s užívateľom napr. vykonanie operácie rotácie scény pre získanie lepšej informácie o vzájomných vzťahoch v grafe.

6.1.3 Metódy a algoritmy vizualizácie v 2D

Základné silovo orientované prístupy majú zásadný obmedzenie dané počtom vykresľovaných/spracovávaných vrcholov. Ich použitie je obmedzená na malé grafy a výsledky sú slabé pre grafy s viac ako niekoľkými stovkami vrcholov. Existuje viacero dôvodov, prečo tradičné algoritmy orientované na silu nefungujú dobre pre veľké grafy.

Jednou z hlavných prekážok ich škálovateľnosti je skutočnosť, že fyzický model má zvyčajne veľa lokálnych miním. Aj s pomocou sofistikovaných mechanizmov na vyhýbanie sa lokálnym minimám nie sú základné algoritmy orientované na silu schopné konzistentne vytvárať dobré rozloženia pre veľké grafy. Rovnako barycentrické¹² metódy nefungujú dobre pre veľké grafy, hlavne kvôli problémom s rozlíšením (pre veľké grafy je minimálna separácia vrcholov veľmi malá, čo vedie k nečitateľným vizualizáciám) [63].

Koncom 90-tych rokov sa objavilo niekoľko techník rozširujúcich funkčnosť silovo riadených metód na grafy s desiatkami tisíc a dokonca stovkami tisíc vrcholov. Hlavnou ideou týchto prístupov je technika viacúrovňového rozloženia, kde je graf reprezentovaný sériou postupne sa zjednodušujúcich štruktúr a usporiadaných v poradí od najjednoduchších po najzložitejšie. Tieto štruktúry môžu byť hrubšie grafy (ako v prístupe Hadanyho a Harela [65], Harela a Korena [66] a Walshawa [67] alebo vertexové¹³ filtrácie ako v prístupe Gajera, Goodricha a Kobourova [68].

6.1.4 Jednoduchý strunový algoritmus

Základná implementácia silovo riadeného algoritmu využíva princíp príťažlivých a odpudivých síl založený na analógií elektro-magnetického silového pôsobenia. Algoritmus formuloval v roku 1984 Eades [62], pričom princíp fungovania je nasledovný [63]:

¹² Barycentrum je ťažisko dvoch alebo viacerých telies, ktoré sa navzájom obiehajú.

¹³ vertex – vrchol (angl., lat.)

Pre reprezentáciu grafu nahradíme vrcholy oceľovými krúžkami a každú hranu nahradíme pružinou, čím vytvoríme mechanický systém. Vrcholy sú umiestnené v určitom počiatočnom usporiadaní a rozmiestnené tak, aby pružinové sily pôsobiace na krúžky-vrcholy posunuli systém do stavu minimálnej energie.

K tomuto základnému princípu sú doplnené dve praktické úpravy:

- 1) sú použité logaritmické silové pružiny; to znamená, že sila vyvíjaná pružinou je:

$$c_1 * \log(d/c_2), \quad (6)$$

kde d je dĺžka pružiny a c_1 a c_2 sú konštanty. Skúsenosti ukazujú, že pružiny Hookeovho zákona (lineárne pružiny) sú príliš silné v prípade, keď sú vrcholy ďaleko od seba. Riešením je použitie logaritmickkej definície sily. V prípade, keď $d = c_2$, nevyvíjajú pružiny žiadnu silu.

- 2) nesusediace vrcholy sa navzájom odpudzujú. Je teda aplikovaná inverzná sila aplikovaného zákona,

$$c_3/d^2, \quad (7)$$

kde c_3 je konštantné a d je vzdialenosť medzi vrcholmi.

Mechanický systém je simulovaný nasledujúcim algoritmom.

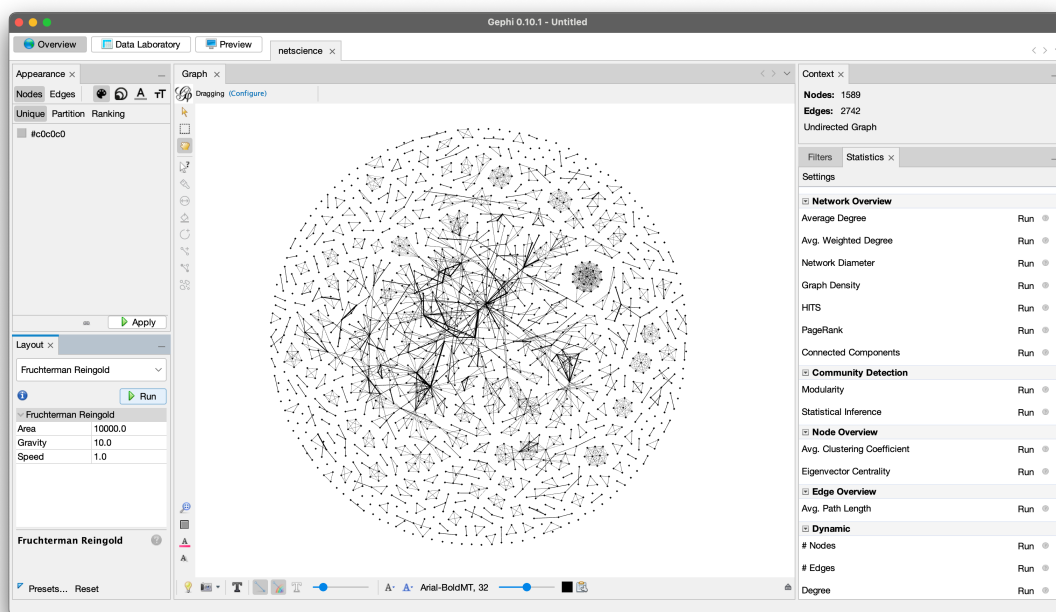
```
1. algoritmus SPRING(G:graf);
2. umiestni vrcholy z množiny G na náhodnú pozíciu;
3. opakuj M krát:
4.     vypočítaj silu pre každý vrchol;
5.     presuň vrchol aplikáciou sily  $c_4 * (\text{sila vypočítaná pre vrchol})$ ;
6. vykresli výsledný graf na výstup.
```

Hodnoty $c_1 = 2$, $c_2 = 1$, $c_3 = 1$, $c_4 = 0,1$ sú vhodné pre väčšinu grafov. Takmer všetky grafy dosahujú minimálny energetický stav po 100-násobnom spustení simulačného kroku, to znamená $M = 100$. Uvedené však, na základe našich experimentálnych zistení, platí iba pre grafy v 2D priestore.

Tento jednoduchý popis princípu fungovania opisuje podstatu pružinových algoritmov a ich prirodzenú jednoduchosť, eleganciu a koncepčnú intuitívnosť. Ciele „esteticky príjemného“ rozloženia boli pôvodne definované dvoma kritériami: 1) všetky dĺžky okrajov by mali byť rovnaké a 2) výsledné rozloženie by malo vykazovať čo najväčšiu symetriu [63].

6.1.5 Algoritmus Fruchterman-Reingold

Algoritmus Fruchterman-Reingold [34] je základným predstaviteľom silovo riadených algoritmov na princípe predstavených vyššie. Výsledok vizualizácie grafu siete spoluautorov (Network Science) pozostávajúceho z 1589 vrcholov a 2742 hrán je pri nastavení parametrov vykresľovanej oblasti, gravitácie a rýchlosti (*Area: 10 000*, *Gravity: 10*, *Speed: 1*) zobrazený na Obrázok 15.



Obrázok 15: Vizualizácia grafu algoritmom Fruchterman-Reingold

Z výslednej vizualizácie je možné identifikovať špecifické rozšírenie základného algoritmu algoritmom Fruchtermana-Reingolda, ktorý pridal k dvom základným kritériám jednoduchého strunového algoritmu tretie. Toto dodatočné kritérium má za cieľ rovnomerné rozloženie vrcholov a považuje vrcholy v grafe za častice resp. telesá, ktoré na seba navzájom pôsobia príťažlivými a odpudivými silami.

Príťažlivé a odpudivé sily sú predefinované na:

$$f_a(d) = d^2 / k, \quad f_r(d) = -k^2 / d, \quad (8)$$

kde f_a je príťažlivá sila, f_r sila odpudivá, d je vzdialenosť medzi dvoma vrcholmi a optimálna vzdialenosť medzi vrcholmi k je definovaná ako:

$$k = C \sqrt{\frac{\text{plocha pre vykreslenie}}{\text{počet vrcholov}}} \quad (9)$$

Tento algoritmus je podobný Eadesovmu algoritmu v tom, že oba algoritmy počítajú príťažlivé sily medzi susednými vrcholmi a odpudivé sily medzi všetkými párami vrcholov. Algoritmus Fruchterman-Reingold pridáva pojem „teplota“, ktorá riadi presun vrcholov tak, že keď sa rozloženie zlepšuje, úpravy sa zmenšujú. Použitie teploty je špeciálnym prípadom všeobecnej techniky nazývanej simulované žihanie [63]. Pseudokód pre algoritmus Fruchterman-Reingold je uvedený nižšie na Obrázok 16.

```

area :=  $W * L$ ; { $W$  and  $L$  are the width and length of the frame}
 $G := (V, E)$ ; {the vertices are assigned random initial positions}
 $k := \sqrt{\text{area}/|V|}$ ;
function  $f_a(x) := \text{begin return } x^2/k \text{ end}$ ;
function  $f_r(x) := \text{begin return } k^2/x \text{ end}$ ;
for  $i := 1$  to  $\text{iterations}$  do begin
    {calculate repulsive forces}
    for  $v$  in  $V$  do begin
        {each vertex has two vectors: .pos and .disp}
         $v.\text{disp} := 0$ ;
        for  $u$  in  $V$  do
            if  $(u \neq v)$  then begin
                { $\delta$  is the difference vector between the positions of the two vertices}
                 $\delta := v.\text{pos} - u.\text{pos}$ ;
                 $v.\text{disp} := v.\text{disp} + (\delta/|\delta|) * f_r(|\delta|)$ 
            end
        end
    end
    {calculate attractive forces}
    for  $e$  in  $E$  do begin
        {each edges is an ordered pair of vertices .v and .u}
         $\delta := e.v.\text{pos} - e.u.\text{pos}$ ;
         $e.v.\text{disp} := e.v.\text{disp} - (\delta/|\delta|) * f_a(|\delta|)$ ;
         $e.u.\text{disp} := e.u.\text{disp} + (\delta/|\delta|) * f_a(|\delta|)$ 
    end
    {limit max displacement to temperature  $t$  and prevent from displacement outside frame}
    for  $v$  in  $V$  do begin
         $v.\text{pos} := v.\text{pos} + (v.\text{disp}/|v.\text{disp}|) * \min(v.\text{disp}, t)$ ;
         $v.\text{pos}.x := \min(W/2, \max(-W/2, v.\text{pos}.x))$ ;
         $v.\text{pos}.y := \min(L/2, \max(-L/2, v.\text{pos}.y))$ 
    end
    {reduce the temperature as the layout approaches a better configuration}
     $t := \text{cool}(t)$ 
end

```

Obrázok 16: Pseudokód algoritmu Fruchterman-Reingold [34]

Pri každej iterácii má základný silový algoritmus zložitosť $O(|E|)$ pre príťažlivé sily a $O(|V|^2)$ pre odpudivé sily. Na zníženie kvadratickej zložitosti odpudivých síl navrhujú Fruchterman a Reingold použiť mriežkový variant ich základného algoritmu, kde sa odpudivé sily medzi vzdialenými vrcholmi ignorujú. Pre riedke grafy s rovnomerným rozložením vrcholov táto metóda umožňuje aproximáciu pre výpočet odpudivých síl zložitosti $O(|V|)$.

6.1.6 Barycentrický algoritmus

Pri návrhu algoritmov pre 3D vizualizáciu sme okrem algoritmu Fruchterman-Reingold využili aj poznatky z implementácie vizualizácie v 2D s použitím barycentrického prístupu podľa Tutteho [61].

Historicky je Tutteho barycentrická metóda z roku 1963 prvým silou riadeným algoritmom na získanie vizualizácie grafu bez kríženia pre daný 3-spojité rovinný graf. Na rozdiel od takmer všetkých ostatných silovo riadených metód, tento algoritmus zaručuje, že výsledná reprezentácia grafu neobsahuje kríženia a všetky strany vizualizovaného grafu sú konvexné [63].

V tomto modeli je sila definovaná hranou (u, v) úmerná vzdialenosti medzi vrcholmi u a v a pružiny majú ideálnu dĺžku nula a teda neexistujú žiadne explicitné odpudivé sily. Sila vo vrchole v je potom definovaná ako:

$$F(v) = \sum_{(u,v) \in E} (p_u - p_v), \quad (10)$$

kde p_u a p_v sú polohy vrcholov u a v . Keďže táto funkcia má triviálne minimum so všetkými vrcholmi umiestnenými na rovnakom mieste, množina vrcholov je rozdelená na pevné a voľné vrcholy. Nastavením parciálnych derivácií silovej funkcie na nulu vzniknú nezávislé sústavy lineárnych rovníc pre x -ovú súradnicu a pre y -ovú súradnicu.

Pseudokód pre Tutteho algoritmus je znázornený na Obrázok 17. Základom algoritmu je myšlienka, že ak je plocha rovinného grafu fixovaná v rovine, potom je možné nájsť vhodné polohy pre zostávajúce vrcholy riešením systému lineárnych rovníc, kde je poloha každého vrcholu reprezentovaná ako konvexná kombinácia polôh jeho susedov.

Barycenter-Draw

Input: $G = (V, E)$; a partition $V = V_0 \cup V_1$ of V into a set V_0 of at least three *fixed* vertices and a set V_1 of *free* vertices; a strictly convex polygon P with $|V_0|$ vertices

Output: a position p_v for each vertex of V , such that the fixed vertices form a convex polygon P .

1. Place each fixed vertex $u \in V_0$ at a vertex of P , and each free vertex at the origin.
2. **repeat**
 foreach free vertex $v \in V_1$ **do**

$$x_v = \frac{1}{\deg(v)} \sum_{(u,v) \in E} x_u$$

$$y_v = \frac{1}{\deg(v)} \sum_{(u,v) \in E} y_u$$

until x_v and y_v converge for all free vertices v .

Obrázok 17: Pseudokód pre Tutteho barycentrický algoritmus [44]

Rovnice v cykle **for** sú lineárne a počet rovníc sa rovná počtu neznámych, čo sa zase rovná počtu voľných vrcholov. Výsledkom riešenia týchto rovníc je umiestnenie každého voľného vrcholu do barycentra jeho susedov. Sústavu rovníc je možné riešiť pomocou Newton-Raphsonovej metódy, pričom výsledné riešenie je navyše jedinečné.

6.1.7 Algoritmus Kamada-Kawai

Algoritmus Kamadu a Kawaia [64] z roku 1989 zaviedol iný spôsob uvažovania o „dobrom“ rozložení grafov. Zatiaľ čo algoritmy Eadesa a Fruchtermana-Reingolda majú za cieľ udržať susedné vrcholy blízko seba a zároveň zabezpečiť, aby vrcholy neboli až príliš blízko seba, Kamada a Kawai využívajú teoretický prístup grafov. *Požadovanú geometrickú (euklidovskú) vzdialenosť medzi dvoma vrcholmi v zobrazení považujú za teoretickú vzdialenosť medzi nimi v príslušnom grafe.* [63]

V takomto modeli by ideálne vykreslenie grafu bolo také, v ktorom sa párové geometrické vzdialenosti medzi nakreslenými vrcholmi zhodujú s teoretickými párovými vzdialenosťami grafu. Keďže tento cieľ nemožno vždy dosiahnuť pre ľubovoľné grafy v 2D alebo 3D euklidovských priestoroch, prístup sa spolieha na nastavenie pružinového systému takým spôsobom, že minimalizácia energie systému zodpovedá minimalizácii rozdielu medzi geometrickými a grafovými vzdialenosťami.

V tomto modeli neexistujú žiadne oddelené príťažlivé a odpudivé sily medzi pármí vrcholov, ale namiesto toho, ak je pár vrcholov (geometricky) bližšie/dalej ako ich zodpovedajúca vzdialenosť grafu, vrcholy sa navzájom odpudzujú/pritahujú na základe definície:

Nech $d_{i,j}$ označuje najkratšiu cestu medzi vrcholom i a vrcholom j v grafe. Potom $l_{i,j} = L \times d_{i,j}$ je ideálna dĺžka pružiny medzi vrcholmi i a j a L je požadovaná dĺžka jednej hrany v zobrazení. [64]

Kamada a Kawai navrhujú hodnotu $L = L_0 / \max_{i < j} d_{i,j}$, kde L_0 je dĺžka strany zobrazovacej plochy a $\max_{i < j} d_{i,j}$ je priemer grafu, t.j. vzdialenosť medzi najvzdialenejším párom vrcholov. Sila pružiny medzi vrcholmi i a j je definovaná ako

$$k_{i,j} = K/d_{i,j}^2, \quad (11)$$

kde K je konštanta.

Riešenie problému s vykreslením vrcholov $|V| = n$ ako častíc $p_1, p_2, \dots, p_n$ v 2D euklidovskom priestore vedie k nasledujúcej funkcii definujúcu celkovú energiu:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{1}{2} k_{i,j} (|p_i - p_j| - l_{i,j})^2. \quad (12)$$

Súradnice častice p_i v 2D euklidovskej rovine sú dané x_i a y_i , čo nám umožňuje prepísať energetickú funkciu ako:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{1}{2} k_{i,j} \left((x_i - x_j)^2 + (y_i - y_j)^2 + l_{i,j}^2 - 2l_{i,j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right). \quad (13)$$

Cieľom algoritmu je nájsť hodnoty premenných, ktoré minimalizujú energetickú funkciu $E(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n)$. Najmä pri lokálnom minime sú všetky parciálne derivácie rovné nule, čo zodpovedá riešeniu $2n$ simultánných nelineárnych rovníc. Preto Kamada a Kawai počítajú stabilnú polohu po jednej častici p_m . Ak zoberieme E ako funkciu iba x_m a y_m , minimum E možno vypočítať pomocou Newton-Raphsonovej metódy tak, že sa v každom kroku algoritmu vyberie častica p_m s najväčšou hodnotou Δm , kde

$$\Delta m = \sqrt{\left(\frac{\partial E}{\partial x_m}\right)^2 + \left(\frac{\partial E}{\partial y_m}\right)^2}. \quad (14)$$

Pseudokód pre algoritmus Kamada-Kawai je znázornený na Obrázok 18.

```

compute  $d_{i,j}$  for  $1 \leq i \neq j \leq n$ ;
compute  $l_{i,j}$  for  $1 \leq i \neq j \leq n$ ;
compute  $k_{i,j}$  for  $1 \leq i \neq j \leq n$ ;
initialize  $p_1, p_2, \dots, p_n$ ;
while ( $\max_i \Delta_i > \epsilon$ )
    let  $p_m$  be the particle satisfying  $\Delta_m = \max_i \Delta_i$ ;
    while ( $\Delta_m > \epsilon$ )
        compute  $\delta x$  and  $\delta y$  by solving the following system of equations:

```

$$\frac{\partial^2 E}{\partial x_m^2}(x_m^{(t)}, y_m^{(t)})\delta x + \frac{\partial^2 E}{\partial x_m \partial y_m}(x_m^{(t)}, y_m^{(t)})\delta y = -\frac{\partial E}{\partial x_m}(x_m^{(t)}, y_m^{(t)});$$

$$\frac{\partial^2 E}{\partial y_m \partial x_m}(x_m^{(t)}, y_m^{(t)})\delta x + \frac{\partial^2 E}{\partial y_m^2}(x_m^{(t)}, y_m^{(t)})\delta y = -\frac{\partial E}{\partial y_m}(x_m^{(t)}, y_m^{(t)})$$

$$x_m := x_m + \delta x;$$

$$y_m := y_m + \delta y;$$

Obrázok 18: Pseudokód algoritmu Kamada-Kawai [63]

Algoritmus Kamada-Kawai je výpočtovo náročný, pričom vyžaduje výpočet najkratšej cesty pre všetky dvojice vrcholov, ktorý možno vykonať so zložitou $O(|V|^3)$ pomocou Floyd-Warshallovho algoritmu alebo $O(|V|^2 \log |V| + |E||V|)$ pomocou Johnsonovho algoritmu. Napriek vyššej časovej a priestorovej zložitosti algoritmus prispieva k jednoduchšej a intuitívnej definícii „dobrého“ rozloženia grafu: *Rozloženie grafu je dobré, ak geometrické vzdialenosti medzi vrcholmi zodpovedajú vzdialenostiam medzi nimi definovanými grafom (hranami).*

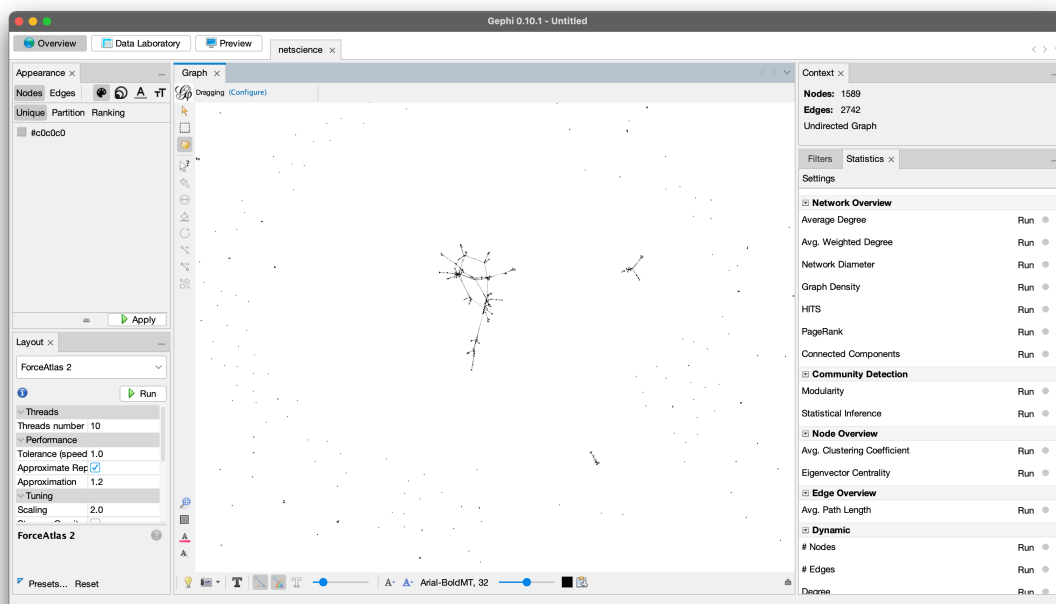
6.1.8 Algoritmus Force Atlas

ForceAtlas2 je silovo riadený algoritmus, ktorý simuluje fyzický systém s cieľom priestorového rozmiestnenia siete/grafu. Uzly sa navzájom odpudzujú ako nabité častice, zatiaľ čo hrany priťahujú svoje uzly ako pružiny. Tieto sily vytvárajú pohyb, ktorý konverguje do rovnovážneho stavu. Cieľom je hypotéza, že táto konečná konfigurácia pomôže interpretácii údajov [38].

Vizualizácia riadená silou je špecifická umiestnením každého uzla v závislosti od ostatných uzlov. Tento proces závisí len od spojení medzi uzlami – hrán, prípadné atribúty uzlov sa neberú do úvahy. Táto stratégia má svoje nevýhody najmä s ohľadom na možnosť zaseknutia sa v lokálnom minime. Algoritmus nie je deterministický a súradnice každého bodu nie sú obrazom žiadnej konkrétnej premennej. Výhodou

algoritmu je, že umožňuje vizuálnu interpretáciu štruktúry. Jeho podstatou je transformácia štruktúrálnej blízkosti na vizuálnu blízkosť, čo uľahčuje výslednú analýzu.

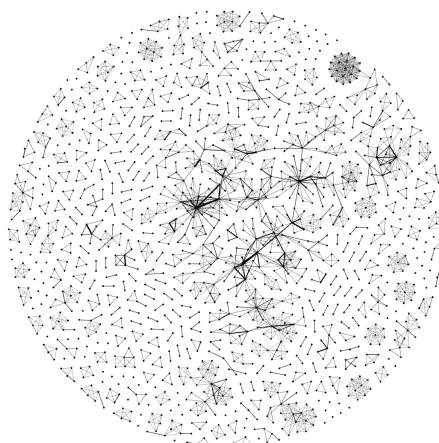
Algoritmus predstavuje praktický prístup k vizualizácii grafov, keďže je optimalizovaný na celkovú rýchlosť a výsledok zobrazenia bez príslušnej matematickej presnosti definície jednotlivých krokov. Modifikáciou algoritmu je jeho verzia *Force Atlas 2*.



Obrázok 19: Vizualizácia algoritmom Force Atlas 2

Obrázok 19 ukazuje výslednú vizualizáciu pre dataset Network Science s použitím algoritmu Force Atlas 2. Na obrázku je vidieť, že pokiaľ vrcholy netvoria prirodzený zhukluster sú umiestnené náhodne na ploche. Tento nedostatok je možné modifikovať úpravou príťažlivej sily resp. veľkosťou plochy pre vykreslenie výslednej vizualizácie.

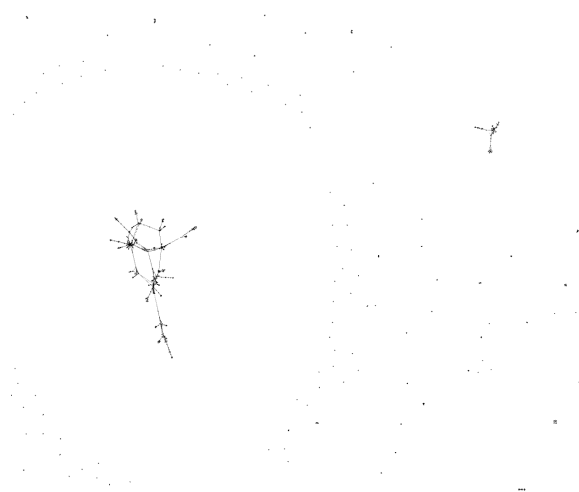
Porovnanie výsledkov algoritmov Fruchterman-Reingold, Force Atlas a Force Atlas 2 vytvorených v nástroji Gephi je na Obrázok 20.



a)



b)



c)

Obrázok 20: Porovnanie algoritmov a) Fruchterman-Reingold, b) Force Atlas, c) Force Atlas 2

6.1.9 Porovnanie výsledkov

Výsledky vizualizácie dát základnými verziami silovo riadených algoritmov sa zásadne líšia, čo je dôsledkom aplikácie rôzneho princípu silového pôsobenia. V rámci práce s 2D algoritmi sme tiež pozorovali určité špecifické vzory, ktoré výsledné vizualizácie obsahujú a na základe ktorých je možné identifikovať typ použitého algoritmu.

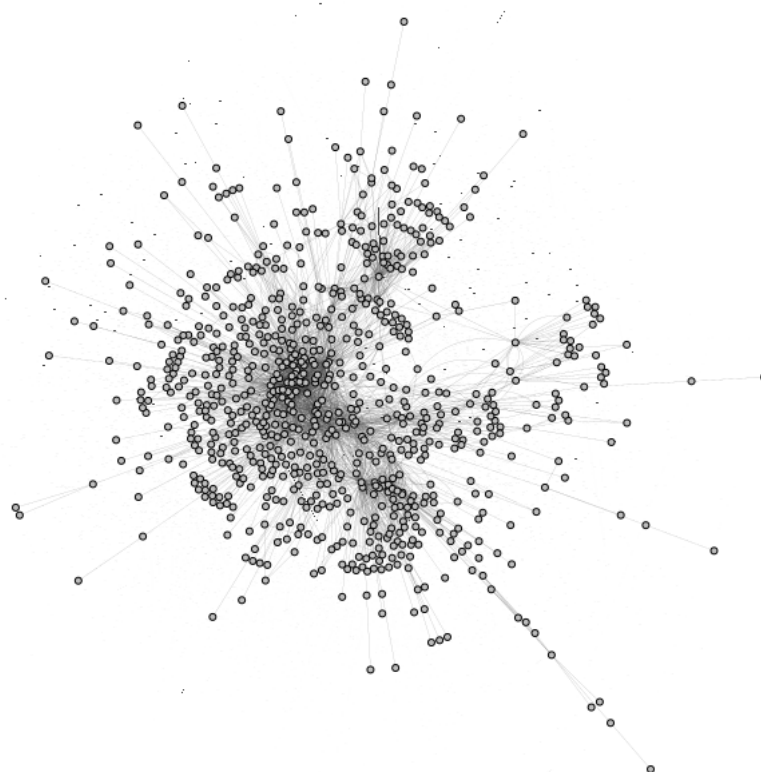
Pre porovnanie sme zvolili nasledovné algoritmy, ktoré sú implementované v knižnici *python-igraph* [69] [70]:

- základný silovo riadený algoritmus implementovaný v knižnici *python-igraph*,
- Fruchterman-Reingold ako základný silovo riadený algoritmus pre 2D vizualizáciu,
- Kamada-Kawai ako pokročilý algoritmus pre silovo riadený prístup k vizualizácii grafov v 2D.

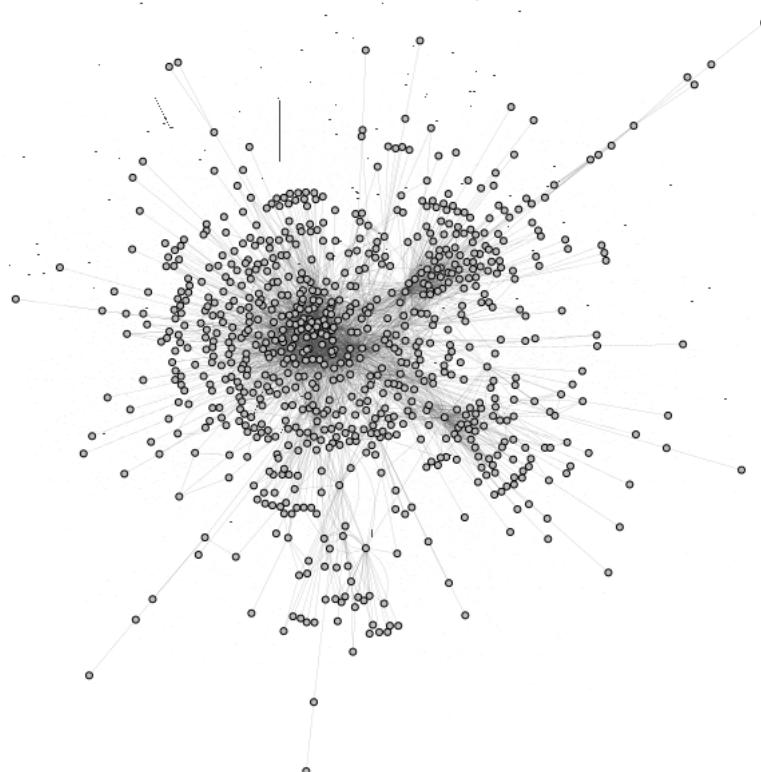
Využili sme knižnicu jazyka Python, keďže nástroj Gephi primárne používa vlastný algoritmus Force Atlas a priebežne vizualizuje výsledky algoritmov s potrebou ich manuálneho ukončenia používateľom. V jazyku Python máme možnosť algoritmus sledovať podrobne, vrátane celkového času potrebného pre prípravu výslednej vizualizácie.

Dataset Game of Thrones

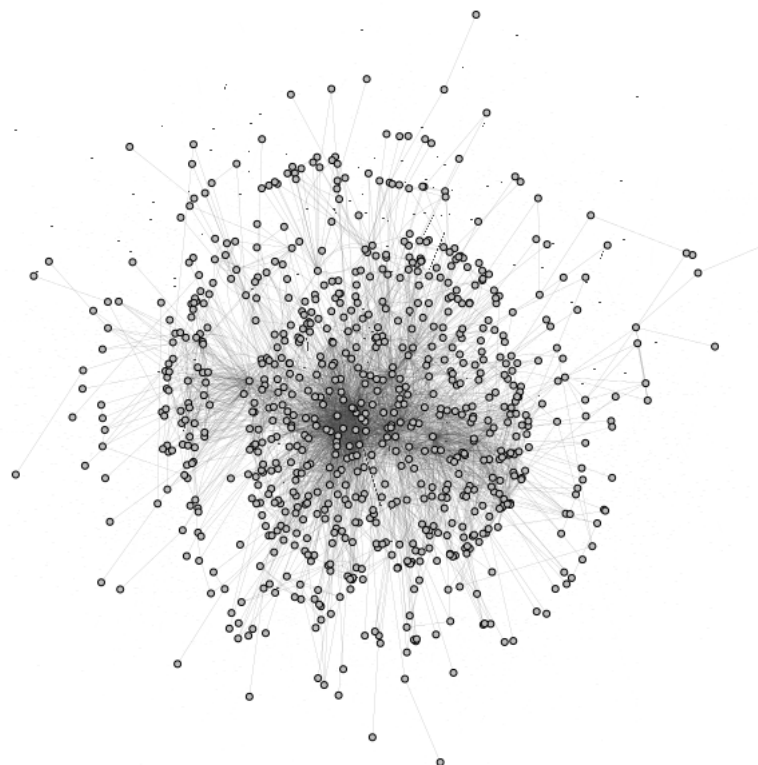
Dataset umožňuje skúmať chronologické poradie kníh, vývoj postáv v celej sérii, analyzovať témy a motívy prítomné v rôznych knihách, skúmať geografické polohy a prostredia a ďalšie aspekty diela pozostávajúceho z piatich kníh (5 samostatných datasetov). Sieť vytvorená zo všetkých piatich datasetov obsahuje 796 vrcholov a 3 909 hrán. V prípade analyzovaného datasetu sú výkonovo časy vizualizácie pre všetky tri algoritmy podobné. Nevýhodou algoritmu Kamada-Kawai je, že nevytvoril prirodzené zhluky tak, ako je to vidieť u prvých dvoch algoritmov.



Obrázok 21: Dataset Game of Thrones – základný algoritmus



Obrázok 22: Dataset Game of Thrones – Fruchterman-Reingold



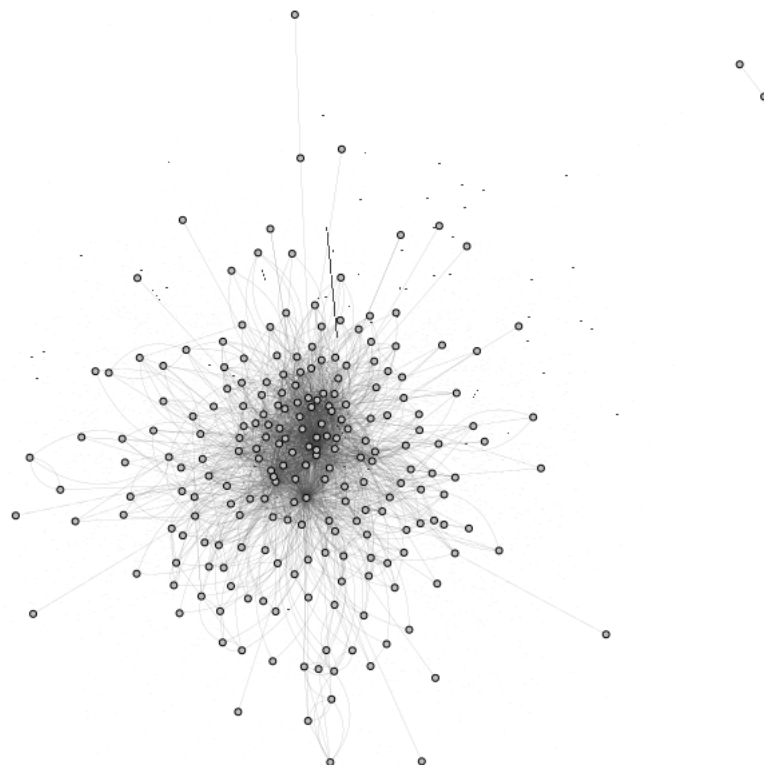
Obrázok 23: Dataset Game of Thrones – Kamada-Kawai

Tabuľka 2: Čas výpočtu pre dataset Game of Thrones v 2D

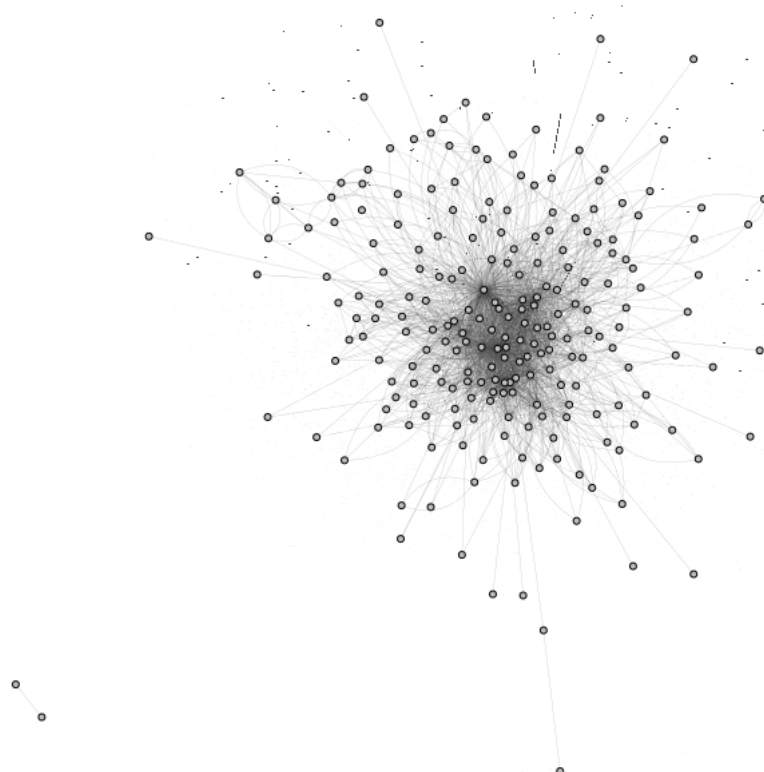
Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Game of Thrones	796 / 3909	3510 ± 62,20	3490 ± 48,60	3350 ± 55,30

Dataset Witcher

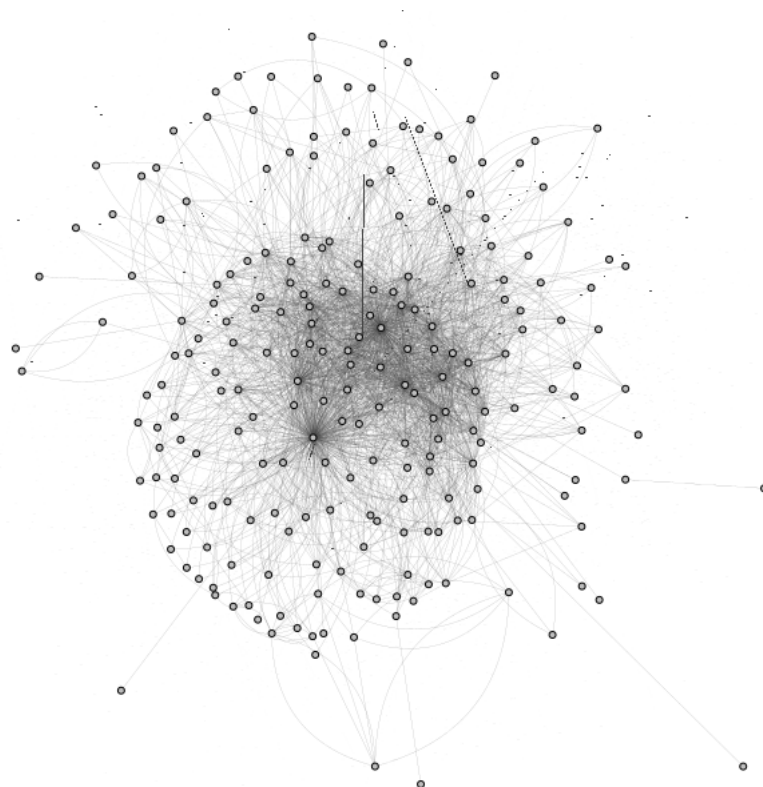
Dataset Witcher je vhodný pre vizualizáciu interakcií postáv – sociálnych interakcií. Dôsledkom uvedenej skutočnosti je existencia väčšieho množstva hrán v porovnaní s vrcholmi. Táto skutočnosť spôsobuje neprehľadnosť v prípade uzlov, ktoré často interagujú s inými uzlami, čo je vidieť pri všetkých troch overovaných algoritmoch. Sieť obsahuje 224 vrcholov a 2 600 hrán. Ako najlepšia vizualizácia sa javí vizualizácia pomocou algoritmu Kamada-Kawai, rýchlosť vykreslenia je pri všetkých troch algoritmoch takmer identická.



Obrázok 24: Dataset Witcher - základný algoritmus



Obrázok 25: Dataset Witcher – Fruchterman-Reingold



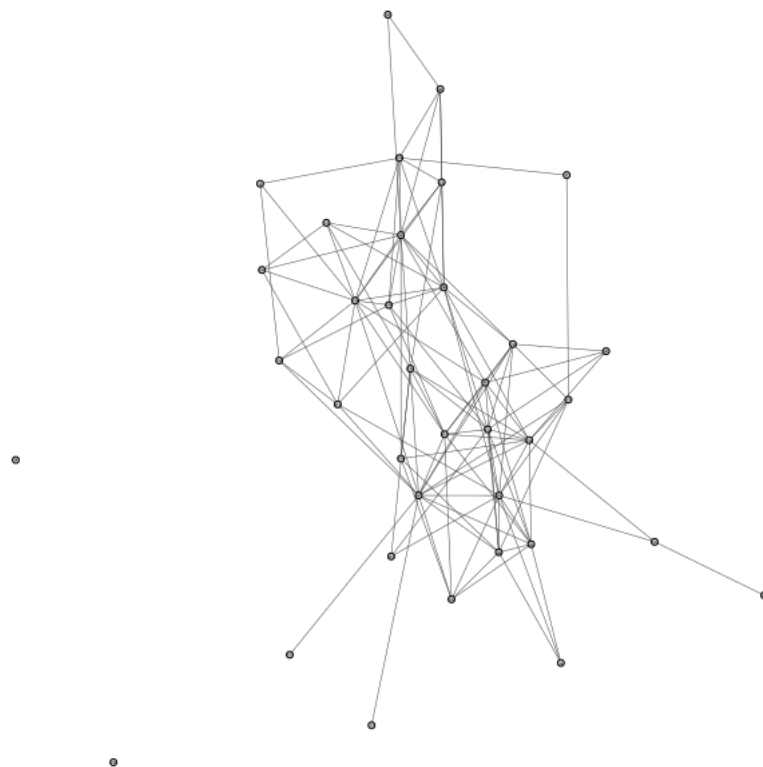
Obrázok 26: Dataset Witcher – Kamada-Kawai

Tabuľka 3: Čas výpočtu pre dataset Witcher v 2D

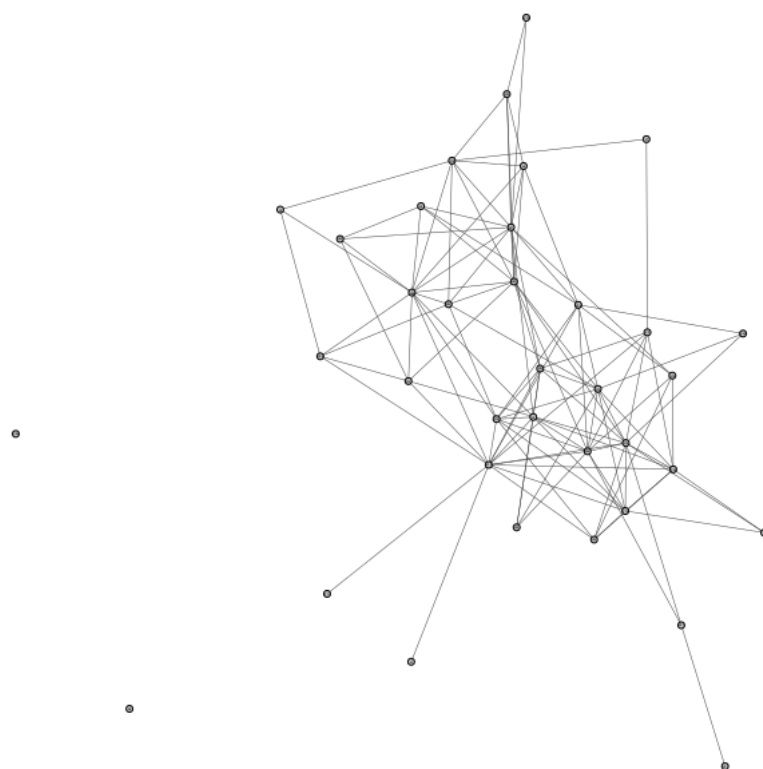
Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Witcher	224 / 2600	1120 ± 39,80	1130 ± 21,50	1130 ± 19,00

Dataset Lazega

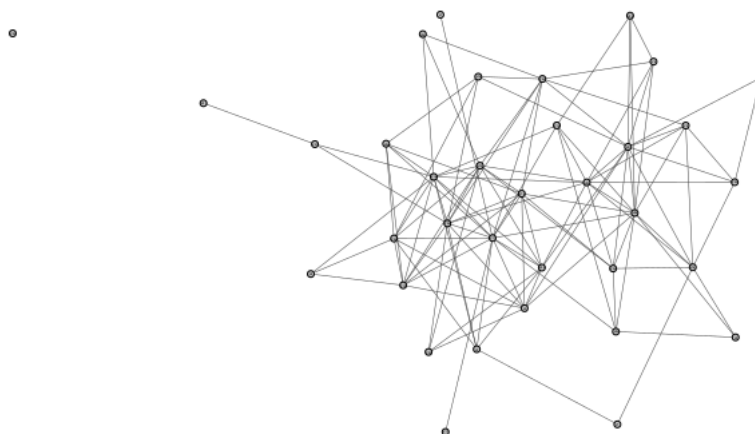
Súbor údajov pochádza zo sieťovej štúdie partnerstva v oblasti práva obchodných spoločností, ktorá sa uskutočnila v severovýchodnej americkej právnickej firme s názvom SG&R, 1988-1991 v Novom Anglicku. Ide teda o vizualizáciu vzťahov – sociálnych interakcií. Sieť obsahuje 36 vrcholov a 115 hrán. Z hľadiska vizualizácie je možné vidieť podobné zobrazenia pre všetky tri porovnávané algoritmy, pričom priestorovo najsymetrickejšie je zobrazenie základného algoritmu a algoritmu Fruchterman-Reingold.



Obrázok 27: Dataset Lazega - základný algoritmus



Obrázok 28: Dataset Lazega - Fruchterman-Reingold



Obrázok 29: Dataset Lazega - Kamada-Kawai

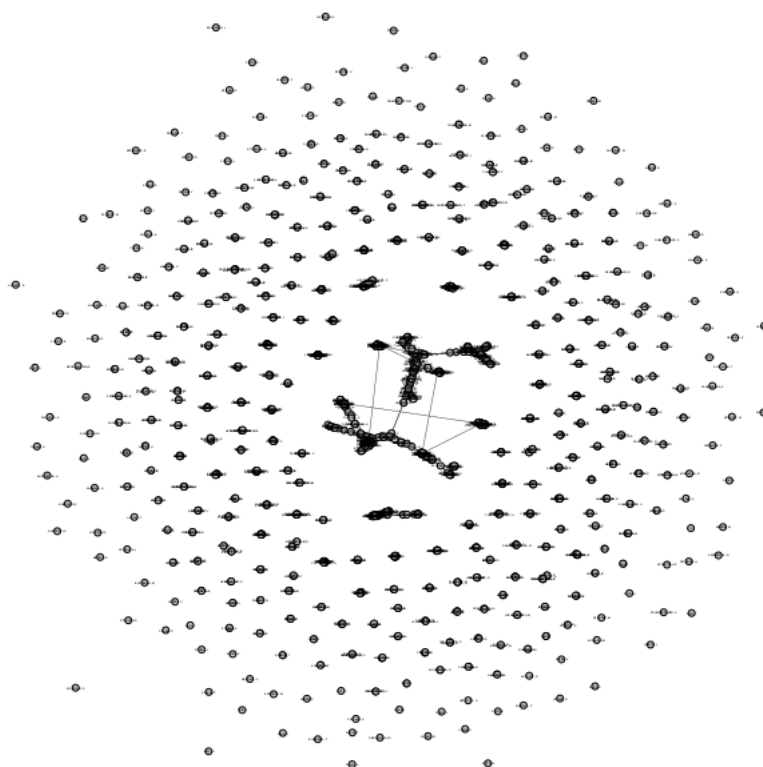
V prípade datasetu Lazega ide o malý graf, pre ktorý dosahujú rýchle výsledky všetky algoritmy.

Tabuľka 4: Čas výpočtu pre dataset Lazega v 2D

Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Lazega	36 / 115	198 ± 3,86	203 ± 11,30	198 ± 8,72

Dataset Network Science

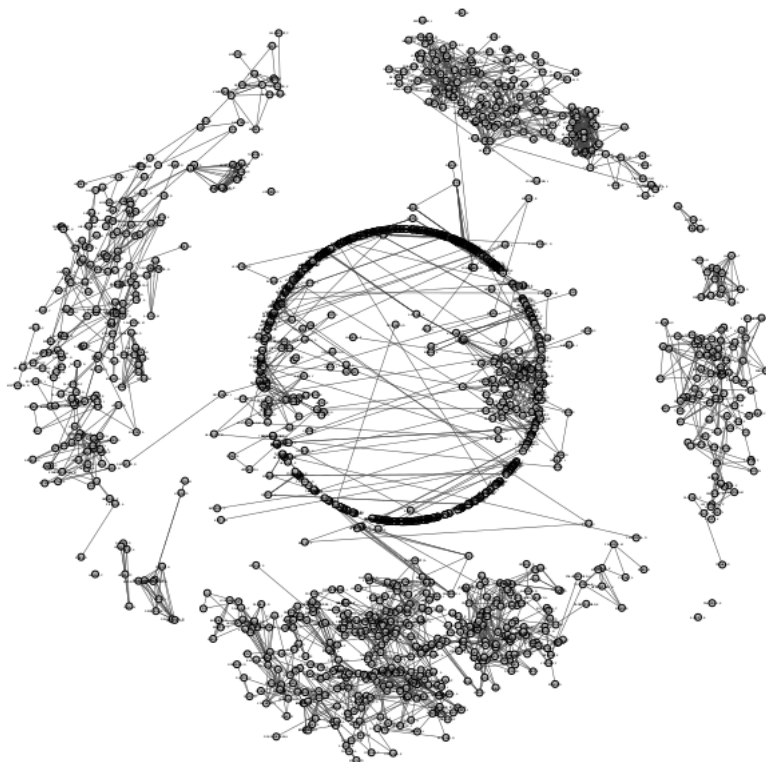
Dataset obsahuje spoluautorskú sieť vedcov pracujúcich na teórii sietí a experimentoch. Sieť obsahuje 1589 vrcholov a 2742 hrán. Výsledná vizualizácia pomocou základného algoritmu umiestnila zhľuky do stredu výsledného zobrazenia, pričom voľné vrcholy zostali v blízko centrálnej oblasti. Algoritmus Fruchterman-Reingold pracuje s voľnými vrcholmi iným spôsobom, pričom ich rozmiestnenie v priestore je viac štruktúrované. Subjektívne najlepšia vizualizácia je výsledkom algoritmu Kamada-Kawai, kde sú zrejmé aj zhľuky voľných vrcholov.



Obrázok 30: Dataset Network Science - základný algoritmus



Obrázok 31: Dataset Network Science - Fruchterman-Reingold



Obrázok 32: Dataset Network Science – Kamada-Kawai

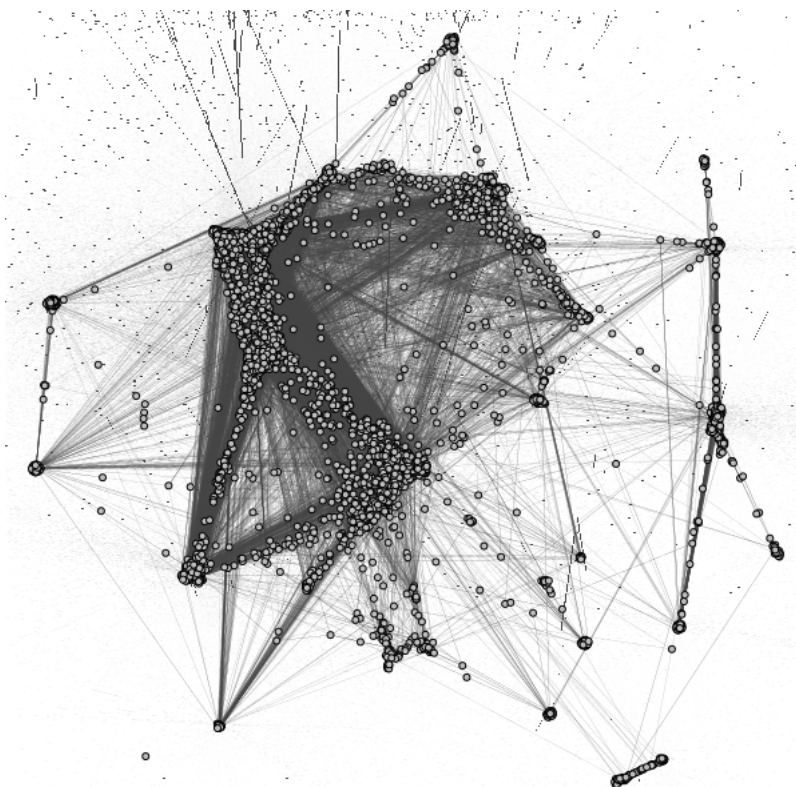
Z pohľadu času výpočtu výslednej vizualizácie je v prípade datasetu s malým počtom prirodzených zhlukov algoritmu Fruchterman-Reingold.

Tabuľka 5: Čas výpočtu pre dataset Network Science v 2D

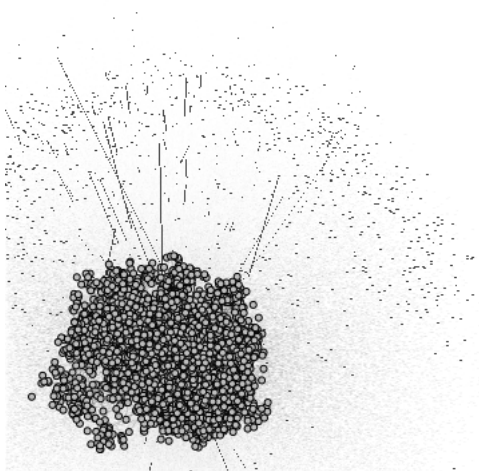
Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Network Science	1589 / 2742	7030 ± 77,10	6010 ± 42,80	7430 ± 104,00

Dataset Pokec

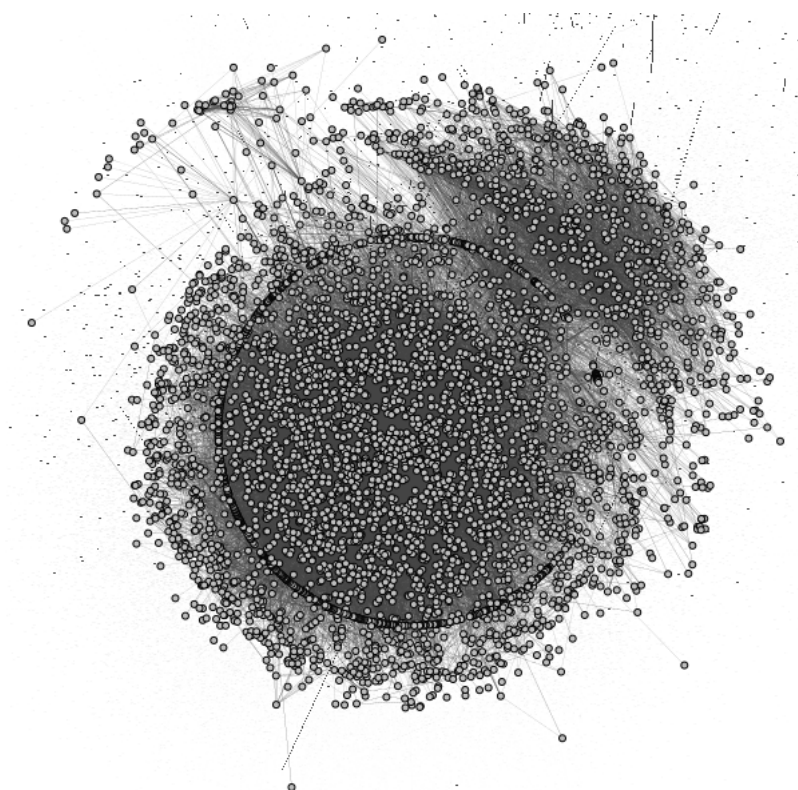
Dataset pokec obsahuje údaje o užívateľoch sociálnej siete Pokec. Ide o extrémne veľkú sieť používanú na analýzu rozsiahlych sietí [49]. Pre porovnanie sme v rámci analýzy výkonnosti a možností vizualizácie vybrali iba podmnožinu 5000 vrcholov, ktorým prislúcha 52 182 hrán. Pre porovnanie s ostatnými algoritmi sme navyše vytvorili dataset s 1000 vrcholmi.



Obrázok 33: Dataset Pokec5000 – základný algoritmus

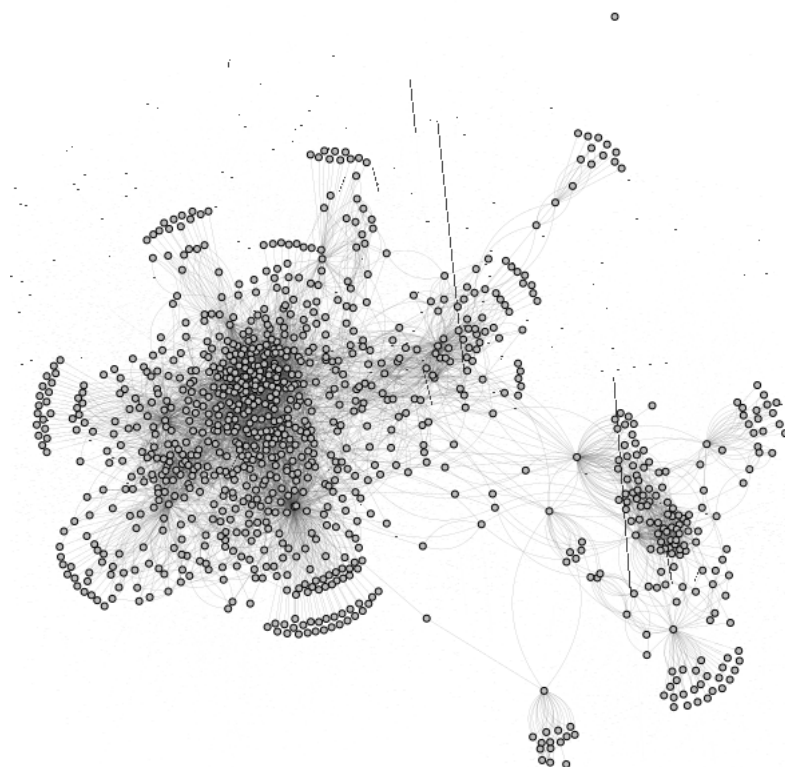


Obrázok 34: Dataset Pokec5000 – Fruchterman-Reingold

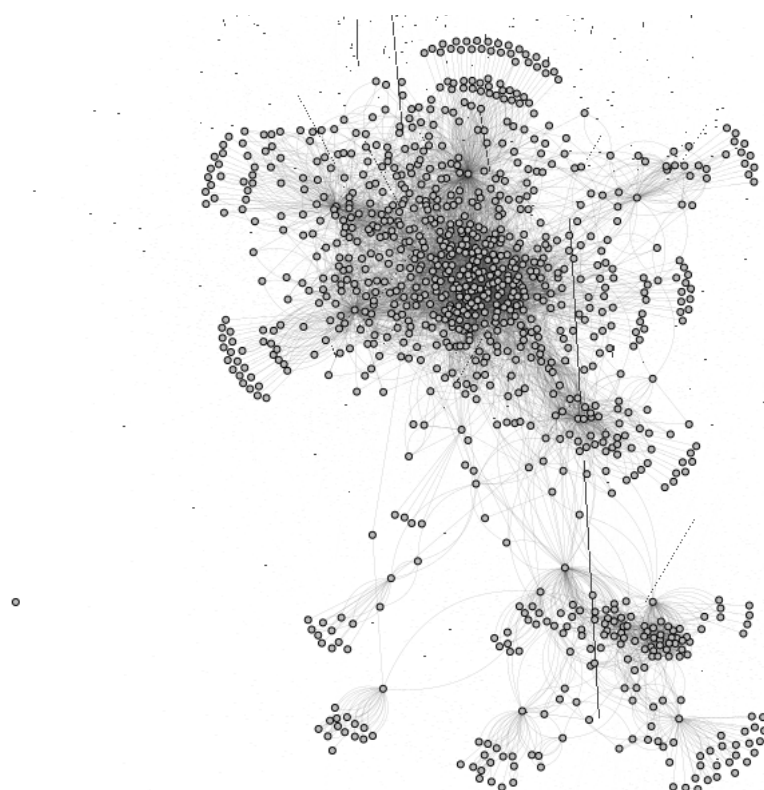


Obrázok 35: Dataset Pokec5000 – Kamada-Kawai

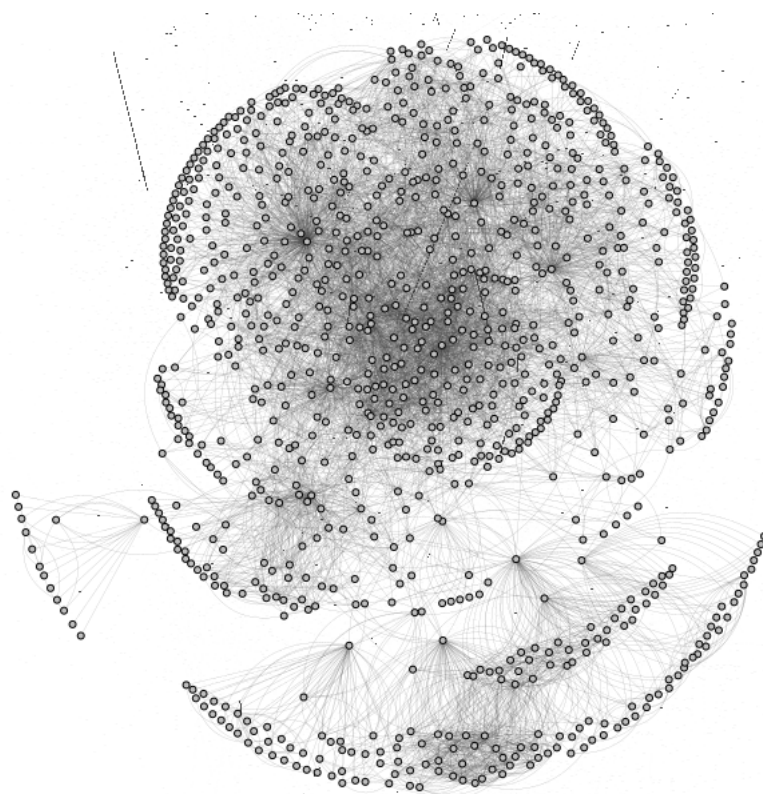
Pre dataset Pokec5000 boli pozorované veľmi chaotické a rozdielne vizualizácie v prípade použitia algoritmu Fruchterman-Reingold, čo znamená, že pri použití tohoto algoritmu nie je možné vychádzať z jednej vytvorenej vizualizácie resp. vizualizovaná informácia nemá dostatočnú vypovedaciu schopnosť. Naproti tomu algoritmus Kamada-Kawai poskytoval veľmi podobné, homogénne vizualizácie avšak za cenu celkové času výpočtu.



Obrázok 36: Dataset Pokec1000 – základný algoritmus



Obrázok 37: Dataset Pokec1000 – Fruchterman-Reingold



Obrázok 38: Dataset Pokec1000 – Kamada-Kawai

V prípade menšieho datasetu Pokec1000 sú vizualizácie homogénne, pričom všetky tri algoritmy poskytujú primeraný výsledok. Subjektívne sa však ako vizuálne najlepšia javí vizualizácia pomocou algoritmu Kamada-Kawai, ktorá definuje prirodzené zhluky (klastre).

Tabuľka 6: Čas výpočtu pre dataset Pokec5000 a Pokec1000 v 2D

Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Pokec5000	5000 / 52182	27800 ± 302,00	21400 ± 396,00	43300 ± 792,00
Pokec1000	1000 / 6297	4980 ± 108,00	4780 ± 130,00	4740 ± 125,00

Tabuľka 7 Časové porovnanie existujúcich algoritmov

Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Game of Thrones	796 / 3909	3510 ± 62,20	3490 ± 48,60	3350 ± 55,30
Witcher	224 / 2600	1120 ± 39,80	1130 ± 21,50	1130 ± 19,00
Lazega	36 / 115	198 ± 03,86	203 ± 11,30	198 ± 08,72
Network Science	1589 / 2742	7030 ± 77,10	6010 ± 42,80	7430 ± 104,00
Pokec5000	5000 / 52182	27800 ± 302,00	21400 ± 396,00	43300 ± 792,00
Pokec1000	1000 / 6297	4980 ± 108,00	4780 ± 130,00	4740 ± 125,00

6.2 Vizualná analýza v trojrozmernom priestore

Vizualizácia dát v trojrozmernom priestore zohráva kľúčovú úlohu v analýze a porozumení komplexných vzťahov v rôznych oblastiach výskumu a aplikácií. Nové výzvy v oblasti vizualizácie dát vyžadujú neustále zdokonaľovanie existujúcich algoritmov a vytváranie nových prístupov, ktoré zabezpečia efektívne a intuitívne vnímanie trojrozmerných dátových štruktúr. V tejto kapitole predstavujeme inovatívny algoritmus založený na princípe silou riadených algoritmov, ktorý umožňuje vizualizáciu v 3D priestore, ale tiež integruje sférické zobrazenia, normalizáciu, zhlukovanie, pružinové spoje a gravitačné interakcie pre vytvorenie robustného nástroja na vizualizáciu komplexných dátových sietí.

Algoritmy sú široko používané v oblasti vizualizácie dát a sú založené na princípe simulovania fyzikálnych síl medzi vrcholmi grafu. Ich efektívnosť v dvojrozmerných priestoroch je dobre zdokumentovaná, ale prenos týchto prístupov do trojrozmerného prostredia si vyžaduje nové techniky a optimalizácie. V našom prístupe sme zdokonalili tradičné silou riadené algoritmy o sférické zobrazenia, čo umožňuje lepšiu reprezentáciu vzťahov medzi vrcholmi v trojrozmernom priestore.

Okrem toho sme implementovali normalizačné techniky, ktoré umožňujú efektívnu manipuláciu s rozsahom hodnôt a zlepšujú konzistenciu vizuálnej prezentácie. Využitie zhlukovania nám umožňuje identifikovať a vizualizovať podskupiny dát s podobnými charakteristikami, zatiaľ čo pružinové spoje a gravitačné interakcie pridávajú dodatočnú dynamiku do vizualizácie, čím zabezpečujú lepšiu interpretáciu štruktúry dátového grafu.

V nasledujúcich častiach kapitoly sa podrobnejšie zameriame na opis navrhnutých algoritmov, ich implementáciu a výsledky, ktoré sú dosiahnuté pri vizualizácii rôznorodých dátových sietí v trojrozmernom priestore. Tento prístup poskytuje nový pohľad na vizualizáciu dát, zohľadňujúc nielen geometrické aspekty, ale aj hierarchické a štruktúrne vlastnosti dát.

Silou riadené algoritmy v 3D priestore prinášajú výhodu presnejšieho zobrazenia vzdialeností medzi vrcholmi. Táto možnosť umožňuje vytváranie vizualizácií, ktoré sú čitateľnejšie a štruktúrovanejšie, čo je významné pre detailné prieskumy dátových vzorov. S výhodami v 3D priestore prichádzajú aj nevýhody. Algoritmy v 3D priestore sú zložitejšie z výpočtového hľadiska, čo môže spôsobovať väčšiu výpočtovú záťaž. Zvýšená komplexita vizualizácie môže zase viesť k strate detailov a zvýšenej náročnosti interakcií s prostredím.

Výhody v 3D Priestore:

- **Presná vzdialenosť:** V 3D priestore majú silou riadené algoritmy väčší priestor na presné zobrazenie vzdialeností medzi vrcholmi.
- **Lepšia čitateľnosť:** Vďaka možnosti rozmiestňovať vrcholy v trojrozmernom priestore môže byť vizualizácia čitateľnejšia a štruktúrovaná.

Nevýhody v 3D Priestore:

- **Zložitejší výpočet:** Algoritmy pre 3D priestor sú často náročnejšie z výpočtového hľadiska.
- **Zvýšená komplexita vizualizácie:** Zväčšením priestoru sa môže zvýšiť aj komplexita vizualizácie.

Väčší Počet Interakcií: Užívatelia môžu potrebovať väčší počet interakcií na presné pohľady a manipuláciu s trojrozmerným priestorom.

6.3 Metodika prípravy dát pre vizualizácie v 3D

Klasické silou riadené algoritmy sú obmedzené na výpočet rozloženia grafu v euklidovskej geometrii, zvyčajne R^2 , R^3 , niektoré novšie implementácie v R^n pre väčšie hodnoty n . Existujú však prípady, keď euklidovská geometria nemusí byť tou najlepšou voľbou, čo je aj výsledkom zistení dizertačnej práce. Určité grafy majú štruktúru, ktorú je lepšie vizualizovať s použitím inej geometrie, napríklad na povrchu gule alebo na anuloidu. Najmä zobrazenia 3D sietí môžu byť zobrazené na guli za účelom mapovania textúr alebo uloženia na torus bez kríženia. Okrem toho sa zistilo, že určité neeuklidovské geometrie, konkrétne hyperbolická geometria, majú vlastnosti, ktoré sú obzvlášť vhodné na usporiadanie a vizualizáciu veľkých tried grafov [71] [72]. Na základe uvedených zistení Kobourov a Wampler opisujú rozšírenia silovo riadených algoritmov na Riemannove priestory [73].

Predspracovanie dát a siete ako celku je podstatnou časťou vizualizácie, pričom predspracovanie môže byť vykonávané s rôznym zameraním. Pri väčšine prípadov je potrebné dáta predspracovať z dôvodu vyčistenia chybných hodnôt, normalizácie alebo iných operácií, ktoré následne uľahčia prácu z dátami ako opisujú autori [8]. Druhým možným dôvodom vykonania predspracovania dát môže byť snaha o odľahčenie algoritmov slúžiac na výpočet informácií z dát, prehľadávanie alebo odľahčenie hardvérových prostriedkov pri ďalšom spracovaní.

6.3.1 Predspracovanie siete

V dnešnej dobe existuje mnoho algoritmov, ktoré majú za úlohu nastaviť zrozumiteľný a prehľadný tvar siete za pomoci algoritmov na výpočet základných charakteristických metrík siete. Výsledné vypočítané hodnoty sú následne použité ako oporný bod pre algoritmy rozloženia siete v priestore. V 2D priestore je tento prístup pomerne bežný a jednoducho aplikovateľný na rôzne druhy dát, v prípade 3D nám ale samotné vypočítanie metrík nestačí. Metriky určujúce vlastnosti siete v 3D priestore môžu byť zavádzajúce, preto je potrebné správne vybrať metriky, s ktorými algoritmy dokážu pracovať efektívne aj v 3D. V prvej verzii našej práce používame silou riadené algoritmy, pre ktoré sú základné metriky len spôsob ako urýchliť výpočet konečnej polohy siete.

6.3.2 Príprava dát

Vizualizácia dát v trojrozmernom priestore zohráva kľúčovú úlohu v analýze a porozumení komplexných vzťahov v rôznych oblastiach výskumu a aplikácií. Nové výzvy v oblasti vizualizácie dát vyžadujú neustále zdokonaľovanie existujúcich algoritmov a vytváranie nových prístupov, ktoré zabezpečia efektívne a intuitívne vnímanie trojrozmerných dátových štruktúr. V tejto kapitole predstavujeme inovatívny algoritmus založený na princípe silou riadených algoritmov, ktorý umožňuje vizualizáciu v 3D priestore, ale tiež integruje sférické zobrazenia, normalizáciu, zhlukovanie, pružinové spoje a gravitačné interakcie pre vytvorenie robustného nástroja na vizualizáciu komplexných dátových sietí.

Prvotným cieľom je zabezpečiť, aby hodnoty hrán vo vizualizovanom grafe boli vhodne normalizované, čím sa maximalizuje interpretovateľnosť a efektívnosť výslednej 3D vizualizácie. Normalizácia dát je predovšetkým potrebná pri parametroch použitých na reprezentáciu síl a pružín, pretože bez normalizácie môžu nastať priestorové komplikácie s vrcholmi, na ktoré je aplikovaná príliš veľká sila.

6.3.3 Škálovanie hodnôt hrán

Škálovanie hodnôt hrán je proces úpravy hodnôt, ktoré reprezentujú vzťahy medzi vrcholmi v grafe. Cieľom tohto procesu je dosiahnuť rovnomerný rozsah hodnôt, ktorý je vhodný pre vizualizáciu a analýzu dátového grafu. Škálovanie hodnôt hrán môže byť dôležité, pretože umožňuje lepšie porovnávanie a interpretáciu vzťahov medzi vrcholmi bez toho, aby sa hodnoty nespravodlivo vynášali v dôsledku ich veľkých variácií. Škálovanie hodnôt hrán je nevyhnutné pre dosiahnutie konzistentných a interpretabilných výsledkov pri vizualizácii v 3D.

6.3.4 Normalizácia parametrov pre hodnoty hrán

Normalizácia parametrov pre hodnoty hrán je proces, ktorý má za cieľ dosiahnuť jednotný a štandardizovaný rozsah hodnôt parametrov, ktoré reprezentujú vzťahy medzi vrcholmi v grafe. Tento postup sa často používa pri príprave dát pre vizualizáciu grafu, či už v 2D alebo 3D priestore. Normalizácia je dôležitá z viacerých dôvodov:

- **Zjednodušenie porovnávania:** Normalizované hodnoty umožňujú jednoduchšie porovnávanie vzťahov medzi vrcholmi, pretože sú škálované na štandardný interval.
- **Minimalizácia vplyvu extrémnych hodnôt:** Normalizácia pomáha eliminovať vplyv extrémnych hodnôt, ktoré by inak mohli nespravodlivo ovplyvniť vizuálne zobrazenie grafu.
- **Zabezpečenie konzistentnosti:** Normalizácia prispieva k dosiahnutiu konzistentného rozsahu hodnôt pre rôzne hrany grafu, čo uľahčuje ich interpretáciu.

6.4 Algoritmus 1: Pružinový algoritmus (Spring Layout Algorithm)

Algoritmus reprezentuje graf ako fyzický systém, kde uzly sú považované za nabité častice vzájomne spojené pružinami. Každý uzol je priradený dvom typom síl: príťažlivé sily a odpudivé sily. Cieľom algoritmu je znížiť celkovú energiu systému pružín presúvaním uzlov do nových polôh. Atrakčná sila (f_a) pôsobí na susedné uzly spojené pružinou, zatiaľ čo odpudivá sila f_r pôsobí na všetky uzly grafu. Sily sú definované vzorcami nižšie:

$$f_a(d) = k_a \cdot \log(d), f_r(d) = k_r \cdot \frac{1}{d^2}$$

(15)

kde k a k_r sú konštanty a d je aktuálna vzdialenosť medzi dvoma uzlami.

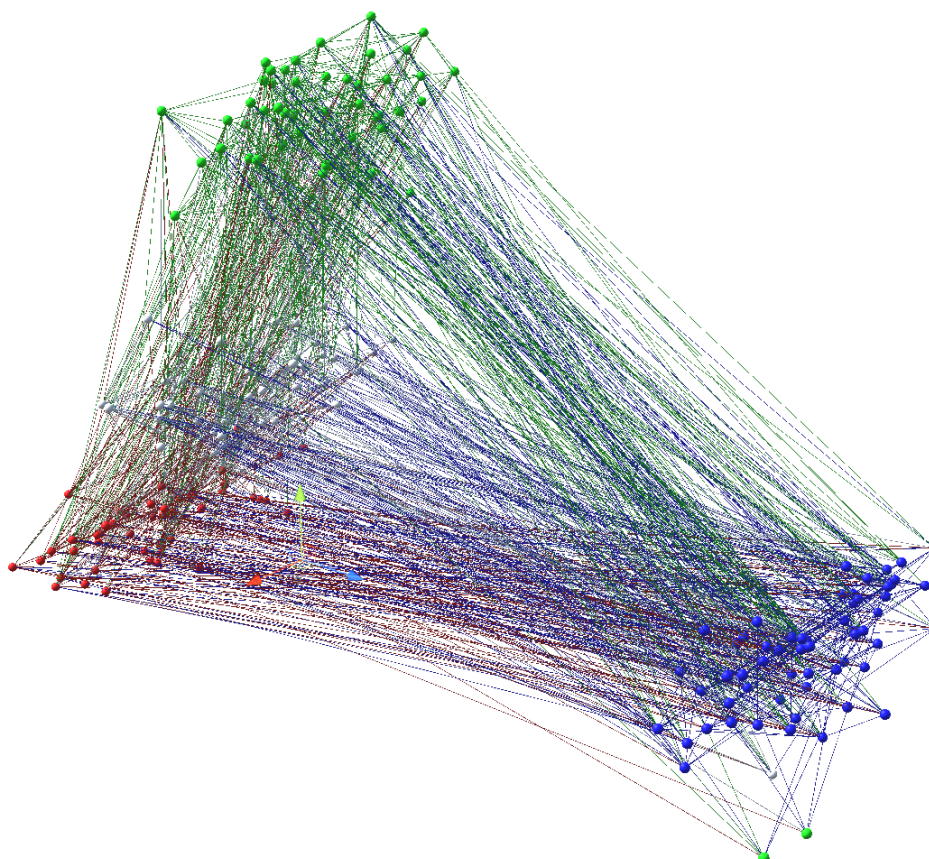
Napriek tomu, že táto metóda produkuje čisté, symetrické usporiadanie pre grafy strednej veľkosti, považuje sa za náročný algoritmus. Jeho časová zložitosť je v mnohých prípadoch vysoká, čím sa stáva algoritmus nepoužiteľný. Okrem toho trpí nedostatočnou predvídateľnosťou, čo znamená, že opakované spustenie algoritmu môže viesť k rôznym výsledkom. To môže predstavovať problémy pri udržiavaní mentálnej mapy používateľa počas interakcie so nestabilnými usporiadaniami. Napriek týmto nevýhodám sa silou riadené algoritmy široko používajú v mnohých vizualizačných

rámcoch. V našej práci bol algoritmus prepracovaný a optimalizovaný s cieľom odstrániť jeho charakteristické nevýhody hlavne pre využitie v 3D priestore.

V praxi môže byť silou riadený graf využitý na zobrazenie sociálnych sietí, vzťahov v biologických systémoch, alebo štruktúry webu. Výhodou je, že tento prístup k vizualizácii grafu je schopný odhaliť vzory a skupiny vrcholov, ktoré by mohli byť ťažko viditeľné v iných formách vizualizácie. Takéto grafy s vynucovaním síl majú svoje využitie aj v analýze dát a prieskume komplexných systémov, pretože ich dynamický prístup k rozmiestňovaniu vrcholov môže pomôcť odhaliť štruktúry a vzťahy, ktoré by inak mohli zostať skryté.

6.4.1 Modifikácia 1

V prvotných experimentoch boli otestované pružinové spojenia medzi uzlami, ktoré reprezentujú príťažlivé sily. Zároveň zabezpečujú aj prislúchajúce odpudivé sily, pretože jedna z vlastností pružín je sťahovanie svojej dĺžky. Prvotný prototyp môžeme vidieť na Obrázok 39, kde je zobrazená vizualizácia náhodne vygenerovaných dát [74].

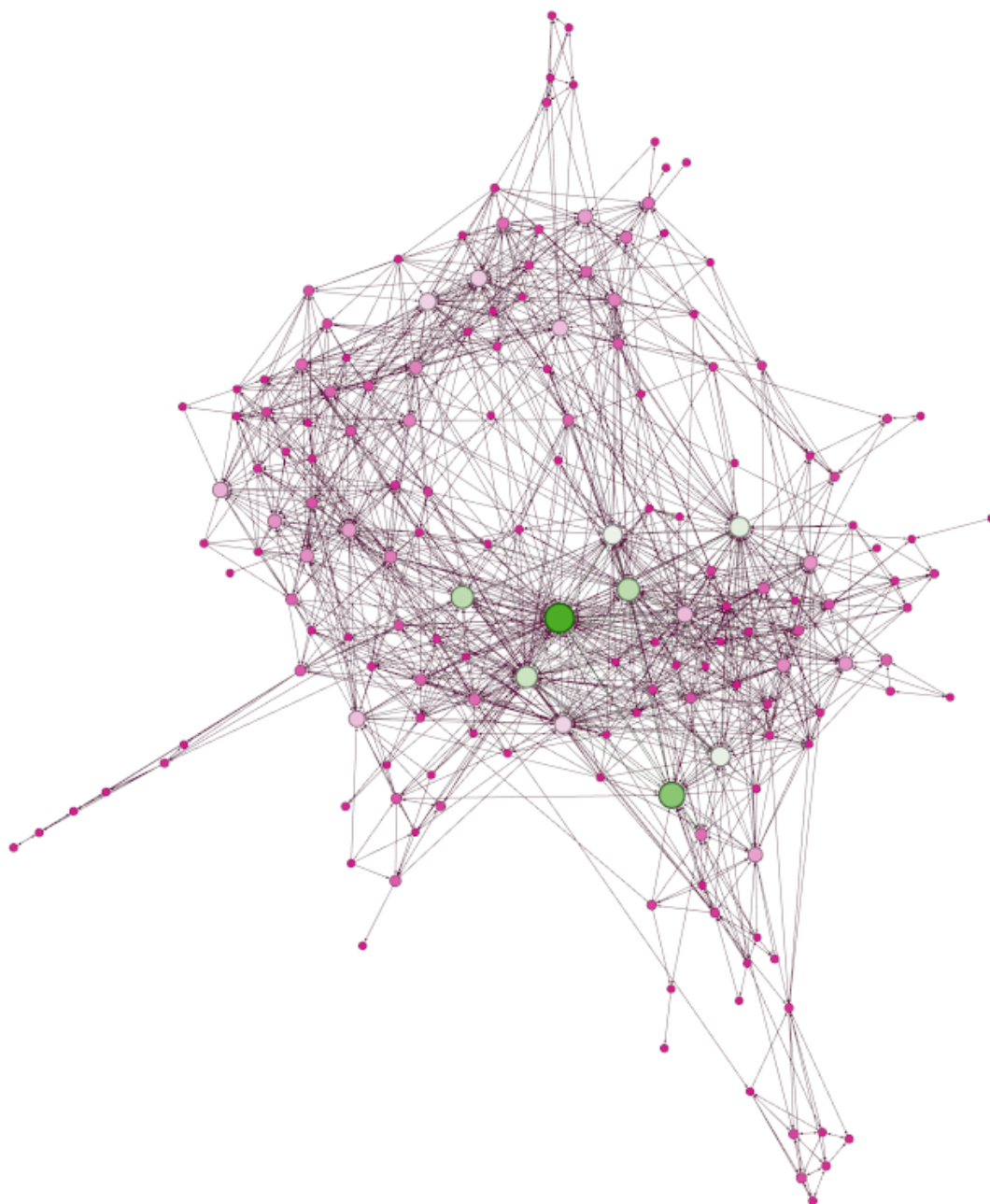


Obrázok 39: Zobrazenie záťažového experimentu

Uzly a hrany sú generované s náhodnými vlastnosťami, ako je farba a poloha, aby sa odstránila možná odchýlka spôsobená akýmkoľvek špecifickým súborom údajov. Náhodné pozície používame na odstránenie počiatočného skreslenia spôsobeného nahromadenou silou, ktorá sa aplikuje, keď sú uzly umiestnené na začiatku scény. Okrem toho na nasmerovanie hrán v grafe je uzlom priradená farba, ktorá predstavuje ich typ. Na základe typu počiatočného a koncového uzla sa vytvoria hrany a nastaví sa sila pružinového spoja. Pružinové spojenia umožňujú rýchle a dostatočne prehľadné zobrazenie grafu, pričom s použitím správnych nastavení kontrolujú rozpínanie uzlov v priestore. Ich nevýhoda spočíva v obmedzenosti vyvíjaných síl v jednom smere a zvýšenej záťaži pre výpočtovú jednotku. Každý pružinový kĺb musí byť v scéne reprezentovaný jedným herným objektom, čo zapríčiňuje zvýšenú záťaž nie len na fyzikálny ale aj vykresľovanie vizuálnych elementov (ang. rendering).

6.5 Algoritmus 2: Silou riadený algoritmus (Force Driven Algorithm)

Rozloženie grafu v priestore je náročná činnosť a vyžaduje určité skúsenosti a zručnosti, ktoré je potrebné aplikovať na použité dáta. Mnohé používané algoritmy usporiadania využívajú ako základný parameter, kombináciu metrík opísaných v kapitole 5.1, pričom je potrebné predspracovanie dát a prípadne manuálne usporiadanie zobrazeného grafu. Silou riadené algoritmy využívajú fyzikálne vlastnosti a procesy na výpočet zrozumiteľného rozmiestnenia uzlov a hrán v priestore, pomocou pôsobiacich síl na jednotlivé elementy grafu.



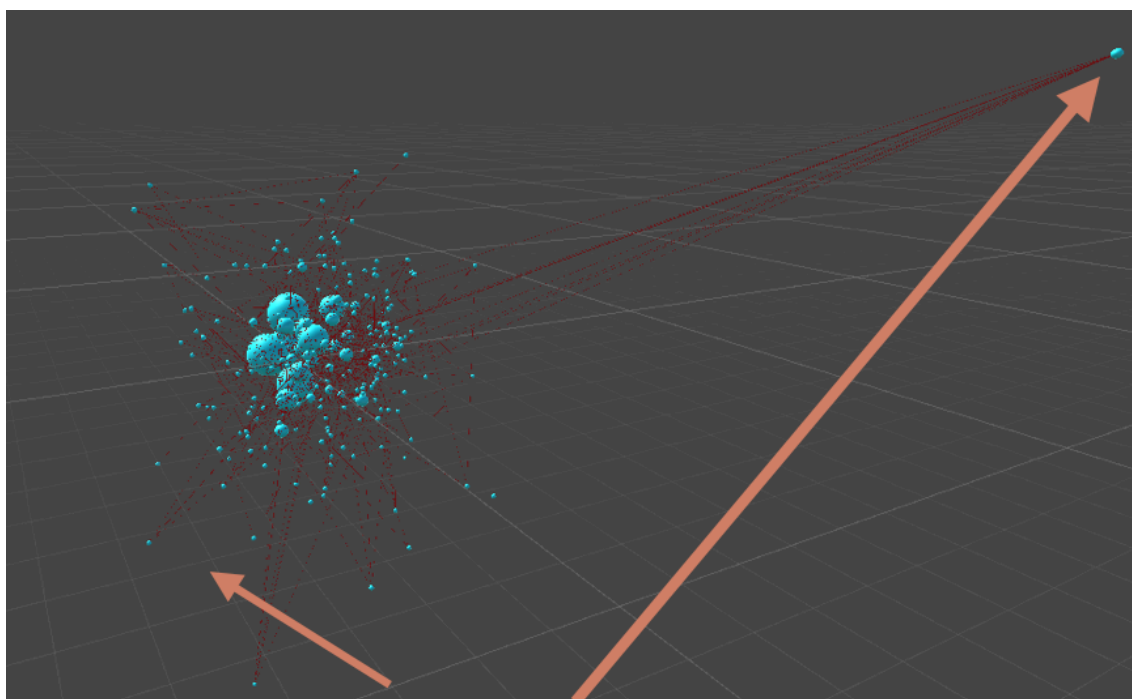
Obrázok 40: Vizualizácia dát v nástroji Gephi

Algoritmy rozloženia orientované na silu vytvárajú grafické zobrazenie použitím systému emulovaných fyzických síl. Gravitácia priradzuje vrcholom hmotnosť v pomere k ich sieťovej centrálnosti, čo umožňuje, aby vrcholy, ktoré sú z hľadiska grafu viac centrálné, boli vizualizované vo fyzicky centrálnych miestach. Zmena mierky mení gravitačnú silu počas simulácie a znižuje prekríženie v porovnaní s gravitáciou bez mierky.

Silou riadené algoritmy poskytujú funkcionality, ktorá zabezpečuje rozloženie grafu v priestore za pomoci fyzikálnych vlastností jednotlivých uzlov a hrán. Výsledné rozloženie

zobrazuje graf v ustálenom stave, kedy sú sily vyrovnané, pričom zaručuje v 2D priestore neprekrývanie uzlov. Jedným z nástrojov používaných na vizualizáciu dát s použitím silou riadených algoritmov je nástroj Gephi, ako môžeme vidieť na Obrázok 40.

Hlavnou vlastnosťou silou riadených algoritmov je fyzikálna príťažlivosť a odpudivosť medzi jednotlivými uzlami reprezentované hranami grafu. Fyzikálne sily umožňujú vytvárať čitateľné a prehľadné rozmiestnenia hrán a uzlov v 3D priestore, bez nutnosti zásahu používateľa do procesu zostavovania grafu. Zvolená metóda aplikovania sily na jednotlivé hrany grafu, nám umožňuje kontrolovať rozpínanie grafu v 3D priestore, pre lepšie zachovanie prehľadnosti celej vizualizácie.



Obrázok 41: Vrcholy odstránené silou za okraj vizualizácie

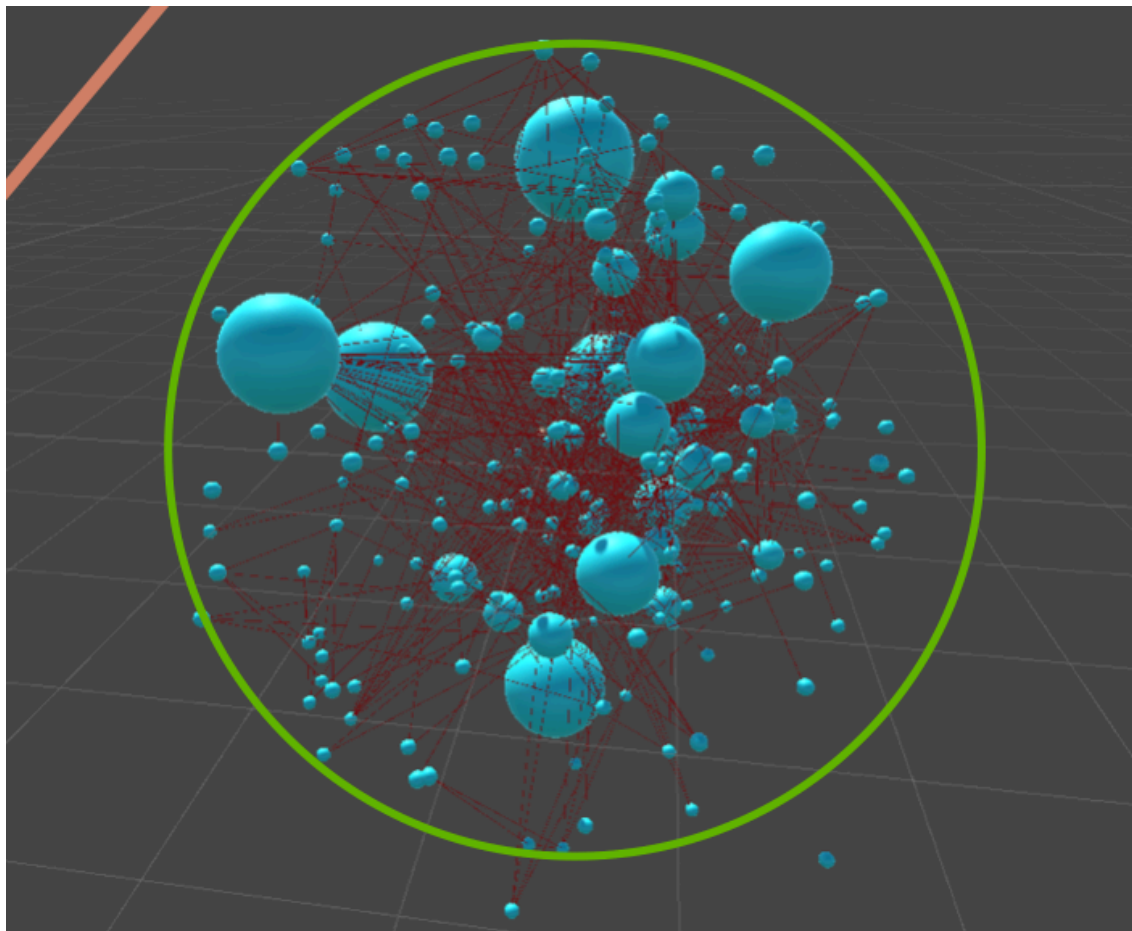
Fyzikálne knižnice Unity3D poskytujú viacero možností reprezentácie sily:

- Funkcie sily
- Pružinové spojenia-kĺby
- Iné knižničné funkcie

Knižnica obsahujúca funkcie na priame aplikovanie síl na dané herné objekty, má veľký potenciál v oblasti silou riadených algoritmov. Na každý uzol v grafe, ktorý je spojený hranou s iným uzlom, je pomocou zadaných funkcií aplikovaná sila v smere zodpovedajúcej smeru hrany.

Keďže na uzly pôsobia rôzne sily, môže nastať situácia, kedy sa uzly zobrazia mimo kamery ako môžeme vidieť na Obrázok 41. V takom prípade je aplikované ukotvenie resp. kotva, ktorá pôsobí na každý uzol v sieti a priťahuje ho smerom k základnému

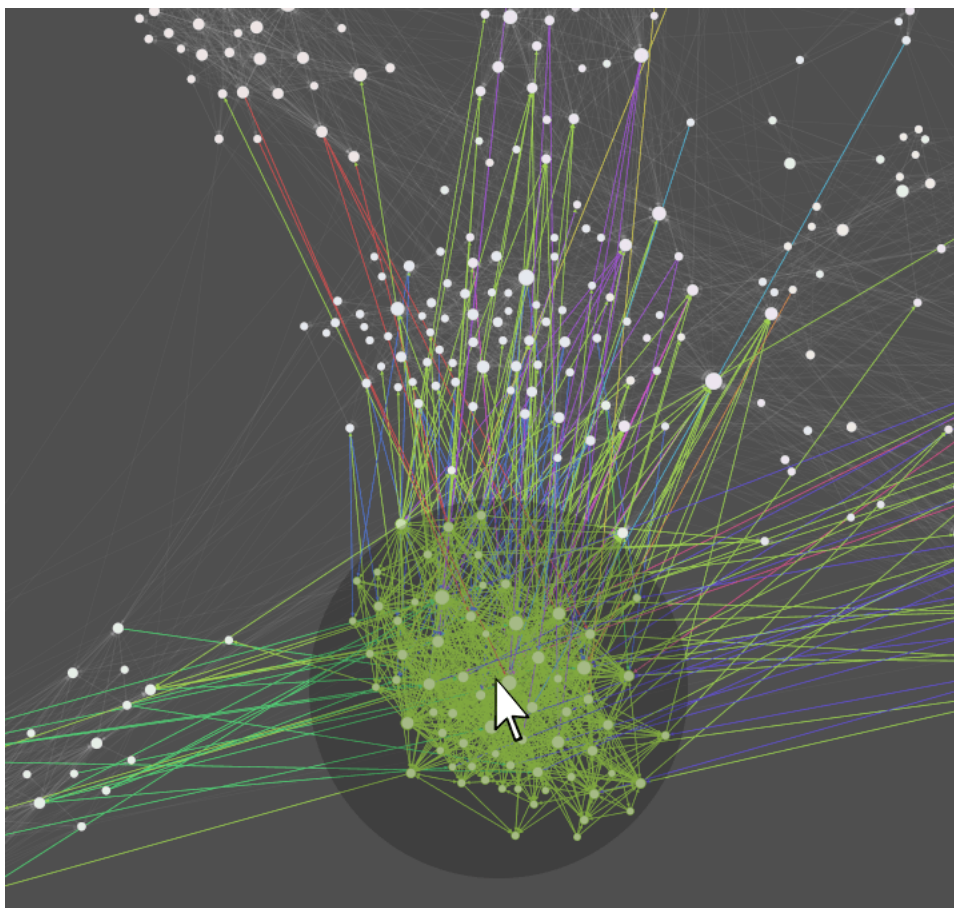
bodú. Aplikovaním kotvy dosiahneme stav ako na Obrázok 42, kde je celá sieť vizualizovaná v zornom poli virtuálnej kamery.



Obrázok 42 Ideálny stav síl

Aplikácia silou riadených algoritmov v 3D priestore je stále predmetom výskumu, pretože v trojrozmernom priestore tieto algoritmy nedokážu sami o sebe zabezpečiť zrozumiteľné zobrazenie z dôvodu pridanej z dimenzie, ktorej odobraním v 2D priestore vytvárame obmedzenie pre pohyb uzlov a hrán. Obmedzenie v podobe pridanej dimenzie je potrebné pretransformovať aj do trojrozmerného priestoru, ale v inej podobe, ktorá zabezpečí, že vizualizácia nebude zakrývať informácie a zároveň bude jasne rozoznateľná pre používateľa [7].

Silovo orientovaný prístup spája vstupný graf s fyzikálnym systémom síl a snaží sa umiestniť vrcholy tak, aby celková energia fyzikálneho systému bola minimálna. Len niekoľko príkladov silovo riadených algoritmov používa hrany s ohybmi, prevažná väčšina silovo riadených algoritmov predstavuje hrany ako priame segmenty, ako môžeme vidieť na Obrázok 43 z nástroja Gephi, a preto sú vypočítané výkresy optimálne z hľadiska sklonu geodetických hrán ako opisujú autori [38].



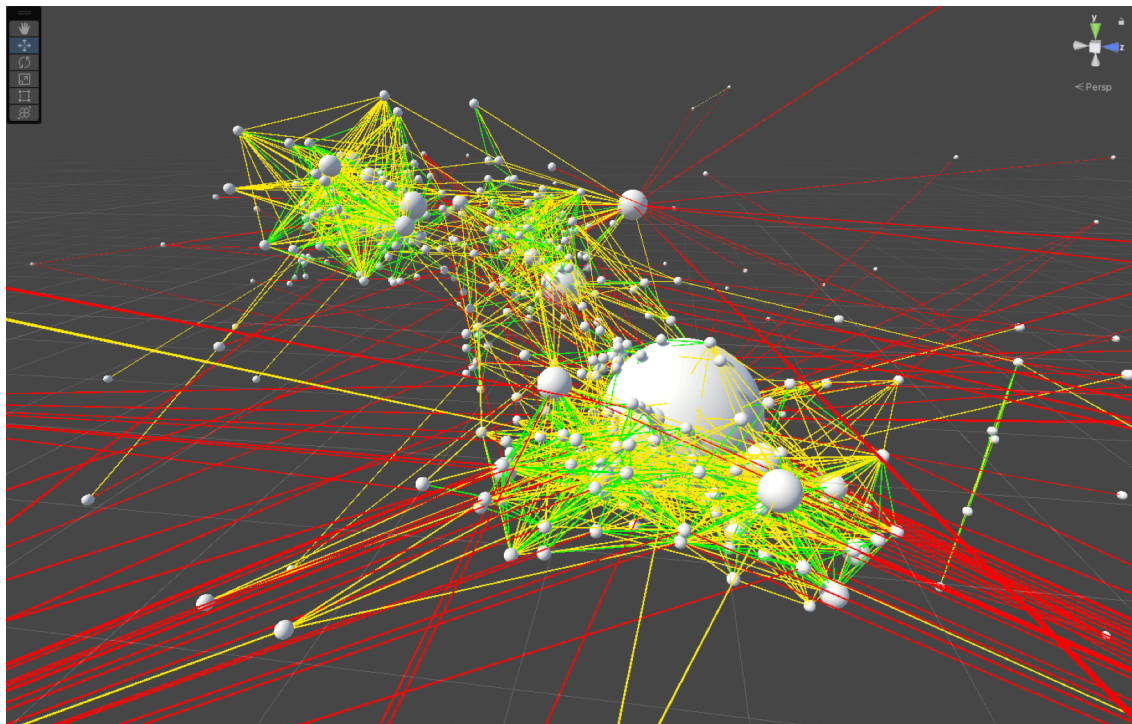
Obrázok 43 Experimentálna vizualizácia dát v Gephi3D

6.6 Algoritmus 3: Pružinovo-silový algoritmus

Silou riadený algoritmus a pružinový algoritmus v 3D priestore vykazujú určité nedostatky a obmedzenia. Jednoduchý pružinový algoritmus je extrémne citlivý a nedochádza k jeho konvergencii do stavu, kedy je možné akceptovať finálnu vizualizáciu. Vyznačuje sa nielen neustálym kmitaním, ale aj nevyváženým stavom celého 3D modelu, ktorý sa v čase často mení. Naopak, silou riadený algoritmus trpí nadmerným vytváraním zhukov a problémom s odpudzovaním vrcholov. Kombináciou týchto dvoch definovaných algoritmov sme boli schopní eliminovať oba pozorované problémy. Vytvorili sme kombinovaný algoritmus pre štandardný euklidovský priestor a jeho modifikáciu so sférickou hranicou, ktorá slúži na zabezpečenie vyššej kompaktnosti a čitateľnosti výslednej vizualizácie.

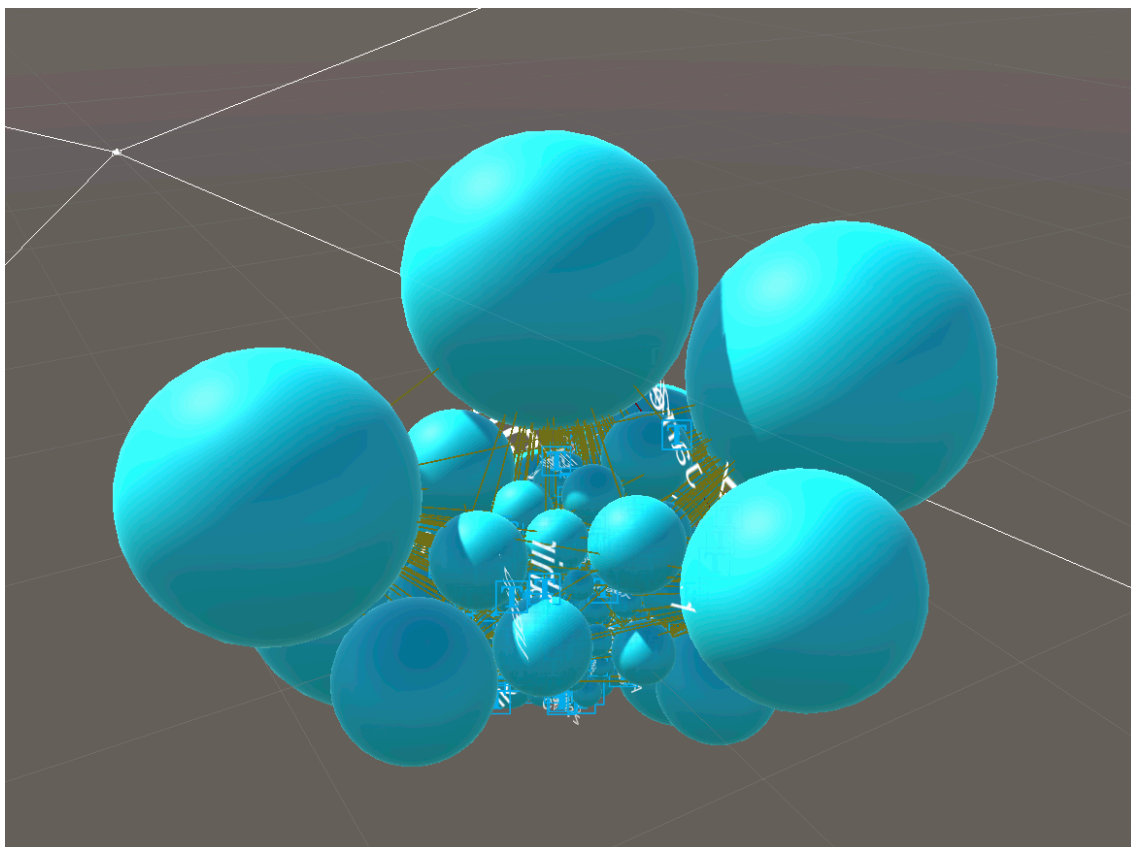
Použitie pružín (pružinových spojov) vytvára systém, v ktorom sú vrcholy priťahované k sebe pôsobením atraktívnych síl. Tento prístup vedie k rýchlejšej stabilizácii siete, pretože pružiny zabezpečujú stabilnú prevádzku silového systému. V tomto prístupe sú hrany reprezentované atraktívnymi silami, ktoré k sebe ťahajú spojené vrcholy. Tento

model je výhodný vďaka rýchlemu spájaniu vrcholov a dobrej internej implementácii vo fyzikálnych prostrediach.



Obrázok 44 Výsledná vizualizácia dát v aplikácii

V prípade kombinácie pružinového a silou riadeného algoritmu dosahujeme vyvážený prístup, kde pružiny a atraktívne sily spolupracujú na stabilizácii grafu. Pri nastavení parametrov síl a pružín je však dôležité nájsť vhodnú rovnováhu medzi rýchlym usadením siete a minimalizáciou zahltenia modelu. Kombinácia týchto postupov tiež zabráni vytvoreniu jedného veľkého zhluku, ktorý by mohol zhoršiť kvalitu výslednej vizualizácie. Spojením týchto dvoch metód docielime harmonickú rovnováhu medzi pružinovými silami a príťažlivými silami, čo vedie k stabilizovanej štruktúre grafu, ako je znázornené na Obrázok 44. Je dôležité vhodne nastaviť silové parametre a pružiny tak, aby dosiahli optimálnu rovnováhu medzi rýchlym ustálením siete a minimalizáciou neprehľadnosti modelu. Kombináciu postupov taktiež zamedzíme vytvoreniu jedného veľkého zhluku, ktorý by mohol narušiť vizualizáciu ako na Obrázok 45.



Obrázok 45 Vytvorený zhuk spôsobený aplikovanými silami

6.7 Algoritmus 4: Pružinovo-silový algoritmus s kontrolovaným rozmiestňovaním

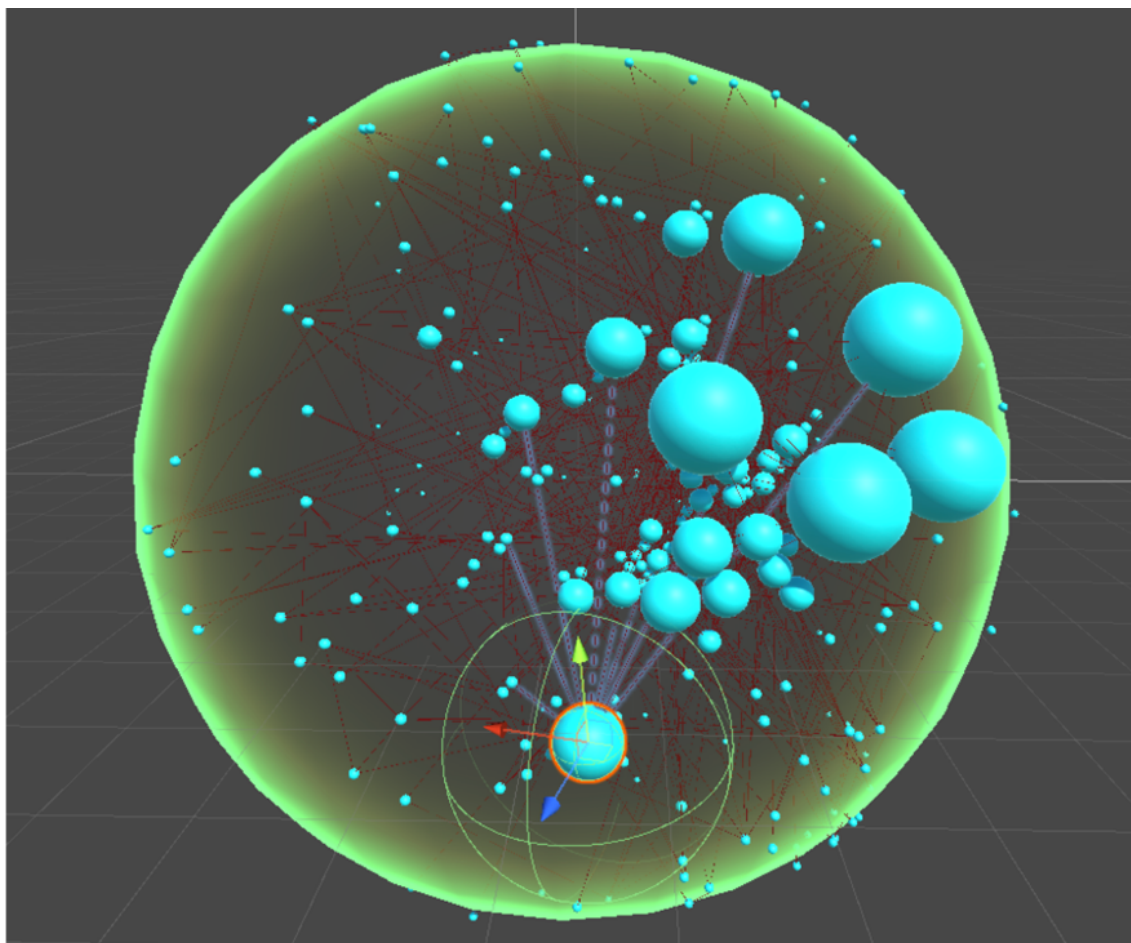
Voľné zobrazenie v priestore je štandard pre 3D vizualizáciu, ale v prípade silou riadených algoritmov nastávajú situácie, v ktorých je potrebné nastaviť obmedzenia priestoru, sily a rozostavenia hrán a uzlov. Pri neregulovaných hodnotách síl môžeme pozorovať správanie grafu v dvoch hlavných prípadoch, ktoré sú neoptimálne pre vizualizáciu ako napríklad:

- Zobrazený graf vytvára jeden nečitateľný zhuk
- Vzdialenosť medzi uzlami sa neustále zväčšuje, čo znamená, že nikdy nedosiahne stabilného stavu

V prípade jedného vytvoreného zhuku dochádza k javu, kedy je suma príťažlivých síl medzi uzlami násobne väčšia ako suma odpudivých síl v grafe. Zhuk pozostávajúci zo všetkých alebo z väčšiny uzlov grafu, sa stáva nečitateľným, z dôvodu zakrytia hrán a uzlov ostatnými elementami grafu. V prípade ak sa vzdialenosť uzlov neustále zväčšuje resp. suma odpudivých síl je väčšia ako suma príťažlivých síl, graf nedokáže v reálnom čase dosiahnuť stabilný stav, a z toho dôvodu je vizualizácia nepoužiteľná.

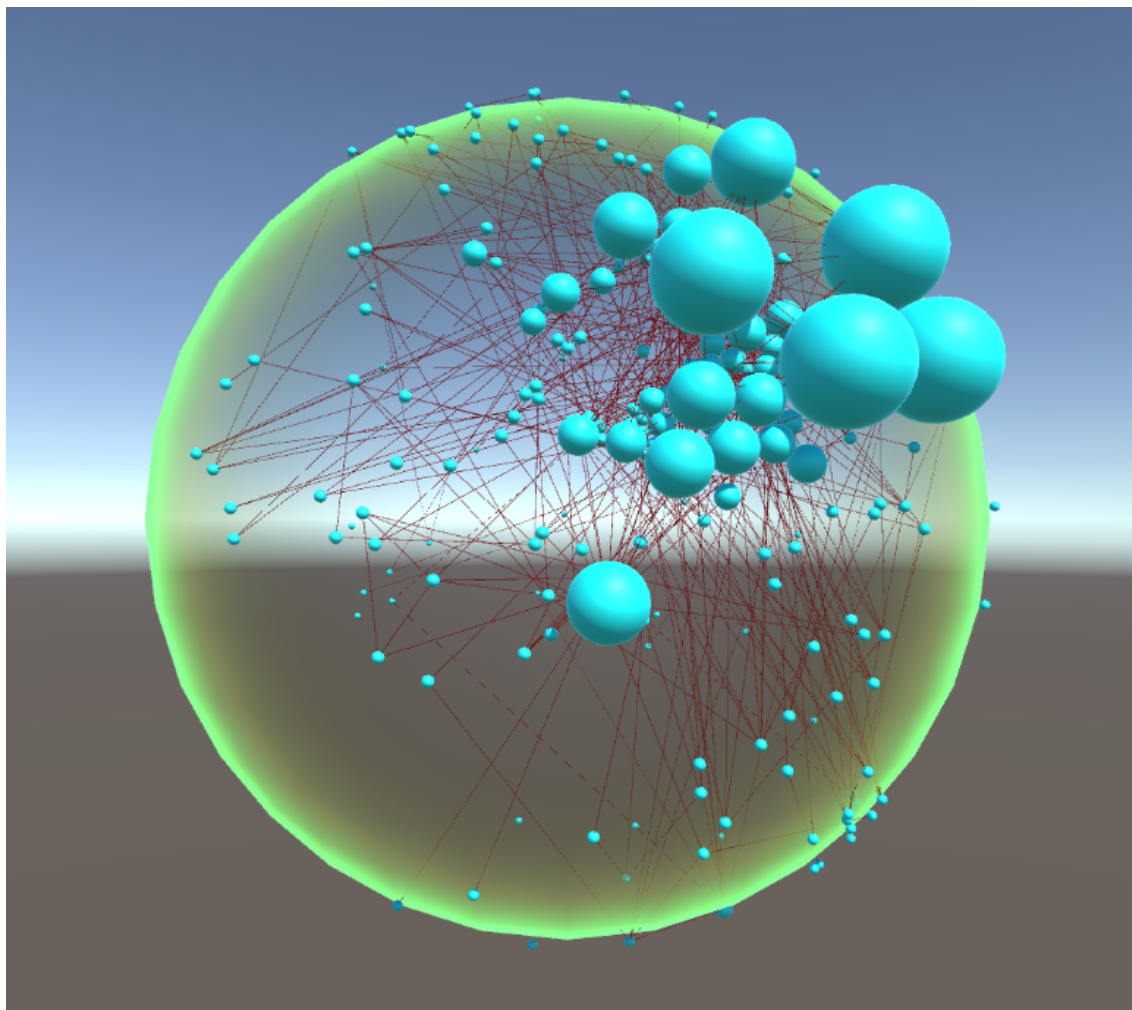
Pre určenie správnej veľkosti obmedzení v 3D priestore, bola vykonaná skupina meraní zameraná na detekciu obmedzení a možností, ktoré poskytuje samotné herné prostredie. Z publikácie [74] vyplývajú doporučené počty použitých herných objektov v jednej scéne, za účelom vytvorenia grafu s udrzaním časovej a výpočtovej zložitosti pod hranicou, kedy sa vizualizácia stáva nepoužiteľnou. Voľné zobrazenie siete v priestore má mnoho nežiadúcich faktorov, ktoré je potrebné detegovať a zabezpečiť. Ako môžeme vidieť na Obrázok 45, vzniknutý zhuk zabraňuje prehľadnosti siete, pretože uzly na ktoré pôsobí menšia príťažlivá sila alebo uzly na ktoré pôsobí priveľká odpudivá sila prekážajú kamere.

Pôsobiace príťažlivé a odpudivé sily v sieti bez zadefinovaných obmedzení, môžu vytvoriť nežiadúce zhuky, ktoré sú nečitateľné pre používateľa. Aby sme predišli situácií, v ktorej zhuk uzlov zakrýva zvyšok vizualizácie, implementovali sme obmedzenie vzdialenosti uzlov medzi sebou. Obmedzenie je reprezentované sférickým polom, ktoré slúži ako stena pre ostatné uzly. Veľkosť sférického poľa je priamo odvodená od veľkosti samotného uzla t.z. čím väčší uzol tým väčšie sférické pole. Tento prístup umožňuje vytvorenie prehľadnejšej vizualizácie ako môžeme vidieť na Obrázok 46.



Obrázok 46 Sféricke obmedzenie

Voľné zobrazenie v priestore prináša nové možnosti ako obohatiť 3D vizualizáciu ale prináša problémy vo forme ustálenia siete v zornom poli kamery. Tento problém je čiastočne vyriešený pre uzly, ktoré sú vytlačené mimo zorného poľa, ale nerieši problém vytvorenia jedného veľkého zhľuku. Jeden zhľuk vzniká v prípade ak sú príťažlivé sily kotvy väčšie akú suma ostatných pôsobiacich síl v sieti. Aby sme zamedzili tomuto nežiadúcemu stavu implementovali sme sférické obmedzenie, ktoré udržiava uzly neustále v rovnakej vzdialenosti od kotvy, pričom im zároveň zabráňuje sa vzdialiť opačným smerom ako môžeme vidieť na Obrázok 47.

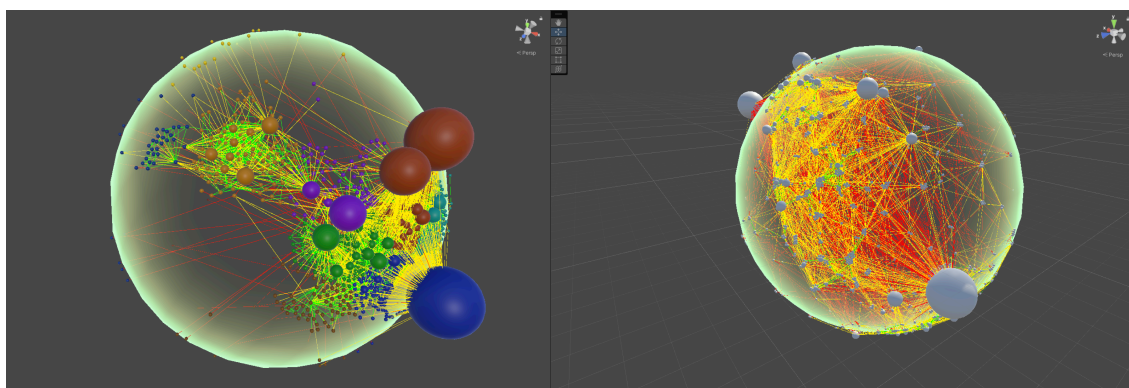


Obrázok 47 Vizualizácia dát so sférickým obmedzením

Na rozdiel od uzlov, hrany majú povolený prechod cez sférickú zábranu, čo vytvára zaujímavý efekt, kde je možné sledovať prepojenia v sieti medzi vybranými uzlami, bez interferencie okolia. Sférická zábrana je z toho dôvodu z väčšiny plochy priehľadná, ale v každom uhle pohľadu si zachováva svoj obrys pre neustálu reprezentáciu zobrazenia na povrchu sféry. Pri vizualizácii grafu v 3D priestore sa často stretávame s potrebou efektívneho zobrazenia vzdialeností medzi vrcholmi. Pre lepšie porozumenie vzdialeností a štruktúry grafu je možné využiť farebné kódovanie hrán na základe

vzdialenosti medzi vrcholmi. V tejto vizualizácii sú hrany farebne ovplyvnené podľa vzdialenosti medzi ich koncovými vrcholmi. Konkrétne, hrany sú zafarbené červeno, ak sú vrcholy ďaleko od seba, zeleno, ak sú vrcholy blízko a žltó, ak sa nachádzajú v priemernej vzdialenosti. Toto farebné kódovanie poskytuje vizuálny indikátor vzájomných vzdialeností vrcholov v 3D priestore ako môžeme vidieť na Obrázok 48.

Porovnaním na obrázku Obrázok 48 môžeme usúdiť, že výhoda tohto prístupu spočíva v tom, že umožňuje užívateľovi rýchlo identifikovať, aké časti grafu sú vzdialené a ktoré sú blízke. Taktiež umožňuje jednoduchú identifikáciu oblastí grafu, ktoré by mohli vyžadovať ďalšiu pozornosť alebo analýzu. Takéto farebné kódovanie vytvára ďalší vrstevnicový aspekt vo vizualizácii, ktorý zvyšuje informačnú hodnotu pre užívateľa.



Obrázok 48 Rôzna veľkosť sférického obmedzenia

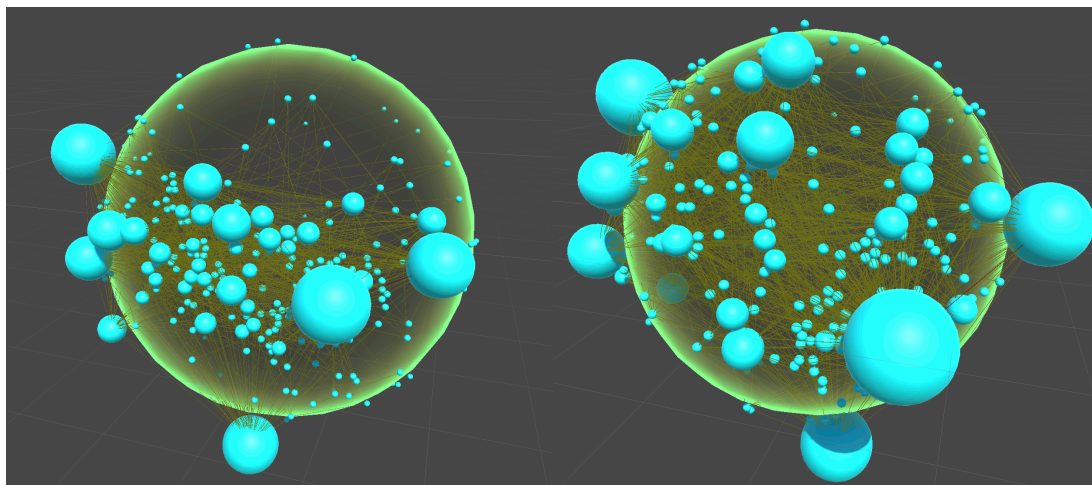
6.7.1 Ukotvenie

Uzly a hrany voľne zobrazení v priestore je potrebné zoskupiť okolo miesta v scéne, na ktoré má používateľ nastavenú kameru. V takom prípade je aplikované ukotvenie resp. kotva, ktorá pôsobí na každý uzol v sieti a priťahuje ho smerom k základnému bodu počas celého priebehu vizualizácie. Aplikovaním kotvy dosiahneme stav, ktorý sme definovali ako žiadaný Obrázok 42, kde je celá sieť vizualizovaná v zornom poli kamery, pričom aplikáciu ukotvenia môžeme pozorovať na Obrázok 47.

6.7.2 Modifikácia 1. Zmena veľkosti sférického obmedzenia

Zvolenie veľkosti sférického obmedzenia závisí od vizualizovaných dát. Je pravdepodobné, že prvotne zvolená veľkosť nebude vyhovovať veľkosti a objemu siete, preto je v aplikácii implementovaná funkcionálna zmena veľkosti sférického obmedzenia počas vykonávania silou riadeného usporiadacieho algoritmu. Zmena veľkosti sféry umožňuje zlepšenie prehľadnosti siete, pretože pri každej zmene, či už zmenšenie sféry alebo jej zväčšenie, sa zároveň mení aj finálne rozloženie uzlov a hrán v priestore. Na obrázkoch Obrázok 49 a Obrázok 50 môžeme pozorovať sieť

zostavenú z rovnakých dát pri rôznej veľkosti sféry. Pre zlepšenie vizualizácie pridáme sféry, ktoré budú slúžiť ako ukotvenie pre vrcholy grafu. Vyberieme vhodnú veľkosť sféry vzhľadom na štruktúru a rozsah hodnôt v dátach, aby sme dosiahli rovnomerný a esteticky príjemný vzhľad. Zároveň musíme prihliadať na stupeň vrcholov, pretože môže priamo ovplyvniť výslednú vizualizáciu takým spôsobom, že vrcholy s veľkým stupňom môžu neprirodzene zakrývať ostatné vrcholy aj napriek odpudivým silám a menším algoritmom na zabezpečenie prehľadnosti ako na Obrázok 45.



Obrázok 49 Malé sférické obmedzenie

Obrázok 50 Veľké sférické obmedzenie

6.7.3 Modifikácia 2 Ohraničenie priestoru v okolí vrcholov

S narastajúcim počtom vrcholov a hrán narastá aj zložitosť vizualizácie. Aj keď 3D priestor poskytuje dostatočný priestor na vizualizáciu veľkého objemu dát, stále sa musíme vysporiadať s problémom prekryvania vrcholov v jednotlivých zhľukoch. Jedným zo spôsobov je implementácia dynamických záporných hrán, ktoré sú aktívne len v prípade, ak sa dva vrcholy priblížia na určenú vzdialenosť. Táto vzdialenosť je definovaná podľa požiadaviek vizualizácie. Dynamické záporné hrany nám pomáhajú obmedziť prekryvanie vrcholov v priestore a zároveň priamo neovplyvňujú ostatné prítlačivé sily v grafe natoľko, aby zmenili vizualizovanú informáciu. Aplikácia dynamických odpudivých síl je ukázaná v kóde nižšie.

```

1. public void Update()
2. {
3.     var nodesInRange = GetNodesInRange(range);
4.     foreach(var node in nodesInRange)
5.     {
6.         var direction = Vectro3.GetDirection(this,node.position);
7.         node.GetRigidbody.ApplyForce(direction);
8.     }
9. }

```

Druhým spôsobom obmedzenia prekryvania vrcholov je implementácia „silového poľa“, ktoré zabezpečuje konštantné vzdialenosti medzi vrcholmi. Silové pole je priamo závislé

na stupni vrcholu, pričom s narastajúcim stupňom sa zväčšuje aj veľkosť poľa. Vedľajším efektom vytvorenia silového poľa môžeme pozorovať, že vrcholy s najvyšším stupňom sú oddialené od povrchu sféry, čo zvyšuje prehľadnosť modelu. Kombinácia silového poľa a ukotvenia na sfére zároveň zabraňuje vrcholom zostať v zákryte, čím umožňuje pozorovať ľubovoľný vrchol aspoň z jedného uhla pohľadu ako môžeme vidieť na obrázku .

6.7.4 Modifikácia zobrazenia hrán

Priamo prepojené hrany, ktoré prechádzajú cez sféru, ponúkajú jednoduchú a priamu cestu medzi vrcholmi, čo uľahčuje interpretáciu a pochopenie vzťahov v grafe. Ich kratšia dĺžka môže prispieť k zníženiu celkovej komplexity vizualizácie. Avšak tento prístup môže narušiť kontinuitu priestoru a viesť k dojmu prestávky v vizuálnej reprezentácii. Okrem toho, priame hrany môžu mať tendenciu prekryvať sa s inými prvkami na sfére, čo môže ovplyvniť čitateľnosť grafu.

Na druhej strane, zaoblené hrany, ktoré idú len po povrchu sféry, udržiavajú kontinuitu priestoru a vytvárajú plynulý vizuálny zážitok. Ich umiestnenie na povrchu sféry môže zlepšiť prehľadnosť vizualizácie tým, že menej zasahujú do významných prvkov na sfére. Avšak ich komplexnejšie modelovanie a manipulácia môžu zvýšiť náročnosť implementácie vizualizačného systému. Okrem toho, tieto hrany môžu mať tendenciu zmenšovať efektívnu vzdialenosť medzi vrcholmi, čo môže ovplyvniť vnímanie vzájomných vzťahov v grafe.

V prípade, keď hrany grafu spájajú vrcholy priamo a prechádzajú cez sféru, môžeme očakávať nasledujúce výhody a nevýhody:

Výhody:

1. **Jednoduchosť:** Priama cesta medzi vrcholmi môže byť jednoduchšia na pochopenie a interpretáciu, pretože nevyžaduje žiadne dodatočné zohľadnenie zakrivenia sféry.
2. **Krátkosť hrán:** Hrany môžu byť kratšie, čo znižuje celkovú zložitosť vizualizácie a uľahčuje sledovanie spojení medzi vrcholmi.

Nevýhody:

3. **Prestávka v kontinuite:** Priame hrany, ktoré prechádzajú cez sféru, môžu vytvoriť dojem prestávky v kontinuite priestoru, čo môže spôsobiť neprirodzený vzhľad vizualizácie.

4. Možné prekrytie: Priame hrany môžu mať tendenciu prekryvať sa s ostatnými prvkami na sfére, ako sú vrcholy, čo môže spôsobiť zhoršenie čitateľnosti vizualizácie.

V prípade zaoblených hrán, ktoré idú len po povrchu sféry, môžeme očakávať nasledujúce výhody a nevýhody:

Výhody:

- **Kontinuita:** Zaoblené hrany na povrchu sféry môžu zachovať kontinuitu priestoru, čo vytvára plynulý vizuálny zážitok.
- **Väčšia prehľadnosť:** Hrany na povrchu sféry môžu menej prekážať významným prvkom na sfére, čo zlepšuje čitateľnosť vizualizácie.

Nevýhody:

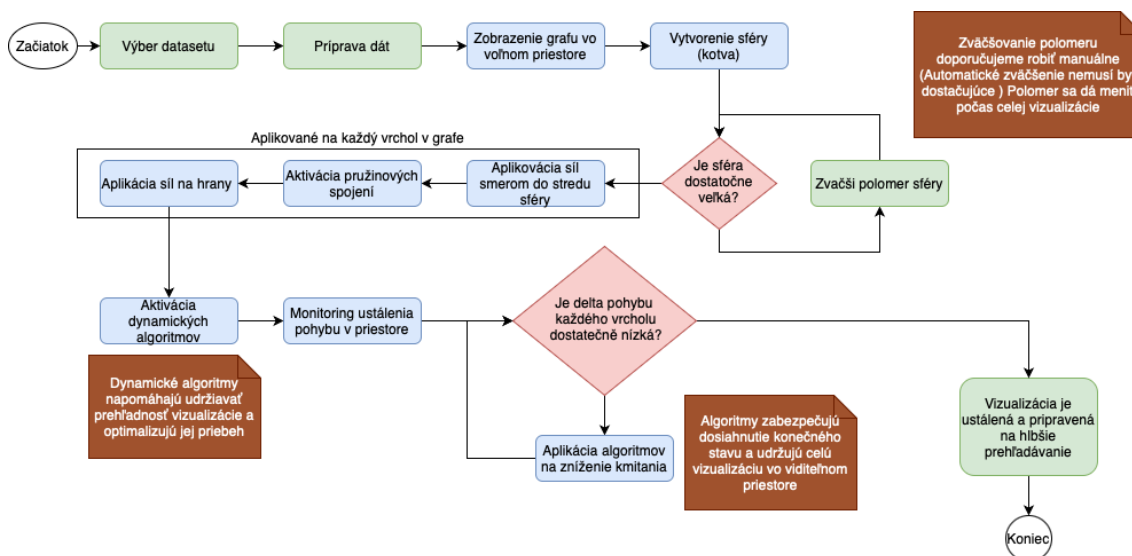
- **Zložitejšie modelovanie:** Zaoblené hrany môžu vyžadovať komplexnejšie modelovanie a manipuláciu, čo môže viesť k zvýšenej náročnosti implementácie vizualizačného systému.
- **Zmenšenie vzdialenosti:** Hrany, ktoré sa prekrywajú s povrchom sféry, môžu mať tendenciu znižovať efektívnu vzdialenosť medzi vrcholmi, čo môže ovplyvniť vnímanie vzájomných vzťahov medzi nimi.

Vo voľbe medzi týmito dvoma prístupmi je dôležité zvážiť konkrétne požiadavky na vizualizáciu a aplikáciu grafu na sfére, aby sme dosiahli optimálnu reprezentáciu, ktorá zohľadňuje potreby užívateľa a zabezpečuje čitateľnosť a pochopenie dát.

6.7.5 Postup vizualizácie

Postup celej vizualizácie je zaznamenaný na diagrame Obrázok 51. Výber datasetu a príprava dát podlieha vytvorenej metodike v prílohe B. Modré časti diagramu znázorňujú vytvorené algoritmy spolu s ich modifikáciami, ktorými je možné dosiahnuť požadovanú úroveň vizualizácie dát, pričom je potrebné dbať na korektné použitie modifikácií pre vybrané dáta. V diagrame je znázornená zastavovacia podmienka, ktorá je definovaná špeciálne pre vytvorené algoritmy. Pri štandardných 2D algoritmoch je vopred daný presný počet iterácií, ktoré sa majú vykonať, a to aj v prípade ak sa vizualizácia dostane do konečného stavu oveľa skôr. Počet iterácií môže byť aj nedostačujúci, a v takom prípade sa vizualizácia zastaví pred dosiahnutím konečného stavu. Vytvorená zastavovacia podmienka zastaví vizualizáciu v prípade ak je priemer všetkých rýchlostí vrcholov v grafe menší ako 1 za vybranú jednotku času, ktorá je vo väčšine prípadov 0.2 sekundy. Pre väčšie grafy sa priemer vzťahuje len na 90%

vrcholov, čo zabezpečuje, že malý počet oscilujúcich vrcholov, ktoré väčšinou disponujú malým počtom hrán, nezabránili zastaveniu vizualizácie.



Obrázok 51 Diagram vizualizácie

6.8 Modifikácie 3D zobrazenia – interaktívne funkcie

Interakcia so dátami v podobe siete je v 3D priestore veľmi dôležitá, pretože umožňuje hlbšie preskúmať a pochopiť dáta. Z toho dôvodu sú implementované dve možnosti ako prehľadávať dátový priestor:

- Monitor a myš
- Monitor a zariadenie Leap motion

Klasické ovládanie pomocou myši umožňuje používateľovi rotovať vizualizovanú sieť, vyberať uzly na ktoré sa práve potrebuje zamerať a zároveň môže posúvať zorné pole kamery. Tento prístup je pre používateľa intuitívny pretože je často používaný v iných nástrojoch na vizualizáciu ako napríklad Gephi opísané v kapitole 6.5. Ovládanie prostredia pomocou myši môže byť v mnohých prípadoch neintuitívne a zavádzajúce, pretože kurzor ako ovládací prvok nedokáže simulovať tretiu dimenziu, čím výrazne redukuje možnosti vizualizácie.

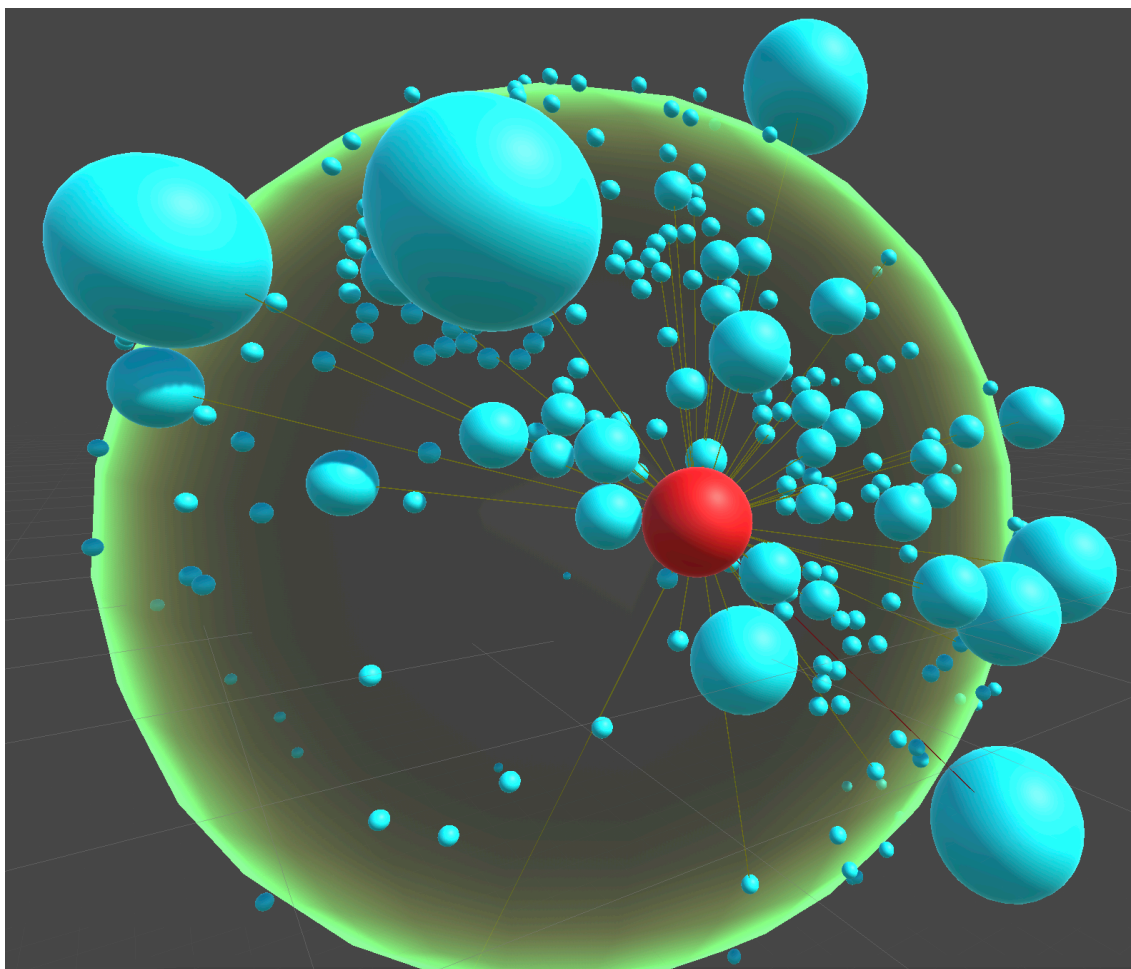
Pre zabezpečenie presnej a jednoduchšej interakcie s vyžitím tretej dimenzie je potrebné použiť zariadenie ako napríklad Leap motion, ktoré spĺňa požiadavky na 3D interakciu v dostatočnej miere pre našu prácu. Zariadenie funguje na princípe snímania rúk pomocou kamier, a následne rozoznáva zadané gestá používateľa, ktoré je možné priradiť jednotlivým úkonom v hernom prostredí. Intuitívne ovládanie pomocou rúk zabezpečuje plynulú a jednoducho ovládateľnú vizualizáciu, pričom umožňuje

vizualizačnej aplikácii používať obmedzenia, ktoré majú za úlohu sprevádzať používateľa prehľadávaným priestorom.

6.8.1 Techniky zamerania a kontextu (Focus + Context)

Techniky zamerania a kontextu sa venujú problému strát kontextu pri približovaní sa k určitým údajom. Predpokladajme, že sme priblížili obrázok. Výsledkom bude, že budeme môcť vidieť len oblasť, ku ktorej sme sa priblížili, a stratíme predstavu o okolitých oblastiach obrázka. Práve tu prichádza do hry technika zamerania a kontextu: dáva používateľom schopnosť vidieť hlavný objekt v detailnom pohľade (zameranie) spolu s prehľadom všetkých okolitých informácií (kontext). Všeobecne platí, že strata kontextu je považovaná za problém v aplikáciách vizualizácie informácií. S cieľom odstrániť tento problém, objavili sa techniky zamerania a kontextu, ktoré používateľovi umožňujú sústrediť sa na niektoré detaily bez straty globálneho kontextu. Táto koncepcia nezastupuje metódy priblíženia a posúvania, ale skôr ich dopĺňa. Väčšina vizualizačných systémov implementuje obe techniky spolu ako nástroj interakcie.

Existuje mnoho prístupov k poskytovaniu zamerania a kontextu. Overview + detail je jedným z najstarších prístupov, pri ktorom sa používajú samostatné oblasti displeja pre rôzne rozlíšenia. Umožňuje používateľom často prepínať medzi rôznymi zobrazeniami. Fish-eye je jednou z najpopulárnejších techník zamerania a kontextu: oblasť záujmu sa stáva väčšou, zatiaľ čo iné časti rozlohy sa postupne zobrazujú s menším detailom. Pri prístupe fish-eye je výpočet hyperbolických súradníc rýchlejší než algoritmus rozlohy, čo sa považuje za problém pri interakcii s vizualizáciou. Skreslenie sa javí ako negatívna dôsledok tejto techniky, ktoré vedie k narušeniu mnohých estetických kritérií riadiacich algoritmus rozlohy, napríklad nežiaduce prieniky hrán. Prekrývanie hrán a uzlov vo vizualizácií je problém, s ktorým sa rôzni autori vysporiadali mnohými spôsobmi ako je opísané v kapitole 6.6. V našej aplikácii sme implementovali techniku, ktorá umožňuje upriamiť pohľad na jeden vybraný uzol, pričom zachová hrany, ktoré sú s ním spojené ako môžeme vidieť na Obrázok 52.



Obrázok 52 Upríamenie pohľadu na jeden uzol

Vybraný uzol je znázornený červenou farbou, pričom všetky ostatné uzly si zachovávajú svoje vlastnosti. Hrany, ktoré priamo nesúvisí s vybraným uzlom sú dočasne skryté, aby nezakrývali zobrazenú informáciu. Pomocou tejto techniky je možné preskúmať jednotlivé uzly postupne a odkrývať spojenia, ktoré by boli inak stratené medzi zvyškom vizualizovaných dát. Ďalšou výhodou tohto prístupu je odľahčenie hardvérových prostriedkov, pretože herné prostredie nemusí vykresľovať nesúvisiace informácie. Na druhú stranu, tento spôsob neumožňuje zobrazenie uzla spolu s hranami vo väčšom kontexte grafu, čo môže mať za následok stratu informácie spojenú s celkovým prepojením v sieti.

6.8.2 Priblíženie a posunutie (Zooming and Panning)

Priblíženie a posunutie sú základné nástroje na preskúvanie veľkého množstva informácií. Posunutie znamená pohyb kamery po scéne, zatiaľ čo priblíženie umožňuje používateľom prepínať medzi abstraktným a detailným pohľadom. Geometrické priblíženie upravuje obrazovú transformáciu a tým umožňuje zväčšovanie alebo zmenšovanie priblíženého grafu. Sémantické priblíženie znamená, že nielen veľkosť

objektov, ale aj zobrazené informácie sa môžu zmeniť pri približovaní sa k určitej oblasti grafu.

Priblíženie v kontexte vizualizácie grafov spočíva v schopnosti zväčšiť alebo zmenšiť pohľad na graf. Geometrické priblíženie nám umožňuje detailnejšie preskúmať konkrétne časti grafu, zatiaľ čo sémantické priblíženie môže meniť aj obsah v závislosti na priblíženom úseku. Posunutie grafu v rámci vizualizačnej plochy poskytuje flexibilitu pri navigácii. Používateľ môže posúvať pohľad na graf, aby preskúmal rôzne časti siete bez potreby meniť mierku priblíženia.

Priblíženie a posunutie sú základnými nástrojmi pri spracovaní veľkých a komplexných grafov. Tieto techniky umožňujú užívateľom flexibilne manipulovať s vizualizovanými dátami, čím podporujú lepšie porozumenie štruktúry siete. Implementácia týchto techník vytvára interaktívne vizualizačné prostredie, ktoré uľahčuje objavovanie a analýzu vzťahov v grafe. Týmto spôsobom priblíženie a posunutie zohrávajú kľúčovú úlohu v procese efektívnej vizualizácie grafov a sietí.

Obe, priblíženie a posunutie, sa navzájom dopĺňajú. Príkladom sú geografické mapy, ako tie používané v službe Google Earth: predpokladajme, že používateľ priblížil oblasť vedľa Frankfurtu v Nemecku. Ak chce zmeniť pohľad na inú oblasť, napríklad Amman v Jordánsku, obvykle musí najskôr priblížiť, aby získal lepší prehľad, potom posunúť na región Amman a nakoniec znovu priblížiť Amman. Vykonanie tohto postupu bez posunutia by trvalo dlho, kým by sa našiel cieľ [22, 58]. V [15] bola predstavená elegantná koncepcia, ktorá vysvetľuje, ako spolu pôsobia priblíženie a posunutie. Navrhovaný koncept sa nazýva diagramy priestorovej škály. Definuje abstraktný priestor tým, že najprv vytvorí viacero kópií pôvodného 2D obrazu. V ďalšom kroku sú tieto kópie naskladané tak, aby vytvorili obrátenú pyramídu, na ktorej je každá kópia umiestnená v určitej úrovni priblíženia. Diagram priestorovej škály môže byť použitý s oboma typmi priblíženia, nielen pre geometrické priblíženie, ale aj pre sémantické priblíženie.

7 DISKUSIA

V našej práci sme vytvorili metodiku pre prípravu dát pre vizualizáciu dát v 3D priestore s použitím silou riadených algoritmov vo fyzikálnom hernom prostredí Unity3D. Predstavujeme vlastné silou riadené algoritmy a ich modifikácie, postupy a doporučená, výber a použitie správnych interakčných nástrojov, ktoré v konečnom dôsledku napomáhajú vytvoriť zrozumiteľné a čitateľné vizualizácie dát v 3D priestore bez nutnosti skúmania a pohľadávania dát vopred. Vytvorené algoritmy používajú vnútorné vlastnosti grafov, čo napomáha jednoduchšej aplikácii na rôzne dáta. Základ algoritmov vychádza z 2D, a preto je dôležité si uvedomiť, aké problémy a výzvy prináša prenos algoritmov do 3D priestoru. V 2D priestore absencia tretieho rozmeru slúži ako obmedzenie, ktoré odstraňuje niektoré nežiadúce stavy vizualizácie, ako napríklad prekrývanie vrcholov v priestore.

Hlavný algoritmus vizualizácie dát v 3D so všetkými jeho modifikáciami, sme vytvorili iteratívnym spôsobom, pričom pridaním nových postupov a obmedzení sa postupne objavovali nové zistenia, ktoré viedli k súčasnemu stavu. V prvej iterácii, sme sa zamerali na reprezentáciu hrán v grafe pomocou pružinových spojení. Výsledky experimentov ukázali obmedzenia 3D priestoru a hlavne samotných fyzikálnych herných prostredí, pričom boli zistené maximálne použiteľné objemy dát, ktoré je možné pomocou týchto nástrojov vizualizovať. Ďalšie podstatné zistenie z vykonaných experimentov v prvej iterácii je, že vrcholy v grafe pri použití pružinových spojení majú tendenciu kmitať do úrovní, kedy už nie je možné graf zastabilizovať. Zároveň sme zistili, že pružinové spojenia, tak ako sú reprezentované v herných prostrediach, sú veľmi náchylné na zmenu ich vnútorných parametrov, ktoré nie sú jednoznačne definované pre druh dát. Preto sme v našej metodike rozhodli o nastavení pružinových spojení ako konštant, pričom jediný parameter ktorý sa mení je normalizovaná sila.

```
1. SpawnNodesOnRandomPositions(nodes);
2. foreach (var node in nodes)
3. {
4.     node.AddEdges(edges[node]);
5.     node.InitializeSpringJoints();
6. }
7. EnableForce();
8. StopAfterOr(time, avgVelocity < velocity);
```

V ďalšej iterácii výskumu, sme zadefinovali hrany grafu ako konštantne pôsobiace sily ako je opísané v kapitole 6.4.1. V tomto experimente sily môžu mať príťažlivý efekt alebo odpudivý efekt, na rozdiel reprezentácie pomocou pružín, v tejto iterácii vytvoríme prepojenia medzi všetkými vrcholmi v grafe podľa pseudokódu:

```
1. public void AddEdges(Edge[] edges)
2. {
```

```

3.     foreach(var node in Graph.nodes)
4.     {
5.         if(node.IsIn(edges))
6.             this.AddEdge(positiveEdge);
7.         else
8.             this.AddEdge(negativeEdge);
9.     }
10. }

```

Výsledky experimentov ukázali, že tento prístup má menšiu tendenciu rozkmitať model ale zároveň prináša nové problémy s udržaním modelu v zornom poli kamery. Aplikované sily na jednotlivé vrcholy závisia od povahy dát a v mnohých prípadoch bol pozorovaný jav, pri ktorom vrcholy, na ktoré bola aplikovaná veľká sila nadobudli také zrýchlenie, že boli nakoniec odtrhnuté od zvyšku modelu. Tento jav môžeme pozorovať aj v 2D prostredí pri algoritmoch ako je napríklad Force Atlas.

Ďalším nežiadúcim javom bolo vytvorenie jedného veľkého zhluku, ktorý obsahoval väčšinu dát, a tým sa stala vizualizácia nečitateľná. Aby sme odstránili spomenuté nedostatky, spojili sme jednotlivé postupy t.z. pružinové spoje a konštantne aplikovanú silu do jedného algoritmu (Pružinovo-silový algoritmus), ktorý je opísaný v kapitole 6.6. Tento algoritmus odstraňuje problémy s kmitaním, čo zabezpečujú konštantne aplikované sily a zároveň pružinové spojenia držia všetky vrcholy v zornom poli kamery. Z jednoduchšej aplikácie algoritmov v 3D, tento algoritmus poskytuje najlepšie výsledky v pomere k jeho zložitosti. Na druhú stranu stále obsahuje určité nedostatky, ktoré môžu zakrývať alebo skresľovať informáciu. Jedným zo závažných problémov je, že tento algoritmus stále nedokáže udržať model vo vybranom bode v priestore. Aj keď odstránime kmitanie jednotlivých vrcholov, model ako celok stále disponuje silou, ktorá ho celý posúva určitým smerom. Model môže nadobudnúť takú rýchlosť, že kamera nedokáže model sledovať v komfortnej rýchlosti. Na odstránenie tohto problému sme v ďalšej iterácii pridali gravitačnú silu t.z. kotvu, ktorá udržiava model v okolí jedného bodu. Kotva pôsobí na každý vrchol v modeli rovnakou konštantnou príťažlivou silou, a zároveň neovplyvňuje samotné vzťahy medzi jednotlivými vrcholmi. Tento prístup je jednoduchý na použitie, ale zároveň pri zle nastavenej konštantnej sile môže celý model skončiť v jednom zhluku ako pri druhej iterácii. Na odstránenie problému jedného veľkého zhluku, bola celá vizualizácia transformovaná na sférické zobrazenie. Sféra v tomto prípade funguje ako zábrana pre vrcholy a zabezpečuje konštantnú vzdialenosť všetkých vrcholov od vybraného bodu v priestore - kotvy. Táto iterácia nášho algoritmu zabezpečuje prehľadnú vizualizáciu pre väčšinu dátových setov a odstraňuje väčšinu nežiadúcich javov z prvých troch iterácií algoritmu.

V ďalších iteráciách algoritmu sme sa zamerali na odstránenie javov, ktoré zabraňujú prehľadnej vizualizácii v 3D ako napríklad prekryvaniu vrcholov alebo neoptimálneho zobrazenia dát na sfére. Na rozdiel od 2D priestoru kde nám absencia jedného rozmeru

napomáha odstrániť problém prekryvania vrcholov, v 3D sme museli zakomponovať nové postupy:

- Dynamické odpudivé sily
- Konštantné vrcholové obmedzenie

Pri použití konštantného vrcholového obmedzenia zabezpečíme každému vrcholu svoj vlastný priestor, pričom ostatné vrcholy sa nemôžu priblížiť do vzdialenosti, ktorá by spôsobovala prekryvanie vrcholov. Pri tomto postupe nastáva problém pri väčších objemoch dát, pretože vrcholy môžu nepriamo ovplyvňovať vzťahy okolo seba tým, že odtláčajú (neprirodzene) všetko smerom od svojho stredu. Na druhú stranu pri menších objemoch sa podľa experimentov, tento postup osvedčil hlavne vzhľadom na výpočtovú zložitosť a jednoduchosť implementácie. Druhým spôsobom je zapojenie dynamických odpudivých síl, ktoré sú aktívne len v prípade, že sa vrcholy priblížia na definovanú vzdialenosť. Každý vrchol má vzdialenosť definovanú podľa svojho stupňa, čo zabezpečuje, že vrcholy s vysokým stupňom nezabraňujú vrcholom s malým stupňom v pohybe po sfére. Postupy je možné kombinovať, ale treba zachovať podmienku, že dynamické odpudivé sily budú aktívne skôr, ako sa vrchol priblíži do vzdialenosti ako je definované v konštantnom obmedzení. V ďalšej iterácii vývoja sme sa venovali hranám grafu, ich zafarbeniu, tvaru a iným vlastnostiam, ktoré je možné aplikovať na vizualizáciu hrán. Zafarbenie hrán môže byť naviazané na rôzne vlastnosti vyplývajúce z dát, pričom v našej verzii je táto vlastnosť naviazaná na euklidovskú vzdialenosť, pre lepšie zachytenie priestorového vnímania. Taktiež je táto vlastnosť v súlade s myšlienkou silou riadených algoritmov, a to takým spôsobom, že priestorová vzdialenosť je len vnútorná vlastnosť grafu t.z. nie je potrebný zásah z vonku na jej určenie, a preto vytvorený algoritmus môže naďalej pracovať klasickým postupom. Pre ešte lepšie prehĺbenie priestorového vnímania, bola zadefinovaná hrúbka hrán nasledovne:

1. Zelené hrany (Vrcholy sú bližšie ako $d1$), hrúbka 1
2. Žlté hrany (Vrcholy sú bližšie ako $d2$), hrúbka 0.5
3. Červené hrany (Vrcholy sú bližšie ako $d3$), hrúbka 0.1

Pričom $d1$, $d2$ a $d3$ závisia od polomeru r sférického obmedzenia kde $d1 = r / 3$, $d2 = r$ a $d3 = 2r$. Hrúbka je vyjadrená v jednotkách Unity3D priestoru. V nasledovnom pseudo-kóde je opísaný vytvorený algoritmus s použitím všetkých modifikácií:

Pseudokód vytvoreného pružinovo-silového algoritmu s pridanými modifikáciami

```
1. public void Visualize()
2. {
3.     List<Node> nodes = DatabaseManager.GetNodes();
4.     List<Edges> edges = DatabaseManager.GetEdges();
5.     // Preprocess data
6.     var normalizedEdges = NormalizeEdges(edges);
7.     // Spawn all GameObjects
8.     SpawnNodesInRandomPositions(nodes);
9.     var anchor = SpawnAnchorAt(position);
10.    var radius = CalculateRadius(nodes, normalizedEdges);
11.
12.    // Create sphere by calculations or set manually in editor
13.    CreateSphereAround(anchor, radius);
14.    InitializeEdges(nodes, normalizedEdges);
15.
16.    // apply forces in this order to prevent jittering
17.    ApplyForce(anchor); // enables anchor for each node
18.    ApplyForce(normalizedEdges, ForceType.SPRING); // enables spring edges
19.    ApplyForce(normalizedEdges, ForceType.FORCE); // enables force edges
20.
21.    // Optional forces
22.    if (useNegativeDynamicForce)
23.        ApplyNegativeDynamicForce(nodes);
24.    if (useNegativeConstForce)
25.        ApplyNegativeConstForce(nodes);
26.
27.    // monitoring
28.    if (calcStatistics)
29.        CalcStatistics();
30.    if (calcClusters)
31.        CalcClusters();
32.
33.    // Constrains.VELOCITY is desired velocity of the model
34.    StartVisualization();
35.
36.    // enable interaction (VR, AR, Leap motion)
37.    EnableInteractionWithModel();
38.    // Stop visualization after set time or if the model has desired velocity
39.    StopAfter(time, Constrains.VELOCITY);
40. }
```

```
1. // ticks every frame
2. // Spring edges are not in update because
3. // they are independent component of nodes
4. // and their forces are calculated by physic engine alone
5. public void NodeUpdate()
6. {
7.     if(anchorActive)
8.         ApplyForceToAnchor();
9.
10.    // node control base logic
11.    foreach (var edge in edgesForce)
12.    {
13.        edge.ApplyForce();
14.        edge.UpdateColor();
15.        edge.UpdatePosition();
16.
17.        if(curvedEdges)
18.            edge.UpdateCurve();
19.    }
20.
21.    // apply negative force to closest nodes
22.    if(!dynamicNegativeEdgesActive) return;
23.    var nodesInRange = this.GetNodesInRange();
24.    foreach (var node in nodesInRange)
25.    {
26.        // direction away from this node
27.        var direction = Vector3.Direction(position, node.position);
28.        node.ApplyForcePush(direction);
29.    }
30.
31.    // add drag to slow nodes movement after time
32.    if(dragTimer)
33.    {
34.        this.rigidBody.drag += dragConst;
35.        dragTimer.Reset();
36.    }
37. }
```

7.1 Porovnanie časovej zložitosti

V tejto časti práce sa zameriame na porovnanie výsledkov nášho algoritmu založeného na 3D vizualizácii dát s existujúcimi algoritmami používanými pre 2D vizualizáciu. Naším cieľom je zistiť, ako sa náš prístup k vizualizácii veľkých dátových množín v trojrozmernom priestore porovnáva s tradičnými algoritmami používanými v dvojrozmernom priestore. V tejto kapitole sa pozrieme na čas potrebný k dosiahnutiu ustálenej vizualizácie nášho algoritmu v porovnaní s existujúcimi metodikami a algoritmami, ktoré sú bežne používané na vizualizáciu dát. Tento pohľad na porovnanie nám umožní lepšie pochopiť výhody a obmedzenia nášho prístupu v porovnaní s existujúcimi technikami.

Tabuľka 8 Časové porovnanie algoritmov

Dataset	Počet vrcholov / Počet hrán	Základný alg. [ms]	Fruchterman- Reingold [ms]	Kamada-Kawai [ms]
Game of Thrones	796 / 3909	3510 ± 62,20	3490 ± 48,60	3350 ± 55,30
Witcher	224 / 2600	1120 ± 39,80	1130 ± 21,50	1130 ± 19,00
Lazega	36 / 115	198 ± 03,86	203 ± 11,30	198 ± 08,72
Network Science	1589 / 2742	7030 ± 77,10	6010 ± 42,80	7430 ± 104,00
Pokec5000	5000 / 52182	27800 ± 302,00	21400 ± 396,00	43300 ± 792,00
Pokec1000	1000 / 6297	4980 ± 108,00	4780 ± 130,00	4740 ± 125,00

Časová zložitosť pre vybrané algoritmy je porovnaná v Tabuľka 8. Z výsledkov Tabuľka 9 v je zrejmé, že vybrané algoritmy v 2D priestore sú rádovo rýchlejšie ako vizualizácia v 3D priestore. Zmena nastáva pri rastúcom objeme vizualizovaných dát ako je zobrazené v Tabuľka 9.

Keďže vybrané 2D algoritmy nedisponujú podmienkou na zastavanie výpočtu vizualizácie, vykonávajú vždy rovnaký počet iterácií na výpočet výsledného stavu grafu. Každá iterácia je výpočtovo náročná a musí byť vykonaná aj keď je graf v ustálenom stave. Predstavený pružinovo silový algoritmus, zastaví vizualizáciu podľa zastavovacej podmienky 6.7.5, a tak zabráni zbytočnému prepočtu ustáleného grafu. Pri narastajúcom počte vrcholov a hrán, ako napríklad pri datasecte Pokec5000, môžeme pozorovať rýchlejšie časy z toho dôvodu, že aplikované sily v grafe sú dostatočne veľké a nedovolia vrcholom sa rozkmitať, čo má za následok rýchlejšie ustálenie vrcholov v priestore.

Tabuľka 9 Časové porovnanie pružinovo-silového algoritmu s kontrolovaným rozmiestňovaním

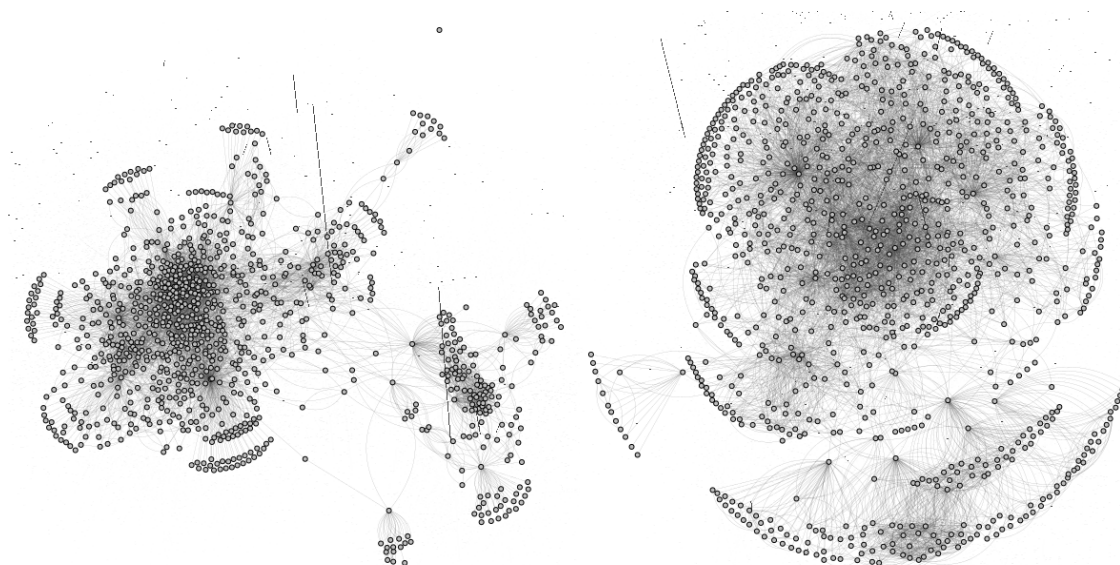
Dataset	Počet vrcholov / Počet hrán	Pružinovo-silový algoritmus [ms]
Game of Thrones	796 / 3909	19244 ± 250
Witcher	224 / 2600	16784 ± 303
Lazega	36 / 115	640 ± 15
Network Science	1589 / 2742	41404 ± 539
Pokec5000	5000 / 52182	16247 ± 1480
Pokec1000	1000 / 6297	83330 ± 15234

7.2 Vizuálne porovnanie

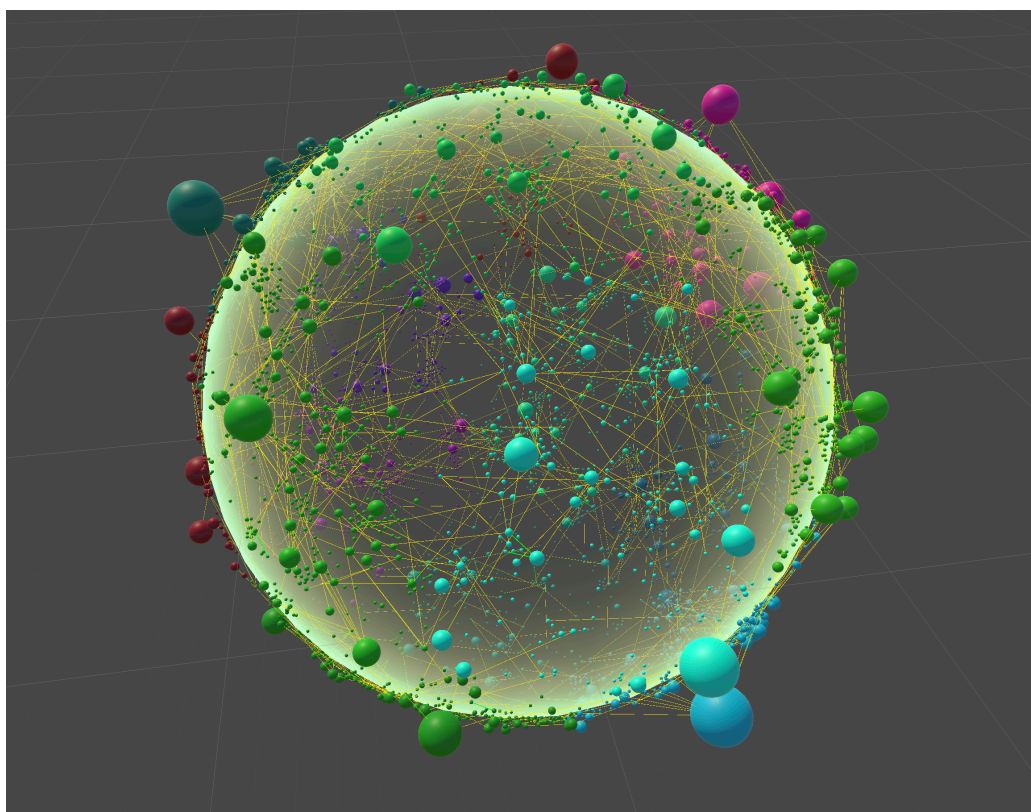
Táto kapitola sa zameriava na vizuálne porovnanie rôznych metód vizualizácie grafov s novo vyvinutou metódou, ktorá kombinuje pružinovo-silový algoritmus s kontrolovaným rozmiestňovaním (PSA-KR). Cieľom je posúdiť využiteľnosť PSA-KR v porovnaní s existujúcimi metódami, ako je jednoduchý pružinový algoritmus (PA), silovo riadený algoritmus (SA) a existujúce vybrané 2D algoritmy.

Vybrané 2D algoritmy na Obrázok 53, majú tendenciu vyvárať jeden veľký zhluk s narastajúcim počtom dát, pričom pozorovateľné pomyselné hranice zhlukov miznú s narastajúcim počtom vrcholov a hrán v grafe. Na Obrázok 53 môžeme vidieť, že vizualizované dáta vytvorili zhluk, ktorý nie je možné hlbšie analyzovať bez vonkajšieho zásahu používateľa alebo bez obmedzenia vizualizovaných dát. PSA-KR na Obrázok 54, vizuálne lepšie odlišuje vytvorené zhluky a umožňuje hlbšiu vizuálnu analýzu. Pozorovaním vizuálneho modelu môžeme usúdiť, že vzniknuté zhluky sú čitateľnejšie a pomyselné hranice zhlukov sú výraznejšie, čo napomáha hlbšej analýze dát. Opísaný jav je výraznejší pri väčšom objeme dát ako môžeme vidieť na Obrázok 58, kde sme pre 2D vizualizáciu pomocou silou riadených algoritmov, pomaly dosiahli hranice použiteľnosti, pričom v PSA-KR stále umožňuje vizuálnu analýzu dát a vzniknutých zhlukov, ako môžeme pozorovať na obrázku Obrázok 59. Je potrebné podotknúť, že vytvorené algoritmy v 3D priestore disponujú značnou mierou možností interakcie s priestorom, kde je možné dáta pozorovať z rôznych uhlov pohľadu a zároveň interagovať so samotným priestorom. Priestorové vnímanie je pre 3D algoritmy neoddeliteľnou časťou vizualizácie a pre používateľa vytvára mentálne pomôcky, ktoré mu napomáhajú vnímať spojitosti vo vizualizovaných dátach. Pre menšie datasety, ktorých príklad môžeme pozorovať na obrázkoch Obrázok 56 a Obrázok 57, nie je spomínaný fenomén na prvý pohľad zreteľný, ale pri interakcií a manipulácii s priestorom, môžeme pozorovať na obrázku Obrázok 57 zreteľné body záujmu, ktoré pridávajú nové informácie o pozorovanom datasete. Samotné vnímanie priestoru

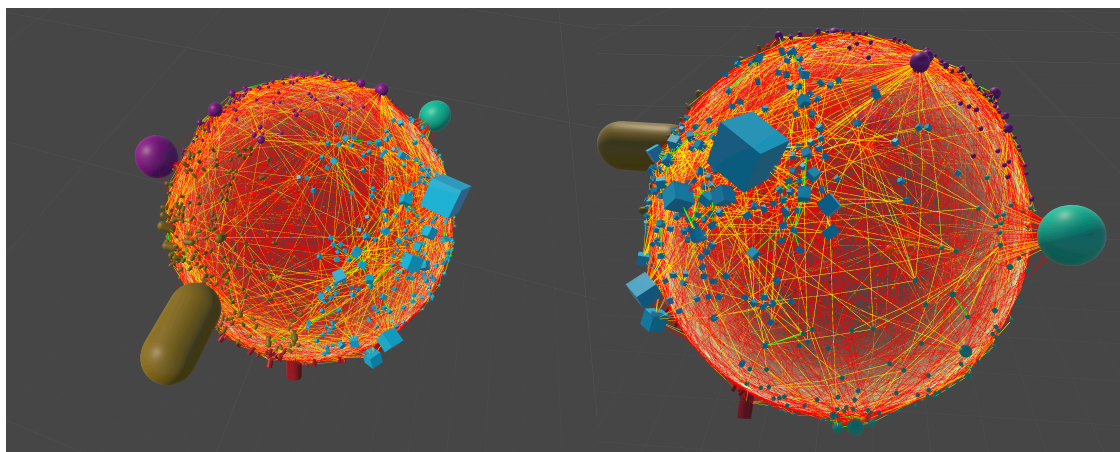
v tomto prípade slúži ako nová skupina vlastností určitých vrcholov, pričom môžeme pozorovať epicentrum pravdepodobne najdôležitejších vrcholov v pozorovanom grafe.



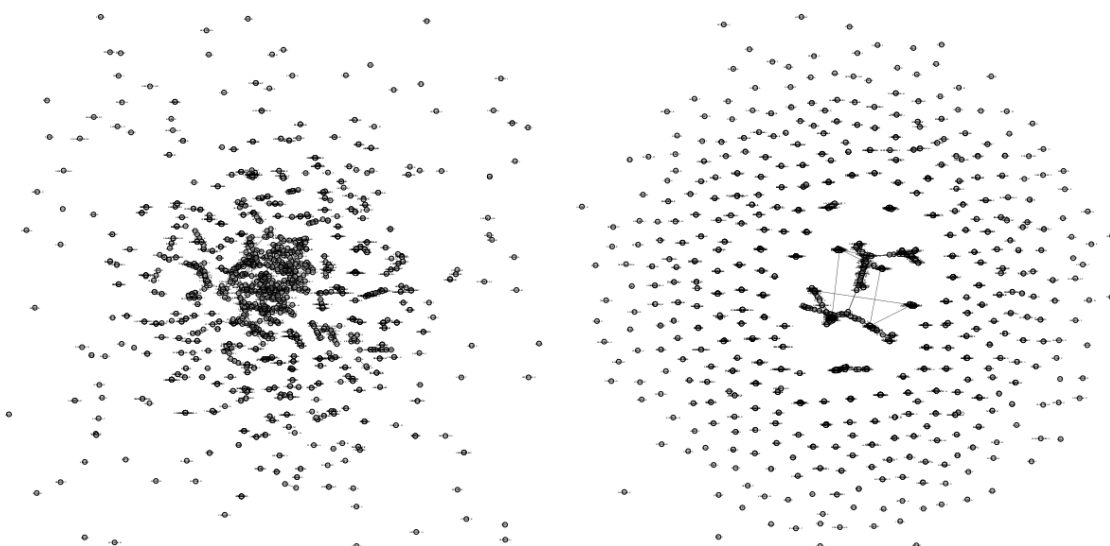
Obrázok 53 Pokec1000 - ForceAtlas2 a Kawada – Kawai



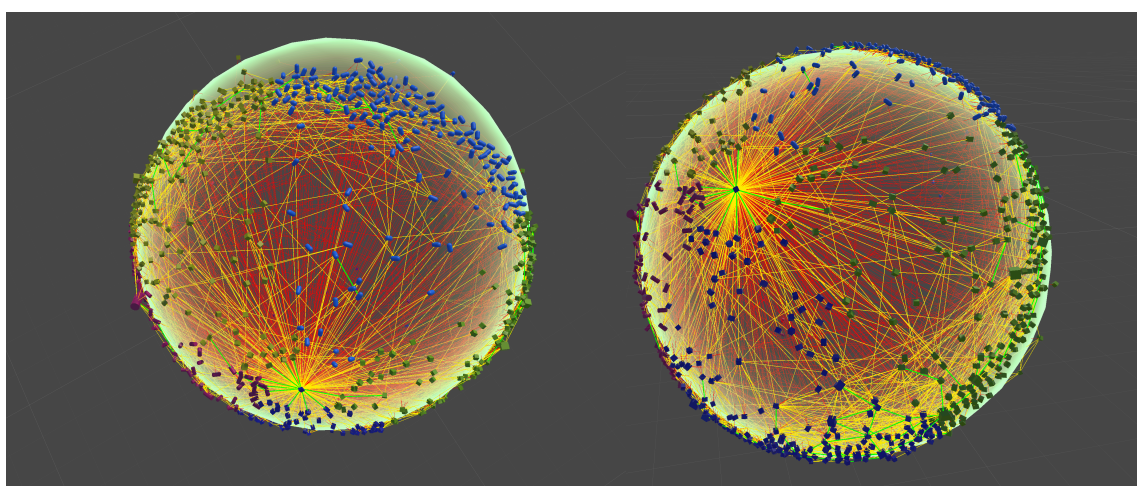
Obrázok 54 Pokec1000 - Pružinovo-silový algoritmus so skrytými hranami



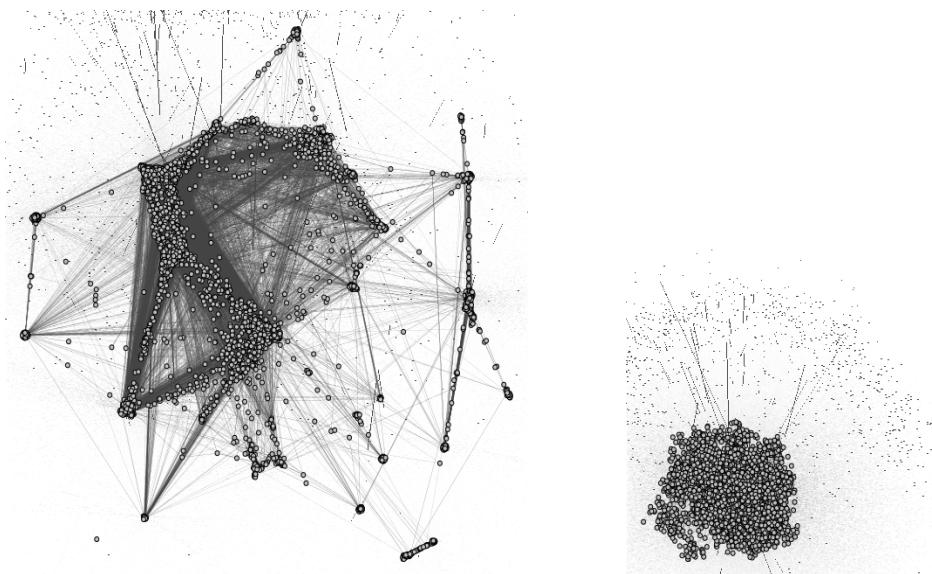
Obrázok 55 Pokec1000 - Pružinovo-silový algoritmus



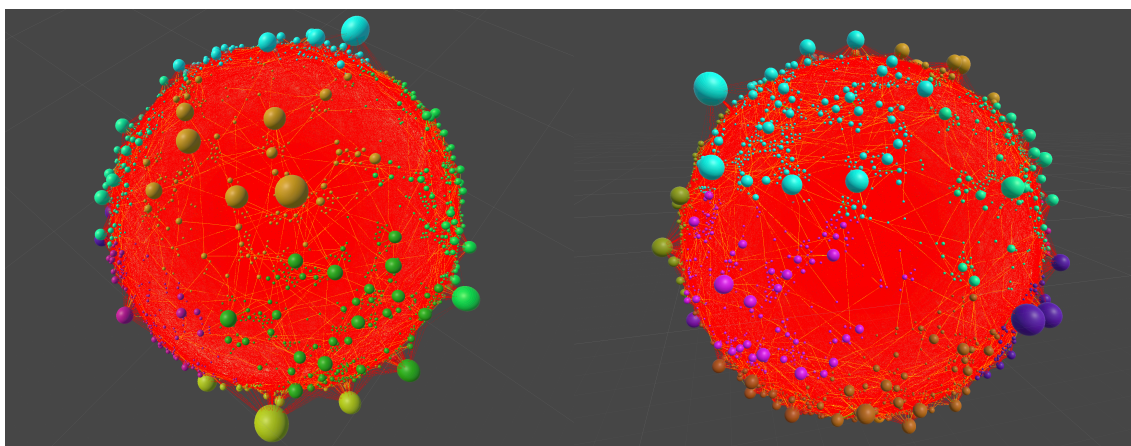
Obrázok 56 Network Science - základný algoritmus a Fruchtreingold algoritmus



Obrázok 57 Network Science - Pružinovo-silový algoritmus



Obrázok 58 Pokec5000 - Fruchterman-Reingold a Force Atlas2



Obrázok 59 Pokec5000 Pružinovo-silový Algoritmus

7.3 Porovnanie zhlukov a priestorovej vzdialenosti

Hodnotenie navrhnutých algoritmov si vyžaduje definovanie metrík pre porovnanie. Keďže použitie kritéria estetickej kvality výslednej vizualizácie je bez výskumu používateľov problematické, zvolili sme kvantitatívne kritériá založené na dvoch základných problémoch vizualizačných algoritmov v 3D priestore. Prvým kritériom je priemerná vzdialenosť vrcholov grafu vo výslednej vizualizácii, druhým je priemerná hustota vytvorených zhlukov. Posledným sledovaným parametrom je priemerná vzdialenosť vytvorených zhlukov.

Priemerná vzdialenosť vrcholov definuje celkovú hustotu grafu. Keďže pre interpretáciu výslednej vizualizácie predstavuje problém ako príliš riedky, tak aj príliš hustý graf, je potrebné sledovať priemernú vzdialenosť vrcholov ako kritérium kvality výslednej

vizualizácie. Pseudo-kód na výpočet priemernej vzdialenosti vrcholov vo výslednom grafe je definovaný v algoritme nižšie.

```

1. // Avg. vertex distance
2. distances = new List()
3. for i in nodes:
4.     for j in nodes:
5.         distances.Add(Distance(i, j))
6. // sum(distances) / distances.count
7. distances.Average()

```

Priemerná hustota klastrov vyjadruje, nakoľko blízko seba sú vo výslednej vizualizácii vrcholy s podobnými vlastnosťami. V prípade malej vzdialenosti medzi vrcholmi v zhľuku je výsledná informácia ťažko čitateľná bez interakcie s používateľom vo forme zväčšovania (ang. zoom). Pri nadmernom zväčšovaní sa následne strácajú informácie o celkovom kontexte vizualizovanej podmnožiny dát. Naopak, pri príliš nízkej hustote nie je zrejmé členstvo vrcholov v rámci zhľuku. Definovaná metrika je popísaná pseudo-kódom v Algoritme nižšie.

```

1. // Avg. cluster density
2. distances = new List()
3. for each c in clusters:
4.     nodes = c.nodes
5.     distancesCluster = new List()
6.     for i in nodes:
7.         for j in nodes:
8.             distancesCluster.Add(Distance(i, j))
9.             distances.Add(distancesCluster.Average())
10. // sum(distances) / distances.count
11. distances.Average()
12.

```

Priemerná vzdialenosť klastrov funguje na podobnom princípe ako vzdialenosť medzi vrcholmi, avšak zohľadňuje skupiny vrcholov. Opäť platí, že výsledná vizualizácia by nemala viesť k extrémnej vzdialenosti medzi zhľukmi, ani k ich nadmernej blízkosti. Pseudo-kód je definovaný v algoritme nižšie.

```

1. // Avg. clusters distance
2. distances = new List();
3. for each centroid in centroids:
4.     for each c in centroids:
5.         if c == centroid continue;
6.         distances.Add(Distance(centroid, c))
7. // sum(distances) / distances.count
8. distances.Average()
9.

```

7.3.1 Pružinový algoritmus

Napriek tomu že, táto metóda vytvára čisté a symetrické usporiadania pre grafy strednej veľkosti, považuje sa za náročný algoritmus. Časová náročnosť býva často vysoká, čo robí algoritmus v mnohých prípadoch nepraktickým. Navyše trpí nedostatočnou prediktabilitou, čo znamená, že opakované spustenia algoritmu môžu viesť k rôznym výsledkom. To môže predstavovať výzvu pri udržiavaní mentálnej mapy používateľa počas interakcie s nestabilnými usporiadaniami.

V počiatočných experimentoch sa testovali pružinové spojenia medzi vrcholmi, ktoré predstavujú atraktívne sily. Zároveň zabezpečujú aj zodpovedajúce odpudivé sily, pretože jednou z vlastností pružín je sťahovanie ich dĺžky. Vrcholy a hrany sú generované s náhodnými vlastnosťami, ako je farba a pozícia, aby sa odstránilo možné skreslenie spôsobené akoukoľvek špecifickou súborom dát. Použili sme náhodné pozície na odstránenie počiatočného skreslenia spôsobeného kumulovanou silou, ktorá sa aplikuje na uzly pri umiestnení na začiatku scény. Na smerovanie hrán v grafe sú vrcholom navyše priradené farby, ktoré reprezentujú ich typ. Na základe typov počiatočného a koncového uzla sa vytvárajú hrany a nastavuje sa pevnosť pružinového spoja. Pružinové spojenia umožňujú rýchlu a dostatočne prehľadnú vizualizáciu grafu, pričom pri použití správneho nastavenia riadia rozloženie uzlov v priestore. Ich nevýhodou sú obmedzené sily pôsobiace v jednom smere a zvýšené zaťaženie výpočtovej jednotky. Každý pružinový spoj musí byť reprezentovaný jedným špecifickým objektom v scéne, čo spôsobuje zvýšené zaťaženie nielen pre fyziku, ale aj pre vykresľovanie [74].

Základná chyba pružinového algoritmu je vysoká náchylnosť na rozkmitanie celého modelu. Toto je základný identifikovaný nedostatok, ktorý viedol k úsiliu o zlepšenie použitej vizualizačnej metódy. Výsledky z experimentov môžeme pozorovať v Tabuľka 10.

Tabuľka 10 - Výsledky experimentov s jednoduchým pružinovým algoritmom (v abstraktných jednotkách používaných v Unity Engine).

Názov datasetu	Priemerná vzdialenosť vrcholov	Priemerná hustota zhlukov	Priemerná vzdialenosť zhlukov
Game of Thrones	348.98	28.65	69.67
Witcher	668.68	27.95	66.71
Lazega	22.01	3.84	26.66
Network Science	166.01	26.89	80.96
Pokec 1000	352.01	26.74	72.03

V praxi možno jednoduchý pružinový algoritmus použiť na zobrazenie sociálnych sietí, vzťahov v biologických systémoch alebo štruktúry webu. Výhodou tohto prístupu k vizualizácii grafov je, že môže odhaliť vzory a skupiny vrcholov, ktoré by v iných formách vizualizácie mohlo byť ťažké vidieť. Tento typ vytvorenej vizualizácie má svoje uplatnenie aj v analýze dát a skúmaní komplexných systémov, pretože ich dynamický prístup k umiestneniu vrcholov môže pomôcť odhaliť štruktúry a vzťahy, ktoré by inak zostali skryté.

7.3.2 Silou riadený algoritmus

Navrhnutý silou riadený algoritmus vytvára grafické reprezentácie pomocou emulovaného systému fyzikálnych síl. Hlavným znakom silou riadených algoritmov je fyzikálna príťažlivosť a odpudzovanie medzi vrcholmi reprezentovanými hranami grafu. Fyzikálne sily umožňujú vytváranie prehľadných a zrozumiteľných rozložení vrcholov a hrán v 3D priestore bez nutnosti zásahu používateľa pri procese zostavovania grafu. Nakoľko na uzly pôsobia rôzne sily, môže dôjsť k situáciám, kedy sa vrcholy zobrazia mimo zorného poľa kamery.

Tabuľka 11 - Výsledky experimentov so silou riadeným algoritmom (v abstraktných jednotkách používaných v Unity Engine).

Názov datasetu	Priemerná vzdialenosť vrcholov	Priemerná hustota zhlukov	Priemerná vzdialenosť zhlukov
Game of Thrones	100.22	7.14	18.86
Witcher	228.06	11.70	17.23
Lazega	20.17	2.41	6.42
Network Science	28.66	5.47	15.09
Pokec 1000	113.23	15.17	60.68

Výsledky silou riadeného algoritmu sú uvedené v Tabuľka 11. V porovnaní s pružinovým algoritmom je zrejmé, že finálna vizualizácia vyžaduje menej priestoru na zobrazenie výsledku, má nižšiu priemernú hustotu klastrov a ešte menšiu priemernú hustotu vnútri klastrov. Tým lepšie vyhovuje potrebám pre akceptovateľnú 3D vizualizáciu.

7.3.3 Pružinovo-silový algoritmus s kontrolovaným rozmiestňovaním

Kombináciou dvoch predtým definovaných prístupov sme dosiahli vyvážený spôsob, pri ktorom pružiny a príťažlivé sily spolupracujú na stabilizácii grafu. Vizualizácia ukazuje, aké dôležité je nastaviť parametre síl a pružín tak, aby sa dosiahla optimálna rovnováha medzi rýchlou stabilizáciou siete a minimalizáciou zahltenia modelu. Kombinácia prístupov tiež zabraňuje vytvoreniu jediného veľkého zhukku. Finálna vizualizácia preukazuje dobrú výkonnosť pri identifikácii zhlukov, ale mnohé vrcholy sú zobrazené mimo oblasti snímanej kamerou v 3D priestore. Na základe tohto faktu sme pristúpili k aplikácii kontrolovaného umiestnenia v kombinovanom algoritme, ktorý považujeme za náš výsledný silovo riadený algoritmus.

Na stanovenie vhodnej metodiky pre nastavenie obmedzení v 3D priestore bola vykonaná séria meraní zameraných na detekciu obmedzení a možností v hernom prostredí. Odporúčané limity pre počet herných objektov v jednej scéne boli odvodené z nášho predchádzajúceho výskumu na vytvorenie grafu pri zachovaní časovej a výpočtovej náročnosti pod prahom, pri ktorom sa vizualizácia stáva nepoužiteľnou.

Vizualizácia siete vo voľnom 3D priestore má niekoľko nevýhod, ktoré je potrebné zohľadniť pri konštrukcii obmedzujúcich podmienok. Napríklad vznik zhlukov zhoršuje prehľadnosť siete, pretože vrcholy ovplyvnené menšími príťažlivými silami alebo vrcholy ovplyvnené nadmernými odpudivými silami blokujú pohľad kamery.

Nekontrolované príťažlivé a odpudivé sily v sieti môžu vytvárať nežiadúce zhluky nečitateľné pre používateľov. Aby sme sa vyhli situáciám, kedy zhuk vrcholov zakrýva zvyšok vizualizácie, implementovali sme obmedzenie vzdialenosti medzi vrcholmi. Toto obmedzenie je reprezentované sférickým obmedzením, ktorá slúži ako stena pre ostatné vrcholy. Veľkosť sférického poľa je priamo odvodená od veľkosti samotného vrcholu – väčšie vrcholy disponujú väčšími sférickými obmedzeniami. Tento prístup umožňuje prehľadnejšiu vizualizáciu.

Na rozdiel od vrcholov, hranám je umožnené prechádzať sférickým obmedzením, čím sa vytvára zaujímavý efekt, kde možno pozorovať spojenia medzi vybranými vrcholmi bez rušenia z okolia. Sférické obmedzenie je preto do značnej miery prehľadné, ale z každého uhla zachováva svoj obrys pre konzistentné zobrazenie na povrchu gule.

Experimentálne výsledky v Tabuľka 12 ukazujú najlepší výkon na základe použitých metrík. Navyše použitím silou riadeného algoritmu s pružinami a kontrolovaným umiestnením sme boli schopní eliminovať nedostatky pozorované v predchádzajúcich implementáciách založených na pružinách alebo všeobecnom silovo riadenom prístupe.

Tabuľka 12 - Výsledky experimentov s pružinovo-silovo riadeným algoritmom s kontrolovaným rozmiestňovaním (v abstraktných jednotkách používaných v Unity Engine).

Názov datasetu	Priemerná vzdialenosť vrcholov	Priemerná hustota zhlukov	Priemerná vzdialenosť zhlukov
Game of Thrones	230.41	11.32	46.25
Witcher	378.91	16.43	42.34
Lazega	73.96	4.86	12.63
Network Science	92.57	6.04	29.33
Pokec 1000	310.08	14.33	55.17

Celkové experimentálne výsledky získané pre definované metriky sú uvedené v Tabuľka 13. Pre jednoduchšiu identifikáciu sú najlepšie hodnoty zvýraznené tučným písmom, pretože sa snažíme nájsť riešenie bez extrémneho umiestnenia. Ako vidno, iteratívnym vývojom sme sa pre väčšinu súborov dát dokázali prekonať pôvodné nedostatky silou riadeného algoritmu pre 3D prostredie, alebo sme sa k najlepšiemu riešeniu veľmi priblížili.

Tabuľka 13 - Prehľad výkonnostných metrík pre jednoduchý pružinový algoritmus, force-directed algoritmus a force-directed algoritmus s pružinami a kontrolovaným rozmiestňovaním

Názov datasetu	Algoritmus	Priemerná vzdialenosť vrcholov	Priemerná hustota zhlukov	Priemerná vzdialenosť zhlukov
Game of Thrones	Pružinový	348.98	28.65	69.67
	Silou riadený	100.22	7.41	18.86
	Pružinovo-silový SKR*	230.41	11.32	46.25
Witcher	Pružinový	668.68	27.95	66.71
	Silou riadený	228.06	11.70	17.23
	Pružinovo-silový SKR*	378.91	16.43	42.34
Lazega	Pružinový	22.01	3.84	26.66
	Silou riadený	20.17	2.41	6.42
	Pružinovo-silový SKR*	73.96	4.86	12.63
Network Science	Pružinový	166.01	26.89	80.96
	Silou riadený	28.66	5.47	15.09
	Pružinovo-silový SKR*	92.57	6.04	29.33
Pokec1000	Pružinový	352.01	26.74	72.03
	Silou riadený	113.23	15.17	60.68
	Pružinovo-silový SKR*	310.08	14.33	55.17

Pri tisícoch vrcholov a hrán je oveľa jednoduchšie pochopiť internú štruktúru dát a ľahko rozlíšiť rôzne typy dát podľa definovaných klastrov. V 2D je možné zlepšiť vizualizáciu ďalšími iteračnými krokmi a vytvoriť užitočnejší model siete.

V práci celkovo vznikli 3 hlavné algoritmy v 3D spolu s ich modifikáciami a to pružinový algoritmus, silou riadený algoritmus a pružinovo-silový algoritmus. Tieto algoritmy poskytujú intuitívne a prehľadné spôsoby vizualizácie na rôznych typoch dát o rôznej veľkosti, pričom najlepšie výsledky dosahuje pružinovo-silový algoritmus s kontrolovaným rozmiestňovaním, pri väčšine pozorovaných datasetoch. Usudzujeme, že vytvorené algoritmy napĺňajú tézu č.2. Vytvorené algoritmy pracujú s predpokladom, že dáta boli predspracované a pripravené podľa metodiky na prípravu dát opísanú v kapitole 6.3 a v prílohe A, ktorá nám zabezpečuje vyrovnané fyzikálne vlastnosti v modeli a obmedzuje nežiadúce stavy vizualizácií, pričom napĺňa tézu č.1. Podľa kapitoly 6.8, usudzujeme, že na interakciu v 3D priestore je potrebné zvoliť vhodné vstupné zariadenia, aby bola zachovaná dostatočná úroveň spätnej väzby vizualizácie a používateľa. Usudzujeme, že opísaný spôsob výberu a použitia vstupných zariadení pre interakciu v 3D napĺňa tézu č.3 a posúva hranice vizualizácie na vyššiu úroveň, ktorú nie je možné dosiahnuť použitím klasických nástrojov ako myš a klávesnica.

ZÁVER

V práci sme preskúmali dôležitosť vizualizácie dát v trojrozmernom priestore a jej úlohu pri analýze a porozumení komplexných vzťahov v rôznych oblastiach výskumu a aplikácií. Naším hlavným cieľom bolo efektívne vizualizovať veľké objemy dát v 3D priestore pomocou herných prostredí a silou riadených algoritmov.

Na základe našich experimentov sme zistili, že využitie fyzikálnych herných prostredí ponúka efektívnu možnosť vizualizovať veľké dátové množiny. Avšak, prevod algoritmov z 2D do 3D prostredia nie je triviálny proces a algoritmy navrhnuté pre 2D priestor môžu byť neefektívne alebo dokonca nepoužiteľné v 3D. Preto sme sa v prvej rade zamerali na vývoj algoritmov špecificky prispôsobených pre vizualizáciu veľkých dátových množín v 3D priestore.

Hlavným prínosom tejto práce je vytvorenie metodiky a algoritmov založených na princípe silou riadených algoritmov, ktoré umožňujú efektívnu vizualizáciu veľkých dátových sietí v 3D. Algoritmy obsahujú modifikácie ako sférické zobrazenia, normalizáciu, zhlukovanie, pružinové spoje a gravitačné interakcie, čím vytvára robustnú techniku na vizualizáciu komplexných dátových sietí. Využitie vytvorených algoritmov prináša inovatívny prístup k vizualizácii dát a môže byť využitý v rôznych oblastiach výskumu a aplikácií, kde je potrebné vizualizovať veľké dátové množiny v trojrozmernom priestore, ako aj v oblastiach analýzy kde vytvorené algoritmy obohacujú pozorované dáta o nové vlastnosti založené na fyzických prepojeniach.

Okrem samotného algoritmu sme sa tiež venovali riešeniu otázky, ako správne pripraviť dáta na vizualizáciu v 3D fyzikálnom hernom prostredí. Zistili sme, že správne upravenie a spracovanie dát, spolu s vhodnými vizualizačnými technikami a postupmi, sú kľúčové pre efektívnu vizualizáciu veľkých objemov dát v 3D priestore. Pre tento účel vznikla metodika pre prípravu dát na vizualizáciu v 3D priestore, ktorá obsahuje postupy, hodnoty a odporúčenia pre správnu úpravu vstupných dát. Využitím vytvorenej metodiky v spojení s vytvorenými algoritmami je možné dosiahnuť vysokú úroveň vizualizácie dát v 3D priestore.

Ďalšou dôležitou témou našej práce bolo využitie vstupných zariadení poskytujúcich 3D vstupy pre zabezpečenie efektívnej interakcie s 3D vizualizáciou. Využitie týchto zariadení umožňuje používateľom interagovať s dátami a získať nové poznatky o vzťahoch medzi nimi, čím sa zvyšuje užitočnosť vizualizačných nástrojov. Zvolenie správnych vstupných zariadení je kľúčové pre dosiahnutie dostatočnej úrovne interakcie,

ktorá poskytuje voľnosť v prehľadávaní dát, a zároveň neobmedzuje používateľa alebo nezakrýva informácie.

Celkovo možno usúdiť, že jednotný postup práce s dátami a ich vizualizáciou v 3D priestore je stále predmetom výskumu. Na základe výsledkov tejto práce sme však zistili, že vizualizácia veľkého objemu dát v 3D priestore je realizovateľná a prináša cenné poznatky v rôznych oblastiach výskumu a aplikácií. Zároveň sme zistili, že správnym spojením prípravy dát, vytvorených algoritmov, výberom vhodných modifikácií a výberom vhodných vstupných zariadení, vytvorené algoritmy dokážu obohatiť dáta o priestorové informácie a nové vlastnosti založené na fyzikálnych vlastnostiach. Obohatené dáta je ďalej možné skúmať a iteratívnym procesom analyzovať a hlbšie porozumieť významu skúmaných dát ako celku. Veríme, že práca v konečnom dôsledku predstavuje nové spôsoby analýzy dát ako aj interakciu dátového analytika s trojrozmerným priestorom, čo považujeme za veľký posun v oblasti vizualizácie dát v 3D priestore a v oblasti silou riadených algoritmov.

Zoznam použitej literatúry

- [1] S. Bassil a R. K. Keller, „Software visualization tools: Survey and analysis,“ rev. *IWPC 2001: Proceedings 9th International Workshop on Program Comprehension*, 2011.
- [2] J. M. Hernández a P. Van Mieghem, „Classification of graph metrics,“ *Delft University of Technology: Mekelweg*, pp. 1-20, 2011.
- [3] R. S. Raghav, S. Pothula, T. Vengattaraman a D. Ponnurangam, „A survey of data visualization tools for analyzing large volume of data in big data platform,“ rev. *International Conference on Communication and Electronics Systems (ICCES)*, 2016.
- [4] West, Douglas a Brent, *Introduction to graph theory*, Prentice hall Upper Saddle River, 2001.
- [5] F. McGee, M. Ghoniem, G. Melancom a B. Pinaud, „The State of the Art in Multi-Layer Network Visualization,“ rev. *Computer Graphics Forum 2019*, 2019.
- [6] Wikipedia - The Free Encyclopedia, „Graph drawing,“ [Online]. Available: https://en.wikipedia.org/wiki/Graph_drawing. [Cit. 30 01 2024].
- [7] J. Mathieu, V. Tommaso, H. Sebastien a B. Mathieu, „Force Atlas 2, a Continuous Graph Layout Algorithm for Handy Network Visualization Design for the Gephi Software,“ *PLOS ONE*, zv. 9, %1. vyd.6, 2014.
- [8] S. M. Anuncia, H. A. Gohel a V. Subbiah, *Data Visualization: Trends and Challenges Toward Multidisciplinary Perception*, Springe Singapore, 2020.
- [9] J. Schmidt, „Visual Data Science,“ rev. *Data Science, Data Visualization, and Digital Twins*, 2022.
- [10] M. Vuckovic, J. Schmidt, T. Ortner a D. Cornel, „Combining 2D and 3D Visualization with Visual Analytics in the Environmental Domain,“ *Information*, zv. 13, %1. vyd.7, 2022.
- [11] B. D. Fisher, „Visual representations and interactions technologies,“ *Illuminating the Path: A Research and Development Agenda for Visual Analytics*, pp. 69-104, 2005.

- [12] F. Amini, S. Rufige, Z. Hossain, Q. Ventura, P. Irani a M. J. McGuffin, „The impact of interactivity on comprehending 2D and 3D visualizations of movement data,“ *IEEE transactions on visualization and computer graphics*, zv. 21, %1. vyd.1, pp. 122-135, 2014.
- [13] N. Elmqvist a P. Tsigas, „A taxonomy of 3d occlusion management for visualization,“ *IEEE transactions on visualization and computer graphics*, zv. 14, %1. vyd.5, pp. 1095-1109, 2008.
- [14] S. Bleisch a S. Nebiker, „Connected 2D and 3D visualizations for the interactive exploration of spatial information,“ rev. *Proc. of 21th ISPRS Congress*, Beijing, China, 2008.
- [15] T. Ortner, *Tight integration of visual analysis and 3D real-time rendering (Ph. D. thesis)*, Technische Universität Wien, 2020.
- [16] C. Nussbaumer Knaflitz, *Storytelling with Data: A Data Visualization Guide for Business Professionals*, Wiley, 2015.
- [17] K. Healy, *Data Visualization: A Practical introduction*, Princeton University Press, 2018.
- [18] A. Kirk, *Data Visualisation: A Handbook for Data Driven Design*, 2nd Edition, SAGE Publications Ltd, 2019.
- [19] B. R. Kent, *3D Scientific Visualization with Blender (IOP Concise Physics)*, Morgan & Claypool, 2015.
- [20] M. Grandjean, „Digital History | Data Visualization | Network Analysis,“ [Online]. Available: <https://www.martingrandjean.ch>. [Cit. 01 2024].
- [21] B. Bollobás, *Modern Graph Theory*, New York, NY: Springer, 1998.
- [22] R. Diestel, *Graph Theory*, Berlin: Springer, 2017.
- [23] J. L. Gross a J. Yellen, *Graph Theory and Its Applications (Textbooks in Mathematics)* 2nd Ed., Chapman and Hall/CRC, 2005.
- [24] M. Newman, *Networks*, 2nd Edition, Oxford University Press, 2018.
- [25] A.-L. Barabási a M. Pósfai, *Network Science*, Cambridge University Press, 2016.

-
- [26] T. H. Cormen, C. E. Leiserson, R. L. Rivest a C. Stein, Introduction to Algorithms, 3rd Edition, MIT Press, 2009.
- [27] M.-Y. Kao, Encyclopedia of Algorithms, 2nd ed., Springer, 2016.
- [28] S. S. Skiena, The Algorithm Design Manual (Texts in Computer Science) 3rd ed., Springer, 2020.
- [29] S. Palúch, Algoritnická teória grafov, Žilina: EDIS, 2020.
- [30] J. Janáček, M. Koháni, A. Szendreyová a L. Buzna, Diskrétna optimalizácia, Žilina: EDIS, 2015.
- [31] M. B. Fronc, Kvantitatívne metódy, Žilina: EDIS, 1998.
- [32] K. Cherven, Mastering Gephi Network Visualization, Packt Pub Ltd, 2015.
- [33] K. Cherven, Network Graph Analysis and Visualization With Gephi, Packt Pub Ltd, 2013.
- [34] T. M. J. Fruchterman a E. M. Reingold, „Graph drawing by force-directed placement,“ *Journal of Software: Practice and Experience*, zv. 21, %1. vyd.11, pp. 1129-1164, 1991.
- [35] Y. F. Hu, „Efficient and high quality force-directed graph drawing,“ *The Mathematica Journal*, zv. 10, pp. 37-71, 2005.
- [36] J. Barnes a P. Hut, „A hierarchical $O(N \log N)$ force-calculation algorithm,“ *Nature*, zv. 324, %1. vyd.4, pp. 446-449, 1986.
- [37] S. Martin, W. M. Brown, R. Klavans a K. Boyack, „OpenOrd: An Open-Source Toolbox for Large Graph Layout,“ rev. *SPIE Conference on Visualization and Data Analysis (VDA)*, 2011.
- [38] M. Jacomy, T. Venturini, S. Heymann a M. Bastian, „ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software,“ *PLOS ONE*, zv. 9, %1. vyd.6, p. <https://doi.org/10.1371/journal.pone.0098679>, 2014.
- [39] M. Bastian, S. Heymann a M. Jacomy, „Gephi: An Open Source Software for Exploring and Manipulating Networks,“ rev. *International AAAI Conference on Weblogs and Social Media*, 2009.

- [40] S. Heymann, „Gephi Wiki,“ 1 3 2015. [Online]. Available: <https://github.com/gephi/gephi/wiki>. [Cit. 1 2024].
- [41] D. Kaufmann a D. Wagner, *Drawing Graphs*, Berlin: Springer, 1998.
- [42] G. di Battista, P. Eades, R. Tamassia a I. Tollis, *Graph Drawing*, Englewood Cliffs: Prentice Hall, 1999.
- [43] J. Tukey, *Exploratory Data Analysis*, New York: Addison-Wesley, 1977.
- [44] E. Tufte, *The Visual Display of Quantitative Information*, Cheshire: Graphics Press, 1983.
- [45] W. Cleveland, *The Elements of Graphics Data*, Monterey: Wadsworth, 1985.
- [46] W. Cleveland, *Visualizing Data*, Summit: Hobart Press, 1993.
- [47] D. Khokhar, *Gephi Cookbook*, Packt Pub Ltd, 2015.
- [48] E. D. Kolaczyk a G. Csárdi, *Statistical Analysis of Network Data with R*, New York: Springer, 2014.
- [49] L. Takac a M. Zabovsky, „Data Analysis in Public Social Networks,“ rev. *International Scientific Conference & International Workshop Present Day Trends of Innovations*, Lomza, Poland, 2012.
- [50] F. Shadhin, „How to Represent a Graph Data Structure in Python,“ 5 2022. [Online]. Available: <https://medium.com/geekculture/how-to-represent-a-graph-data-structure-in-python-9f0df57e33a2>. [Cit. 01 2024].
- [51] H. Melcher, „Python Graph Libraries,“ 03 12 2022. [Online]. Available: <https://wiki.python.org/moin/PythonGraphLibraries>. [Cit. 1 2024].
- [52] G. Sabidussi, „The centrality index of a graph,“ *Psychometrika*, zv. 31, pp. 581-683, 1966.
- [53] L. Freeman, „A set of measures of centrality based on betweenness,“ *Sociometry*, zv. 40, %1. vyd.1, pp. 35-41, 1977.
- [54] P. Bonacich, „Factoring and weighting approaches to status scores and clique identification,“ *Journal of Mathematical Sociology*, zv. 2, %1. vyd.1, pp. 113-120, 1972.

- [55] L. Katz, „A new status index derived from sociometric analysis,“ *Psychometrika*, zv. 18, %1. vyd.1, pp. 39-43, 1953.
- [56] J. Kleinberg, „Authoritative sources in a hyperlinked environment,“ *Journal of ACM*, zv. 46, %1. vyd.5, pp. 604-632, 1999.
- [57] E. Lazega, „Lazega lawyers network data,“ 2001. [Online]. Available: <https://search.r-project.org/CRAN/refmans/sand/html/lazega.html>. [Cit. 1 2024].
- [58] M. E. J. Newman, „Finding community structure in networks using the eigenvectors of matrices,“ *Physical Review E*, zv. 74, 2006.
- [59] G. Di Battista, P. Eades, R. Tamassia a I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Englewood Cliffs: Prentice Hall, 1999.
- [60] U. Brandes, *Drawing on physical analogies.*, *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science* ed., zv. 2025, M. Kaufmann a D. Wagner, Ed., Springer-Verlag, 2001, pp. 71-86.
- [61] W. T. Tutte, „How to draw a graph.,“ rev. *Proc. London Math. Society*, 1963.
- [62] P. Eades, „A heuristic for graph drawing,“ rev. *Congressus Numerantium*, 1984.
- [63] S. G. Kobourov, „Force-Directed Drawing Algorithms,“ rev. *Handbook of Graph Drawing and Visualization*, R. Tamassia, Ed., CRC Press, 2013, pp. 383-408.
- [64] T. Kamada a S. Kawai, „An algorithm for drawing general undirected graphs,“ *Inform. Process. Lett.*, zv. 31, pp. 7-15, 1989.
- [65] R. Hadany a D. Harel, „A multi-scale algorithm for drawing graphs nicely,“ *Discrete Applied Mathematics*, zv. 113, %1. vyd.1, pp. 3-21, 2001.
- [66] D. Harel a Y. Koren, „A fast multi-scale method for drawing large graphs,“ rev. *Graph Drawing. GD 2000. Lecture Notes in Computer Science*, zv. 1984, J. Marks, Ed., Berlin, Heidelberg, Springer, 2002, pp. 179-200.
- [67] C. Walshaw, „A multilevel algorithm for force-directed graph drawing,“ *Journal of Graph Algorithms and Applications*, zv. 7, %1. vyd.3, pp. 253-285, 2003.
- [68] P. Gajer, M. T. Goodrich a S. G. Kobourov, „A fast multi-dimensional algorithm for drawing large graphs,“ *Computational Geometry: Theory and Applications*, zv. 29, %1. vyd.1, pp. 3-18, 2004.

- [69] G. Csardi a T. Nepusz, „python-igraph,“ 2024. [Online]. Available: <https://python.igraph.org/en/stable/>. [Cit. 01 2024].
- [70] G. Csardi a T. Nepusz, „The igraph software package for complex network research,“ *InterJournal, Complex Systems*, zv. 1695, 2006.
- [71] J. Lamping, R. Rao a P. Pirolli, „A focus+context technique based on hyperbolic geometry for visualizing large hierarchies,“ rev. *Proceedings of Computer Human Interaction*, 1995.
- [72] T. Munzner, „H3: Laying out large directed graphs in 3D hyperbolic space,“ rev. *Proceedings of IEEE Symposium on Information Visualization*, 1997.
- [73] S. G. Kobourov a K. Wampler, „Non-Euclidean spring embedders,“ *IEEE Transactions on Visualization and Computer Graphics*, zv. 11, %1. vyd.6, pp. 757-767, 2005.
- [74] A. Brezáni a M. Zábovský, „3D Graph Visualization Performance using Physical Engine,“ *INTERNATIONAL JOURNAL OF INFORMATION TECHNOLOGIES, ENGINEERING AND MANAGEMENT SCIENCE*, zv. 2, %1. vyd.6, 2022.
- [75] E. Lazega, *The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership.*, Oxford: Oxford University Press, 2001.

Prílohy

ZOZNAM PRÍLOH

PRÍLOHA A	METODIKA PRE PRÍPRAVU DÁT NA VIZUALIZÁCIU V 3D	III
PRÍLOHA B	ANALÝZA 2D ALGORITMOV – KÓDY	V

Príloha A | Metodika pre prípravu dát na vizualizáciu v 3D

1. **Vyberieme a upravíme dáta na grafovú štruktúru**
2. **Vyberieme dáta, ktoré budú reprezentovať vrcholy v grafe**
(Používateľov, značky áut, osoby v systéme a iné.)
3. **Vyberieme dáta, ktoré budú reprezentovať prítiažlivé sily-hrany**
(Dáta, ktoré budú reprezentovať vzťahy, pretože prítiažlivé sily sú hlavnou príčinou tvorby zhlukov a je potrebné definovať dopredu, aký účel má vizualizácia)
 - a. **Vyberieme dáta, ktoré budú reprezentovať odpudivé sily-hrany** (Nepovinné) (V prípade ak dáta nie sú vybrané, odpudivé sily budú pôsobiť medzi každou dvojicou vrcholov kde neexistuje hrana)
4. **Vyberieme dáta alebo metriku, ktorá bude reprezentovať trenie**
(Trenie spôsobuje, že vrcholy v grafe budú menej náchylné na zmenu svojej polohy a pôsobiace sily budú menej ovplyvňovať posun vrcholu v 3D priestore, pretože vyššia hodnota trenia zabraňuje aby vrchol nadobudol priveľkú rýchlosť. Napríklad môžeme trenie zadefinovať podľa metriky – stupeň vrcholu, pričom takéto nastavenie zabezpečí, že vrcholy s menším stupňom nebudú presúvať vrcholy s vyšším stupňom. Ak dáta alebo metrika nie je zvolená, trenie je nastavené pre každý vrchol na hodnotu 1 v abstraktných jednotkách Unity3D)
5. **Vyberieme dáta alebo metriku, ktorá bude reprezentovať hmotnosť** (Podobne ako vlastnosť „trenie“. Hmotnosť zabezpečuje dominantnosť vrcholu v priestore, pričom zabraňuje alebo podporuje jeho presun v 3D priestore. Čím je hodnota hmotnosti vyššia, tým väčšiu silu je potrebné aplikovať na jeho presun, ale na rozdiel od trenia, aj vrcholy s veľkou hmotnosťou môžu jednoducho nadobudnúť vysokú rýchlosť a tým rozkmitať model. Ak dáta alebo metrika nie je zvolená, hmotnosť je nastavené pre každý vrchol na hodnotu 1 v abstraktných jednotkách Unity3D)
6. **Vyberieme dáta alebo metriku, ktorá bude reprezentovať tvrdosť pružín** (Tvrdosť pružín obmedzuje kmitanie modelu. Zvolené dáta alebo metrika, by mala mať spomaľujúci charakter. Napríklad komunikujúci priatelia na sociálnej sieti budú by mohli mať nastavenú hodnotu tvrdosti vyššiu, aby sme zabezpečili ich spoločnú polohu v priestore. Ak dáta alebo metrika nie je zvolená, tvrdosť je nastavené pre každý vrchol na hodnotu 1 v abstraktných jednotkách Unity3D)
7. **Vyberieme dáta alebo metriku, ktorá bude reprezentovať veľkosť vrcholov** (Vizuálna prehľadnosť. Ak dáta alebo metrika nie je zvolená, veľkosť je nastavené pre každý vrchol na hodnotu 1 v abstraktných jednotkách Unity3D)

8. **(Nepovinné)** Vyberieme dáta pre iné vlastnosti ako je farba vrcholov, názov a iné.
9. **Normalizujeme dáta na abstraktné jednotky Unity3D**
 - a. **Príťažlivé sily na škálu 0-100** (*Pre kategorizáciu 0 - 1*)
 - b. **Odpudivé sily na škálu 0-100** (*Pre kategorizáciu 0 - 1*)
 - c. **Trenie na škálu 1-500** (*Pre veľké datasety sú preferované vyššie hodnoty*)
 - d. **Hmotnosť na škálu 1-1000** (*V prípade ak je použitá modifikácia konštantného zvyšovania hmotnosti tak je potrebné škálu zmeniť na 1-300*)
 - e. **Tvrdosť pružín na škálu 1-10**

Vybrané a normalizované dáta je teraz možné použiť na vytvorené algoritmy.

Príloha B | Analýza 2D algoritmov – kódy

B.1 Vizualizácia vzťahov medzi právnikmi (R)

```
1. # Lazega Network Visualization
2. # =====
3. #
4. # This data set comes from a network study of corporate law partnership
5. # that was carried out in a Northeastern US corporate law firm, referred to
6. # as SG&R, 1988-1991 in New England. It includes (among others) measurements
7. # of networks among the 71 attorneys (partners and associates) of this firm,
8. # i.e. their strong-coworker network, advice network, friendship network, and
9. # indirect control networks. Various members' attributes are also part of the
10. # dataset, including seniority, formal status, office in which they work,
11. # gender, lawschool attended, individual performance measurements (hours
12. # worked, fees brought in), attitudes concerning various management policy
13. # options, etc.
14. # Source: https://search.r-project.org/CRAN/refmans/sand/html/lazega.html
15.
16. # Format
17. # -----
18. # lazega is an igraph graph object, undirected. It has the following vertex
19. # attributes:
20. # 'name',
21. # 'Seniority',
22. # 'Status' (all 1, meaning partner),
23. # 'Gender' (1 is man, 2 is woman),
24. # 'Office' (1 is Boston, 2 is Hartford, 3 is Providence),
25. # 'Years' (years with the firm),
26. # 'Age', 'Practice' (1 is litigation, 2 is corporate), and
27. # 'School' (1 is Harvard or Yale, 2 is University of Connecticut, 3 is other). See
28. # the reference below for more.
29. # elist.lazega is a data frame containing an edge list of the network.
30. # v.attr.lazega is a data frame containing the vertex attributes only.
31.
32. install.packages("igraph")
33. install.packages("sand")
34. install.packages("Matrix")
35.
36. library(igraph)
37. library(sand)
38. library(Matrix)
39.
40.
41. elist.lazega
42. v.attr.lazega
43.
44. g.lazega <- graph_from_data_frame(elist.lazega,
45.                                   directed = "FALSE",
46.                                   vertices = v.attr.lazega)
47.
48. g.lazega$name <- "Lazega Lawyers"
49.
50. # Statistical overview
51. vcount(g.lazega)
52. # > vcount(g.lazega)
53. # [1] 36
54.
55. ecount(g.lazega)
56. # > ecount(g.lazega)
57. # [1] 115
58.
59. vertex_attr_names(g.lazega)
60. # > vertex_attr_names(g.lazega)
61. # [1] "name"      "Seniority" "Status"    "Gender"    "Office"    "Years"
62. # [7] "Age"      "Practice"  "School"
63.
```

```

64. # Graph preparation
65. adj.lazega <- as_adjacency_matrix(g.lazega)
66.
67. image(Matrix(adj.lazega))
68.
69. g.lazega.adj <- graph_from_adjacency_matrix(adj.lazega)
70. g.edgelist <- graph_from_edgelist(g.lazega.adj)
71.
72. g.lazega2 <- graph_from_data_frame(g.edgelist,
73.                                   directed = "FALSE",
74.                                   vertices = v.attr.lazega)
75.
76. # Graph visualization
77. # -----
78.
79. data(lazega)
80.
81. # Node color, shape and size
82. colbar <- c("red", "dodgerblue", "goldenrod")
83. v.colors <- colbar[V(lazega)$Office]
84. v.shapes <- c("circle", "square")[V(lazega)$Practice]
85. v.size <- 3.5*sqrt(V(lazega)$Years)
86.
87. # Plot
88. set.seed(42)
89. l <- layout.fruchterman.reingold(lazega) # force-directed layout
90.
91. igraph_options(vertex.color = v.colors,
92.                vertex.shape = v.shapes,
93.                vertex.size = v.size,
94.                vertex.label = lazega$name)
95. plot(lazega, layout = l)
96. title("Lazega Lawyers")
97.

```

B.2 Spracovanie datasetu Game of Thrones v 2D (Python)

Načítanie grafu a úprava vstupných dát:

```

1. import igraph as ig
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. data_files = ["GoT/book1.csv", "GoT/book2.csv", "GoT/book3.csv", "GoT/book4.csv",
6.               "GoT/book5.csv"]
7. data = pd.concat((pd.read_csv(filename) for filename in data_files))
8. names = pd.Categorical(data[["Source", "Target"]].values.ravel())
9.
10. vertex_names = pd.DataFrame()
11. vertex_names["name"] = names
12. vertex_names["code"] = names.codes
13. vertex_dict = pd.Series(vertex_names["code"].values,
14.                          index=vertex_names["name"]).to_dict()
15. data["SourceId"] = data["Source"].map(vertex_dict)
16. data["TargetId"] = data["Target"].map(vertex_dict)
17. data_graph = data[["SourceId", "TargetId", "weight", "book"]]
18.
19. g = ig.Graph.DataFrame(data_graph)
20. g.summary()
21.
22. 'IGRAPH D-W- 796 3909 -- \n+ attr: book (e), weight (e)'

```

Základná vizualizácia grafu:

```

1. %%time
2.

```

```

3. # Graph plot
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. g.vs["label_size"] = 2
7. g.vs["color"] = "gray"
8. ig.plot(g,
9.         target=ax,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,
14.        edge_arrow_width=2,
15.        )
16. plt.show()

```

Algoritmus Fruchterman-Reingold:

```

1. %%time
2.
3. # Graph plot Fruchterman-Reingold
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("fruchterman_reingold")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,
14.        edge_arrow_width=2,
15.        )
16. plt.show()

```

Algoritmus Kamada-Kawai:

```

1. %%time
2.
3. # Graph plot Kamada-Kawai
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("kamada_kawai")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,
14.        edge_arrow_width=2,
15.        )
16. plt.show()

```

B.3 Spracovanie datasetu Witcher v 2D (Python)

Načítanie grafu a úprava vstupných dát:

```

1. import igraph as ig
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. data = pd.read_csv("witcher_network.csv")
6. names = pd.Categorical(data[["Source", "Target"]].values.ravel())
7.

```

```

8. vertex_names = pd.DataFrame()
9. vertex_names["name"] = names
10. vertex_names["code"] = names.codes
11. vertex_dict = pd.Series(vertex_names["code"].values,
                           index = vertex_names["name"]).to_dict()
12. data["SourceId"] = data["Source"].map(vertex_dict)
13. data["TargetId"] = data["Target"].map(vertex_dict)
14. data_graph = data[["SourceId", "TargetId", "Weight", "book"]]
15.
16. g = ig.Graph.DataFrame(data_graph)
17. g.summary()

```

IGRAPH D--- 224 2600 -- \n+ attr: Weight (e), book (e)'

Základná vizualizácia grafu:

```

1. %%time
2.
3. # Graph plot
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. g.vs["label_size"] = 2
7. g.vs["color"] = "gray"
8. ig.plot(g,
9.         target=ax,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,
14.        edge_arrow_width=2,
15.        )
16. plt.show()

```

Algoritmus Fruchterman-Reingold:

```

1. %%time
2.
3. # Graph plot Fruchterman-Reingold
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("fruchterman_reingold")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,
14.        edge_arrow_width=2,
15.        )
16. plt.show()

```

Algoritmus Kamada-Kawai:

```

1. %%time
2.
3. # Graph plot Kamada-Kawai
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("kamada_kawai")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,

```

```

14.         edge_arrow_width=2,
15.     )
16. plt.show()

```

B.4 Spracovanie datasetu Lazega v 2D (Python)

Načítanie grafu:

```

1. import igraph as ig
2. import matplotlib.pyplot as plt
3.
4. g = ig.read("lazega.gml")
5. g.summary()

```

```
'IGRAPH UN-- 36 115 -- \n+ attr: Age (v), Gender (v), Office (v), Practice (v), School (v), Seniority (v), Status (v), Years (v), id (v), name (v)'
```

Základná vizualizácia grafu:

```

1. %%time
2.
3. # Graph plot
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. g.vs["label_size"] = 2
7. g.vs["color"] = "gray"
8. ig.plot(g,
9.         target=ax,
10.        vertex_label=g.vs["name"],
11.        vertex_label_size=0,
12.        vertex_size=5,
13.        edge_width=0.5,
14.    )
15. plt.show()

```

Algoritmus Fruchterman-Reingold:

```

1. %%time
2.
3. # Graph plot Fruchterman-Reingold
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("fruchterman_reingold")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label=g.vs["name"],
11.        vertex_label_size=0,
12.        vertex_size=5,
13.        edge_width=0.5,
14.    )
15. plt.show()

```

Algoritmus Kamada-Kawai:

```

1. %%time
2.
3. # Graph plot Kamada-Kawai
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("kamada_kawai")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,

```

```

10.         vertex_label=g.vs["name"],
11.         vertex_label_size=0,
12.         vertex_size=5,
13.         edge_width=.5,
14.     )
15. plt.show()

```

B.5 Spracovanie datasetu Network Science v 2D (Python)

Načítanie grafu:

```

1. import igraph as ig
2. import matplotlib.pyplot as plt
3.
4. g = ig.read("netscience.gml")
5. g.summary()

```

'IGRAPH U--- 1589 2742 -- \n+ attr: id (v), label (v), value (e)'

Základná vizualizácia grafu:

```

1. %%time
2.
3. # Graph plot
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. g.vs["label_size"] = 2
7. g.vs["color"] = "gray"
8. ig.plot(g,
9.         target=ax,
10.        vertex_label=g.vs["label"],
11.        vertex_label_size=0,
12.        vertex_size=5,
13.        edge_width=0.5,
14.    )
15. plt.show()

```

Algoritmus Fruchterman-Reingold:

```

1. %%time
2.
3. # Graph plot Fruchterman-Reingold
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("fruchterman_reingold")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label=g.vs["label"],
11.        vertex_label_size=0,
12.        vertex_size=5,
13.        edge_width=0.5,
14.    )
15. plt.show()

```

Algoritmus Kamada-Kawai:

```

1. %%time
2.
3. # Graph plot Kamada-Kawai
4. fig, ax = plt.subplots(figsize=(15,15))
5.

```

```

6. layout = g.layout("kamada_kawai")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.        vertex_label=g.vs["label"],
11.        vertex_label_size=0,
12.        vertex_size=5,
13.        edge_width=.5,
14.        )
15. plt.show()

```

B.6 Spracovanie datasetu Pokec v 2D (Python)

Načítanie grafu:

```

1. import igraph as ig
2. import matplotlib.pyplot as plt
3. import pandas as pd
4.
5. data = pd.read_csv("soc-pokec-relationships.txt", sep="\t", header=None)
6. data.rename(columns={0: "SourceId", 1: "TargetId"}, inplace=True)

```

```

1. maing = ig.Graph.DataFrame(data)
2. maing.summary()

```

```
'IGRAPH D--- 1632804 30622564 -- '
```

```

1. vs = maing.vs.select(range(5000))
2. g = maing.subgraph(vs)
3. g.summary()

```

```
'IGRAPH D--- 5000 52182 -- '
```

Základná vizualizácia grafu:

```

1. %%time
2.
3. # Graph plot
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. g.vs["label_size"] = 2
7. g.vs["color"] = "gray"
8. ig.plot(g,
9.         target=ax,
10.        vertex_label_size=0,
11.        vertex_size=5,
12.        edge_width=0.1,
13.        edge_arrow_size=5,
14.        edge_arrow_width=2,
15.        )
16. plt.show()

```

Algoritmus Fruchterman-Reingold:

```

1. %%time
2.
3. # Graph plot Fruchterman-Reingold
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("fruchterman_reingold")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,

```

```
10.         vertex_label_size=0,
11.         vertex_size=5,
12.         edge_width=0.1,
13.         edge_arrow_size=5,
14.         edge_arrow_width=2,
15.     )
16. plt.show()
```

Algoritmus Kamada-Kawai:

```
1. %%time
2.
3. # Graph plot Kamada-Kawai
4. fig, ax = plt.subplots(figsize=(15,15))
5.
6. layout = g.layout("kamada_kawai")
7. ig.plot(g,
8.         target=ax,
9.         layout=layout,
10.         vertex_label_size=0,
11.         vertex_size=5,
12.         edge_width=0.1,
13.         edge_arrow_size=5,
14.         edge_arrow_width=2,
15.     )
16. plt.show()
```