

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-182905-74343

Bc. Peter Vašek

Využitie dynamického zhľukovania založeného na hustote v bioinformatike

Diplomová práca

Študijný program: Inteligentné softvérové systémy
Študijný odbor: 9.2.5 Softvérové inžinierstvo
Miesto vypracovania: Ústav informatiky, informačných systémov a
softvérového inžinierstva, FIIT STU, Bratislava
Vedúci práce: Mgr. Peter Laurinec, PhD.

Dátum: Apríl 2019

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný odbor: Inteligentné softvérové systémy

Autor: Bc. Peter Vašek

Diplomová práca: Využitie dynamického zhlukovania založeného na hustote v bioinformatike

Vedúci diplomovej práce: Mgr. Peter Laurinec, PhD.

apríl, 2019

Pokroky v spôsobe merania vlastností buniek zaraďili bioinformatiku medzi vedné oblasti spracúvajúce veľké objemy dát. Jedným z hlavných spôsobov spracovania dát je analýza zhlukov. Ide o metódu vytvárajúcu v zvolenej množine podmnožiny podobných elementov. Medzi metódy zhlukovania patrí zhlukovanie založené na hustote, ktoré analyzuje priestor a na základe hustoty jeho elementov.

Diplomová práca sa venuje možnostiam využitia zhlukovania založeného na hustote v oblasti cytometrických dát. Sú to dáta popisujúce vlastnosti buniek a často sa reprezentujú ako vysoko dimenzionálne body v n-rozmernom priestore. Práca sa pri ich analýze snaží o efektívnu variáciu algoritmu DB-SCAN založenú na mriežke, použiteľnú pre vysoko dimenzionálne dáta. Tiež rieši možnosť dynamického zhlukovania, ktoré umožňuje udržiavať existujúci model stavu pacienta miesto neustáleho opakovania výpočtovo náročného zhlukovania na objemných dátach. Okrem stavu doménovej oblasti analyzuje aj možnosti parallelizácie výsledného riešenia kvôli možnosti distribuovaného počítania algoritmu. Prednosťou takéhoto algoritmu bude odolnosť voči chybám v dátach, ktoré sa pri meraniach nezriedkavo vyskytujú.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Intelligent Software Systems

Author:Bc. Peter Vašek

Master's Thesis: Density Based Dynamic Clustering in Bioinformatics

Supervisor: Mgr. Peter Laurinec, PhD.

2019, April

Advancements in methods of cell measurement shifted bioinformatics among science fields processing big data. One of the most conventional ways of data processing is cluster analysis. It is a method, that transforms chosen set to a list of subsets with similar elements. Cluster analysis methods includes density based cluster analysis which analyzes density of its elements in space. Diploma thesis is aimed at possibilities of usage of density based clustering in processing of cytometric data. Those are data describing characteristics of cells usually represented as points in n-dimensional space. It is seeking effective variation of grid based DBSCAN algorithm that can be used even with high dimensional data. Thesis also considers option of dynamic clustering, allowing us to hold one model of patient status and enriching it with input instead of repeating computations everytime data changes. Work analyse both actual state of area and possibilities of parallelisation of solution. Advantage of such algorithm would be tolerance towards errors in input data, which can happen.

Obsah

1	Úvod	1
2	Opis problémovej oblasti	3
2.1	Bioinformatika	3
2.2	Cytometria	3
2.2.1	Gén	5
2.2.2	Proteín	5
2.2.3	Aminokyselina	5
2.2.4	Genotyp	5
2.2.5	Fenotyp	5
2.2.6	Expresivita génov	5
2.3	Analýza zhlukov	6
2.4	Gating	7
2.4.1	Automatický Gating	7
2.4.2	SPADE	7
2.4.3	t-SNE	8
2.4.4	Wanderlust	9
2.4.5	Citrus	10
2.4.6	PhenoGraph	10
2.4.7	FlowSOM	10
2.4.8	FLOCK	11
2.5	Algoritmy analýzy zhlukov	12
2.5.1	K-Means	12
2.5.2	Hierarchické zhlukovanie	14
2.5.3	DBSCAN	15
2.5.4	ρ - approximate DBSCAN	19
2.5.5	Dynamický DBSCAN	21
2.5.6	GDPAM	24
2.5.7	OPTICS	25
2.5.8	Zhodnotenie popísaných algoritmov	25
3	Návrh riešenia	27
3.1	Špecifikácia požiadaviek	27

4 Opis riešenia	29
4.1 Popis vstupných dát	29
4.2 Implementácia	30
4.2.1 Štruktúra programu	30
4.2.2 Vlastnosti programu	32
4.2.3 Popis výstupných dát	34
4.3 Paralelizovateľnosť výpočtov	34
4.4 Dynamickosť výpočtov	36
4.5 Dátové štruktúry	36
4.5.1 Union-Find	36
4.5.2 PCA	37
4.5.3 Prehľadávanie stromu do šírky	37
4.6 Validácia výsledkov	37
4.6.1 Rand index	38
4.6.2 Fowlkes-Mallows index	38
4.6.3 Jaccard index	39
5 Vyhodnotenie	40
5.1 Testovacie dátá	40
5.2 Testovacia konfigurácia	41
5.3 Výsledky zhlukovania	42
5.4 Paralelizácia riešenia	44
5.5 Dynamické zhlukovanie	44
5.6 Odhadovanie parametrov pre zhlukovanie	46
5.7 Vizualizácia výsledných zhlukov	49
6 Zhodnotenie	57
6.1 Ďalšie možnosti rozvoja	57
7 Technická dokumentácia	59
7.1 Načítanie a analýza fcs dát	60
7.2 Vykonanie zhlukovania	60
7.3 Práca s výsledkami	60
Literatúra	62
A Elektronické médium	67
B Plán práce na riešení projektu	68

Kapitola 1

Úvod

Technický pokrok v dnešnej dobe stále viac tlačí na prepájanie vedných disciplín. Vznikajú nové odvetvia, napríklad bioinformatika, v ktorých sa skúsení odborníci snažia porozumieť, čo vie ponúknuť kolega z druhej strany. Jednou z najväčších výziev vplývajúcich na kvalitu života človeka je porozumenie ľudskému "zdrojovému kódu", DNA.

Témou tejto práce je využíte jednej z metód analýzy zhlukov - zhlukovanie založené na hustote - v doméne cytometrických, teda bunkových dát. Analýza zhlukov nám slúži na automatické vyhodnotenie podobnosti prvkov v množinách. V doméne bioinformatiky sa používa napr. pri pokročilých vyšetreniach pacientov alebo výrobe liekov. Neustále sa zlepšujúce technické vybavenie slúžiace na meranie vlastností buniek produkuje mohutné vysoko dimenzionálne dátové množiny. Aby sme udržali krok pri spracovaní, potrebujeme inovačne používané postupy. Analýza zhlukov založená na hustote je jedna z možných foriem zhlukovania cytometrických dát. Jej výhodou oproti iným zhlukovacím metódam je, že dokáže identifikovať zhluky ľubovoľných tvarov, nepotrebuje dopredu poznať počet hľadaných zhlukov a automaticky odfiltruje chyby a šum v dátach. Nevýhodou je náročné stanovenie hranice požadovanej hustoty.

Štandardným algoritmom v doméne zhlukovania založeného na hustote je DBSCAN 2.5.3. Cieľom tejto práce je vytvoriť na jeho základe riešenie pre analýzu vysoko dimenzionálnych dát použiteľné v oblasti cytometrie. Pri návrhu je potrebné brať do úvahy výpočtovú zložitosť, preferovať zachovanie presnosti výsledkov oproti aproximačným výpočtom, možnosť dynamického spracovania dát bez potreby neustáleho reštartovávania algoritmu pri zmene dát a interpretovanie výsledkov formou vizualizácie. Dôležitá súčasť práce je podrobná analýza aktuálne používaných metód, ich výhod a nevýhod. Porovnanie algoritmov nie je priamočiare, v doménovej oblasti máme viačero typov úloh. Niekoľko odborníci snažia identifikovať malú populáciu vzácnych buniek, inokedy zase identifikovať každého účastníka vzorky.

KAPITOLA 1. ÚVOD

Prehľad práce: kapitola číslo dva obšírne skúma doménovú oblasť. Venuje sa ako základným pojmom bioinformatiky, tak metódam tzv. gatingu vykonávaného biológmi pri spracovaní bunkových dát a algoritmom používaným pri analýze zhľukov. Tretia kapitola obsahuje špecifikáciu požiadaviek a návrh na výsledok riešenia práce. V štvrtnej kapitole opisujeme riešenie od vstupných dát, použitých dátových štruktúr cez implementáciu až po možnosti validácie výsledkov. Piata kapitola pokrýva oblasť výhodnotenia. Opisuje obsah a formu testovacích dát, kategórie testovania, dosiahnuté výsledky a ukážky z vizualizácií. V siedmej kapitole je krátke zhodnotenie vykonanej práce a námety na ďalší rozvoj. V siedmej kapitole je technická dokumentácia artefaktov projektu nasledovaná zoznamom použitých zdrojov. V prílohách sa nachádza elektronický nosič a plán práce riešenia projektu spolu s jeho výhodnotením.

Kapitola 2

Opis problémovej oblasti

2.1 Bioinformatika

Bioinformatika, mladý odbor prepojujúci informatiku s biológiou (predovšetkým molekulárnu), sa explicitne objavila na svetovej scéne v 90-tych rokoch 20. storočia [6]. Zaoberá sa navrhovaním a aplikovaním metód na spracovanie veľkých objemov biomedicínskych dát, ktoré sa začali hromadiť vďaka dostupnosti veľkokapacitných databáz a príslušnej infraštruktúry. Významným miľníkom bolo pre zber takýchto dát rozšírenie internetu, vďaka ktorému je možné centrálnie zhromažďovať dátá z veľkého množstva zdrojov roztrúsených kdekoľvek po svete.

2.2 Cytometria

Pojem cytometria označuje proces merania fyzických a chemických charakteristík biologických buniek [35]. Od ich objavenia v 17. storočí až do tridsiatych rokov 20. storočia ľudia nemali prístroje ktoré by umožňovali viac, než len jednoduché vizuálne pozorovania prostredníctvom postupne sa vyvíajúcich optických zariadení. Skutočné začiatky tejto disciplíny v prvej polovici 20. storočia prišli s elektrónovými mikroskopmi a v sedemdesiatych rokoch 20. storočia s jednoduchými triedičmi buniek (*angl. cell sorters*) umožňujúcimi mechanicky izolovať pre ďalšie štúdium zo vzorky bunky s požadovaným vlastnosťami [36]. V sedemdesiatych rokoch už boli tieto nástroje komerčne dostupné, aj keď počet organizácií, ktoré si ich mohli dovoliť, bol spočiatku nízky. Postupné zavádzanie novších, energeticky menej náročných postupov a zlepšujúce sa možnosti počítacového spracovania výsledkov viedli ku dnešnému hromadnému využívaniu analýzy bunkových dát. Základné úlohy cytometrie sú:

- Určiť, či sa vo vzorke nachádza hľadaný typ bunky.
- Určiť početnosť hľadaného typu buniek vo vzorke.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

- Určiť typ sledovanej bunky.
- Určiť funkčné charakteristiky sledovanej bunky

Napriek tomu, že v dnešnej dobe dokážeme sledovať aj elektrické či akustické vlastnosti buniek, alebo ich radiáciu, na splnenie vyššie uvedených úloh sa najčastejšie používa optické meranie a o väčšine cytometrov sa dá uvažovať ako o špecializovaných mikroskopoch.

Cytometria ako praktická disciplína sa uplatňuje napr. v imunológií, kde sa pomocou nej sledujú počty T-lymfcytov u ľudí napadnutých HIV [2], či aktivita tumoru u pacientov s diagnostikovanou rakovinou [24].

Parameter, ktorý cytometria u bunky sleduje, môže byť fyzický (veľkosť bunky, tvar, svetelný rozptyl a pod), chemický (obsah DNA, výskyt proteínov, sekvencia nukleotidov a pod). Parametre sa delia na vonkajšie a vnútorné, na základe toho, či ku ich sledovaniu je potrebné použiť činidlo (*angl. reagent*), ktoré sa v žargóne cytometrie označuje ako sonda (*angl. probe*) [35]. Tiež sa delia na štrukturálne a funkcionálne. Najvýznamnejšie spôsoby merania vlastností buniek v cytometrií sú:

- Prietoková cytometria (*angl. Flow Cytometry*) [35] - Merané bunky prichádzajú do meracieho zariadenia v prúde, často sa v ňom na základe sledovaného atribútu rovno separujú. Metóda sa používa prevažne v imunológií a hematopatológií. Klúčové atribúty meraní sú rýchlosť analýzy, citlivosť merania a počet bunkových parametrov snímateľných súčasne.
- Mikrospektrofotometria (*angl. Micro-spectrophotometry*) - Spojenie mikroskopu a spektrometra, umožňuje merať optické spektrum mikroskopických objektov. Výhodou je, že meranie nepoškodzuje/neovplyvňuje vzorku, čo je pri sledovaní biologických vzoriek dôležité.
- Skenovacia cytometria (*angl. Scanning Cytometry*) [29] - Skenovanie fluorescencie (umelo vyvolanej podľa typu merania) buniek pomocou laseru prebieha veľkou rýchlosťou (až 5000 buniek/min) pri zachovaní vysokej citlivosti a presnosti merania.
- Hmotnostná cytometria (*angl. Mass Cytometry*) [26] - Metóda je založená na hmotnostnej spektrometrií a umožňuje pomocou rôznych stabilných izotopov značkovať viacero sledovaných prvkov v bunke (napr. prítomnosť špecifických proteínov). Pre každú bunku je na rozdiel od prietokovej cytometrie možné odmerať desiatky parametrov (prítomných proteínov). Výstupom merania buniek je dátový súbor vo formáte fcs (z *angl. flow cytometry standard*), ktorý je opísaný v kapitole 4.1. Nevýhodami metódy sú pomalá rýchlosť spracovania biologickej vzorky a vyššie náklady na prístroje.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

2.2.1 Gén

Pojem má dva významy. Prvý je všeobecné označenie dedičnej vlohy, jednotka informácie o vlastnosti organizmu. Druhým významom je označenie úseku DNA so špecifickou funkciou. Rozlišujeme gény štrukturálne, regulačné, enzýmy produkujúce a pod [15].

2.2.2 Proteín

Proteín, odborný termín pre bielkovinu [19], je označenie pre refaz aminokyselín. Aby sa sekvencia aminokyselín mohla označiť za bielkovinu, musí sa skladať zo 100 a viac aminokyselín (2-10 je oligopeptid, 11-100 polypeptid). Poradie aminokyselín v refazci (primárna štruktúra) určuje chemické vlastnosti bielkoviny.

2.2.3 Aminokyselina

Za aminokyselinu sa v chémii považuje akákoľvek molekula obsahujúca karboxylovú skupinu ($-COOH$) a aminoskupinu ($-NH_2$). Delia sa na proteinogénne/neproteinogénne, esenciálne/neesenciálne a ketogénne/glukogénne [19]. V molekulárnej biológií sa týmto pojmom označujú proteinogénne (bielkovinotvorné, pri translácií sa zabudovávajú do bielkovinových reťazcov) α -L-aminokyseliny, ktorých je 24, a považujú sa za základné stavebné zložky všetkých proteínov.

2.2.4 Genotyp

Genotyp je súbor génov vo forme DNA definujúcich všetky zdedené vlohy jedinca [19]. Zahŕňa aj všetky variácie a polymorfizmy, ktorými sa jedinci v daných génoch líšia.

2.2.5 Fenotyp

Vonkajším prejavom, viditeľným či merateľným, genotypu je fenotyp. Fenotyp je okrem genotypu organizmu určený ešte prostredí, v ktorom prebieha vývoj organizmu [19].

2.2.6 Expresivita génov

O tom, či sa u jedinca prejaví formou fenotypu nejaký jeho genotyp rozhodujú faktory penetrácia a expresivita génu. Pri penetrácii sa určuje pravdepodobnosť, že sa gén vôbec prejaví (napr. pri dominantnom géne sa môže pravdepodobnosť rovnať 1, počíta sa ako pomer jedincov v populácii s daným genotypom ku počtu jedincov v populácii s daným fenotypom), pri expresivite sa určuje sila, s akou sa prejaví [18]. Expresivitu génu môžu

ovplyvňovať rôzne modifikátory, potlačovače (*angl. suppressors*), či epistáza (aktivitu daného génu prekrýva aktivita iného génu).

2.3 Analýza zhlukov

Úloha analýzy zhlukov spočíva v identifikovaní vnútornej štruktúry dátovej množiny bez akýchkoľvek predchádzajúcich doménových znalostí. Vytvárajú sa zhluky - skupiny elementov, ktoré sú si vzájomne podobné. Zhluky by sa mali vzájomne čo najviac odlišovať a dohromady reprezentovať všetky kategórie zúčastnených vstupných elementov. Podľa toho, ako definujeme zhluk, poznáme niekoľko druhov analýzy zhlukov:

- Modely s použitím ťažiska (*angl. Centroid models*) - každý zhluk má stanovené centrum (či už pomyselné súradnice, alebo niektorý z bodov zhluku) a obsahuje všetky body, ktoré majú k danému centru bližšie než k centrám ostatných zhlukov. Príkladom je K-means (kapitola 2.5.1).
- Modely založené na pravdepodobnosti (*angl. Distribution models*) - prvky sú do zhlukov zatriedené podľa pravdepodobnostného priestorového rozdelenia, napr. zmes viacozmerných normálnych rozdelení (*angl. Gaussian mixture models*).
- Hierarchické modely (*angl. Connectivity-based model*) - buduje sa graf susednosti elementov vstupnej dátovej množiny - tzv. dendrogram (kapitola 2.5.2). Model sa vytvára buď delením celej vstupnej množiny na podmnožiny až po úroveň jednotlivých prvkov, alebo naopak, spájaním od úrovne jednotlivých prvkov až po zhluk obsahujúci celú vstupnú množinu.
- Modely založené na hustote (*angl. Density models*) - stanový sa hranica hustoty, od ktorej sa priestor obsahujúci prvky vstupnej množiny považuje za zhluk. Model automaticky identifikuje vopred neznámy počet zhlukov ľubovoľného tvaru a odfiltruje šum z medzizhlukových priestorov. Hlavným predstaviteľom množiny týchto algoritmov je DBSCAN (kapitola 2.5.3).
- Samo organizujúce sa mapy [20] (*angl. Self Organizing Maps*) - metóda použitím neurónovej siete s učením bez učiteľa (*angl. Unsupervised learning*) mapuje vysoko dimenzionálne údaje na dvoj, prípadne troj-rozmernú vizualizovateľnú mriežku zhlukov. V režime učenia sa mapa postupne prijíma elementy vstupnej dátovej množiny, meria Euklidovu vzdialenosť ku jednotlivým zhlukom v mriežke a upravuje váhy svojich zhlukov zmenou ich pozície smerom ku vstupnému umiestneniu spracovávaného bodu. Bližšie zhluky sa posúvajú viac, čím ďalej sa zhluk

nachádza od vstupného elementu, tým menej je ovplyvnený. Po spracovaní celej trénovacej množiny už ďalšie vstupné elementy neupravujú pozície zhlukov, miesto toho sa klasifikujú pomocou najbližšieho zhluku v mape.

Pri niektorých algoritnoch sa môžu prvky prideľovať do množín s určitou pravdepodobnosťou, prípadne je prípustné, aby patrili do viacerých zhlukov súčasne - skupina fuzzy zhlukovacích algoritmov.

2.4 Gating

Identifikácia a oddelenie populácií buniek s rovnakým fenotypom, tzv. gating [35], sa v cytometrií používa vo fáze predspracovania dát. "Brána" (*angl. gate*) predstavuje množinu obmedzení, ktorá buď odfiltruje všetky bunky ktoré obmedzenia spĺňajú (exkluzívna brána) alebo tie, ktoré obmedzenia nespĺňajú (inkluzívna brána). Z vedeckého hľadiska ide o dôležitý krok, ktorý vplýva na kvalitu výsledku bunkovej analýzy. Nedeterminizmus výsledkov gatingu prináša hlavne subjektivita pracovníka rozhodujúceho na základe svojich vedomostí o tom, ktoré bunky ďalej analyzovať a ktoré z pozorovania vylúčiť. Stanovenie filtrovacích pravidiel sa najčastejšie robí na základe parametrov bunky ako je jej veľkosť, životaschopnosť (reakcia na podnety, mŕtve bunky preč), prípadne markerov (napr. tagov na sledovanom proteíne) vlastnosti ktorú chceme v analýze sledovať. Najjednoduchšia verzia gatingu spracúva len jeden parameter, je ale možné ohraničovať dátu aj na dvojrozmernom histograme (4 kvadranty, vlastnosti buniek + +, - +, - -, + -), prípadne na viacrozmernom, kde ale komplexita rýchlo rastie. Okrem manuálneho gatingu sa pochopiteľne rozvíjajú aj automatické postupy, kde subjektívneho a pomalého človeka pri rozhodovaní nahradza počítač.

2.4.1 Automatický Gating

Problémom existujúcich algoritmov na automatický gating populácií buniek je, že v súčasnosti nedosahujú požadovanú úroveň presnosti [31]. Malé populácie vzácných buniek, napr. kmeňových, môžu byť vyraodené ako šum, prípadne zlúčené s iným, väčším zhlukom. Všeobecne sa dá povedať, že metódy automatického gatingu podliehajú klasickým problémom vyplývajúcim z metód zhlukovania, na ktorých sú založené.

2.4.2 SPADE

SPADE (*angl. spanning-tree progression analysis for density-normalized events*) patrí ku základným algoritmom používaným pri automatickom gatingu, predovšetkým pri prietokovej a hmotnostnej cytometrií. Využíva kombináciu

zredukovania dát vzorkovaním (*angl. downsampling*), zhlukovania a použitia minimálnej kostry grafu na poskytnutie 2D vizualizácie aj viacdimenziólnych bunkových dát. V základnej verzií je nedeterministický a viacnásobné spustenie nad rovnakými dátami nedá zakaždým rovnaký výsledok. V článku [28] autor navrhuje nahradie aktuálne používaný algoritmus na downsampling deterministickým, na hustote založeným algoritmom a aktuálne používaný zhlukovací algoritmus deterministickým k-means algoritmom [38]. Odlišnosť jednotlivých behov klasického k-means algoritmu je spôsobená náhodným zvolením inicializačných súradníc centier zhlukov. V práci je navrhnutý algoritmus založený na metóde PCA (popísanej v kapitole 4.5.2), ktorý inicializuje centrá zhlukov pre každý dataset vždy rovnako.

Postup algoritmu SPADE, ktorého vstupom sú dáta viacerých skenovaných populácií buniek s vyznačenými proteínovými markermi:

1. Downsampling založený na hustote pre každú populáciu zvlášť, aby sa z každej získalo približne rovnako veľa buniek. Pre každú bunku sa vypočíta lokálna hustota podľa počtu buniek v najbližšom okolí a s určitou pravdepodobnosťou sa každá z nich môže odstrániť (zdroj nedeterminizmu).
2. Spojenie všetkých takto získaných buniek do jednej množiny.
3. Zhlukovanie do zámerne väčšieho počtu zhlukov než sa očakáva počet populácií buniek vo vzorke (typicky 100-300 zhlukov). Účelom tohto kroku je dosiahnuť, aby každá subpopulácia bola rozdená do viac než jedného zhluku, čo zároveň znamená, že v každom zhluku očakávame len populáciu jedného typu. Ako zhlukovací algoritmus sa používa obyčajný k-means (nedeterministický) alebo urýchlená varianta aglomeratívneho hierarchického zhlukovania (ktorá tiež obsahuje prvok náhody).
4. Zostrojenie minimálnej kostry grafu zo vzniknutých zhlukov.
5. Manuálne pozorovanie výsledkov odborníkom. Vytvára sa zobrazenie grafu vo viacerých verziách, vždy s bunkami zafarbenými inak, na základe iného markera. V prípade hmotnostnej spektrometrie, ktorá má typicky významne viac sledovaných proteínov než prietoková, sú takýchto grafov desiatky. Interpretácia výsledkov je jednoduchšia ako pri manuálnom gatingu, stále ale nie je plne automatická.

2.4.3 t-SNE

Technika t-SNE (*angl. t-Distributed Stochastic Neighbor Embedding*) [40] patrí medzi metódy nelineárnej redukcie dimenzionality dát za účelom vizualizácie. Princípom je namapovanie multidimenzionálnych objektov do dvoj,

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

alebo trojrozmerného priestoru tak, že objekty ktoré sú si podobné budú vo výsledku s vyššou pravdepodobnosťou blízko seba. Postup algoritmu:

1. Vypočíta sa pravdepodobnosť podobnosti všetkých dvojíc multidimensionálnych objektov.
2. Vytvorí sa dvoj(troj)rozmerná mapa do ktorej sa umiestnia body na základe pravdepodobnosti susednosti z kroku 1.
3. Pomocou Kullback-Leiblerovej divergencie [22] sa minimalizuje rozdiel medzi nimi.

Na určenie podobnosti dvojíc bodov počas prvého kroku algoritmu sa okrem Euklidovskej vzdialenosťi môžu použiť aj iné metriky vzdialenosťí. Jednou z distribuovaných implementácií algoritmu je viSNE [3].

2.4.4 Wanderlust

Na rozdiel od doteraz spomenutých vizualizačných algoritmov Wanderlust [3] berie do úvahy "vývojovú trajektóriu" populácie buniek, čo otvára možnosti pre ďalšie prípady použitia. Napr. premena medzi rôznymi štádiami života bunky bude zachytená ako úbytok markerov naviazaných na bunky v prvom štádiu a zvýšenie počtu markerov naviazaných na bunky v nasledujúcim štádiu. Problémom takéhoto spracovania dát je predovšetkým štatistický šum (či už technického pôvodu alebo spôsobený nepresnosťou biologickej vzorky) ktorý môže spôsobiť tzv. skrat (*angl. short-circuit*) - párs buniek je spojený na základe priestorovej blízkosti, aj keď vo vývojovej fáze sú štádiá buniek ďaleko od seba. Ďalším problémom je odstránenie vzácných malých populácií buniek ako šumu, pretože sa tak stráca vývojový krok a trajektória nie je presná. Postup algoritmu:

1. Vytvorí sa kNN graf, každá bunka predstavuje uzol.
2. Náhodne sa zvolí množina štartovacích buniek, pre každú sa vytvorí 1-k-NNG (1-out-of-k-nearest-neighbor graph).
3. Pre každú bunku v každom grafe sa hľadá trajektória (najpodobnejší sused, váha podľa priestorovej vzdialenosťi).
4. Po ukončení tvorby trajektórie vo všetkých grafoch sa vypočíta priemerná a stanový sa ako trajektória celej vzorky.

2.4.5 Citrus

Citrus (*angl. cluster identification, characterization, and regression*) [8] je nástroj pre automatizáciu gatingu vyžadujúci od používateľa len definovanie hľadaných vlastností cieľovej množiny buniek. Postup:

1. Kvôli šetreniu výpočtovým časom sa náhodne vyberie používateľom definovaný počet buniek z každej vzorky.
2. Hierarchickým zhľukovaním sa identifikujú populácie buniek s rovnakými fenotypmi (na základe podobnosti markerov).
3. Vytvoria sa zhľuky podobných buniek, ktoré obsahujú aspoň 5 percent celkového počtu buniek. Bunky môžu patriť aj do viacerých zhľukov súčasne.
4. Vlastnosti každého zhľuku sa počítajú na základe vlastností každého jedného prvku zhľuku.
5. Citrus na základe zadaného štatistického modelu vyhodnotí, ktoré zhľuky korelujú so zadaným modelom a vygeneruje pre používateľa výstup.

2.4.6 PhenoGraph

Algoritmus PhenoGraph [9] je určený pre spracovanie vysoko dimenzionálnych bunkových dát pomocou grafových metód. Na základe Jaccardovho indexu vytvára k-NNG ktorého prvky následne spája Louvainovou metódou pre identifikáciu skupín. Výsledný počet zhľukov s podobnými fenotypmi stanoví automaticky.

2.4.7 FlowSOM

FlowSOM (*angl. SOM = Self-Organizing Map*) [14] je ďalšou z rady metód na zhľukovanie a vizualizáciu dát z prietokovej a hmotnostnej cytometrie. Používa samo organizujúce sa mapy, minimálnu kostru grafu a dvojúrovňové zhľukovanie. Postup:

1. Načítanie vstupných dát a predspracovanie: Okrem možnosti spojenia viacerých vstupných fcs súborov do jednej množiny dát, na základe ktorej sa vytvorí jeden model, sa tiež normalizujú hodnoty v stĺpcach. Po odčítaní priemernej hodnoty daného atribútu (priemerná hodnota v stĺpci bude tým pádom rovná nule) sa predelí hodnota štandardnou odchýlkou atribútu, čo spôsobí celkovú štandardnú odchýlku rovnú jednej.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

2. Vytvorenie samo organizujúcej sa mapy: Mapa sa skladá z k uzlov, pričom každý je definovaný ako d dimenzionálny bod. Na začiatku sa uzly náhodne inicializujú bunkami datasetu. Ako funkcia susednosti sa používa Chebyshevova vzdialenosť. Mapa sa trénuje pomocou pridávania náhodne vybratých buniek vstupnej vzorky a následným aktualizovaním všetkých susedných uzlov.
3. Výsledná mapa sa vizualizuje pomocou minimálnej kostry grafu. Uzly sa spájajú tak, aby suma hmotností jednotlivých vetiev grafu bola čo najmenšia. Vznikne spojený acyklický graf.
4. Zhlukovanie uzlov (buď so zadaným počtom požadovaných zhlukov, alebo na základe tzv. lakťového pravidla).

Silnou stránkou metódy je spracovanie ľubovoľne veľkého počtu dimenzií do dvojrozmerného priestoru, čo je praktické pri vizualizácii výsledkov.

2.4.8 FLOCK

Jedným z prvých algoritmov reagujúcich na zvýšený počet meraných atribútov (spôsobený nástupom pokročilejších meracích zariadení) bol v roku 2011 FLOCK [30] (*angl. FLow Clusteringwithout K*). Autori identifikovali problém s použitím K-means algoritmu (opísaný v kapitole 2.5.1) pri multidiemználnych dátach z dôvodu zvyšujúceho sa rizika upadnutia do lokálneho optima. Ako riešenie navrhli použiť pre časť výpočtu zhlukovanie založené na hustote. Postup algoritmu:

1. Predspracovanie dát: Načítanie .fcs súborov a normalizácia pre každý stĺpec tabuľky, aby boli atribúty rovnocenné.
2. Zhlukovanie založené na hustote:
 - (a) Každá dimenzia v priestore s bodmi sa rozdelí na rovnaký počet úsekov, čím sa vytvorí mriežka s d dimenzionálnymi rovnako veľkými kvádrami
 - (b) Pre každý kváder sa vypočíta hustota na základe počtu bodov ktoré doň spadajú.
 - (c) Podľa zadanej hraničnej hustoty sa vytvoria centrá zhlukov a pripoja sa ku nim všetky susediace kvádre.
 - (d) Určí sa centrum zhluku ako bod s priemernými súradnicami pre prvky daného zhluku.
3. Zhlukovanie s použitím ťažiska: Okolo centier získaných v predošлом kroku na základe hustoty sa vykoná klasický k-means (2.5.1)

Vstupom od používateľa sú zadané dva parametre:

- Počet dielov, na ktoré sa delí každá dimenzia.
- Hraničná hustota pre určenie zhlukov v 2. c) kroku.

Zaujímavosťou FLOCK algoritmu je predovšetkým zrešnenie viacerých metód zhlukovania, ktoré využíva silné stránky jednotlivých metód a potláča ich nedostatky.

2.5 Algoritmy analýzy zhlukov

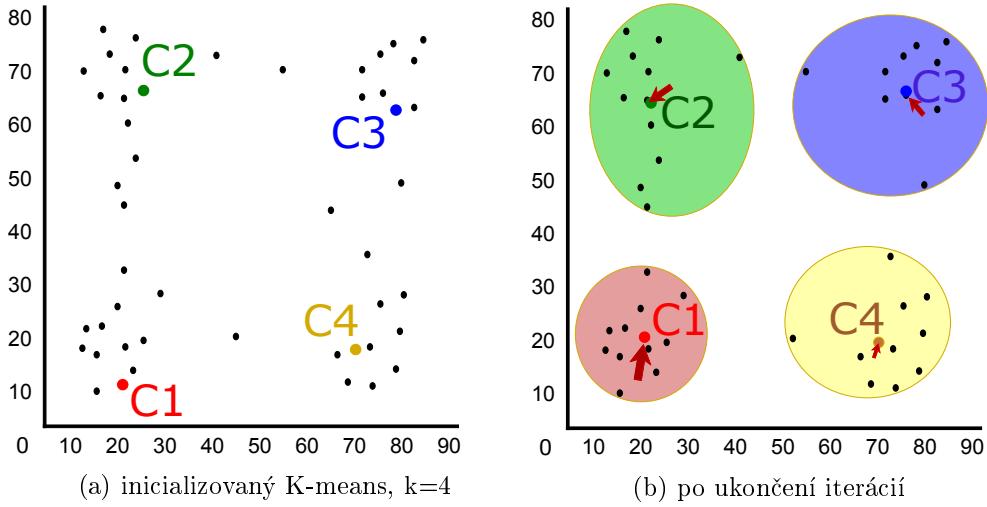
2.5.1 K-Means

Analýza algoritmu K-means je prebratá z analýzy v rámci mojej bakalárskej práce [23].

Jeden z najznámejších predstaviteľov ľažiskových zhlukovacích modelov, algoritmus K-means [25], potrebuje pred použitím poznáť parameter k - počet zhlukov, ktorý je zvolený používateľom. Kroky algoritmu sú nasledovné:

1. Náhodne sa vytvorí k centier zhlukov.
2. Každý prvk sa dátovej množiny sa priradí k preňho najbližšiemu centru. Výsledkom je k zhlukov.
3. Pre každý zhluk sa vypočíta nová pozícia centra ako bod s najmenším súčtom vzdialenosí od prvkov zhluku. Pozícia centra zhluku sa zmení na túto novú hodnotu.
4. Opakovanie od kroku 2 dovtedy, kým sa pozícia centier všetkých zhlukov neprestane meniť medzi iteráciami.

Výsledkom je k disjunktných množín (zhlukov), ktoré dohromady obsahujú všetky prvky vstupnej množiny.



Obr. 2.1: Vytvorenie zhlukov pomocou K-Means

Algoritmus má svoj pôvod v oblasti spracovania signálu a komprezíí dát [5]. Pre vstupnú množinu dát $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ kde prvok $\mathbf{x}_i = (x_1, \dots, x_d)$ a d je počet dimenzií, sa dá minimalizačná funkcia, ktorou sa dosiahne optimálne priradenie prvkov do zhlukov, zapísť ako:

$$J = \arg \min \sum_{i=1}^k \sum_{j=1}^n \|\mathbf{x}_j^{(i)} - \mathbf{c}_i\|^2. \quad (2.1)$$

k označuje počet zhlukov ($k \leq n$), $x_j^{(i)}$ je prvok patriaci do i -teho zhluku a c_i je centrum i -teho zhluku.

V prípade, že je zadaný počet zhlukov k , počet dimenzií prvkov je d a máme n prvkov, pre i iterácií je zložitosť algoritmu $O(kdni)$.

Pre priebeh algoritmu je kľúčové zvoliť správne počet zhlukov. V prípade malého počtu bude vzdialenosť bodov v zhluku od centra príliš veľká, v prípade príliš veľkého počtu zhlukov zase bude posun centra zhluku medzi jednotlivými iteráciami len malý.

Výhodou algoritmu je jeho jednoduchosť a rýchlosť. Nevýhodou:

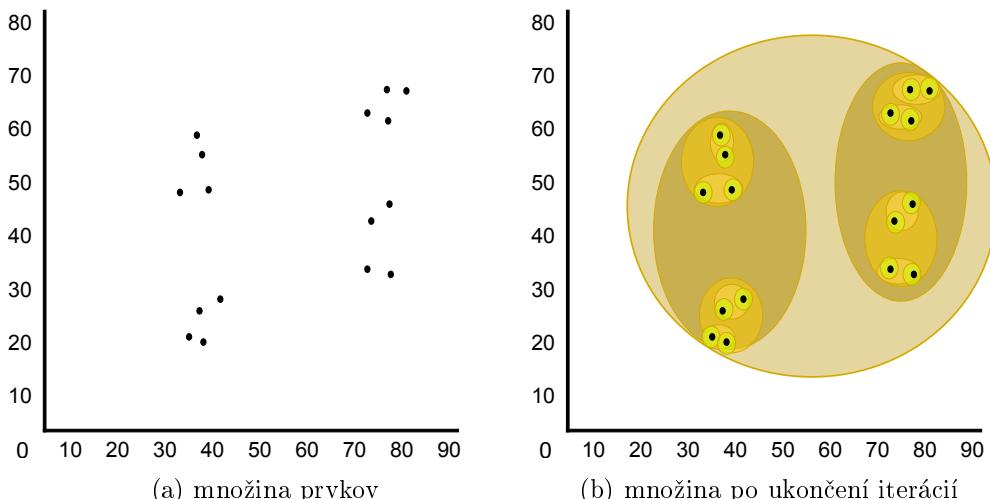
- Vplyv prvkov, ktorých vlastnosti sa výrazne vymykajú priemeru, na kvalitu výsledku.
- Čažkosti pri spracovaní zhlukov, ktoré nemajú sférický tvar.
- Pri výpočte vzdialenosťí prvkov od centra je možné použiť len Euklidovu vzdialenosť.

2.5.2 Hierarchické zhľukovanie

Analýza hierarchického zhľukovania je prebratá z analýzy v rámci mojej bakálarskej práce [23].

Ako názov napovedá, hierarchické zhľukovanie patrí medzi hierarchické modely zhľukovania. Na rozdiel od ostatných modelov, výsledkom nie sú na vzájom disjunktné množiny prvkov, ale hierarchia prekrývajúcich sa zhľukov(množín) výrazne rozdielnej mohutnosti. Čím menšiu mohutnosť zhľuk má, tým podobnejšie prvky sa v ňom nachádzajú. Zhľukovanie sa vykonáva jedným z dvoch spôsobov, deliacim (*angl. Divisive*) alebo zoskupujúcim (*angl. Agglomerative*).

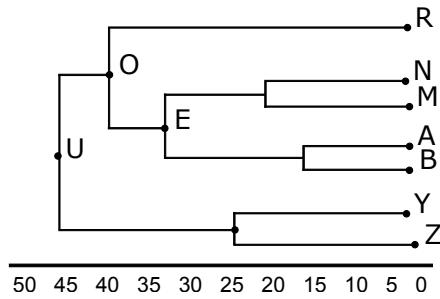
Pri deliacom začneme s jediným zhľukom, ktorý obsahuje všetky vstupné prvky. Problémom je rozhodnúť, ako zhľuk rozdeliť. Jedným z riešení je najst' prvak, ktorý sa najviac odlišuje, vytvoriť mu nový zhľuk, a potom všetky prvky starého zhľuku, podobné viac prvku v novom zhľuku presunúť tam. Postup sa opakuje až kým každý prvak nemá vlastný zhľuk na samom spodku hierarchie. Zložitosť tohto postupu je $O(2^n)$ [16].



Obr. 2.2: Vytvorenie zhľukov pomocou Hierarchického zhľukovania

Pri zoskupujúcim spôsobe začneme s jednotlivými prvkami, ktorým nájdeme najbližšieho suseda. Metrík pre výhodnotenie susednosti je viacero (napr. Euklidovská vzdialenosť, Manhattanská vzdialenosť), ich výber má vplyv na tvar výsledných zhľukov. Spájanie najpodobnejších zhľukov na každej úrovni hierarchie opakujeme, kým nedostaneme jeden zhľuk obsahujúci všetky prvky. Zložitosť je menšia ako pri deliacom spôsobe, $O(n^2 \log(n))$ [16].

Výsledkom oboch spôsobov je binárny strom, ktorý sa zobrazuje pomocou grafu nazývaného dendogram.



Obr. 2.3: Hierarchia zhlukov

Veľkou prednosťou hierarchického zhlukovania je, že nie je potrebné zadávať žiadne vstupné parametre. Nevýhodou je ale pomerne vysoká výpočtová náročnosť ktorá neumožňuje použiť tento model pre ľubovolne veľké vstupy.

2.5.3 DBSCAN

Analýza algoritmu DBSCAN je prebratá z analýzy v rámci mojej bakalárskej práce [23].

Už od svojho uvedenia v roku 1996 patrí DBSCAN [11] (*angl. Density-Based Spatial Clustering of Applications with Noise*) medzi hlavných predstaviteľov zhlukovacích modelov založených na hustote. Oproti iným modelom má niekoľko výhod:

- Nepotrebuje vopred poznať počet zhlukov vo vstupnej množine.
- Automaticky zanedbáva anomálie v dátach.
- Nemá problém so zhlukmi nepravidelných tvarov, ohraničenie zhluku je tvorené nižšou hustotou dát, než aká je potrebná na priradenie do zhluku.

Pred použitím je potrebné algoritmu zadať dva parametre, maximálnu medziprvkovú vzdialenosť ε a minimálny počet susedov prvku $minPts$.

Maximálna medziprvková vzdialenosť určuje vzdialenosť, po ktorú sa dva prvky rátajú ako susedné. Vyplýva z nej ε -okolie bodu, množina bodov susediacich s určitým bodom. Definuje sa ako [11]:

$$N_\varepsilon(p) = \{q \in X \mid dist(p, q) \leq \varepsilon\}, \quad (2.2)$$

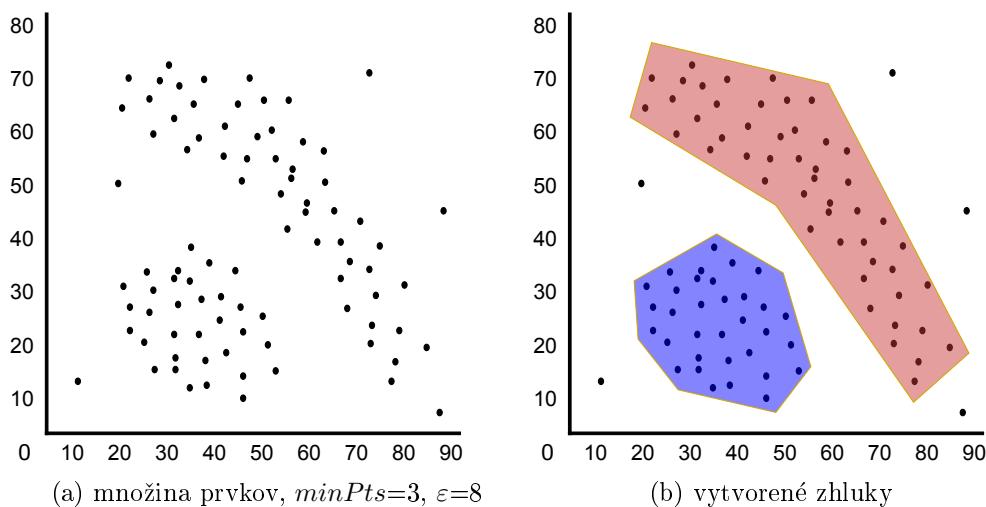
kde p je daný bod, X je množina vstupných bodov, q je bod patriaci do ε -okolia (N_ε) a funkcia $\text{dist}(p, q)$ vracia vzdialenosť medzi dvoma bodmi. Minimálny počet susedov prvku zase rozhoduje o tom, koľko prvkov musí byť vo vzdialosti menšej než ε aby sa prvak rátal medzi jadrové prvky zhľuku. Prvky zhľuku sa delia na jadrové a dosiahnuteľné. Jadrové sú také body, pre ktoré platí $|N_\varepsilon(p)| \geq \text{minPts}$. Z bodu sú dosiahnuteľné všetky také body, ktoré patria do ε -okolia, $q \in N_\varepsilon(p)$ a zároveň $|N_\varepsilon(q)| \geq \text{minPts}$, alebo do ε -okolia niektorého dosiahnuteľného bodu. Pre jadrové body v rámci zhľuku je dosiahnuteľnosť symetrická, vo vzťahu jadrový a dosiahnuteľný bod je asymetrická. Vstupná množina dát ešte môže obsahovať tzv. šum (*angl. Noise*). Sú to body, ktoré po prebehnutí zhľukovania nepatria ku žiadnemu zhľuku.

Samotný algoritmus prebieha nasledovne:

1. Algoritmus vyberie náhodný nespracovaný bod $p \in X$ a definuje množinu bodov v ε -okolí bodu $N_\varepsilon(p)$.
2. Ak je mohutnosť množiny $N_\varepsilon(p) < \text{minPts}$, zvolí sa ďalší bod a postup sa opakuje. Pre $N_\varepsilon(p) \geq \text{minPts}$, sa založí nový zhľuk.
3. Spustí sa expandovanie zhľuku so štartom v bode p (*angl. Seed*). Pre každý bod $q \in N_\varepsilon(p)$ sa vytvorí $N_\varepsilon(q)$. Podľa mohutnosti $N_\varepsilon(q)$ sa určí, či ide o jadrový, alebo dosiahnuteľný bod. Pridaním bodu q do zhľuku sa $N_\varepsilon(q)$ stáva dosiahnuteľným z p a zaraďuje sa do množiny testovaných bodov.
4. Po odhalení celého zhľuku algoritmus prejde naspäť na krok 1.

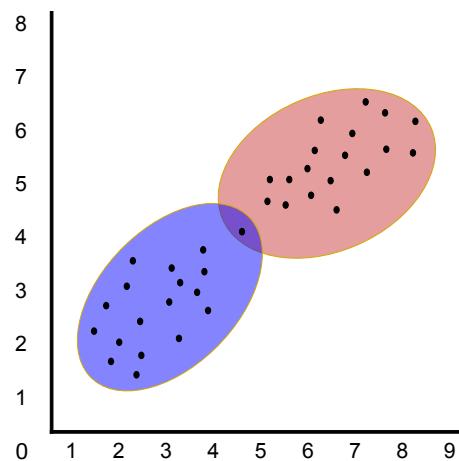
Všetky vytvorené zhľuky sú navzájom od seba oddelené medzerou o veľkosti minimálne ε . Prvky, ktoré nie sú súčasťou žiadneho zhľuku sa označujú ako šum.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI



Obr. 2.4: Vytvorenie zhlukov DBSCAN

DBSCAN má aj svoje slabšie stránky. Určenie parametrov ε a minPts vyžaduje vopred poznať rozloženie a vlastnosti dát vo vstupnej množine. Tiež nie je možné dobre spracovať dátá, ktorých jednotlivé zhluky majú významne rozdielne hustoty - ε je stanovené na pevnú pre celý vstup. DBSCAN nie je plne deterministický. Môže nastať situácia, keď okrajový bod zhluku je dosiahnuteľný z viac než len jedného zhluku. V takom prípade bude patriť k zhluku, ktorý ho dosiahne ako prvý, čo závisí od poradia v akom sa rozvíjajú jednotlivé prvky množiny. Pre jadrové prvky je algoritmus deterministický bez ohľadu na poradie rozvíjania prvkov.



Obr. 2.5: Nedeterminizmus v DBSCAN, bod na okraji je dosiahnuteľný via cerými zhlukmi

Výpočtová zložitosť DBSCAN je v najhoršom prípade $O(n^2)$. Každému prvku $p \in X$ sa počíta vzdialosť ku všetkým ostatným prvkom pri určovaní $N_\varepsilon(p)$. Použitím priestorového indexovania, napríklad pomocou R-stromu s hĺbkou $\log(n)$ zabezpečíme, že nie je potrebné počítať pre každý prvek vzdialosť ku všetkým ostatným $n - 1$ prvkom. Priemerná zložitosť vďaka tomu klesne na $O(n \log(n))$.

Existuje mnoho modifikácií pôvodného DBSCAN, medzi najznámejšie patria:

- C-DBSCAN [32] -(*angl. Constraint-driven DBSCAN*) umožňuje pomocou pridania podmienok (*angl. Constraints*) na spracovanie vstupných dát vniestť do výpočtu dodatočné informácie o doméne. Podmienky pre vzťahy prvkov môžu byť typu "musí byť spojené" (*angl. Must-Link*) alebo typu "nesmie byť spojené" (*angl. Cannot-Link*). Ak sú podmienky relevantné, zrýchľujú výpočet, ale predovšetkým zvyšujú kvalitu vytvorených zhlukov.
- GDBSCAN [33] -(*angl. Generalized DBSCAN*) o susednosti dvoch prvkov sa dá rozhodnúť aj na základe predikátov (symetrických, reflexívnych). Pri získávaní informácií o susedných prvkoch v rámci ε -okolia môžeme použiť aj iné metriky ako spočítanie prvkov, napr. určenie priemernej hodnoty niektorého spoločného atribútu.
- GFDBSCAN [21] -(*angl. Generalized Fuzzy DBSCAN*) používa sa pri vyhodnocovaní nejednoznačných atribútov, napr. pri profilovaní zákazníckych preferencií. Prvky klasifikuje ako jadrové s určitou pravdepodobnosťou.
- HPDBSCAN [17] -(*angl. Highly Parallel DBSCAN*) paralelná verzia DBSCAN s využitím rozhrania OpenMP, MPI a indexovania pomocou R-stromu.
- MDBSCAN [41] -(*angl. Multi-level DBSCAN*) zameriava sa na spracovanie dát v ktorých môžu mať zhluky odlišné hustoty. Podľa hodnoty ε vytvorí graf susedností prvkov a stanoví dve hodnoty minimálneho počtu susedov(\min{Pts}_1 , \min{Pts}_0), ktoré pri výpočte berie do úvahy. Zhlukovanie má lepšie výsledky pre dátá s rozdielnou hustotou pri zachovaní výpočtovej náročnosti porovnatnej s DBSCAN.
- PDBSCAN [43] -(*angl. Parallel DBSCAN*) hlavný proces s použitím MPI pridelí všetkým pripojeným zariadeniam úseky vstupných dát, ktoré majú spracovať. Následne prijíma a spája ich výsledky. Výpočet je dobre škálovateľný, pretože procesy nemajú žiadne zdieľané premenné. Pozornosť sa venuje správnemu rozdeleniu úsekov, prvky pravdepodobne patriace do jedného zhluku by mali byť počítané na jednom zariadení, aby sa znížila potreba komunikácie.

- PARDICLE [27] -(*angl.* Parallel Approximate Density-based Clustering) paralelný heuristiký algoritmus pre DBSCAN ktorý dosahuje takmer rovnakú kvalitu výsledných zhľukov ($\text{Omega-Index} \geq 0.99$) ako originálny algoritmus. Pracuje s hodnotou lokálnej hustoty pre každý prvok, ktorú odhaduje s použitím už určenej hustoty spracovaných susedov prvkmu. Pri hľadaní hustoty susedov prvku nepočítá vždy všetkých susedov, v závislosti od lokálnej hustoty vyberá minimálne minPts prvkov (aby nedošlo k omylom ohľadom klasifikácie jadrových bodov) a určí ich lokálnu hustotu. Algoritmus sa spolieha na to, že v hustých oblastiach existuje viacero ciest, ktorými je prvok dosiahnuteľný a nie je potrebné počítať všetky.

2.5.4 ρ - approximate DBSCAN

V článku [12] autori preukázali, že výpočtová zložitosť základného algoritmu DBSCAN vo viac než dvojrozmernom priestore nemôže byť $O(n \log(n))$, ale prinajlepšom $\Omega(n^{4/3})$. Ako dôkaz ukazujú prepojenie problému DBSCAN s geometrickým problémom navrhnutým J. Hopcroftom [10] - ak máme v priestore náhodne rozmiestnené body a priamky, nedokážeme určiť, či sa zvolený bod nachádza na niektornej priamke rýchlejšie ako $\Omega(n^{4/3})$ v najlepšom prípade. Navrhujú preto novú metódu, ρ - approximate DBSCAN, ktorá vďaka použitiu zanedbateľnej aproximácie naozaj dosahuje zložitosť $O(n \log(n))$, ba v priemernom prípade dokonca $O(n)$ pre ľubovoľne veľké d .

Algoritmus je založený na viac úrovňovej mriežke so základnou veľkosťou strany bunky $\varepsilon/\sqrt{2}$ v dvojrozmernom a ε/\sqrt{d} vo viacrozmernom priestore. Každá neprázdna bunka v základnej mriežke sa ďalej rekurzívne delí na 2^d menších až po hranicu veľkosti strany $\varepsilon/\rho/\sqrt{d}$. Táto minimálna veľkosť zabezpečuje, že pri hľadaní susedných buniek zvolenej bunky môžeme urýchlene prehliadať strom hierarchie buniek odmietaním všetkých buniek s $\text{dist}(p, c_i) > \varepsilon$ a prijímaním tých, ktoré celým objemom spadajú do vzdialenosťi menšej než $\varepsilon + \rho$. Značenie jadrových bodov a koncept vzájomnej hustotnej dosiahnutelnosti (*angl.* density-reachability) bodov zostáva oproti DBSCAN nezmenený, pribúda však koncept ρ - approximate hustotnej dosiahnutelnosti - bod p_i je dosiahnuteľný z bodu p_j ak $\text{dist}(p_i, p_j) < (\varepsilon * (1 + \rho))$ pre $i, j \in N$.

Aproximácia vstupuje do výpočtu vo forme vstupného parametra ρ , ktorým môže byť ľubovoľné kladné číslo, najčastejšie veľmi malé desatinné (0.001). ρ - approximate DBSCAN garantuje, že výsledok bude niekde medzi výsledkami základného DBSCAN s parametrami $(\varepsilon, \text{MinPts})$ a $(\varepsilon * (1 + \rho), \text{MinPts})$. Spolieha sa na to, že existuje dostatočne veľký interval pre ε , ktorý dáva podobné, dobré hodnoty a preto jeho úprava o hodnotu ρ významne neovplyvní výsledok.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

V dvojrozmernom priestore algoritmus dosiahne priemerný čas $O(\text{MinPts} * n)$, prinajhoršom $O(n \log(n))$ nasledovne:

1. Vytvorí sa pravidelná mriežka so stranou štvorca $\varepsilon/\sqrt{2}$. Každý štvorec sa rekurzívne delí na 2^d menších, až po hranicu veľkosti $\varepsilon\rho/\sqrt{d}$.
2. Každý štvorec sa vyhodnotí. Ak je počet bodov v ňom $\geq \text{MinPts}$, všetky body sa budú pridávať naraz do toho istého zhluku. Ak menej, jednotlivo sa vyhodnotia počty ich susedov. V prípade, že je v mriežke aspoň jeden bod, ktorý má za hranicou mriežky dosť susedov aby bol jadrový, môžeme pridať do zhluku všetky body v mriežke - prinajhoršom budú okrajovými bodmi zhluku.
3. Pomocou Union-Find algoritmu (opísaného v kapitole 4.5.1) sa štvrčeky v mriežke pospájajú do zhlukov.

Pri viac než dvojrozmernom priestore je postup veľmi podobný. Priestor sa rozdelí do d rozmernej mriežky s veľkosťou hrany ε / \sqrt{d} a všetkým bunkám sa určí počet bodov. S prázdnymi sa ďalej nepracuje, neprázdné sa delia na 2^d rovnako objemných menších mriežok. Opakuje sa vyhodnotenie počtu bodov v týchto o jednu úroveň menších d rozmerých kockách a opäť sa delí na menšie, až kým nie je hrana kocky najviac $(\varepsilon * (1 + \rho)) / \sqrt{d}$. Vznikne stromová štruktúra d rozmerých kociek, pre každú sa určí $\text{count}(c)$ - počet bodov v jej hraniciach. Bunka bez potomkov je list stromu.

Vytváranie zhlukov prebieha nasledovne: na začiatku je počítadlo počtu zhlukov numOfClust rovné nule. Prechádzame postupne všetky zatiaľ nezaradené body a hľadáme, do akého zhluku patria. Môžu nastať tri prípady:

1. d rozmerá kocka je celá mimo gule s polomerom $(\varepsilon * (1 + \rho))$ so stredom v bode q a neberie sa do úvahy.
2. Bunka je plne pokrytá, numOfClust sa zvyšuje a zakladáme nový zhluk, do ktorého priradíme aj všetky susedné bunky.
3. Bunka je prekrytá čiastočne. Skontroluje sa, či sa jedná o list stromu. Ak nie, všetky jej podbunky sa pridajú do zásobníka a vyhodnotia sa podľa týchto troch krokov. Ak bunka je listom, pridáva sa do výsledného zhluku.

Po prebehnutí priradovania bodov do zhlukov nám zostáva množina nezaradených bodov - šum, ktorý sa do výsledku nedostane.

Metóde ρ - approximate DBSCAN vyčítajú v odbornej literatúre [34] predovšetkým nevhodne zvolené parametre základnej verzie DBSCAN v porovnávacích testoch, nevhodnosť použitia mriežky pri vysoko dimenzinálnych

dátach a nemožnosť použiť pre určovanie vzdialenosťi bodov inú ako Euklidovskú vzdialenosť.

2.5.5 Dynamický DBSCAN

Dynamické zhlukovanie založené na hustote nepracuje jednorázovo nad nemennou, statickou množinou dát, naopak, je možné priebežne pridávať či odoberať prvky a aktualizovať existujúci výsledok. Problém je, že zhluk C do ktorého patrí bod p_i sa nedá definovať len sledovaním bodu p_i . Ak príde ku zmene vlastnosti bodu p_i , môže, ale nemusí sa zmeniť zhluk C . Do úvahy pripadá jeho spojenie s iným zhlukom, prípadne rozpadnutie na dva menšie zhluky.

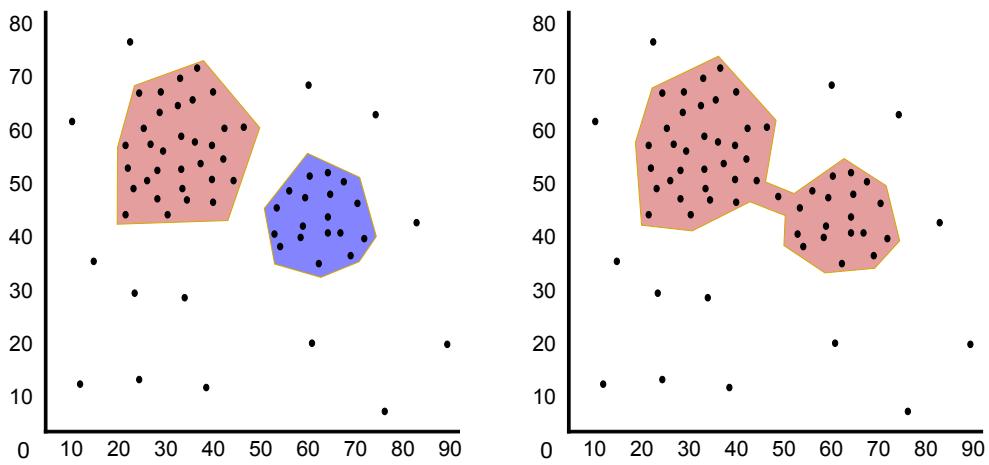
Pridanie možnosti vložiť nový prvok (*insert*) a ubrať existujúci prvok (*delete*) pre body množiny spracovávanej ρ - approximate DBSCAN upravuje zložitosť opäť na prinajlepšom $\Omega(n^{4/3})$ [13]. V ďalšom článku preto autori pôvodného algoritmu navrhujú metódu ρ - double - approximate DBSCAN ktorá pomocou ďalšej malej aproximácie opäť dosiahne $O(|Q|)$, kde Q je mohutnosť množiny vstupných prvkov.

Zavádzajú novú operáciu - *Cluster-Group-By* (skrátene *C-group-by*), ktorá pre množinu prvkov Q , ktorá je podmnožinou celej dátovej množiny P , vracia zoznam prvkov Q a ich príslušnosť ku zhlukom. Napr. $((q1, q2, q3, q4, q5) \rightarrow (q1)(q2, q3)(q4, q5))$.

V prípade, že sa do *C-group-by* pošle celá množina P , výsledkom je klasické zhlukovanie. Operácia je ale určená primárne na dopyty zaujímajúce sa o zhlukovú príslušnosť niekoľkých prvkov (sú / nie sú v rovnakom zhluku).

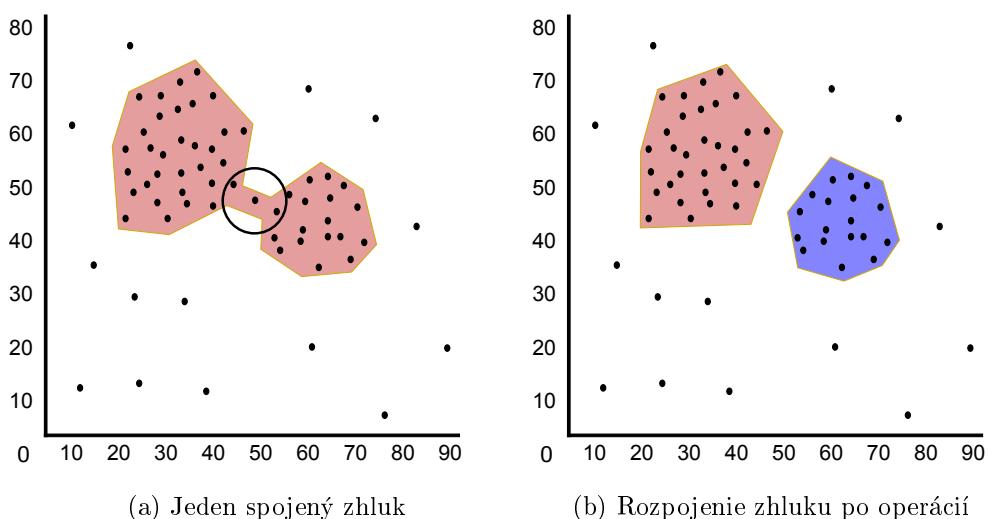
Vkladanie (operácia *insert*) nových prvkov je jednoduché. Zoberú sa všetky body v ϵ okolí vloženého bodu a ich zhluky sa spoja. Opäť môžeme použiť algoritmus Union-Find (opísaný v kapitole 4.5.1).

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI



Obr. 2.6: Priebeh operácie *insert*

Pri mazaní prvkov (operácia *delete*) sa zoberú všetky hrany medzi bodom ktorý ideme zmazať a jeho ε susedmi a medzi susedmi navzájom. Vymažú sa hrany vedúce na mazaný bod a zostávajúce sa skontrolujú. Ak minimálna kostra grafu spojenia susedov obsahuje hranu väčšiu než ε , zhluk sa rozpadol.



Obr. 2.7: Priebeh operácie *delete*

ρ - double - approximate DBSCAN sa zakladá na relaxovaní určovania jadrových bodov. Ak je v ε -ovom okolí bodu p_i aspoň $MinPts$ bodov, určite sa jedná o jadrový bod, ak nie je v $(\varepsilon * (1 + \rho))$ okolí bodu p_i aspoň $MinPts$

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

bodov, určite sa nejedná o jadrový bod. Iné možnosti sa nesledujú. Základom je opäť mriežka so stranou ε/\sqrt{d} , bunky sú susedné ak sú ich vzájomne najbližšie body maximálne ε vzdialené. Vyrába sa graf mriežky $G = (V, E)$ kde V je množina jadrových buniek (jadrové bunky sú bunky, ktoré obsahujú aspoň jeden jadrový bod) a E množina hrán. Každá hrana spájajúca bunky c_1, c_2 splňa:

- Ak c_1, c_2 sú jadrové bunky obsahujúce jadrové body p_1, p_2 ,
- Tak p_1, p_2 sú v rovnakom zhluku a c_1, c_2 sú spojené v rovnakom komponente.

Na vykonanie funkcie $Cluster - Group - By$, na ktorej stojí ρ - double - approximate DBSCAN algoritmus potrebujeme nasledovné prostriedky:

1. Štruktúra jadrových bodov - pre každý bod zaznamenáme či je, alebo nie je jadrový.
2. ρ approximate ε emptiness funkcia - vracia pre zadaný bod p a jadrovú bunku c hodnotu:
 - 1 ak bunka má bod q taký, že $dist(q, p) < \varepsilon$,
 - 0 ak žiadny jadrový bod z bunky c nespĺňa $dist(q, p) < (\varepsilon*(1+\rho))$.

Okrem hodnoty 1 v prípade úspechu vracia aj daný bod, ktorý podmienku spĺňa. Pre každý jadrový zhluk sa udržiava štruktúra jeho jadrových bodov, ktorá nám pomáha pri udržiavaní aktuálnosti hrán susednosti buniek pri vkladaní a odoberaní bodov.

3. Štruktúra spojených komponentov - podporuje možnosti:
 - Pridanie hrany medzi susednými bunkami c_1 a c_2 ($edgeInsert(c_1, c_2)$),
 - Odobratie hrany ($edgeRemove(c_1, c_2)$) a zrušenie priameho susedstva buniek,
 - Získanie Id komponentu (buniek pospájaných hranami) do ktorého ľubovoľný bod patrí.

S týmito štruktúrami sa následne dá vykonávať operácia $C - group - by$. Na zopakovanie, jej vstupom je množina bodov, napr. $(q1, q2, q3, q4, q5)$ a výstupom priradenie vstupných bodov do zhlukov v rámci datasetu, napr. $(q1)(q2, q3)(q4, q5)$.

Algoritmus má priebeh:

1. Rozdelenie vstupnej množiny bodov Q na základe štruktúry jadrových bodov na podmnožiny Q_1 - jadrové body a Q_2 - nie jadrové body.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

2. Pre každý jadrový bod z množiny Q_1 sa získa pomocou štruktúry spojených komponentov Id komponentu do ktorého patrí jadrová bunka v ktorej bod leží. Založí sa zhluk s takýmto Id a priradí sa doň daný komponent.
3. Pri bodoch z množiny Q_2 sa rozhoduje o tom, či budú priradené k najbližšej jadrovej bunke, alebo označené ako šum. Pre každú sa volá ρ approximate ε emptiness funkcia - ak je výsledok 1, bod sa priraduje do rovnakého zhluku ako bod, ktorý funkcia na dôkaz vráti. Ak je výsledok 0, bod sa označí za šum.

Hlavným problémom metódy ρ - double - approximate DBSCAN je použitie vo vyššom počte dimenzií. Autori v článku kvôli výpočtovej zložitosti pracujú s maximálnym počtom dimenzií rovným 7.

2.5.6 GDPAM

Úpravou klasického DBSCAN algoritmu smerom ku lepšej použiteľnosti na vysoko dimenzionálnych dátach sa v článku [7] zaoberal kolektív autorov z Univerzity Čínskej akadémie vied v Pekingu. Pri práci s vysoko dimenzinálnymi datasetmi pri DBSCAN založenom na mriežke identifikovali dva hlavné problémy:

1. Geometrický rast počtu susedných buniek pri každej ďalšej dimenzií.
2. Reťazenie spájania buniek do zhlukov na základe hrán. Platí tu:
 - Symetria - ak je bunka c_1 susedom bunky c_2 , znamená to, že aj bunka c_2 bude susedom bunky c_1 .
 - Tranzitivita - ak je c_1 sused s c_2 a c_2 s c_3 , aj c_1 je sused s c_3 . Nemalo by byť potrebné to znova skúmať.

Navrhovaným riešením, ktoré by umožnilo použiť zhlukovanie založené na hustote aj pri vyššom počte dimenzií, je zavedenie bitmapového indexovania susednosti buniek. Pre každú dimenziu sa vytvára bitová mapa s počtom riadkov k a počtom stĺpcov N . Počet riadkov k je stanovený podľa počtu unikátnych začiatočných koordinátov buniek obsahujúcich nejaké body v danej dimenzií. Počet stĺpcov N je daný počtom neprázdných buniek. Následne sa dá v sade máp vyhľadávať pomocou vzorca:

$$\text{NeighbourRange}(g, i) = [g.\text{pos}[i] - \lceil \sqrt{d} \rceil, g.\text{pos}[i] + \lceil \sqrt{d} \rceil]$$

kde g predstavuje bunku ktorej hľadáme susedov, i predstavuje i -tu dimenziu, $g.\text{pos}[i]$ predstavuje pozíciu riadku tabuľky v ktorom ma g bunka hodnotu 1 a \sqrt{d} odmocninu z celkového počtu dimenzií datasetu. Výsledkom bude interval riadkov tabuľky, v ktorých hľadáme susedné bunky - hodnota

v tabuľke je rovná 1. Zavolením tohto dopytu pre každú dimenziu získame i vektorov dĺžky N s prípustnými hodnotami 0 a 1. Logickým *AND* všetkých vektorov dostaneme výslednú množinu susedov ľubovoľnej bunky. Čím vyšší počet dimenzií, tým viac operácií nám tento postup šetrí.

2.5.7 OPTICS

Analýza algoritmu OPTICS je prebratá z analýzy v rámci mojej bakalárskej práce [23].

OPTICS (*angl. Ordering points to identify the clustering structure*) [4] rieši problém detegovania zhlukov v prípade, že vstupné dátá nemajú homogénnu hustotu. Pre každý bod $p \in X$ definuje jadrovú vzdialenosť (*angl. core distance*) $dist_{core}(p)$ rovnú vzdialenosť ku jeho najbližšiemu susedovi r , a vzdialenosť dosiahnutelnosti (*angl. reachability distance*) rovnú väčšej z hodnôt $dist_{core}(p)$ bodu a reálnej vzdialenosť $dist(p, r)$. Na základe týchto dvoch hodnôt je možné usporiadať výsledok zhlukovania do dendrogramu tak, že je možné dodatočne stanoviť parametre na minimálnu hustotu prvku. Väčšou hodnotou minimálnej hustoty sa napr. z výsledku ako zhluk definuje len prienik dvoch rôzne hustých zhlukov. Keď sa táto hodnota zníži, bude sa už za zhluk považovať hustejší z nich, a pri ešte nižšej hodnote bude zhlukom ich zjednotenie.

2.5.8 Zhodnotenie popísaných algoritmov

Všetky vyššie uvedené algoritmy majú svoje výhody aj nevýhody. Zhrňme si ich:

1. K-means (2.5.1):

- Výhody: rýchly, jednoduchý,
- Nevýhody: ťažké určiť k , problémy so zhlukmi iného než sférickeho tvaru.

2. Hierarchické zhlukovanie (2.5.2):

- Výhody: nie sú potrebné vstupné parametre,
- Nevýhody: výpočtovo náročné, výsledný dendrogram treba ďalej interpretovať.

3. DBSCAN (2.5.3):

- Výhody: identifikuje zhluky nepravidelných tvarov, odfiltruje šum a anomálie, nie je potrebné poznať počet zhlukov,
- Nevýhody: ťažké určiť ϵ a $minPts$ pomocou ktorých sa definuje hraničná hustota zhluku.

KAPITOLA 2. OPIS PROBLÉMOVEJ OBLASTI

4. ρ - approximate DBSCAN (2.5.4):

- Výhody: zrýchlenie klasického DBSCAN pomocou mriežky, pridanie možnosti aproximácie ρ parametrom (percentuálne zväčšenie ϵ) urýchľujúce výpočet pri expanzii zhlukov,
- Nevýhody: pri vyšších počtoch dimenzií je rázia manipulácie s mriežkou vysoká.

5. ρ - double - approximate DBSCAN (2.5.5):

- Výhody: pridanie dynamického zhlukovania, podpora operácií *insert* a *delete*,
- Nevýhody: aj v tejto vylepšenej verzii stále problém s vyšším počtom dimenzií kvôli manipulácii s mriežkou.

6. GDPAM (2.5.6):

- Výhody: použitie bitových máp umožňuje použiť DBSCAN založený na mriežke aj pri vyššom počte dimenzií,
- Nevýhody: zložitosť určovania parametrov ϵ a $minPts$.

7. OPTICS (2.5.7):

- Výhody: nie je potrebné zadávať parameter ϵ , identifikuje štruktúru prítomnosti aj rôzne hustých zhlukov,
- Nevýhody: výsledok je potrebné ďalej interpretovať.

Napriek ťažkostiam pri určovaní ϵ a $minPts$ parametrov algoritmov založených na hustote prínosy tejto metódy, predovšetkým:

- nie je treba poznáť počet zhlukov,
- ľubovoľný tvar hľadaných zhlukov,
- automatické filtrovanie šumu,

zadávajú dôvod, aby sme ju pri analýze cytometrických dát skúsili aplikovať.

Kapitola 3

Návrh riešenia

V rámci témy "Využitie dynamického zhlukovania založeného na hustote v bioinformatike" by sme sa chceli venovať použitiu dynamického ρ - double - approximate DBSCAN (opísané v časti 2.5.5) pri spracovaní cytometrických dát (2.2). V prvom kroku implementujeme nové riešenie založené na hustote, ktorého cieľom bude automatický gating ekvivalentný krokom vzorkovania a hierarchického/ťažiskového zhlukovania v algoritme SPADE (2.4.2). Použitie zhlukovania založeného na hustote odstráni problém s neznámym počtom zhlukov a so šumom, problémom ale bude správna konfigurácia zhlukovacích parametrov definujúcich požadovanú hustotu. Čiastočnú predstavu o rozložení zhlukov a ich hustote by mohol priniesť algoritmus OPTICS (2.5.7), ktorému stačí zadať parameter $minPts$ - počet susedných bodov. Vyskúšaním rôznych hodnôt bude možné odhadnúť, akú hustotu asi vstupné dátá majú. Výzvou bude popasovať sa s vysokou dimenzionalitou dát. Pokúsime sa zistiť, či by nebolo možné niektoré atribúty bodov vyradiť ako málo významné. Na riešenie vysokého počtu dimenzií vstupných dát použijeme algoritmus GDPAM (2.5.6) a modifikujeme ho aj pre použitie v dynamickom zhlukovaní. Výsledok porovnáme s existujúcimi riešeniami a výsledné zhluky vizualizujeme.

3.1 Špecifikácia požiadaviek

- Implementovať algoritmus ρ - approximate DBSCAN.
- Implementovať dynamický algoritmus ρ - double - approximate DBSCAN.
- Optimalizovať algoritmus ρ - double approximate DBSCAN pre použitie vo vyšších dimensiach pomocou bitových máp algoritmu GDPAM (2.5.6).
- Odhaliť nedostatky implementovaných algoritmov a modifikáciou vytvoriť vlastné riešenie.

KAPITOLA 3. NÁVRH RIEŠENIA

- Vytvoriť R package s výsledným algoritmom pre použitie C++ zdrojového kódu v rámci programovacieho jazyka R.
- Vytvoriť paralelnú OpenMP verziu algoritmu a otestovať rozsah zrýchlenia.
- Vyhodnotiť rádovo prípustné hodnoty parametrov ϵ a $minPts$ pre vysoko dimenzionálne cytometrické dátá.
- Vyhodnotiť kvalitu výsledných zhľukov cytometrických dát s dostupným správnym výsledkom pomocou externých validačných indexov.
- Vizualizovať výsledky zhľukovania.

Kapitola 4

Opis riešenia

Nasledujúca kapitola je venovaná detailom vývoja riešenia, ktoré by splnilo požiadavky z kapitoly špecifikácie požiadaviek (3.1).

4.1 Popis vstupných dát

Vstupom pre opísané zhľukovacie algoritmy môžu byť akékoľvek súbory so štruktúrou:

1. na každom riadku sa nachádza jeden prvok vstupnej množiny, jeho jednotlivé atribúty sú oddelené prázdnym znakom,
2. v každom stĺpci sa nachádza nameraná množina hodnôt vybraného atribútu.

Špecifická doména tejto práce sú cytometrické dáta. Na každom riadku sa bude nachádzať meranie jednej bunky a každý stĺpec bude predstavovať expresivitu proteínu naprieč datasetom. Štandardným typom súboru používaným v tejto doméne je .fcs (z *angl. flow cytometry standard*). Formát pozostáva z hlavičky, ktorá obsahuje metadáta merania - informácie o zariadení, ktoré skenovalo bunky apod. a samotného tela - $m \times n$ matice s počtom riadkov m rovným počtu meraných buniek a počtom stĺpcov n rovným počtu meraných atribútov. Prvý riadok matice väčšinou obsahuje názvy atribútov (proteínov) v stĺpcoch a je potrebné ho pri načítavaní hodnôt preskočiť.

Na otváranie fcs súborov sa bude v rámci projektu používať R package Bioconductor, ktorý už má prostriedky na manipuláciu so súbormi implementované. Pomocou tohto balíčka z .fcs súboru exportujeme maticu nameraných hodnôt. R skript na vytvorenie vstupu pre algoritmus sa nachádza na elektronickom nosiči v prílohe A.

Počet dimenzií vstupných dát sa dá okresať o atribúty nesúvisiace s biologickými vlastnosťami meraných buniek.

4.2 Implementácia

Zdrojový kód algoritmu je vyvýjaný v jazyku C++ vo verzii C++11 s použitým komplátorem MinGW vo verzii MinGW-W64-builds-4.3.0 a vývojovým prostredím CLion 2018.2.5. Pri manipulácií s fcs súbormi a integrácií C++ zdrojového kódu do R balíka sa používa verzia jazyka R 3.5.1 vo vývojovom prostredí RStudio 1.1.423. Verzie jednotlivých R balíkov budú spomenuté v častiach opisujúcich ich použitie.

Voľba jazyka C++ je daná jeho rýchlosťou, voľba jazyka R širokou použiteľnosťou v oblasti práce s dátami. Nasledujúca sekcia bude obsahovať podrobnejší popis implementačnej časti práce.

4.2.1 Štruktúra programu

Základná štruktúra finálnej verzie C++ algoritmu vyzerá nasledovne:

```
setAlgParameters( $\epsilon$ , minPts, inpFile, outFile, saveState);
loadInputPoints(inputFile);
initCellLevel(inputPoints);
initGdpamTables(cellLevel);
setCellPositions(gdpamTables, cellLevel);
for ( $i = 0$  ;  $i < numOfCells$  ;  $i++$ ) do
    getNeighbours(cells[i], gdpamTables);
    if(evaluateNeighbours(neighbours));
    createUnionFindNode(cells[i]);
end
processUnionFindNodes(cells);
result = createClusters(cells);
```

Algoritmus 1: Kostra finálneho algoritmu

Na umožnenie tohto priebehu je potrebné zaviesť do kódu nasledovné objekty a ich vzťahy:

1. Bod (*angl. Point*) - jeden prvok vstupnej množiny, okrem súradníc v priestore pre každú dimenziu objekt uchováva informáciu o priradení do zhluku a či je jadrovým bodom - či je v jeho okolí aspoň $minPts$ ďalších bodov.
2. Bunka (*angl. Cell*) - d rozmer náštev kocka v rámci priestoru vstupného datasetu s veľkosťou hrany ϵ/\sqrt{d} . Celý vstupný priestor je rozdelený do buniek. Každá bunka pozná množinu bodov ktoré do nej patria, svoje id, informácie o svojej veľkosti, súradničach, či sa ráta medzi jadrové bunky a informácie o svojej pozícii v Union Find strome (4.5.1).

KAPITOLA 4. OPIS RIEŠENIA

3. Level - množina buniek s rovnakou veľkosťou hrany (vo finálnom algoritme množina všetkých buniek) vyplňajúcich stavový priestor.
4. Zhluk (*angl. Cluster*) - množina susedných buniek s jedinečným identifikátorom. Poskytuje možnosť vypísať prvky zhluku.
5. HyperGridBitmap - tabuľka s pozíciami buniek v jednotlivých dimenzích. Po nastavení tabuľiek v každej dimenzií je podľa algoritmu (2.5.6) možné v čase $O(d(2\sqrt{d}+1)) = O(d^{3/2})$, kde d je počet dimenzií, zistiť susedov ľubovoľnej bunky.
6. DataSet - objekt predstavujúci načítaný dátový vstup.

Štruktúra algoritmu ρ - approximate DBSCAN z ktorého sme čiastočne vychádzali:

```
getInputPoints(inputFile);
setAlgParameters( $\epsilon$ , minPts, inpFile, outFile);
initFirstCellLevel(inputPoints);
assignPoints(inputPoints);
evaluateCells(firstCellLevel);
recursiveInitOfGridLevels(minCoords, maxCoords);
for (i = 0 ; i < numOfLevels ; i++) do
    | assignPointsToGrid(levels[i], inputPoints);
    | evaluateCells(levels[i]);
end
for (i = 0 ; i < numOfLevels ; i++) do
    | for (j = 0 ; j < grid[i]→cells.size() ; j++) do
        | neighbourCells = getNeighbourCoreCells(grid[i]→cells[j]);
        | addEdges(grid[i]→cells[j], neighbourCells);
    | end
end
for (i = 0 ; i < inputPoints ; i++) do
    | if (!(inputPoints[i].inCluster())) inputPoints[i].processEdges();
    | inputPoints[i].createClusterIfNeeded();
end
```

Algoritmus 2: Kostra ρ - approximate DBSCAN algoritmu

Objekty používané v tomto algoritme sú z veľkej časti rovnaké, poskytujú ale navyše niekoľko metód:

1. Bod (*angl. Point*) - poskytuje metódy na získanie množiny susedných bodov (body v okolí do vzdialenosťi ϵ) a na získanie vzdialenosťi ku inému bodu alebo bunke.

KAPITOLA 4. OPIS RIEŠENIA

2. Bunka (*angl. Cell*) - uchováva navyše informácie o hranách so susednými bunkami, a o tom, či je jadrovou bunkou. Jadrová bunka je taká bunka, ktorá obsahuje aspoň jeden jadrový bod. Objekt bunky poskytuje metódy na expandovanie bunky - rozdelenie aktuálnej bunky na 2^d menších, s obmedzením na minimálnu veľkosť hrany ϵ / \sqrt{d} . Ďalej je tu možnosť priradiť body bunky do buniek vzniknutých jej expandovaním a možnosť pridať bunku do niektorého zhluku.
3. Level - poskytuje metódy na priradenie bodov datasetu do buniek na danom leveli a možnosť odfiltrovať všetky prázdne bunky.
4. Hrana (*angl. Edge*) - dvojica ukazovateľov na bunky, ktoré sú spolu v zhluku. Množinu svojich hrán si udržiava každá bunka zvlášť.
5. Zhluk (*angl. Cluster*) - množina susedných buniek pospájaná hranami.

Druhou časťou riešenia je vizualizácia výsledkov zhlukovania. Vizualizáciu sme sa rozhodli realizovať pomocou interaktívneho grafického panelu (*angl. dashboard*) vytvoreného v jazyku R knižnicou “flexdashboard”. Tá umožňuje pomocou formátu R Markdown zadefinovať požadované grafické elementy a spustí ich vo vlastnom okne na vystavenej lokálnej url adrese.

4.2.2 Vlastnosti programu

Po prvotnom načítaní bodov vstupnej množiny sa na udržiavanie všetkých polí prvkov v programe kvôli rýchlosťi používajú vektory ukazovateľov. Ako ukazovatele sa používajú tzv. smart pointre, ktoré zabraňujú únikom pamäte.

Program má v aktuálnej verzií niekoľko kľúčových oblastí. Zoradené podľa času vykonávania:

1. Načítanie bodov - okrem samotného načítania sa tiež pre každú dimenziu hľadá priestorové rozpätie (minimálna a maximálna hodnota atribútu v danej dimensií). Vstupné dátá by v tejto fáze už mali byť znormované, čo zabezpečuje, že sa hodnoty jednotlivých expresív génov rádovo neodlišujú. Načítavanie vstupných dát má lineárnu priestorovú aj výpočtovú zložitosť.
2. Stanovenie parametrov - pre algoritmus potrebujeme zadať 2 parametre:
 - ϵ - najdôležitejšia z hodnôt, okrem určenia hustoty má významný vplyv na rýchlosť výpočtu. Na vytvorenie predstavy o správnej hodnote ϵ nám môže pomôcť algoritmus OPTICS (opísaný v časti

2.5.7) alebo je ju možné stanoviť na základe doménových znalostí, ak vieme ako veľmi sa odlišujú prvky hľadaných zhlukov.

- $\min Pts$ - minimálny počet susedov bodu aby sa rátal ako jadrový. Opäť môže vyplynúť z doménovej znalosti, ak vieme stanoviť očakávaný počet výskytu buniek z hľadaných populácií, prípadne vieme, aké najmenšie zoskupenie ešte chceme brať ako relevantný vzor.

3. Vytváranie mriežky a priraďovanie bodov:

- (a) Vytvorenie levelu - vo finálnom algoritme vytvárame iba jeden level s bunkami o veľkosti hrany $\epsilon\sqrt{d}$. V ρ - approximate DBSCAN to je najprv začiatočný, 0-tý level pokrývajúci celý priestor vstupnej množiny a následne sa v každom ďalšom leveli berú všetky bunky okrem prázdných a vytvára sa pre každú 2^d potomkov. Počet buniek rastie geometricky, počet krovok je ale obmedzený hranicou veľkosti bunky na nízke prirodzené číslo. Inicializáciu buniek v stavovom priestore sme optimalizovali - namiesto vypĺňania celého priestoru bunkami a následného odfiltrovania prázdných vytvárame bunky iba na koordinátoch, kde dôjde ku pokrytiu aspoň jedného bodu. Tento prístup nám okrem iného umožnil používať ľubovoľnú hodnotu parametra ϵ , pri ktorom sme predtým sledovali, či nám kvôli exponenciálnemu rastu pri vysokých dimenziách nepretečú premenné kvôli počtu buniek.
- (b) Priradenie bodov bunkám levelu - tento krok sa vykonáva iba v ρ - approximate DBSCAN, vo finálnom algoritme prichádza ku priraďovaniu bodov vo fáze vytvárania buniek. Každá bunka nižzej úrovne pozná svojho rodiča a dokáže si z jeho množiny bodov priradiť tie, ktoré spadajú do jej priestoru. Výpočtová zložitosť tohto kroku je lineárna, stále pracujeme s tým istým vektorom vstupných bodov, z pamäťového hľadiska ide iba o kopírovanie ukazovateľov.
- (c) Určenie jadrových buniek - v bunkách s veľkosťou hrany ϵ/\sqrt{d} sú všetky body v rovnakom zhluku. Ak má bunka rovnako alebo viac než $\min Pts$ bodov, určite bude súčasťou zhluku. Ak má bunka menej než $\min Pts$ ale viac než 0 bodov, môže byť priradená ako susedná ku jadrovej bunke (jej body budú okrajovými bodmi zhluku) alebo vyradená ako šum. Zložitosť kroku je lineárna, každú bunku je potrebné spracovať raz.
- (d) Pospájanie jadrových buniek do zhlukov - susednosť jadrových buniek môžeme skúmať pomocou algoritmu GDPAM (2.5.6), alebo klasicky cez hrany susednosti. Zložitosť je za použitia hrán buniek kvadratická, určovanie susednosti buniek sa vykonáva voči

všetkým ostatným. V prípade použitia algoritmu GDPAM (2.5.6) klesá na $O(d^{3/2})$.

4. Získanie výsledkov zhlukovania - každá bunka má nastavené unikátne id na základe ktorého algoritmus Union-Find 4.5.1 vytvorí taký počet oddelených stromov, ako je počet zhlukov identifikovaných v datasete. Zložitosť kroku je pri použití vyvážených stromových štruktúr logaritmická.

4.2.3 Popis výstupných dát

Výstupy zhlukovacieho algoritmu sú dva.

1. Štandardné rozdelenie vstupných bodov do zhlukov - textový súbor s $m \times n + 1$ maticou bodov. Na m riadkoch sú vypísané všetky merané biologické bunky a v $n + 1$ stĺpcach ich atribúty, pričom posledný stĺpec (+1) obsahuje číslo zhluku do ktorého bod patrí. Hodnota -1 indikuje, že bod neboli priradený do žiadneho zhluku a z výsledku je odfiltrovaný ako šum.
2. Koordináty buniek použitých v rámci zhlukovania - jedná sa o vzorovanie celej vstupnej množiny namapovaním na intervale s veľkosťou $\epsilon\sqrt{d}$. Rovnako ide o textový súbor s $m \times n + 1$ maticou, na m riadkoch sú ale vypísané priestorové bunky použité pri zhlukovaní. V n stĺpcach sa nachádzajú priestorové súradnice začiatku bunky a $n+1$ obsahuje index výsledného zhluku. Tento výstup je obzvlášť praktický pri následnej vizualizácii, keďže rozdelenie priestoru na intervale významne zníži počet zobrazovaných elementov a rozdelenie na intervale zabráni ich prekrývaniu sa.

Ďalším, voliteľným výstupom zhlukovania je uloženie stavu stavového priestoru na konci zhlukovania. V prípade, že používateľ napr. vykoná mnoho hodín trvajúce zhlukovanie datasetu pacienta o veľkosti niekoľko miliónov buniek, môže si želať vyexportovať výsledný stav zhlukov, ktorý použije ako východiskový pri ďalších zhlukovaniach dát pre daného pacienta. Tento súbor teda môže byť zároveň aj vstupným súborom, ktorým si algoritmus nainicializuje stavový priestor ešte pred pridaním nových dát.

4.3 Paralelizovateľnosť výpočtov

Z paralelizačných možností používame len základnú paralelizáciu v rámci zdieľanej pamäte. Pre použitie viacerých vlákien procesora na rozdelenie spracovania vzájomne nezávislých výpočtov v rámci C++ kódu využívame framework OpenMP.

V rámci algoritmu sme identifikovali niekoľko miest súcich na paralelizáciu:

KAPITOLA 4. OPIS RIEŠENIA

1. Inicializácia GDPAM tabuliek - for cyklus s počtom vykonaní rovným počtu dimenzií. Zrýchlenie má význam len pri netriviálnom počte dimenzií.
2. Nastavenie pozícíí buniek v GDPAM tabuľkách - vnorené for cykly s počtom vykonaní $poetdimensi * poetbuniek$.
3. Získanie zoznamu susedných buniek V GDPAM tabuľkách - vnorené for cykly s počtom vykonaní $poetdimensi * poetbuniek$. Je potrebné dať si pozor a agregáciu výsledkov vykonať v rámci kritickej oblasti. Pre optimalizáciu nechávame každé vlákno zapisovať priebežné výsledky do lokálnej premennej a zaznačujeme výsledok do globálnej až na konci výpočtu.
4. Spájanie buniek do zhľukov pomocou algoritmu Union-Find - for cyklus s počtom susedov danej bunky. Zrýchlenie má význam len pri vyšom počte dimenzií, čo o spôsobuje vyšší počet susedných buniek.
5. Nastavovanie identifikačného čísla zhľuku bodom buniek zhľuku - zrýchlenie má význam len pri hustých datasetoch s vyššou hodnotou $minPts$, pri ktorých sa v bunkách nachádza väčšie množstvo bodov.
6. Meranie vzdialenosťi dvoch bodov - for cyklus s počtom vykonaní rovným počtu dimenzií (Euklidova vzdialenosť). Zrýchlenie má význam len pri netriviálnom počte dimenzií.
7. Nastavenie pracovných koordinátov pri inicializácii stavového priestoru - for cyklus s počtom vykonaní rovným počtu dimenzií. Zrýchlenie má význam len pri netriviálnom počte dimenzií.
8. Hľadanie existujúcich buniek pre body pri inicializácii stavového priestoru - for cyklus s počtom vykonaní rovným počtu vstupných bodov. Je potrebné dať si pozor pri vytváraní nových buniek v prípade že sa vhodná existujúca nenájde - počas doby vytvárania novej bunky mohlo iné vlákno vytvoriť bunku v priestore potrebnom pre nás skúmaný bod. Okolo bodu ukladania novej bunky do vektora buniek sa preto nachádza kritická oblasť ešte raz kontrolujúca potrebnosť vytvorenia bunky.

Optimalizácia programu z hľadiska použitých dátových štruktúr je robená primárne pre čisté C++, v rámci Rcpp OpenMP negarantuje dosahovanie rovnakých výsledkov. Percentuálny rýchlosťný zisk vyplývajúci z paralelizácie je popísaný v kapitole vyhodnotenie (5.4).

4.4 Dynamickosť výpočtov

Niekteré zhlukovacie algoritmy vyžadujú pri každej zmene vstupnej množiny opakovanie výpočtu. Závisí to od vplyvu, aký má na príslušnosť ku zhluku jeden prvok množiny. Napr. v prípade ľažiskových zhlukovacích metód, kde je zhluk vytvorený okolo pomyselného centra, má zmene vlastností prvku zhluku vplyv na pozíciu centra, tým na všetky ostatné prvky v zhluku a dokonca aj na prvky iných zhlukov. Pri našom zhlukovaní na základe hustoty to neplatí. Úprava vlastností prvku má vplyv iba na jeho bezprostredné okolie a môže viesť maximálne ku vzniku alebo rozpadu jedného zhluku. Implementovali sme preto možnosť cez parameter zhlukovacej funkcie požadovať:

- Uloženie výsledného stavového priestoru zhlukovania.
- Načítanie zvoleného predtým uloženého stavového priestoru ešte pred načítaním novej dátovej množiny - spôsobí to *insert* novej množiny dát do existujúceho stavu zhlukovania.
- Načítanie zvoleného predtým uloženého stavového priestoru, následné odstránenie množiny prijatých bodov a opäťovné uloženie nového stavu. Jedinečným identifikátorom bodov naprieč rôznymi datasetmi sú jeho súradnice.

Táto funkcia má význam z hľadiska šetrenia výpočtovým časom.

4.5 Dátové štruktúry

Zoznam dátových štruktúr a algoritmov použitých pri implementácii:

4.5.1 Union-Find

Na spájanie veľkého počtu disjunktných elementov do skupín sa často používa algoritmus Union-Find [39]. Dátová štruktúra, ktorú spravuje, umožňuje vykonávať nad jednotlivými elementmi operácie:

- Union - ako argument dostáva dva elementy, spojí množiny v ktorých sa tieto elementy nachádzajú.
- Find - argumentom je element, pre ktorý vráti id množiny v ktorej sa element nachádza.

Pomocou týchto dvoch operácií sa elementy umiestňujú do stromov. Zložitosť algoritmu je $O(\log(n))$, pričom operácia Union má zložitosť len $O(1)$.

KAPITOLA 4. OPIS RIEŠENIA

4.5.2 PCA

Analýza hlavných komponentov (*angl. Principal Component Analysis*) slúži na zredukovanie počtu atribútov objektu snažiac sa zachovať čo najviac z ich výpovednej hodnoty [1]. Technika je efektívna aj v prípade, že koncový počet atribútov je omnoho menší ako zdrojový. Okrem použitia kvôli vizualizácii dát v dvoj alebo troj rozmernom priestore sa môže vhodne použiť aj pri redukcii dimenzionality z dôvodu uľahčenia výpočtov ktoré vysoká dimenzionalita datasetu často výrazne ovplyvňuje. V našej práci techniku PCA používame len pri vizualizácii dát.

4.5.3 Prehľadávanie stromu do šírky

Pre urýchlenie vyhľadávania prvkov poľa v pamäti môžme jednoducho použiť dátovú štruktúru v podobe grafu stromu s n potomkami. Strom môže obsahovať koreň (začiatokný uzol), uzly s potomkami a uzly bez potomkov, tzv. listy stromu. Ak je strom usporiadany, pri hľadaní konkrétneho elementu sa postupuje od koreňa smerom nadol, pričom sa vždy vieme rozhodnúť, ktorým potomkom máme pokračovať, až kým element nenájdeme. V prípade neusporiadaných stromov môžeme postupovať pri prehľadávaní všeobecne a to dvoma spôsobmi:

1. Prehľadávanie do hĺbky - začína sa v koreni. V každom uzle pokračujeme do nenavštíveného elementu nižšej úrovne, pokiaľ nenarazíme na list stromu. Následne sa vraciame v hierarchii nahor, kým opäť nevieme pokračovať do nenavštíveného uzla nižšej úrovne.
2. Prehľadávanie do šírky - opäť sa začína v koreni. Do fronty sa vložia všetci potomkovia aktuálneho uzla a vyberie sa z nej nový uzol, u ktorého sa postup opakuje. Týmto spôsobom sa postupne prechádzajú všetky úrovne hĺbky od najmenšej (koreň) po najväčšiu (listy stromu).

4.6 Validácia výsledkov

Pri zhľukovaní dát často nepoznáme správne riešenie zatriedenia zhľukov. Aby sme mohli posúdiť kvalitu výsledku, prípadne porovnať jednotlivé iterácie zhľukovania, môžeme použiť validačné indexy. Delíme ich na:

- Interné - posudzujú zhľuky bez akýchkoľvek vonkajších informácií. Medzi známejšie patria metóda Siluety, Davies-Bouldin index, či pre zhľukovanie založené na hustote použiteľný DBCV index.
- Externé - spoliehajú sa na doložený správny výsledok a porovnávajú, nakoľko sa mu ten vytvorený podobá.

V práci pre vyhodnotenie použijeme externé validačné indexy z R knižnice "clues" opísané na nasledujúcich riadkoch.

4.6.1 Rand index

Rand index je jedným z jednoduchších postupov na porovnanie prvkov dvoch množín. Vytvára dvojice elementov porovnávaných množín a triedi ich do 4 možných kategórií:

- a - párs elementov sa nachádza v oboch množinách v tom istom zhluku.
- b - párs elementov je v rovnakom zhluku v prvej množine a v rozdielnom v druhej množine.
- c - párs elementov je v rozdielnych zhlukoch v prvej množine a v rovnakom zhluku v druhej množine.
- d - párs elementov sa ani v jednej množine nenachádza v tom istom zhluku.

Index zobrazuje pomer prvkov správne v rovnakom/rozdielnom zhluku voči všetkým prípadom, matematicky:

$$R = \frac{a + b}{a + b + c + d}. \quad (4.1)$$

Rand index je možné upraviť (*angl. Adjusted Rand index*) pomocou kontingenčnej tabuľky aby viac zdôraznil vlastnosť podobnosti na ktorej nám záleží. V našej práci používame okrem štandardného Rand indexu aj:

- Hubert and Arabie's adjusted Rand index
- Morey and Agresti's adjusted Rand index

Hodnota indexu je reálne číslo z intervalu $< 0, 1 >$, čím vyššie, tým lepšia podobnosť množín.

4.6.2 Fowlkes-Mallows index

Fowlkes–Mallows index sleduje príslušnosť dvojíc prvkov do rovnakých zhlukov vo vlastnom výsledku zhlukovania a v správnom priloženom riešení. Zatrieduje dvojice do prípadov:

- TP (*angl. True positive*) - prípad, keď sa dvojica prvkov nachádza v rovnakom zhluku v oboch zhlukovaniach.
- FP (*angl. False positive*) - prípad, keď sa dvojica prvkov vo vlastnom výsledku zhlukovania nachádza v rovnakom zhluku, v originálnom výsledku sú ale prvky v zhlukoch odlišných.

- FN (*angl. False negative*) - prípad, keď sa dvojica prvkov vo vlastnom výsledku zhlukovania nachádza v rozličných zhlukoch, v originálnom výsledku sú ale v zhluku rovnakom.
- TN (*angl. True negative*) - prípad, keď sa dvojica prvkov nenachádza spolu v jednom zhluku ani v jednom výsledku zhlukovania.

Výpočet hodnoty indexu matematicky:

$$FM = \sqrt{\frac{TP}{TP + FP} * \frac{TP}{TP + FN}}. \quad (4.2)$$

Oproti Rand indexu lepšie vyhodnocuje málo súvisiace dátá - ak zadáme ako vstup dva nesúvisiace zhlukovacie výsledky, hodnota sa bude blížiť ku nule kvôli nízkemu počtu TP prípadov. Rand index, keďže má v čitateli aj TN prípad, by stále ukazoval vysoké výsledné skóre. Hodnota FM indexu sa pohybuje v intervale $< 0, 1 >$, čím vyššie skóre, tým lepší výsledok zhlukovania.

4.6.3 Jaccard index

Jaccardov index vyjadruje pomer prieniku množín ku ich celkovému objemu. Matematicky:

$$J(m_1, m_2) = \frac{|m_1 \cap m_2|}{|m_1| + |m_2| - |m_1 \cap m_2|}. \quad (4.3)$$

Použitie je vhodné, keď chceme sledovať mieru prekrytie atribútov viacerých množín. Index nezachytáva vzťah prvkov, ktoré sa nenachádzajú ani v jednej z množín. Prípustné hodnoty sú reálne čísla z intervalu $< 0, 1 >$, kde hodnota 1 znamená totožné množiny a hodnota 0 disjunktné.

Kapitola 5

Vyhodnotenie

Kapitola popisuje na akom zariadení sme testovali, aké dáta sme použili a aké výsledky sme s nimi dosiahli v jednotlivých častiach navrhnutého riešenia.

5.1 Testovacie dáta

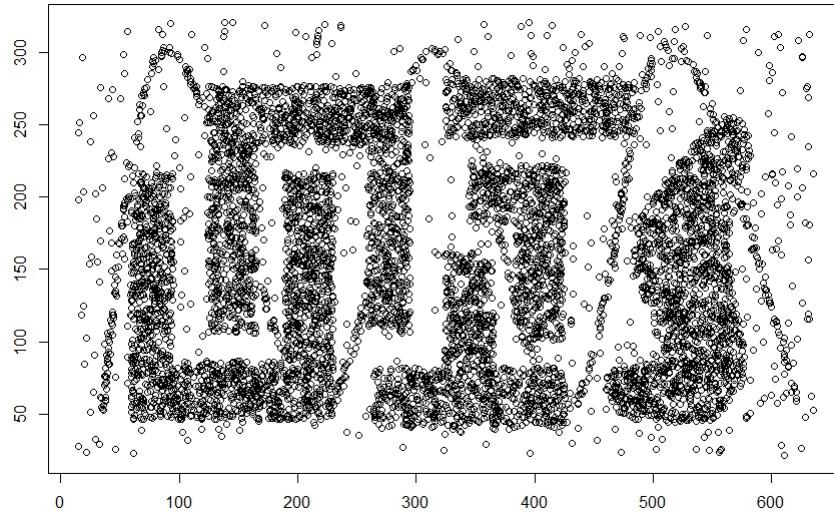
Získať cytometrické dáta na internete nie je náročné, veľké množstvo datasetov prietokovej ale aj hmotnostnej cytometrie sa nachádza napríklad na komunitou udržiavanej stránke “ <https://flowrepository.org/>” [37]. Väčším problémom je nájsť datasety s označením zhľukov vhodným pre použitie v rámci automatickej dátovej analýzy. Jedným z mála voľne dostupných zdrojov takýchto dát je článok [42] zaobrájúci sa porovnávaním výsledkov množstva algoritmov používaných v domébovej oblasti. Zo zdrojov uvedených v tejto práci sme získali 5 testovacích datasetov:

- Levine_13dim.fcs - 167,044 buniek, 13 atribútov, 24 manuálne identifikovaných zhľukov
- Levine_32dim.fcs - 265,627 buniek, 32 atribútov, 14 manuálne identifikovaných zhľukov
- Samusik_all.fcs - 841,644 buniek, 39 atribútov, 24 manuálne identifikovaných zhľukov
- Mosmann_rare.fcs - 396,460 buniek, 14 atribútov, 1 zhľuk - vzácna populácia buniek
- Nilsson_rare.fcs - 44,140 buniek, 13 atribútov, 1 zhľuk - vzácna populácia buniek

Tieto dáta sú normalizované, výsledné zhľuky označené a okrem proteínov (atribútov na základe ktorých zhľukovanie prebieha) obsahujú aj ďalšie informácie súvisiace s cytometrickým meraním.

Ďalším používaným datasetom slúžiacim na overenie základnej funkcionality zhlukovania založeného na hustote je štandardný:

- t4.8k.dat - 8000 objektov, 2 atribúty, bez správneho výsledku zhlukovania



Obr. 5.1: t4.8k.dat - ukážkové dátá pre zhlukovanie založené na hustote

K dispozícii sme počas vývoja mali tiež datasety rozpracované v rámci spolupráce fakulty Informatiky a Informačných Technológií STU s Biomedicínskym centrom Slovenenskej akadémie vied. Jedná sa o niekoľko desiatok až stoviek multidimenziorných datasetov s neznormalizovanými a neoznačenými dátami. Aj keď sme na nich jednotlivé časti riešenia testovali, ich vyhodnotenie nie je súčasťou tejto práce a posúdenie kvality zhlukov vyžaduje doménového experta.

5.2 Testovacia konfigurácia

Na testovanie výsledkov práce sme použili stroj s parametrami:

- CPU: Intel Core i7-2600, 4 jadrá s HT, 3.4GHz (Turbo 3.8GHz)
- Hlavná pamäť: 16GB
- Disk: SSD 500GB
- Operačný systém: Win10 Pro 64bit

Všetky časy behu algoritmov a výsledné indexy sú aritmetickým priemerom troch výsledkov.

5.3 Výsledky zhlukovania

Testovacie dátá svojim obsahom môžeme rozdeliť na dve kategórie:

- Správna klasifikácia čo najväčšieho počtu buniek - mnohozhlukové dátasety Levine_13dim.fcs, Levine_32dim.fcs a Samusik_all.fcs.
- Identifikácia malej vzácnej populácie v rámci homogénneho zvyšku - Mosmann_rare.fcs, Nilsson_rare.fcs.

Pokusy o identifikáciu vzácných buniek neboli úspešné. Nájsť v rámci 14-dimenziálneho 400 000 datasetu (Mosmann_rare.fcs) populáciu o veľkosti 100 buniek bolo nad sily nášho algoritmu. Bližšie dôvody rozvedieme v záverečnej diskusii.

Snahy o zatriedenie buniek do zhlukov algoritmom založeným výhradne na hustote dopadli nasledovne:

Čas behu		24s	7m 53s	2s		48m 16s				
Dataset	Algoritmus (vstupné parametre)	FlowSOM	PhenoGraph (kNN = 30)	K-Means (k = 15, k = 24, k = 35)		Density-based ($\epsilon = 1.1$, minPts = 500)				
Levine_13dim (167 044 x 13)										
Validačné indexy		Rand HA Rand MA Rand Fowlkes-Mallows Jaccard	0.7796 0.3839 0.3839 0.5594 0.3159	0.9756	0.921	0.963	0.944	0.8230		
				0.640	0.781	0.630	0.5407	0.5407		
				0.634	0.781	0.630	0.5408	0.5408		
				0.694	0.805	0.681	0.6809	0.6809		
				0.519	0.668	0.491	0.4636	0.4636		

Obr. 5.2: Porovnanie výsledkov zhlukovania pre Levine_13dim.fcs

Pre porovnanie sme ku nášmu algoritmu založenému na hustote vybrali na ťažiskách založený K-Means (2.5.1), na grafoch založený PhenoGraph (2.4.6) a na samo organizujúcich sa mapách založený FlowSOM (2.4.7). Pri najmenej dimenzionálnom z testovacích datasetov môžeme vidieť významne vyšší čas spracovania, inak ale v dosiahnutých výsledkoch nezaostáva.

KAPITOLA 5. VYHODNOTENIE

Čas behu		46s	38m 31s	9s		1h 36m 33s
Dataset	Algoritmus (vstupné parametre)			K-Means (k = 7, k = 14, k = 21)		
Levine_32dim (265 627 x 32)	FlowSOM	PhenoGraph (kNN = 30)		K-Means (k = 7, k = 14, k = 21)		Density-based ($\epsilon = 1.2$, minPts = 1000)
Validačné indexy						
Rand	0.8821	0.8995	0.942	0.917	0.878	0.7385
HA Rand	0.6681	0.5541	0.789	0.652	0.429	0.4652
MA Rand	0.6682	0.5540	0.789	0.652	0.429	0.4652
Fowlkes-Mallows	0.7595	0.6508	0.825	0.724	0.548	0.6751
Jaccard	0.5863	0.4309	0.700	0.534	0.317	0.4557

Obr. 5.3: Porovnanie výsledkov zhlukovania pre Levine_32dim.fcs

Pri druhom testovacom súbore prudko rastie počet dimenzií čo sa podpíše predovšetkým na výpočtovom čase. Suverénne najlepšie hodnoty validačných indexov dosahuje algoritmus K-Means, keď mu ako parameter k posúvame správny počet zhlukov v datasete. Za zmienku stojí aj fakt, že prednosť nášho prístupu založenom na hustote - filtrovanie šumu - je v týchto datasenoch len zbytočným spomalnením. Správne predspracovaný dataset obsahuje len živé a identifikované bunky. V prípade, že by sme pracovali s nenormalizovanými dátami obsahujúcimi aj záznamy neplatných meraní alebo meraní mŕtvyh buniek, ostatné 3 algoritmy by boli vyvodené z miery viac ako náš.

Čas behu		2m 17s	5h 31m 9s	26s		9h 43m 36s
Dataset Samusik_all (841 644 x 39)	Algoritmus (vstupné parametre)			K-Means (k = 15, k = 24, k = 35)		
Samusik_all	FlowSOM	PhenoGraph (kNN = 30)		K-Means (k = 15, k = 24, k = 35)		Density-based ($\epsilon = 2$, minPts = 1500)
Validačné indexy						
Rand	0.8163	0.9716	0.925	0.914	0.906	0.1407
HA Rand	0.4912	0.8771	0.638	0.535	0.458	0
MA Rand	0.4912	0.8771	0.638	0.535	0.458	0
Fowlkes-Mallows	0.6358	0.8942	0.692	0.619	0.566	0.3751
Jaccard	0.4169	0.8075	0.513	0.404	0.331	0.1407

Obr. 5.4: Porovnanie výsledkov zhlukovania pre Samusik_all.fcs

Najväčším z testovacích datasetov čo do počtu buniek, to do počtu dimenzií je Samusik_all. Expresivita génov (hodnota atribútov) nadobúda veľkosť rádovo v jednotkách, čo vymedzuje všetkých vyše 800 000 bodov do malého a veľmi hustého stavového priestoru. Stanoviť správne hodnoty vstupných parametrov pre náš algoritmus bolo náročné, a kvôli výpočtovej náročnosti sme si nemohli pomôcť inak použitým algoritmom OPTICS (2.5.7). Náš najlepší dosiahnutý výsledok dokázal identifikovať len jeden zhluk, a časť vstupu vyradil ako šum. Na zlepšenie výsledku by bolo potrebné vyladiť predovšetkým parameter ϵ , pri ktorom sa aj drobná zmena v takto hustom priestore viditeľne prejaví.

KAPITOLA 5. VYHODNOTENIE

Celkovo sa dá zhrnúť, že pri správne určených parametroch je okrem výšieho výpočtového času prístup založený na hustote použiteľný. Poskytuje navyše niektoré vlastnosti, ktoré sa pri analýze dát expertovi môžu hodniť - napr. identifikácia zhlukov nepravidelného tvaru, či odfiltrovanie extrémnych hodnôt v podobe šumu.

5.4 Paralelizácia riešenia

Popis paralelizovateľných častí kódu sa nachádza v kapitole (4.3). Implementáciou paralelizácie sme dosiahli nasledovné zrýchlenie:

Dataset	Parametre		Čas behu		Zrýchlenie
	ϵ	minPts	bez OpenMP	s OpenMP	
Levine_13dim (167 044 x 13)	1.1	500	1h 24m 41s	48m 16s	75,45%
Levine_32dim (265 627 x 32)	1.2	1000	2h 20m 24s	1h 36m 33s	45,40%

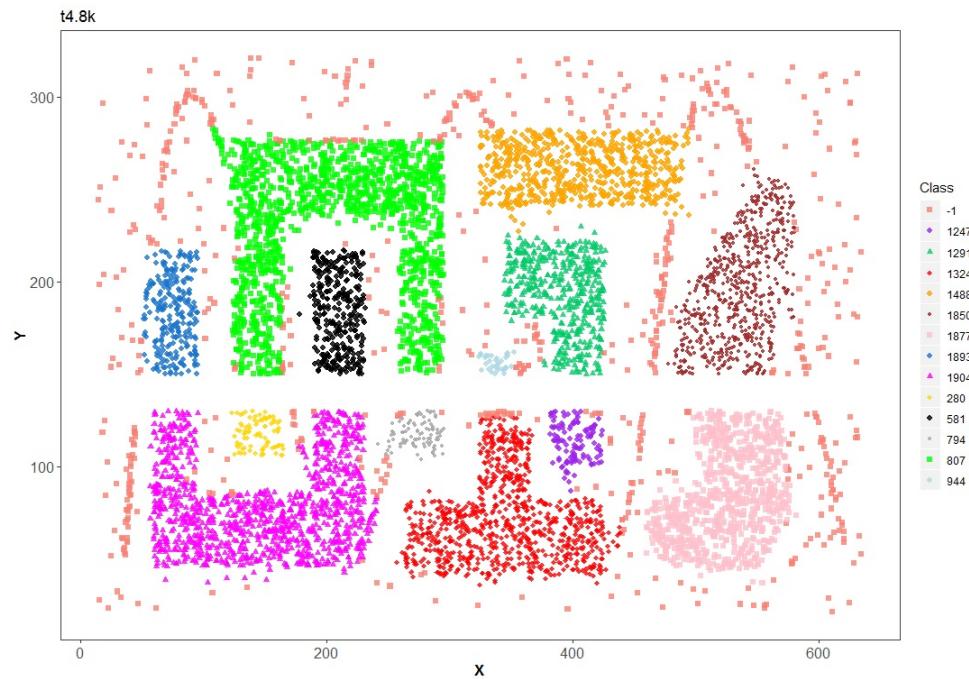
Obr. 5.5: Tabuľka priemerných časov zhlukovania a výsledného zrýchlenia

Význam paralelizácie rastie s počtom dimenzií a celkovým počtom analyzovaných buniek. Pri malých datasetoch môže byť dokonca rézia spojená s prístupom do kritických oblastí väčšia, než výpočtový zisk.

5.5 Dynamické zhlukovanie

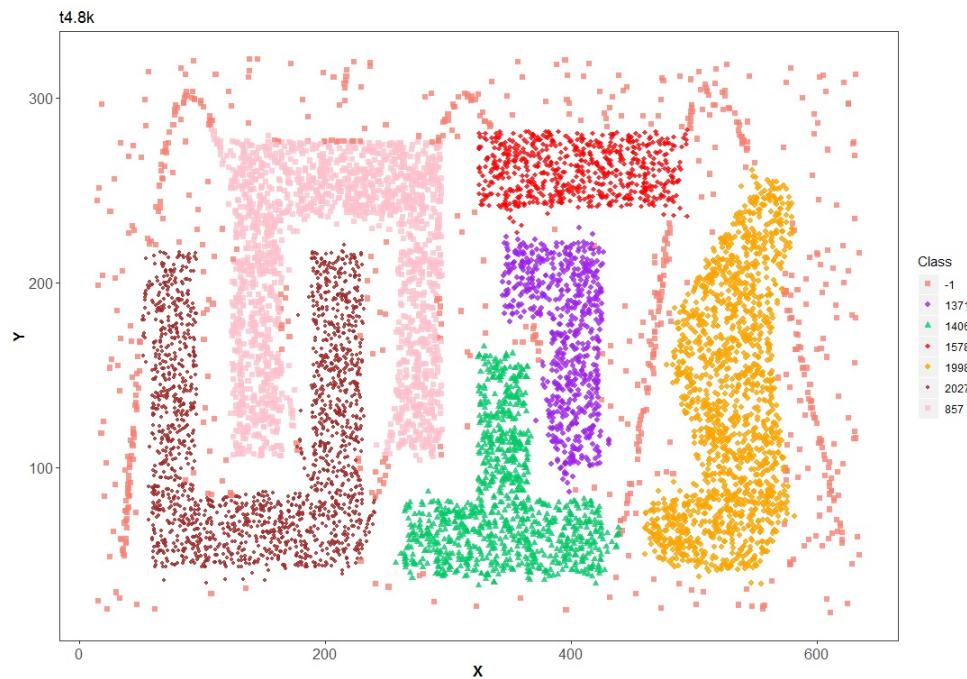
Na vyžiadanie algoritmus uloží výslednú konfiguráciu stavového priestoru (hierarchia buniek vyplnená bodmi) do zvláštneho súboru, štandardne rovnému názvu vstupného súboru s príponou indikujúcou účel súboru. Následne si takyto uložený súbor môže používateľ pred budúcim zhlukovaním načítať, a namiesto začiatku od nuly rozširovať rovno výsledok minulého zhlukovania. Príklad na našej algoritmus testujúcej množine:

KAPITOLA 5. VYHODNOTENIE



Obr. 5.6: t4.8k.dat s páskom chýbajúcich bodov

Môžeme vidieť, že chýbajúci pás je dostatočne široký, aby oddelil body správne patriace do jedného zhluku. V prípade že takýto súbor použijeme ako východzí a dodáme doň body majúce vplyv na usporiadanie zhlukov, výsledkom bude rýchly prepočet pridávaných zhlukov a nový výsledok. Rovnakým spôsobom je možné ubrať body nachádzajúce sa v zhlukovaní.

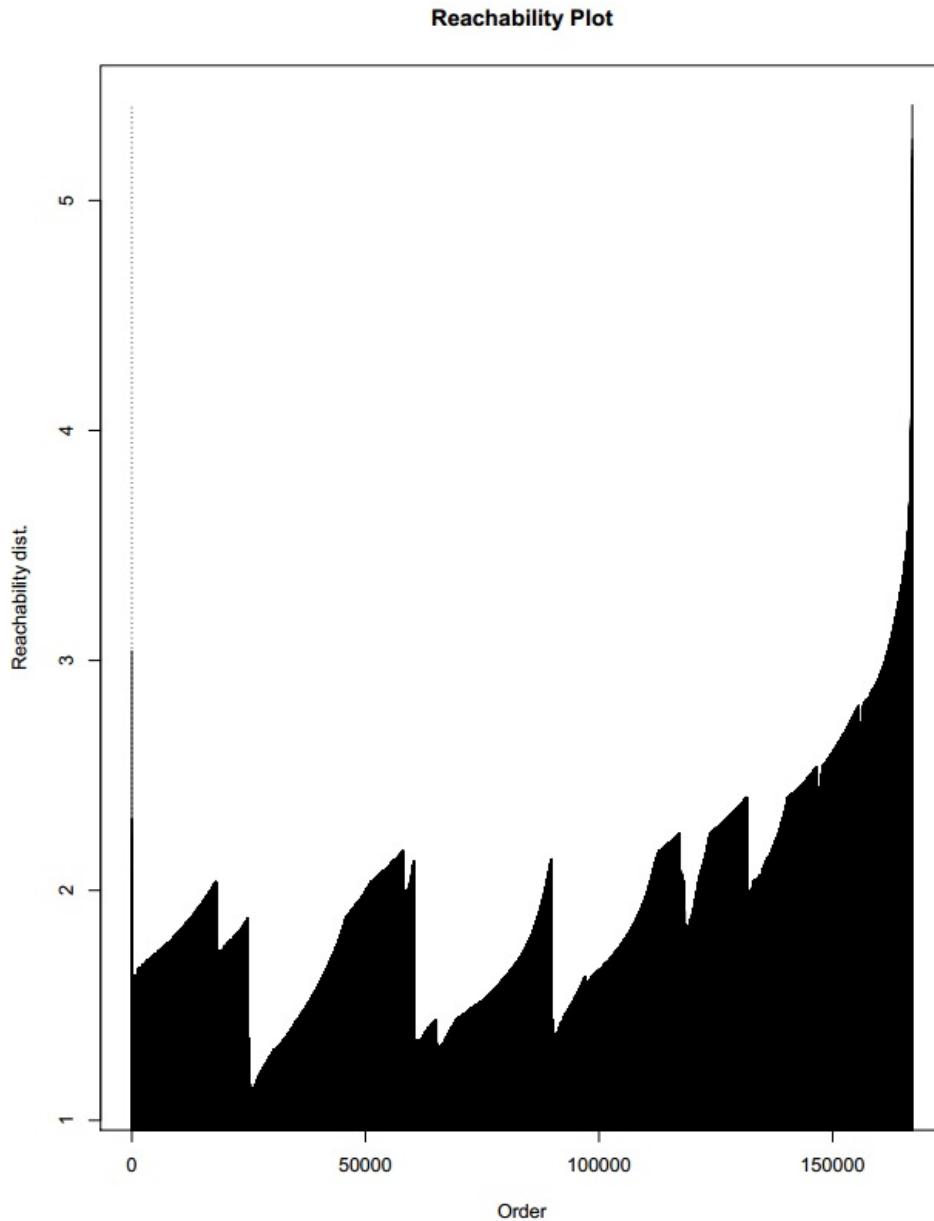


Obr. 5.7: t4.8k.dat s doplnením bodov

Otvoreným problémom pri našom dynamickom zhlukovaní je, ako správne namapovať do priestoru dátá rôznych vstupných súborov. V prípade že by sme postupne prijímalí dátá pre jedného pacienta merané napríklad na dvoch rôznych prístrojoch s rôznou priemernou odchýlkou meranej expressivity, body v priestore by neboli "zladenéä" ku efektu dopĺňania hustoty zhlukov by nedošlo.

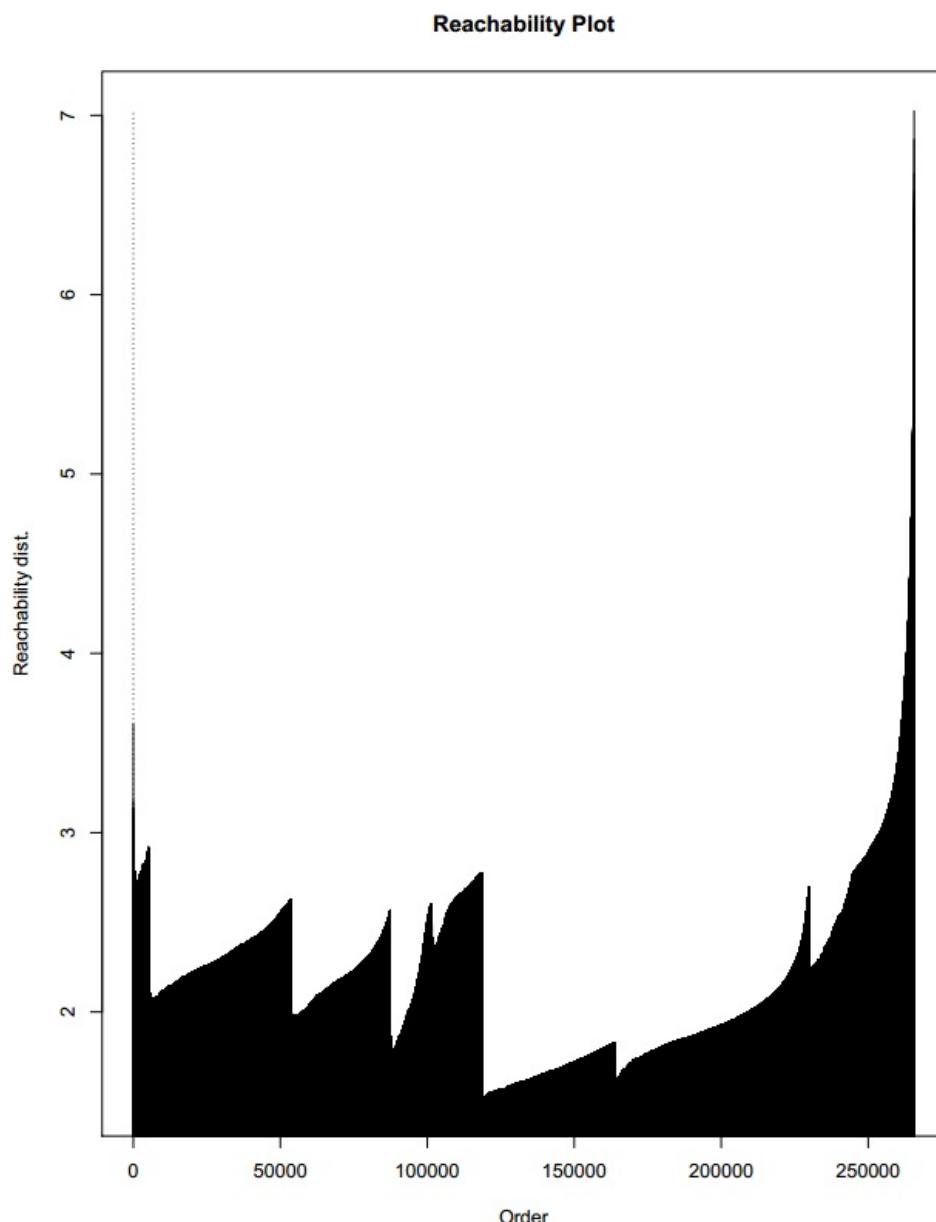
5.6 Odhadovanie parametrov pre zhlukovanie

Najväčším problémom nášho prístupu zhlukovania založeného na hustote je odhadnutie správnych parametrov. Z výsledkov (5.3) vyplýva, že aj v prípade vysoko dimenzionálnych dát je možné nastaviť parametre tak, že v priestore vystúpia hustejšie oblasti ako zhluky. Pre potreby tejto práce sme okrem skúšania viacerých hodnôt a porovnania výsledkov používali ako základ pre odhadovanie grafický výstup algoritmu OPTICS (2.5.7). Tento algoritmus je rovnako ako náš založený na hustote, má však podstatne väčšiu výpočtovú zložitosť. Jeho výstup usporiadavajúci body na základe hustotnej dosiahnutelnosti v priestore vyzerá napr. pre náš testovací dataset Levine_13dim.fcs nasledovne:



Obr. 5.8: OPTICS s $\varepsilon = 10$, $minPts = 500$ pre dataset Levine_13dim.fcs

Na obrázku môžeme vidieť usporiadanie pri sebe ležiacich bodov podľa vzdialenosťi, ktorú potrebujú na dosiahnutie počtu susedov $minPts$. Dokážeme ľahko identifikovať hranicu ε zachytávajúcu taký počet zhlukov, ako je počet vrcholov (*angl. Peak*) v grafe, ktoré pretne. Pri nižšom počte prvkov a dimenzií je táto forma analýzy hustoty datasetu použiteľná, pri vyššom však trvá násobne viac ako samotné zhlukovanie.

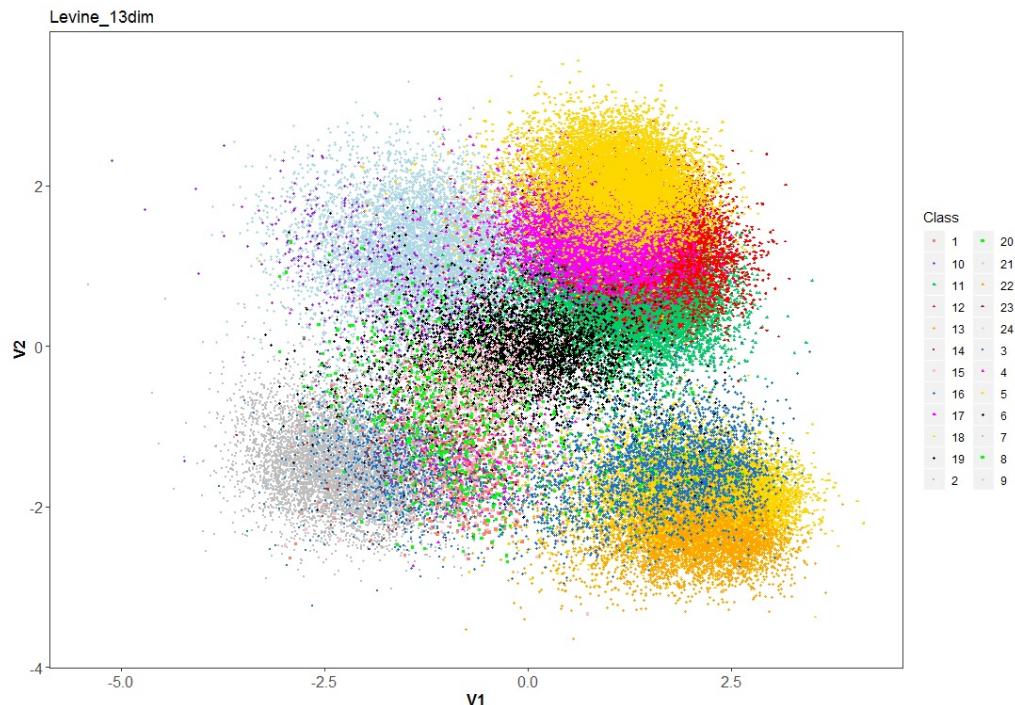


Obr. 5.9: OPTICS s $\varepsilon = 15$, $\text{minPts} = 1000$ pre dataset Levine_32dim.fcs

Pri cytometrických dátach doménový expert môže aj bez takého výstupu stanoviť veľkosť najmenšej očakávanej populácie, či mieru podobnosti prvkov napr. podľa toho, či dataset zachytáva vývojové štádia kmeňových buniek alebo vzorku krvi. Grafický výstup OPTICS algoritmu nám tiež ukazuje, že by malo zmysel pri zhlukovaní použiť prístup umožňujúci nastavenie rôznych hraníc hustoty pre rôzne časti datasetu (tzv. multi density prístup).

5.7 Vizualizácia výsledných zhľukov

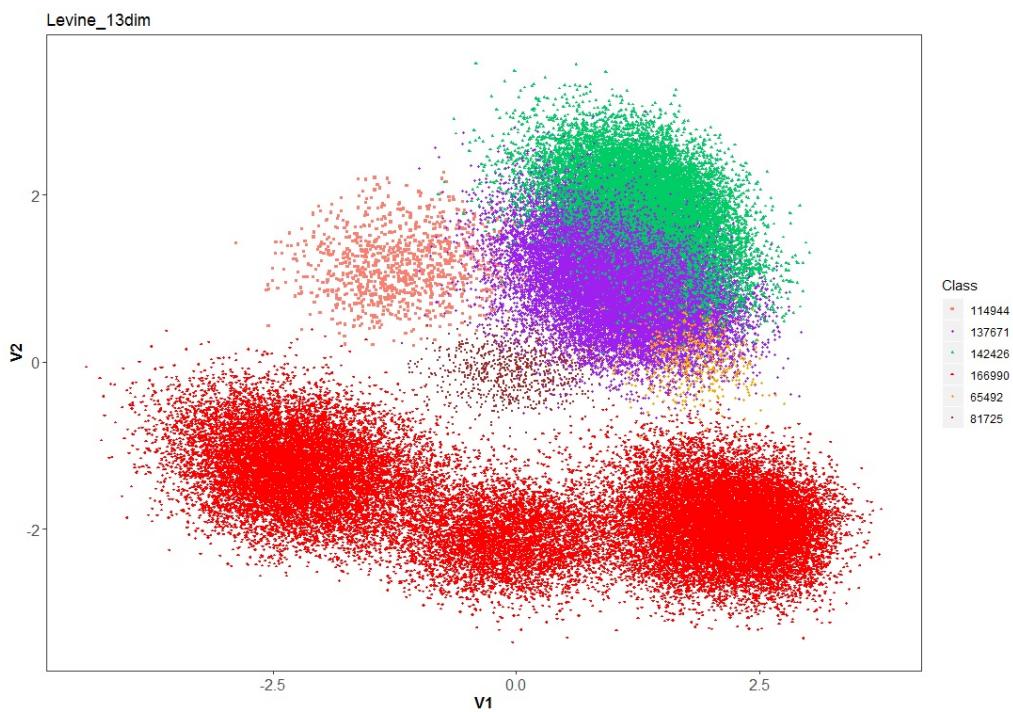
Problémom pri vizualizácii cytometrických dát je, že atribúty nie sú anonymné a nepodstatné, naopak, na základe hodnôt kombinácie atribútov sa stanovuje typ bunky. Môžeme sice použiť prístupy redukujúce dimenzionality ako napr. PCA (4.5.2):



Obr. 5.10: PCA s originálnymi zhľukmi pre dataset Levine_13dim.fcs

Ako môžeme vidieť, zobrazenie je neprehľadné a porovnávať viacero takýchto zobrazení navzájom je napr. ešte vďaka rôznemu farebnému mapovaniu takmer nemožné.

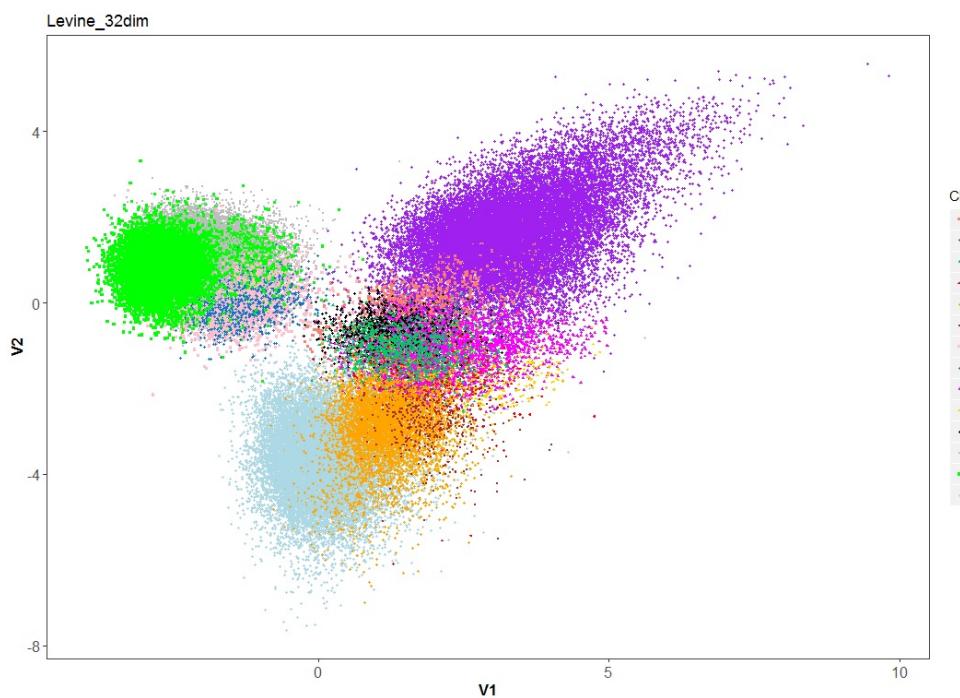
KAPITOLA 5. VYHODNOTENIE



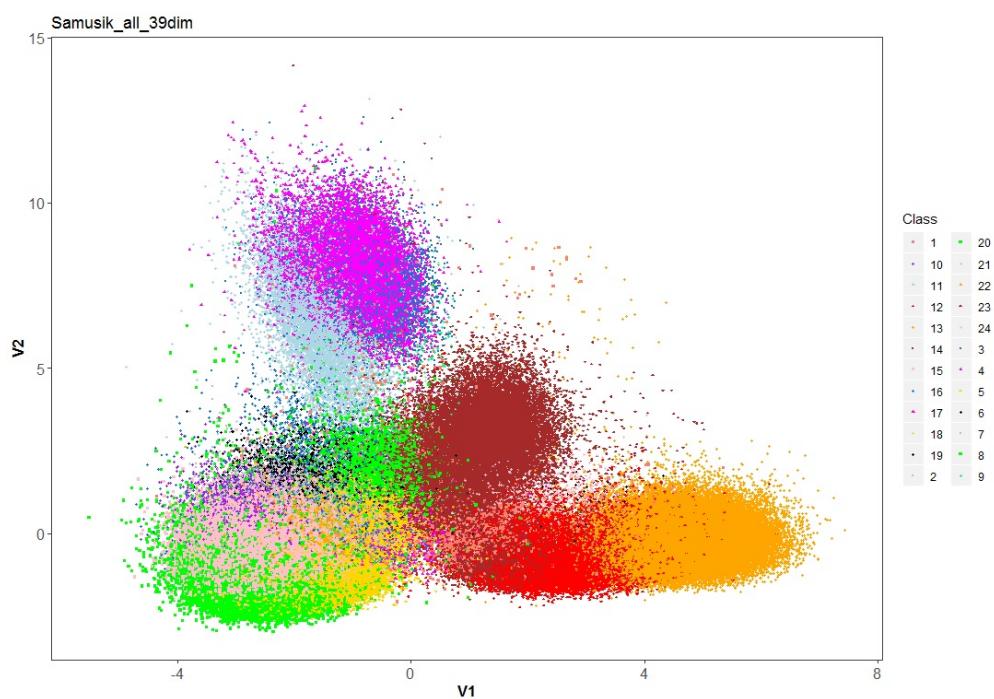
Obr. 5.11: PCA s našimi zhľukmi pre dataset Levine_13dim.fcs

Z obrázka sa dá vyčítať počet zhľukov, informácia o uložení v priestore ale pri transformácii zaniká. Treba zdôrazniť, že sme použili len 13 dimenzionálny dataset a v hmotnostnej cytometrií sú štandardom 40+ dimenzionálne.

KAPITOLA 5. VYHODNOTENIE



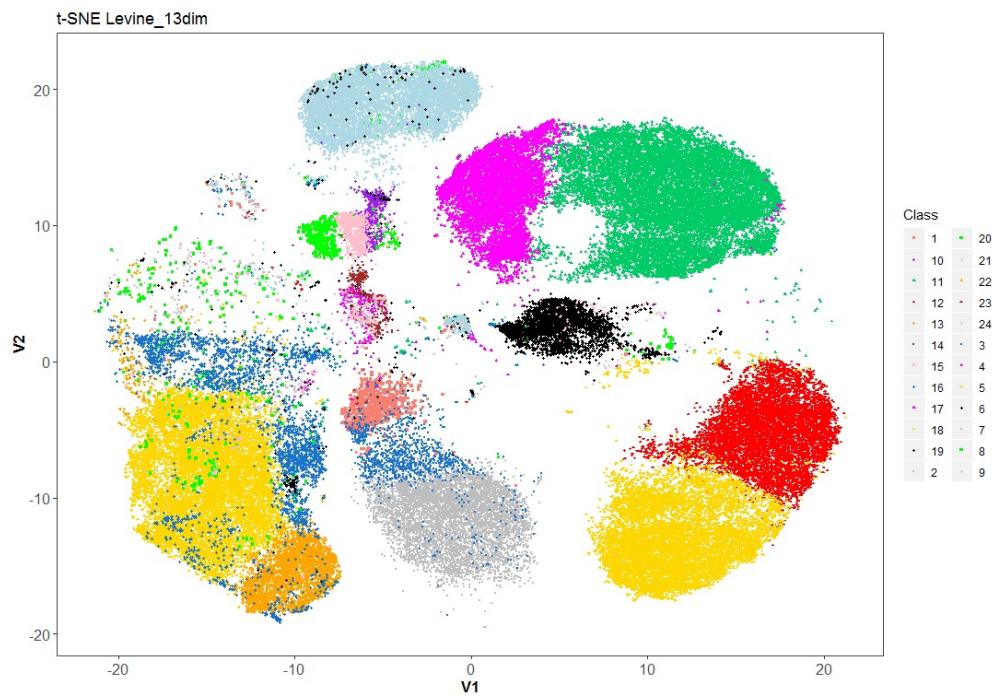
Obr. 5.12: PCA pre 32 dimenzionálny dataset Levine_32dim.fcs



Obr. 5.13: PCA pre 39 dimenzionálny dataset Samusik_all.fcs

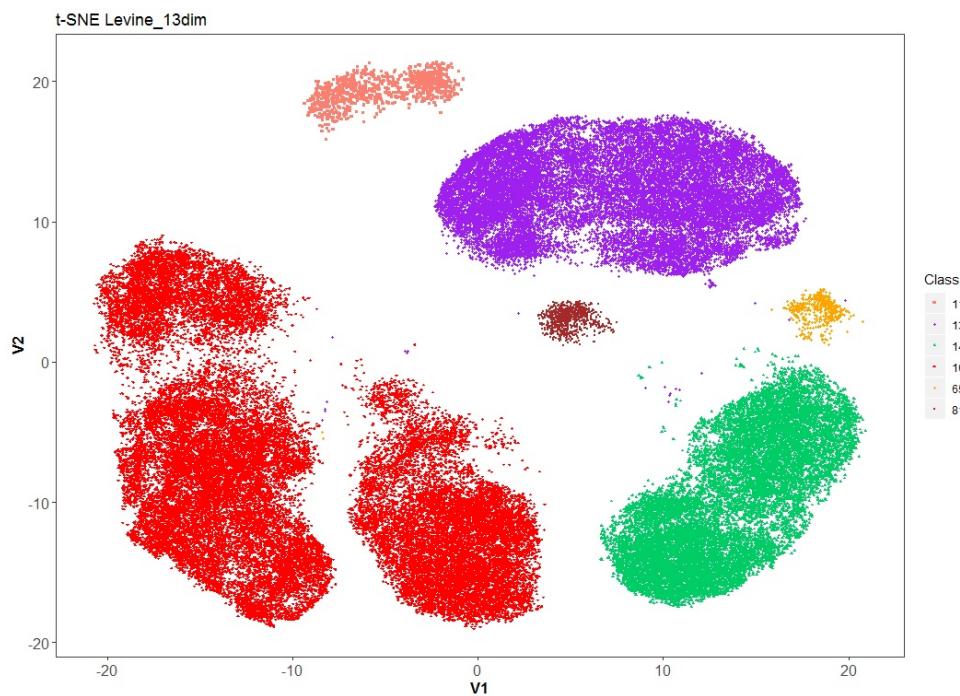
KAPITOLA 5. VYHODNOTENIE

Pokročilejšou formou redukcie dimenzií je algoritmus t-SNE (2.4.3):

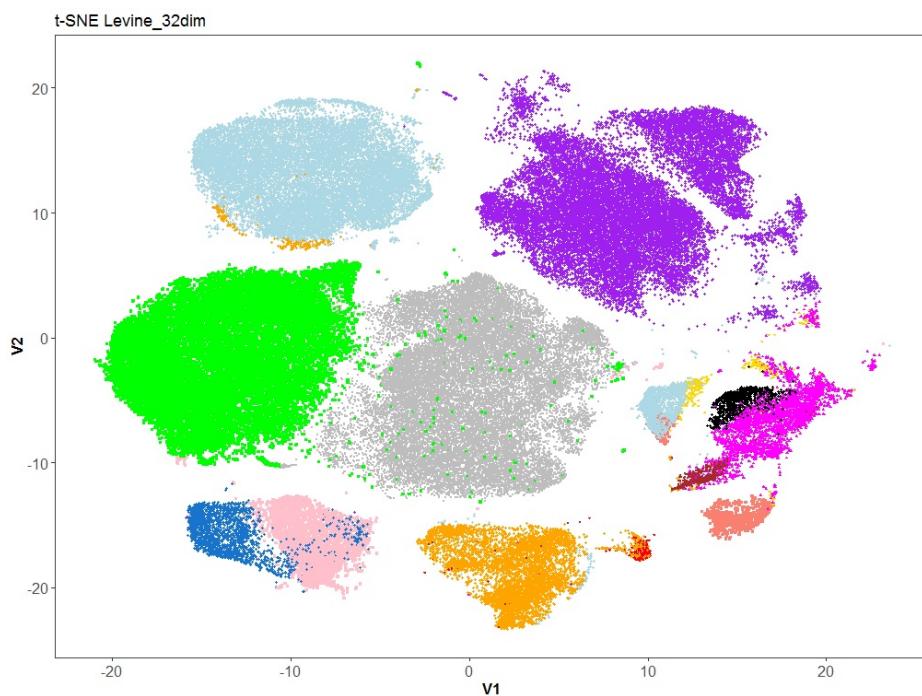


Obr. 5.14: t-SNE s originálnymi zhľukmi pre dataset Levine_13dim.fcs

KAPITOLA 5. VYHODNOTENIE

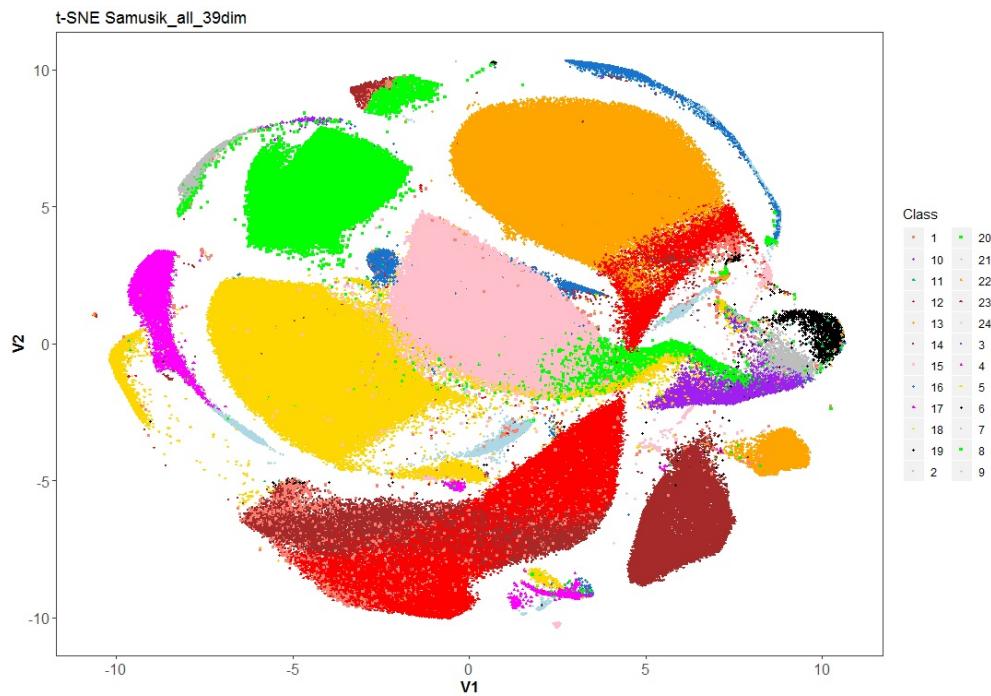


Obr. 5.15: t-SNE s našimi zhľukmi pre dataset Levine_13dim.fcs



Obr. 5.16: t-SNE pre 32 dimenzionálny dataset Levine_32dim.fcs

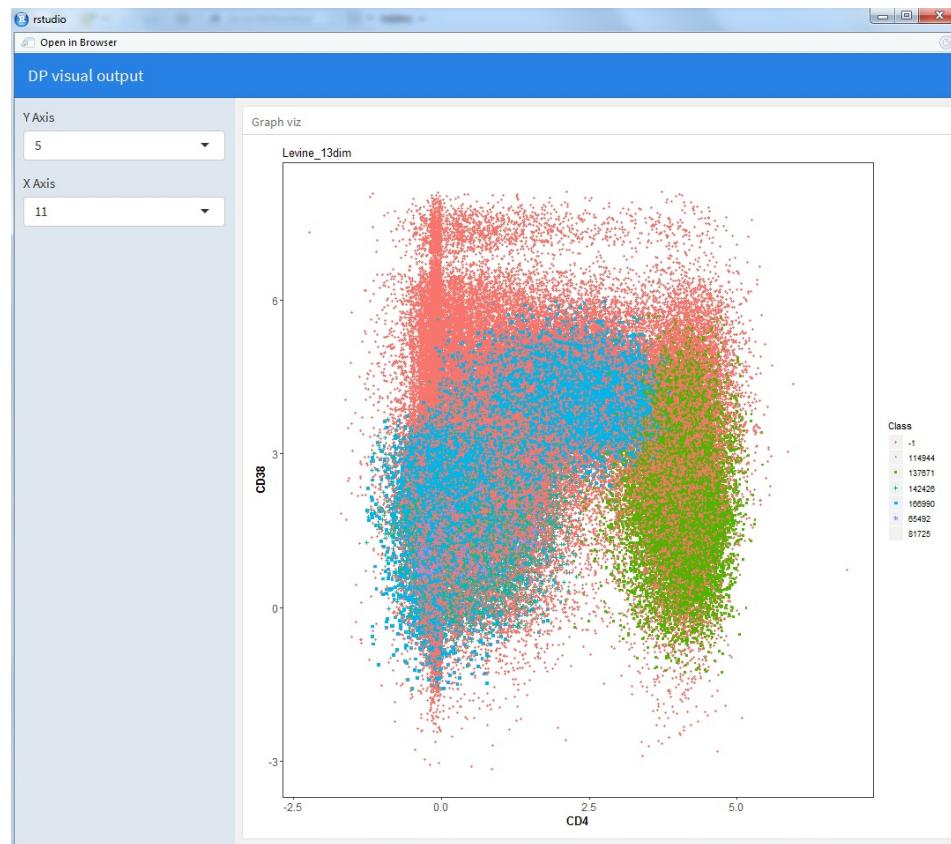
KAPITOLA 5. VYHODNOTENIE



Obr. 5.17: t-SNE pre 39 dimenzionálny dataset Samusik_all.fcs

Napriek významne lepšiemu priestorovému oddeleniu jednotlivých zhľukov na vizualizácii stále nie je možné selektívne manipulovať priestorom tak, aby zobrazil požadovanú skupinu atribútov - expresívít génov. V našom vlastnom riešení pomocou R knižnice “flexdashboard” sme sa rozhodli ponúknut’ používateľovi možnosť vybrať si, na základe ktorých atribútov si želá namapovať množinu na x a y os. Takýmto spôsobom sa uberajú aj vizualizácie algoritmov ako SPADE (2.4.2), či FlowSOM (2.4.7).

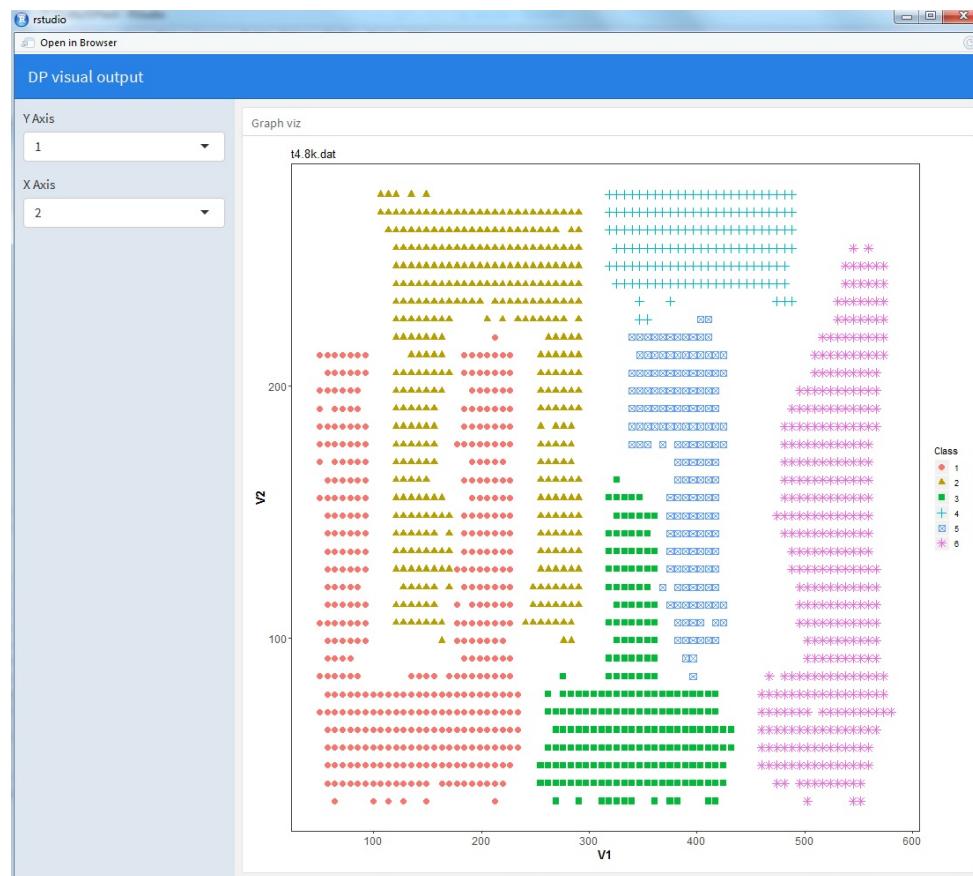
KAPITOLA 5. VYHODNOTENIE



Obr. 5.18: Shiny dashborad pre výsledok zhlukovania

Nečakanou výhodou pri vizualizovaní veľkého počtu bodov je, že v rámci algoritmu pracujeme s hierarchiou buniek ktorým body priradujeme. Vytvára to východiskový bod pre vzorkovanie (*angl.* Downsampling) ktoré vo výslednej vizualizácii priradí každému intervalu v priestore len jeden výsledný bod na základe príslušnosti ku zhluku.

KAPITOLA 5. VYHODNOTENIE



Obr. 5.19: Shiny dashborad pre výsledok zhľukovania z hľadiska priestorových buniek algoritmu

Grafický panel je ľahko rozšíriteľný o ďalšie ovládacie prvky a používateľ si ho môže jednoducho prispôsobiť.

Kapitola 6

Zhodnotenie

V práci sme sa snažili za účelom automatizácie gatingu cytometrických dát vytvoriť prístup zhlukovania založeného na hustote použiteľný aj vo vysoko dimenzionálnych dátach. V oblasti sme nenašli žiadny algoritmus ktorý by sa pri výpočte spoliehal výhradne na informácie o hustote, aj keď napr. FLOCK (2.4.8) si hustotou čiastočne pomáha. Pomocou moderných foriem zhlukovania založeného na hustote používaných v iných doménach (použitie mriežky, hyperdimenzionálne bitmapy, dynamické zhlukovanie) sme zoštrojili algoritmus schopný pri správne zadaných parametroch identifikovať bunkové populácie na úrovni porovnatnej s konkurentmi. Problémom algoritmu je výpočtová zložitosť, ktorú sme sa snažili minimalizovať paralelizáciou riešenia. Aj keď to z testov na niekoľkých označených vysoko dimenzionálnych a objemných datasetoch nemusí byť vidieť, zhlukovanie založené na hustote má svoje silné stránky. Najjednoznačnejšími sú možnosť zachytávať zhluky ľubovoľných tvarov, nevyžadovanie informácie o počte hľadaných zhlukov či automatické odfiltrovanie šumu.

Celá oblasť práce s cytometrickými dátami je stále silno závislá od doménových expertov. Okrem číselných označení bodov v množinách vedecký pracovníci potrebujú interaktívny a pružný náhľad na analyzované dátá. Výstup nášho algoritmu preto vizualizujeme pomocou knižníc jazyka r ako grafický panel s možnosťou mapovania na základe zvolených atribútov a sprehľadnenia dát pomocou vzorkovania na základe výsledných zhlukov.

6.1 Ďalšie možnosti rozvoja

V rámci témy zhlukovania založeného na hustote sme narazili na množstvo ďalších smerov bádania, ktoré by mohli nezanedbateľne vylepšiť aktuálny prototyp. Patria sem:

- Použitie viacerých hustôt pri analýze zhlukov (*angl. Multi density clustering*).

KAPITOLA 6. ZHODNOTENIE

- Použitie ďalších metrík vzdialenosť medzi bodmi v priestore. Pre použitie vo vysokých dimenziách sú mnohé lepšie, než teraz použitá Euklidova (napr. Chebyshevova vzdialenosť).
- Pridanie obmedzení (*angl. constraints*) do procesu zhlukovania [32] vychádzajúceho z algoritmu DBSCAN. Už počas zhlukovania by sa vďaka doménovým znalostiam mohol kláňať väčší dôraz na podobnosť objektov na základe vybraných atribútov.
- Použitie aproximácie na urýchlenie zhlukovania. Jednoznačnú prioritu má ale v tejto doméne presnosť výsledkov.

Kapitola 7

Technická dokumentácia

Výsledkom práce sú:

- C++ projekt s core algoritmom.
- R knižnica obsahujúca vo forme Rcpp core algoritmus. Pre modifikovanie kódu a väčšiu rýchlosť sa ale odporúča spúšťanie priamo C++ programu.
- Zbierka R skriptov na manipuláciu so vstupnými dátami, predspracovanie pre vstup algoritmu, vyhodnotenie a vizualizáciu výsledkov.

Proces manipulácie s našim riešením môžeme rozdeliť na 3 pomyselné časti:

1. Načítanie a analýza fcs dát, ich transformácia do podoby použitejnej ako vstup algoritmu (či už R knižnice, alebo surového vstupu do C++ programu).
2. Vykonanie / vykonávanie zhlukovania.
3. Práca s výsledkami.

Nevyhnutnou podmienkou pre vykonanie hociktorej z častí je nainštalovanie jazyka R (<https://www.r-project.org/>), ideálne v aktuálnej verzií 3.6.0. Ďalším odporúčaným softvérom je:

- R Studio (<https://www.rstudio.com/products/rstudio/>), ideálne v aktuálnej verzií 1.2.1335.
- C++ kompilátor s podporou OpenMP.
- C++ IDE podľa vlastnej chuti.

7.1 Načítanie a analýza fcs dát

Pre načítanie fcs súborov v R jazyku potrebujeme nainštalovanú knižnicu "flowCore". Následne si vieme uložiť do premennej ľubovoľný vstupný cytometrický súbor ako:

```
fcs_read = read.FCS("path");
```

Matica expresivity génov v načítanom súbore je premenná objektu fcs.read@exprs. V fcs.read@parameters@data\$name si vieme pozrieť, ktorý atribút predstavuje ktorý proteín. Ak chceme získať predstavu o rozložení hustoty v stavovom priestore, môžeme použiť algoritmus OPTICS. Ten sa nachádza v knižnici "dbscan". Kód pre túto časť manipulácie s dátami pokrýva skript "readFCSplotOPTICS.R" priložený na elektronickom nosiči.

7.2 Vykonanie zhlukovania

Zhlukovanie je možné spustiť buď ako funkciu R knižnice DPDensityBased::metaCluster() s prípustnými argumentmi:

- file - meno vstupného súboru.
- eps - parameter eps pre zhlukovanie.
- minPts - parameter minPts pre zhlukovanie.
- outputFile - explicitné stanovenie názvu výstupného súboru.
- saveStatusOutputFile - súbor do ktorého chceme dostať výsledný stav zhlukovania pre budúce dynamické použitie.
- loadFromFile - súbor z ktorého chceme načítať stav zhlukovania pred začatím zhlukovania.

Alebo klasickým spuštením DPDensityBased.exe so vstupnými parametrami rovnakými ako vyššie pri R volaní. priložené binárky programu nemusia fungovať pod inou architektúrou alebo operačným systémom a odporúča sa radšej si vybuildovať projekt nanovo.

7.3 Práca s výsledkami

Po prebehnutí algoritmu a obdržaní výstupného súboru máme niekoľko možností:

- Vizualizácia cez dashboard priložený v DPDensityBased knižnici vočlaním ::visualize() s parametrom premenou výstupného zhlukovania.

KAPITOLA 7. TECHNICKÁ DOKUMENTÁCIA

- Vyhodnotenie kvality zhľukovania validačnými indexami Rand, Jaccard a Fowlkes-Mallows. Nástrojom na to je priložený skript “process-ResultIndexes.r”.
- Priama vizualizácia cez ggplot knižnicu, nástrojom je priložený skript “visualization.R”.
- Vizualizácia pomocou PCA alebo t-SNE, priložené skripty “pcaScript.Rä” “Rtsne.R”
- Pre skúsenejších možnosť porovnať hodnoty s hodnotami iných zhľukovacích algoritmov. V repertoári máme skript pre K-Means, flowSOM a RphenoGraph, nainštalovať si ich ale je potrebné manuálne, keďže Bioconductor zatiaľ oficiálne nepodporuje najnovšiu verziu Rtools.

Literatúra

- [1] *Principal Component Analysis*. Springer-Verlag, 2002.
- [2] Diana I. Albu, Danielle Califano, and Dorina Avram. Flow cytometry analysis of transcription factors in t lymphocytes. In *Methods in Molecular Biology*, pages 377–390. Humana Press, 2010.
- [3] El-Ad David Amir. visne and wanderlust, two algorithms for the visualization and analysis of high-dimensional single-cell data. 2014.
- [4] Mihael Ankerst, Markus M. Breunig, Hans peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. pages 49–60. ACM Press, 1999.
- [5] Hans-Hermann Bock. Origins and extensions of the k-means algorithm in cluster analysis. *Electronic Journal for History of Probability and Statistics*, 4, December 2008.
- [6] Erik Bongcam-Rudloff, Teresa Attwood, Andreas Gisel, and Nils-Einar Eriksson. Concepts, historical milestones and the central place of bioinformatics in modern biology: A european perspective. In Mahmood A. Mahdavi, editor, *Bioinformatics*, chapter 1. InTech, Rijeka, 2011.
- [7] Thapana Boonchoo, Xiang Ao, and Qing He. An efficient density-based clustering algorithm for higher-dimensional data. *CoRR*, abs/1801.06965, 2018.
- [8] Robert V. Bruggner, Bernd Bodenmiller, David L. Dill, Robert J. Tibshirani, and Garry P. Nolan. Automated identification of stratifying signatures in cellular subpopulations. *Proceedings of the National Academy of Sciences*, 111(26):E2770–E2777, jun 2014.
- [9] Joseph A. DiGiuseppe, Jolene L. Cardinali, William N. Rezuke, and Dana Peer. PhenoGraph and viSNE facilitate the identification of abnormal t-cell populations in routine clinical flow cytometric data. *Cytometry Part B: Clinical Cytometry*, sep 2017.
- [10] Jeff Erickson. On the relative complexities of some geometric problems. In *CCCG*, 1995.

LITERATÚRA

- [11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.
- [12] Junhao Gan and Yufei Tao. Dbscan revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’15, pages 519–530, New York, NY, USA, 2015. ACM.
- [13] Junhao Gan and Yufei Tao. Dynamic density based clustering. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD ’17, pages 1493–1507, New York, NY, USA, 2017. ACM.
- [14] Sofie Van Gassen, Britt Callebaut, Mary J. Van Helden, Bart N. Lambrecht, Piet Demeester, Tom Dhaene, and Yvan Saeys. FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A*, 87(7):636–645, jan 2015.
- [15] Niklas Markus Gericke and Mariana Hagberg. Definition of historical models of gene function and their relation to students’ understanding of genetics. *Science & Education*, 16(7):849–881, Aug 2007.
- [16] Sean Gilpin, Buyue Qian, and Ian Davidson. Efficient hierarchical clustering of large high dimensional datasets. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM ’13, pages 1371–1380. ACM, 2013.
- [17] Markus Götz, Christian Bodenstein, and Morris Riedel. HpdbSCAN: Highly parallel dbSCAN. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, MLHPC ’15, pages 2:1–2:10, 2015.
- [18] Anthony J.F. Griffiths, Jeffrey H. Miller, David T. Suzuki, Richard C. Lewontin, and William M. Gelbart. *Introduction to Genetic Analysis*. W. H. Freeman, 2000.
- [19] M. KODÍČEK. výkladový slovník [online]. http://vydavatelstvi.vscht.cz/knihy/uid_es-002/ebook.html, 2007.
- [20] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1 – 6, 1998.
- [21] Ch. Bala Koteshwariah, N. Raghu Kisore, and V. Ravi. A fuzzy version of generalized dbSCAN clustering algorithm. In *Proceedings of the Second ACM IKDD Conference on Data Sciences*, CoDS ’15, pages 128–129. ACM, 2015.

LITERATÚRA

- [22] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.
- [23] Vašek P. Laurinec P. Analýza zhlukov velkých dátových množín. <http://opac.czrp.sk/?fn=detailBiblioForm&sid=82D5162064037F62A49985F7A2EE>, 2017.
- [24] Ann Lévesque, Alain Paquet, and Michel Pagé. Measurement of tumor necrosis factor activity by flow cytometry. *Cytometry*, 20(2):181–184, jun 1995.
- [25] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.
- [26] Olga Ornatsky, Dmitry Bandura, Vladimir Baranov, Mark Nitz, Mitchell A. Winnik, and Scott Tanner. Highly multiparametric analysis by mass cytometry. *Journal of Immunological Methods*, 361(1-2):1–20, sep 2010.
- [27] Md. Mostofa Ali Patwary, Nadathur Satish, Narayanan Sundaram, Fredrik Manne, Salman Habib, and Pradeep Dubey. Pardicle: Parallel approximate density-based clustering. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’14, pages 560–571, 2014.
- [28] Qiu Peng. Toward deterministic and semiautomated spade analysis. *Cytometry Part A*, 91(3):281–289.
- [29] Piotr Pozarowski, Elena Holden, and Zbigniew Darzynkiewicz. Laser scanning cytometry. In *Cell Imaging Techniques*, pages 165–192. Humana Press, 2006.
- [30] Yu Qian, Chungwen Wei, F. Eun-Hyung Lee, John Campbell, Jessica Halliley, Jamie A. Lee, Jennifer Cai, Y. Megan Kong, Eva Sadat, Elizabeth Thomson, Patrick Dunn, Adam C. Seegmiller, Nitin J. Karandikar, Christopher M. Tipton, Tim Mosmann, Iñaki Sanz, and Richard H. Scheuermann. Elucidation of seventeen human peripheral blood b-cell subsets and quantification of the tetanus response using a density-based method for the automated identification of cell populations in multidimensional flow cytometry data. *Cytometry Part B: Clinical Cytometry*, 78B(S1):S69–S82, 2010.
- [31] Peng Qiu, Erin F Simonds, Sean C Bendall, Kenneth D Gibbs, Robert V Bruggner, Michael D Linderman, Karen Sachs, Garry P Nolan, and Sylvia K Plevritis. Extracting a cellular hierarchy from high-dimensional

LITERATÚRA

- cytometry data with SPADE. *Nature Biotechnology*, 29(10):886–891, oct 2011.
- [32] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. *the 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 216–223, 2007.
 - [33] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.
 - [34] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbSCAN. *ACM Trans. Database Syst.*, 42(3):19:1–19:21, July 2017.
 - [35] Howard M. Shapiro. *Practical Flow Cytometry*. Wiley-Liss, 2003.
 - [36] Howard M. Shapiro. The evolution of cytometers. *Cytometry Part A*, 58A(1):13–20, February 2004.
 - [37] Josef Spidlen, Karin Breuer, Chad Rosenberg, Nikesh Kotecha, and Ryan R. Brinkman. Flowrepository: A resource of annotated flow cytometry datasets associated with peer-reviewed publications. *Cytometry Part A*, 81A(9):727–731, 2012.
 - [38] Ting Su and J. Dy. A deterministic method for initializing k-means clustering. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 784–786, Nov 2004.
 - [39] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975.
 - [40] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. 9:2579–2605, 11 2008.
 - [41] Shimei Wang, Yun Liu, and Bo Shen. Mdbscan: Multi-level density based spatial clustering of applications with noise. In *Proceedings of the The 11th International Conference on The Changing Face of Knowledge Management Impacting Society*, KMO ’16, pages 21:1–21:5. ACM, 2016.
 - [42] Lukas M. Weber and Mark D. Robinson. Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. *Cytometry Part A*, 89(12):1084–1096, 2016.

LITERATÚRA

- [43] Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Min. Knowl. Discov.*, 3(3):263–290, September 1999.

Príloha A

Elektronické médium

Na disku sa nachádzajú nasledovné položky:

- priečinok /DP obsahuje elektronickú verziu práce vo formáte pdf.
- priečinok /Data obsahuje vstupné súbory spomenuté v práci.
- priečinok /CppCLionProject obsahujúci zdrojový kód C++.
- priečinok /RPackage obsahujúci zdrojové súbory s kódom C++ upraveným do R knižnice a sprievodné skripty slúžiace na manipuláciu s dátami.
- súbor readme.txt obsahujúci technickú dokumentáciu

Príloha B

Plán práce na riešení projektu

DP1 - LS 2017/2018:

- 1-5 týždeň: na základe prieskumu DP0 ďalšie vyhľadávanie a študovanie odbornej literatúry
- 6-8 týždeň: ujasňovanie si súvislostí a návrh prototypu
- 9-10 týždeň: implementácia jednoduchého prototypu zhlukovacieho algoritmu
- 11-12 týždeň: písanie dokumentu DP1
- Vyjadrenie ku napĺňaniu plánu: Práca na projekte sa rozbiehala pozvoľna, definícia cesty na dosiahnutie cieľov zadania práce sa často menila. Celkový objem práce mierne zaostával za očakávaniami a výslednú analýzu bude treba počas ďalšieho vývoja dopĺňať.

DP2 - ZS 2018/2019:

- 1-6 týždeň: implementácia pokročilejšieho prototypu algoritmu
- 7-8 týždeň: riešenie implementačných problémov
- 9-10 týždeň: konzultovanie možností ďalšieho postupu implementácie
- 11-12 týždeň: písanie dokumentu DP2
- Vyjadrenie ku napĺňaniu plánu: Tempo práce bolo vďaka zväčšenej frekvencií konzultácií vyššie ako pri DP1, nepodarilo sa ale stihnúť pokročilejšie otestovať implementovaný prototyp.

PRÍLOHA B. PLÁN PRÁCE NA RIEŠENÍ PROJEKTU

DP3 - LS 2018/2019:

- 1-3 týždeň: finalizácia implementácie riešenia - zakomponovanie parallelizácie, dynamického spracovania a bitových máp pre prácu s hramy vo vyšších dimenziách.
- 3-5 týždeň: rozsiahle testovanie a porovnávanie výsledkov s inými algoritmami
- 6-12 týždeň: validácia výsledkov a finalizácia dokumentu
- Vyjadrenie ku napĺňaniu plánu: Finalizácia implementácie sa očakávane nečakane predĺžila. Testovanie a porovnávanie s podobnými algoritmami bolo brzdené neaktualizovanými r knižnicami projektu Biocconductor. Časť práce na DP sa našťastie prenesla na školské predmety Testovanie softvéru a Vizualizácia dát. Celkové hodnotenie semestra je, že som bol celý čas týždeň pozadu.