VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA VLASTNOSTÍ ZHLUKOVACÍCH ALGORITMOV

ANALYSIS OF CLUSTERING METHODS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ŠIMON LIPTÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IVANA BURGETOVÁ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Lipták Šimon, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Analýza vlastností shlukovacích algoritmů**
Analysis of Clustering Methods
Kategorie: Data mining

Zadání:

1. Prostudujte různé shlukovací algoritmy a jejich teoretické vlastnosti.
2. Po dohodě s vedoucí vyberte vhodné implementace dostupných shlukovacích algoritmů, jejichž vlastnosti budou dále analyzovány.
3. Vyhledejte dostupné datové sady, které by byly vhodné pro analýzu shlukovacích algoritmů.
4. Po dohodě s vedoucí vyberte vhodné datové sady pro testování vlastností shlukovacích algoritmů.
5. Navrhněte a implementujte aplikaci pro testování vybraných shlukovacích algoritmů.
6. S využitím vybraných datových sad otestujte různé vlastnosti vybraných shlukovacích algoritmů. Vytvořte vhodnou prezentaci zjištěných vlastností.
7. Zhodnoťte dosažené výsledky a porovnejte je s očekávanými teoretickými předpoklady.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1
- Aggarwal, Ch. C., Reddy, Ch. K.: Data Clustering Algorithms and Applications, CRC Press, 2014, ISBN 978-1-4665-5821-2

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Burgetová Ivana, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 23. října 2018

Analýza vlastností zhlukovacích algoritmov

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pani Ing. Ivany Burgetovej, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Šimon Lipták
13. mája 2019

Poďakovanie

Touto cestou by som sa chcel poďakovať pani Ing. Ivane Burgetovej, Ph.D. za poskytnutie odbornej literatúry a odbornú pomoc pri písaní tejto diplomovej práce.

Abstrakt

Cieľom tejto diplomovej práce bolo zoznámiť sa so zhlukovou analýzou, zhlukovacími metódami a ich teoretickými vlastnosťami. Bolo potrebné si vybrať zhlukovacie algoritmy, ktorých vlastnosti budú analyzované, vyhľadať a vybrať dátové sady, na ktorých budú tieto algoritmy spúšťané. Taktiež cieľom bolo navrhnutie a implementovanie aplikácie, ktorá bude vyhodnocovať a zobrazovať výsledky zhlukovania vhodným spôsobom. Následne bola vykonaná analýza dosiahnutých výsledkov a ich porovnanie s teoretickými predpokladmi.

Abstract

The aim of this master's thesis was to get acquainted with cluster analysis, clustering methods and their theoretical properties. It was necessary select clustering algorithms whose properties will be analyzed, find and select data sets on which these algorithms will be triggered. Also, the goal was to design and implement an application that will evaluate and display clustering results in an appropriate manner. The last step was to analyze the results and compare them with theoretical assumptions.

Kľúčové slová

zhluk, zhluková analýza, zhlukovacie metódy, požiadavky na zhlukovú analýzu, ohodnotenie zhlukovania, aplikácia pre zhlukovanie dát

Keywords

cluster, cluster analysis, clustering methods, requirements for cluster analysis, evaluation of clustering, application for data clustering

Citácia

LIPTÁK, Šimon. *Analýza vlastností zhlukovacích algoritmov*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivana Burgetová, Ph.D.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Diplomová práca sa zaoberá analýzou vlastností zhlukovacích algoritmov, medzi ktoré patrí časová náročnosť a ohodnotenie kvality zhlukovania pomocou siluetového koeficientu. V tejto diplomovej práci bola vytvorená aplikácia pre prácu s algoritmami a dátovými sadami. Diplomová práca sa skladá z 10 častí, ktoré sú popísané v nasledujúcich odstavcoch.

V druhej časti je popísaná teória zhlukovej analýzy, termíny, ktoré boli použité pri tejto analýze. Ďalej sú popísané požiadavky na zhlukovú analýzu, ako je škálovatelnosť, schopnosť pracovať s rôznymi typmi atribútov, schopnosť správne určiť vstupné parametre, schopnosť vytvárania zhlukov rôznych tvarov, schopnosť spracovávať dáta, ktoré obsahujú šum a odľahlé hodnoty, necitlivosť na poradie vstupných dát a schopnosť spracovávať novovložené dáta, schopnosť vytvárať zhluky na základe rôznych obmedzení, interpretovateľnosť a použiteľnosť, schopnosť spracovávať vysoko-dimenzované dáta, rozdeľovacie kritéria, separácia zhlukov, podobnosť zhlukov a zhlukovací priestor. Taktiež sú tu popísané dátové štruktúry, ako sú dátová matica a podobnostná matica a premenné, ktoré sa využívajú pri zhlukovaní dát, medzi ktoré patria: binárne, intervalové, kategorické, ordinálne, pomerové a premenné zmiešaných typov. Ďalej je rozpracované ohodnotenie zhlukovacích metód, kde sú popísané metódy, ktoré určujú ako kvalitné zhluky vytvorili algoritmy.

V tretej časti sú rozpracované jednotlivé zhlukovacie metódy, ktoré sú rozdelené na metódy založené na rozdeľovaní (k-Means, k-Medoids a CLARANS), hierarchické metódy (Chameleon), metódy založené na modeloch (Expectation-Maximization a SOM), metódy založené na hustote (DBSCAN) a ďalšie menej známe zhlukovacie metódy (metódy založené na mriežke, metódy pre zhlukovanie vysoko-dimenzovaných dát, metódy pre zhlukovanie grafových a sieťových dát a metódy založené na obmedzeniach).

V štvrtej časti sú popísané dátové sady, ktoré boli vybrané a takisto algoritmy a implementácie týchto algoritmov, ktoré boli použité v aplikácii. Celkovo bolo vybraných 5 dátových sád a 3 algoritmy, medzi ktoré patria Chameleon, DBSCAN a SOM.

Piata časť sa zaoberá návrhom a implementáciou aplikácie. V tejto časti sú popísané jednotlivé fázy návrhu, prvky, ktoré aplikácia používa a spôsob implementácie aplikácie.

V šiestej časti sú rozobrané experimenty, ktoré boli vykonané pri hľadaní správnych parametrov pre jednotlivé algoritmy, tak aby výsledky pre každú dátovú sadu boli čo najlepšie. Experimenty sú rozpísané pre každý algoritmus a pre každú dátovú sadu.

Siedma časť sa zaoberá overovaním časovej zložitosti algoritmov pre každú dátovú sadu a porovnávaním výsledkov s teoretickými znalosťami.

V ôsmej časti sú popísané experimenty s overovaním kvality zhlukovania pomocou siluetového koeficientu. V tejto časti je zhodnotené, či siluetový koeficient je vhodný pre ove-

rovanie, či sa zhluky dát vytvorili správne a či je možné pomocou siluetového koeficientu určiť správne parametre algoritmov.

Posledná časť zahŕňa porovnanie rôznych implementácií algoritmu Chameleon, ktoré boli použité v tejto diplomovej práci.

Kapitola 2

Analýza témy

V tejto kapitole sa vysvetľuje termín zhluková analýza. Ďalej popisuje požiadavky na zhlukovú analýzu, aké typy dát sa využívajú v zhlukovej analýze a ako sa zhlukovanie vyhodnocuje.

2.1 Zhluková analýza

Predtým, ako sa začneme venovať zhlukovej analýze, musíme si priblížiť, čo to vlastne je zhluk. Zhluk je množina dát alebo objektov, ktoré sú podobné ostatným objektom alebo dátam v množine a súčasne sú odlišné od dát z ostatných zhlukov [5]. Zhluková analýza alebo zhlukovanie je proces hľadania najlepšieho rozdelenia objektov do skupín, teda zhlukov. Výsledkom samotného zhlukovania je teda množina zhlukov. Je známe, že rôzne zhlukovacie metódy môžu generovať rozdielne výsledné zhluky na rovnakej dátovej sade [12]. Na základe tohto zistenia je zhlukovanie užitočné pri objavovaní nových, zatiaľ neznámych, zhlukov pri daných dátových sadách.

Zhluková analýza sa využíva v rôznych odvetviach a aplikáciách, medzi ktoré patrí rozpoznávanie obrazových vzorov, biológia, bezpečnosť, prieskum trhu, dátová analýza, spracovanie obrazu. Zhlukovanie môže byť využité aj pri predspracovaní dát pre ďalšie algoritmy, ktoré sa zameriavajú hlavne na klasifikáciu a charakterizáciu a využívajú práve vytvorené zhluky. Webové vyhľadávanie patrí medzi ďalšiu kategóriu, v ktorej sa využíva zhlukovanie, ktoré pomáha vytriediť výsledky vyhľadávania do skupín a prezentovať výsledky prehľadným a ľahkým spôsobom.

Zhlukovanie sa nazýva aj segmentácia dát, pretože veľké dátové sady rozdeľuje do zhlukov podľa podobnosti objektov. Zhluková analýza môže byť taktiež použitá aj na detekciu odľahlých objektov, ktoré sa nachádzajú ďaleko od zhlukov. Detekcia odľahlých objektov alebo hodnôt sa využíva pri monitorovaní kriminálnej aktivity, napríklad pri odcudzení kreditnej karty. Zhluková analýza sa ďalej využíva aj pri dolovaní dát z databáz, štatistike, strojovom učení, pri získavaní informácií, marketingu a iných oblastiach. Zhluková analýza využíva učenie bez učiteľa, čo znamená, že nepotrebuje žiadne preddefinované triedy ani trénovacie množiny dát a z tohto dôvodu využíva formu učenia na základe pozorovania [12].

2.2 Požiadavky na zhlukovú analýzu

Medzi požiadavky na zhlukovú analýzu, ktoré môžu byť použité napríklad pri porovnávaní rôznych zhlukovacích metód, patria:

- **Škálovateľnosť** – existuje veľké množstvo dátových sád, niektoré sú veľké, ktoré môžu obsahovať milióny alebo miliardy objektov, niektoré sú malé s niekoľko stovkami objektov. Zhlukovacie algoritmy pracujú veľmi dobre na malých dátových sadách, avšak, napríklad vo webovom vyhľadávaní pracujeme s veľkými dátovými sadami, pri ktorých, keď spustíme zhlukovací algoritmus len na malej vzorke, môže dôjsť ku skresleniu výsledku, preto sú potrebné vysoko škálovateľné zhlukovacie algoritmy.

- **Schopnosť pracovať s rôznymi typmi atribútov** – veľké množstvo algoritmov je navrhnutých tak, aby spracovávalo numerické dáta, ale existujú aplikácie, ktoré požadujú spracovávanie iných typov dát, napríklad binárne, nominálne, ordinálne alebo zoskupenia týchto typov. V súčasnosti je stále väčší a väčší dopyt po algoritmoch, ktoré spracovávajú komplexné dáta, ako napríklad grafy, dokumenty alebo obrázky.

- **Schopnosť správne určiť vstupné parametre** – niektoré zhlukovacie algoritmy požadujú určenie vstupných parametrov od užívateľa, ako je napríklad počet zhlukov. Niekedy nie sme schopní určiť tieto parametre, hlavne vo vysoko dimenzovaných dátach. Toto rozhodovanie môže značne ovplyvniť kvalitu zhlukov.

- **Schopnosť vytvárania zhlukov rôznych tvarov** – pod pojmom zhluk si najčastejšie predstavíme dáta, ktoré sú zoskupené v nejakom kruhu. Túto teóriu potvrdzujú algoritmy, ktoré sú založené na euklidovskej [12] alebo manhattenskej vzdialenosti [4]. Avšak, potrebujeme aj zhlukovacie algoritmy, ktoré generujú zhluky nielen kruhových tvarov, ale aj iných tvarov, napríklad detekcia šírenia lesného požiaru.

- **Schopnosť spracovávať dáta, ktoré obsahujú šum a odľahlé hodnoty** – v reálnom svete sa stretávame s dátami, ktoré obsahujú šum, odľahlé hodnoty, neznáme hodnoty alebo chybné dáta. Zhlukovacie algoritmy môžu byť citlivé práve na tieto zmienené dáta, čo môže spôsobiť zníženú kvalitu vytvorenia zhlukov, preto potrebujeme algoritmy, ktoré vedia pracovať aj s takýmto typom dát.

- **Necitlivosť na poradie vstupných dát a schopnosť spracovávať novovložené dáta** – niektoré aplikácie produkujú dáta za behu a niektoré zhlukovacie algoritmy nie sú schopné spracovať novopridané dáta už do existujúcich štruktúr alebo zhlukov a je nutné znovu spustiť algoritmus aj s novými dátami, aby sa zhluky prepočítali. Niektoré zhlukovacie algoritmy môžu byť citlivé na poradie vstupných dát a môžu vytvárať rôzne zhluky na základe poradia vstupných dát, preto sú nutné aj zhlukovacie algoritmy, ktoré nie su citlivé na poradie vstupných dát a sú schopné pracovať aj s novovloženými dátami.

- **Schopnosť zhlukovania na základe rôznych obmedzení** – zhlukovanie môže pomôcť aj v reálnom svete, kde potrebujeme vytvoriť zhluky aj napriek prekážkam, ktoré sa v reálnom svete vyskytujú. Nie všetky zhlukovacie metódy dokážu pracovať s týmito prekážkami alebo obmedzeniami, preto potrebujeme aj zhlukovacie algoritmy, ktoré sa dokážu vysporiadať aj s týmto problémom.

- **Schopnosť spracovávať vysoko-dimenzionálne dáta** – existuje veľké množstvo dátových sád, ktoré majú dve alebo tri dimenzie a veľké množstvo zhlukovacích algoritmov na nich pracuje dobre. Ale máme aj dátové sady, ktoré obsahujú niekoľko tisícok dimenzií, a preto potrebujeme zhlukovacie algoritmy, ktoré dokážu pracovať aj s takýmito dátovými sadami.
- **Interpretovateľnosť a použiteľnosť** – očakávame, že výsledky zhlukovania budú interpretovateľné, komplexné a použiteľné.
- **Rozdeľovacie kritéria** – zhlukovacie algoritmy zoskupujú dáta z dátových sád do zhlukov podľa rôznych kritérií. Niektoré zhlukovacie algoritmy vytvárajú zhluky, kde neexistuje žiadna hierarchia a všetky zhluky sú na rovnakej úrovni. Ale máme aj algoritmy, ktoré vedia rozdeliť zhluky hierarchicky, napríklad pri dolovaní dát z textu.
- **Separácia zhlukov** – rozdelenia do zhlukov vykonávajú zhlukové algoritmy rôzne. Dáta môžu patriť iba jednému zhluku a vtedy máme jednoznačne rozdelené zhluky, ale niektoré dáta v zhlukoch môžu patriť aj do viacerých zhlukov súčasne, napríklad pri zhlukovaní dokumentov do tém. Dokument môže súvisieť s rôznymi témami.
- **Podobnosť zhlukov** – niektoré zhluky si môžu byť navzájom podobné, čo sa určuje napríklad euklidovskou vzdialenosťou, vektorovým priestorom, pri iných zhlukoch zase konektivitou založenou na hustote. Tieto skutočnosti sú dôležitým prvkom pri navrhovaní zhlukovacích metód.
- **Zhlukovací priestor** – pri jednoduchých a malých dátových sadách je zhlukovací priestor malý a zhlukovacie algoritmy prehľadávajú celý takýto priestor pri hľadaní zhlukov. Pri vysoko-dimenzovaných dátových sadách to môže spôsobiť vytvorenie nekvalitných zhlukov, preto je priestor rozdelený na rôzne podpriestory [12].

2.3 Typy dát v zhlukovej analýze

Zhlukovacie metódy nevyužívajú len jeden typ dát, s ktorým pracujú. Zvyčajne vykonávame zhlukovanie nad dátovou sadou, ktorá obsahuje n objektov a každý objekt môže byť charakterizovaný p atribútmi. Dátové štruktúry, ktoré využívajú zhlukovacie algoritmy sú dátová matica a podobnostná matica.

- **Dátová matica** – reprezentuje n objektov pomocou p atribútov. Dátová matica má podobu relačnej tabuľky alebo matice n x p.

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

- **Podobnostná (vzdialenostná) matica** – zobrazuje vzdialenosti pre všetky dvojice z n objektov. Často býva reprezentovaná ako n x n matica.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & d(n,3) & \dots & 0 \end{bmatrix},$$

  kde *d(i,j)* reprezentuje vzdialenosť medzi objektmi i a j. Veľmi podobné objekty majú vzdialenosť blízku 0. Trojuholníkovú vzdialenostnú maticu dostávame vďaka vlastnostiam, pre ktoré platí *d(i,i)=0* a *d(i,j)=d(j,i)* [11].

V zhlukovej analýze používame rôzne typy premenných, s ktorými pracujú zhlukovacie algoritmy. Medzi najpoužívanejšie patria: binárne, intervalové, kategorické, ordinálne, pomerové premenné a premenné zmiešaných typov.

- **Binárne premenné** – môžu nadobúdať iba dve hodnoty, buď *True* alebo *False*. Existujú dva typy binárnych premenných: symetrické a asymetrické binárne premenné.

  1. **Symetrické binárne premenné** – v týchto premenných majú obe hodnoty rovnakú dôležitosť, napríklad *muž-žena* [10]. Pokiaľ chceme určiť vzdialenosť dvoch objektov pomocou symetrických binárnych premenných, použijeme koeficient:

$$d(i,j) = \frac{r+s}{q+r+s+t}, \tag{2.1}$$

     kde r je počet premenných, ktoré majú hodnotu *True* pre objekt i a hodnotu *False* pre objekt j, s je počet premenných, ktoré majú hodnotu *False* pre objekt i a hodnotu *True* pre objekt j, q je počet premenných, ktoré majú hodnotu *True* pre obidva objekty, t je počet premenných, ktoré majú hodnotu *False* pre obidva objekty [11].

  2. **Asymetrické binárne premenné** – v týchto premenných je jedna hodnota dôležitejšia ako druhá, napríklad pozitívny a negatívny výsledok medicínskych testov [10]. Typicky sa dôležitejší výsledok považuje za hodnotu *True* a menej dôležitý za hodnotu *False*. Pre určenie vzdialenosti medzi dvomi objektmi použijeme koeficient:

$$d(i,j) = \frac{r+s}{q+r+s}. \tag{2.2}$$

- **Intervalové premenné** – medzi tieto premenné patria premenné typu výška, váha alebo teplota a podobne. Pre vyjadrenie podobnosti objektov pomocou týchto premenných sa používajú vzdialenostné funkcie troch typov, Euklidovská [12], Manhattanská [4] a Minkowského vzdialenostná funkcia [20]. Musíme si dávať pozor na jednotky pri týchto premenných, pretože pri zmene jednotiek môže dôjsť k inému rozdeleniu dát v zhlukoch. Preto musíme štandardizovať dáta, aby mali rovnakú váhu. Štandardizácia sa vykonáva niekoľkými spôsobmi, napríklad pomocou strednej odchýlky a hodnoty z-score.

1. **Stredná odchýlka s_f:**

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|), \tag{2.3}$$

kde $x_{1f}, ..., x_{nf}$, je n hodnôt premennej f a m_f je stredná hodnota f, ktorá sa dá vypočítať ako:

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \ldots + x_{nf}). \tag{2.4}$$

2. **Z-score z_{if}:**

$$z_{if} = \frac{x_{if} - m_f}{s_f}. \tag{2.5}$$

Po štandardizácii sa následne určí podobnosť objektov pomocou vzdialenostných funkcií. Najčastejšie sa používa Euklidovská a Manhattanská vzdialenostná funkcia [11].

- **Kategorické premenné** – označujú sa aj ako nominálne premenné a používajú sa napríklad ako názvy značiek áut alebo pobočiek bánk [10]. Kategorické premenné môžu nadobúdať viac stavov a sú zovšeobecnením binárnych premenných. Vzdialenosť alebo mieru odlišnosti medzi dvomi objektmi i a j môžeme vypočítať na základe koeficientu:

$$d(i, j) = \frac{p - m}{p}, \tag{2.6}$$

kde m je počet zhôd a p je celkový počet premenných [11].

- **Ordinálne premenné** – sú premenné, ktoré môžu nadobúdať viac stavov a zároveň sú usporiadané do poradia. Napríklad hodnosti v armáde: *vojak, slobodník, desiatnik, ..., generál*. Ordinálne premenné môžeme spracovávať podobne ako nominálne premenné. Musíme priradiť jednotlivým hodnotám číselné ohodnotenie $1, \ldots, M_f$, následne musíme transformovať hodnoty na rozsah z intervalu $(0, 1)$ pomocou koeficientu z_{if} nasledovne:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}, \tag{2.7}$$

kde r_{if} je číselná hodnota z intervalu $1, \ldots, M_f$. Podobnosť objektov potom môžeme určiť z hodnôt z_{if} pomocou ľubovoľnej vzdialenostnej funkcie [11].

- **Pomerové premenné** – sú premenné, ktoré obsahujú hodnoty meraní na nelineárnej stupnici, ako je napríklad exponenciálna stupnica, ktorá sa približuje formule: Ae^{Bt} alebo Ae^{-Bt}, kde A a B sú pozitívne konštanty a t reprezentuje čas. Typickým príkladom takýchto premenných je rozklad rádioaktívneho prvku. Pomerové premenné môžeme spracovávať niekoľkými spôsobmi.

  1. Pomerové premenné môžeme spracovávať ako intervalové, ale môže dôjsť ku skresleniu výsledkov.
  2. Na pomerové premenné f, ktoré majú nejakú hodnotu x_{if} pre objekt i, môžeme aplikovať logaritmickú transformáciu:

$$y_{if} = log(x_{if}). \tag{2.8}$$

Potom hodnoty y_{if} môžeme spracovávať ako intervalové premenné.

3. Pomerové premenné môžeme spracovávať ako ordinálne premenné a ich rozsah spracovať ako intervalové premenné.

Medzi najúčinnejšie metódy patria posledné dve. Použiteľnosť danej metódy závisí priamo od danej aplikácie [11].

- **Premenné zmiešaných typov** – v reálnych databázach sa často vyskytujú objekty, ktoré obsahujú premenné viacerých typov. Spracovávať tieto premenné môžeme dvoma spôsobmi:

  1. Premenné zmiešaných typov môžeme spracovávať tak, že zoskupíme premenné rovnakého typu a vykonáme zhlukovú analýzu pre každý typ zvlášť. Táto metóda nie je veľmi účinná, lebo spracováva každý typ premenných zvlášť.
  2. Môžeme spracovať všetky premenné dohromady a vykonať zhlukovú analýzu nad týmito premennými. Táto technika spája rozličné premenné do jednej podobnostnej matice. Podobnosť (vzdialenosť) *d(i,j)* medzi objektmi *i* a *j* je možné definovať takto:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}, \tag{2.9}$$

  kde $\delta_{ij}^{(f)} = 0$, ak x_{if} alebo x_{jf} chýba v databáze, alebo $x_{if} = x_{jf} = 0$ a f je asymetrická binárna premenná, vo všetkých ostatných prípadoch $\delta_{ij}^{(f)} = 1$. Príspevok premennej f k podobnosti objektov i a j ($\delta_{ij}^{(f)}$) je možné vypočítať podľa typu premennej *f*.

  - Ak je f je intervalová premenná, tak $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$, kde h prechádza cez všetky prístupné objekty premennej *f*.
  - Ak je f binárna alebo kategorická premenná, tak $d_{ij}^{(f)} = 0$, ak sa hodnota $x_{if} = x_{jf}$, vo všetkých ostaných prípadoch $d_{ij}^{(f)} = 1$.
  - Ak je f ordinálna premenná, tak vypočítame hodnoty r_{if} a $z_{if} = \frac{r_{if}-1}{M_f-1}$ a následne hodnoty z_{if} spracujeme ako intervalové premenné.
  - Ak je *f* pomerová premenná, tak vykonáme logaritmickú transformáciu a tieto transformované dáta ďalej spracujeme ako intervalové premenné.

Podľa vyššie uvedených krokov vieme spracovať premenné zmiešaných typov a vykonať zhlukovú analýzu [11].

2.4 Ohodnotenie zhlukovania

Zhlukovacie algoritmy vytvárajú zhluky dát. Aby sme vedeli, ako kvalitne zhlukovacie algoritmy vytvorili zhluky, existujú rôzne ohodnotenia zhlukovacích metód, napríklad: *ohodnotenie tendencie zhlukovania*, *určenie počtu zhlukov*, *meranie kvality zhlukovania* a *časová zložitosť*.

2.4.1 Ohodnotenie tendencie zhlukovania

Ohodnotenie tendencie zhlukovania určuje, či vstupná dátová sada má nenáhodnú štruktúru, ktorá môže viesť k produkcii zmysluplných zhlukov. Túto vlastnosť môžeme určovať pomocou Hopkinsovej štatistiky, ktorá sa počíta nasledovne.

Máme vzorku n bodov $p_1, \ldots, p_n$ z dátovej sady D. Pre každý jeden bod p_i nájdeme najbližšieho suseda p_i $(1 \leq i \leq n)$ a zmeriame vzdialenosť x_i medzi bodom p_i a jeho najbližším susedom:

$$x_i = min_{v \in D} \{dist(p_i, v)\}. \tag{2.10}$$

Máme vzorku n bodov $q_1, \ldots, q_n$ z dátovej sady D. Pre každý bod q_i nájdeme najbližšieho suseda q_i $(1 \leq i \leq n)$ z dátovej sady $D - \{q_i\}$ a zmeriame vzdialenosť y_i medzi bodom q_i a jeho najbližším susedom v $D - \{q_i\}$:

$$y_i = min_{v \in D, v \neq q_i} \{dist(q_i, v)\}. \tag{2.11}$$

Následne spočítame Hopkinsovu štatistiku podľa vzorca:

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}. \tag{2.12}$$

Pokiaľ $H < 0.5$, potom je nepravdepodobné, že dátová sada D obsahuje štatisticky významné zhluky [12].

2.4.2 Určenie počtu zhlukov

Určenie počtu zhlukov je podstatnou časťou, pretože niektoré algoritmy vyžadujú na vstupe počet zhlukov a počet zhlukov ovplyvňuje aj presnosť a kvalitu zhlukovania. Správny počet zhlukov je ťažké určiť, lebo je závislý na tvaroch a rozsahu v dátovej sade a taktiež môže byť závislý na požadovanej rozlíšiteľnosti od používateľa.

Jednou z najjednoduchších metód je metóda, kde sa počet zhlukov určí na hodnotu $\sqrt{\frac{n}{2}}$, kde n je počet objektov v dátovej sade a v jednom zhluku sa očakáva $\sqrt{2n}$ objektov.

Ďalšou metódou, ako určiť počet zhlukov je *elbow metóda*, ktorá volí počet zhlukov tak, aby pridanie ďalšieho zhluku neprinieslo výrazné zlepšenie modelu. Ak zvolíme nejaké číslo $k > 0$, tak môžeme pomocou nejakého zhlukovacieho algoritmu vytvoriť k zhlukov a môžeme vypočítať odchýlku v rámci zhluku $(var(k))$ a vykresliť graf, kde prvý bod zakrivenia odporúča hodnotu správneho počtu zhlukov.

Krížová validácia (Cross-validation) je metóda, ktorá najprv rozdelí dátovú sadu na m častí a $m-1$ častí použije na vytvorenie zhlukovacieho modelu a zvyšok použije na testovanie kvality zhlukovania. Používa vzdialenosť medzi objektmi testovacej sady a najbližšími centroidmi a meria ako dobre vyhovuje model testovacej sade. Pre nejaké $k > 0$ sa zopakuje tento proces m-krát, aby sa každá časť použila ako testovacia sada. Počet zhlukov sa určí podľa toho, kde celková kvalita zhlukov bola najlepšia [12].

2.4.3 Meranie kvality zhlukovania

Meranie kvality zhlukovania môže byť rozdelené do dvoch kategórií podľa parametra *ground truth*, ktorý sa považuje za ideálne zhlukovanie. Ak je parameter ground truth dostupný, používajú sa vonkajšie metódy (*extrinsic methods*). Ak nie je dostupný, používajú sa vnútorné metódy (*intrinsic methods*).

1. **Vonkajšie metódy** – hlavnou úlohou tejto metódy je priradiť skóre $Q(C, C_g)$ do zhlukov C, vzhľadom na parameter ground truth C_g. Meranie je efektívne, ak spĺňa 4 kritériá: homogénnosť zhlukov, úplnosť zhlukov, uchovávanie malých zhlukov a rag bag (kritérium rag bagu je, že heterogénny objekt pri radený do čistého zhluku by mal byť viac penalizovaný, ako keď je priradený do rag bagu - označenie objektov, ktoré nemôžu byť spojené s inými objektmi, napríklad šum) [12].

2. **Vnútorné metódy** – tieto metódy vyhodnocujú, ako dobre sú zhluky oddelené a ako sú kompaktné. Ako meradlo používame rôzne indexy: siluetový koeficient (*Silhouette coefficient*), Dunnov index separácie (*Dunn separation index*) alebo Davies-Bouldinov index.

   Siluetový koeficient – pre každý objekt z dátovej sady určíme priemernú vzdialenosť objektu o od ostatných objektov $a(o)$ a minimálnu priemernú vzdialenosť objektu $b(o)$ od všetkých zhlukov, do ktorých objekt o nepatrí. Potom spočítame siluetový koeficient $s(o)$ ako:

   $$s(o) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}}. \tag{2.13}$$

   Siluetový koeficient nadobúda hodnoty od -1 po 1. Siluetový koeficient dosahuje hodnotu 1, ak zhluk obsahujúci objekt o je kompaktný a objekt o je ďaleko od ostatných zhlukov [12].

   Dunnov index separácie – je určený pomocou priemeru zhluku $\Delta(\Omega_i)$ a vzdialenosťou medzi zhlukmi $\delta(\Omega_i, \Omega_j)$. Priemer zhluku sa vypočíta nasledovne:

   $$\Delta(\Omega_i) = max_{x,y \in \Omega_i} ||x - y||, \tag{2.14}$$

   vzdialenosť medzi zhlukmi je určená pomocou vzorca:

   $$\delta(\Omega_i, \Omega_j) = min_{x \in \Omega_i, y \in \Omega_j} ||x - y||, \tag{2.15}$$

   výsledný Dunnov index separácie je určený vzorcom:

   $$r = min_i min_{j, i \neq j} \frac{\delta(\Omega_i, \Omega_j)}{max_k \Delta(\Omega_k)}. \tag{2.16}$$

   Pokiaľ chceme dosiahnuť čo najlepšie rozdelenie do zhlukov, musí byť Dunnov index separácie čo najvyšší [7].

   Davies-Bouldinov index – preferuje zhluky, ktoré sa nachádzajú ďaleko od seba a sú kompaktné. Je podobný Dunnovmu indexu separácie a je určený pomocou priemeru zhluku, ktorý je definovaný nasledovne:

   $$diam(c_i) = \left(\frac{1}{n_i} \sum_{x \in c_i} ||x - z_i||^2 \right)^{1/2}, \tag{2.17}$$

   potom Davies-Bouldinov index je definovaný pomocou vzorca:

   $$DB_k = \frac{1}{k} \sum_{i=1}^{k} max_{j=1,\ldots,k, i \neq j} \left\{ \frac{diam(c_i) + diam(c_j)}{||c_i - c_j||} \right\}, \tag{2.18}$$

   kde n_i sú body, z_i sú centroidy zhlukov c_i. Cieľom je získať zhluky s minimálnou vzdialenosťou objektov v rámci zhluku, preto najlepší počet zhlukov dostaneme, keď bude Davies-Bouldinov index čo najmenší [24].

2.4.4 Časová zložitosť

Časová zložitosť udáva teoretické zobrazenie závislosti doby behu algoritmu na veľkosti vstupných dát. Časovú zložitosť je možné vyjadriť *Big-O* notáciou, napríklad *O(n)*, čo znamená, že čas, ktorý algoritmus potrebuje na svoje vykonanie, je lineárne závislí na počte prvkov. Existuje niekoľko časových tried, do ktorých je možné algoritmus zaradiť. Najčastejšie vyskytujúce sa triedy sú zobrazené v tabuľke 2.1 [6].

Big-O notácia	**názov času**
$O(1)$	konštantný
$O(\log n)$	logaritmický
$O(n \log n)$	linearitmický
$O(n)$	lineárny
$O(n^2)$	kvadratický
$O(n^3)$	kubický

Tabuľka 2.1: Prehľad tried časovej zložitosti algoritmov.

Kapitola 3

Zhlukovacie metódy

Existuje veľké množstvo zhlukovacích algoritmov, ktoré môžu byť rozdelené rôznymi spôsobmi do skupín. Niekedy je ťažké rozhodnúť, do ktorej skupiny algoritmy patria, lebo môžu obsahovať vlastnosti rôznych metód. V tejto kapitole je popísané jedno z možných rozdelení [12]. Do tohto rozdelenia patria: metódy založené na rozdeľovaní, hierarchické metódy, metódy založené na hustote, metódy založené na mriežke, metódy založené na modeloch, metódy pre zhlukovanie vysoko-dimenzovaných dát, metódy pre zhlukovanie grafových a sieťových dát a metódy založené na obmedzeniach.

3.1 Metódy založené na rozdeľovaní

Rozdeľovanie je základnou a najjednoduchšou verziou zhlukovej analýzy, kde sa n objektov z dátovej sady D organizuje do k zhlukov. Rozdeľovací algoritmus rozdeľuje objekty do k zhlukov, kde $n \geq k$. Zhluky sú vytvorené tak, aby si objekty v nich boli čo najviac podobné, zatiaľ čo objekty rôznych zhlukov si boli navzájom odlišné. Nevýhodou tejto metódy je to, že musíme zadať počet zhlukov k, do ktorých má rozdeľovací algoritmus rozdeliť objekty z dátovej sady. Medzi rozdeľovacie algoritmy patrí algoritmus k-Means, k-Medoids, CLARANS a ďalšie [12].

3.1.1 k-Means

K-Means je zhlukovacia metóda, ktorá je založená na centrálnom bode. Centrálny bod môže byť určený ako stredná hodnota objektov alebo ako ťažisko zhluku. Vzdialenosť medzi objektmi sa určuje pomocou Euklidovskej vzdialenosti a objekty sa priraďujú do zhlukov podľa vzdialenosti od centrálneho bodu. Po priradení objektu do zhluku sa centrálny bod musí znovu prepočítať. V každom cykle sa objekty prerozdeľujú do zhlukov. Kvalita zhlukov sa meria pomocou štvorca chyby (*squared error*) E, definovaná ako:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2, \tag{3.1}$$

kde p je bod v priestore reprezentujúci daný objekt a c_i je centrálny bod zhluku C_i. Pomocou koeficientu E sa snažíme dostať čo najkompaktnejšie a najkvalitnejšie zhluky dát.

Algoritmus 1: k-Means

Vstup : k: počet zhlukov, D: dátová sada, ktorá pozostáva z n objektov
Výstup: množina k zhlukov

1 *vyberieme ľubovoľné objekty z dátovej sady D a označíme ich ako centrálne body*;
2 **repeat**
3 | *objekt (znovu) priradíme do zhluku, ku ktorému je najviac podobný na základe strednej hodnoty v zhluku*;
4 | *prepočítame stredy zhlukov, čo je stredná hodnota objektov v každom zhluku*;
5 **until** *nenastane žiadna zmena*;

Aby sme získali lepšie výsledky, môžeme algoritmus spustiť niekoľkokrát s rôznymi stredmi zhlukov. Časová zložitosť algoritmu k-Means je $O(nkt)$, kde n je celkový počet objektov, k je počet zhlukov a t je počet iterácií, zvyčajne $k \ll n$ a $t \ll n$. Táto metóda je relatívne dobre škálovateľná a efektívna pri práci s veľkými dátovými sadami [12].

Existuje niekoľko druhov k-Means algoritmov, ktoré sa líšia napríklad vo výbere počiatočných stredov zhlukov, výpočtu stredov zhlukov a podobne [11].

3.1.2 k-Medoids

Tento zhlukovací algoritmus je podobný algoritmu k-Means, ale je viac odolný voči šumu a odľahlým hodnotám. K-Medoids na rozdiel od algoritmu k-Means využíva ako stred zhluku skutočný bod (*medoid*). Ide o bod, ktorý sa nachádza najbližšie k ostatným objektom, čiže má najnižšiu vzdialenosť k ostatným bodom a zároveň sa nachádza, čo najbližšie ku stredu zhluku.

Ako aj pri algoritme k-Means, aj pri tomto algoritme musíme zvoliť počet zhlukov k, ktoré sa majú vytvoriť. Následne vyberieme náhodne k objektov z dátovej sady a označíme ich ako počiatočné zhluky a najlepších zástupcov zhlukov. Následne sa analyzujú všetky kombinácie reprezentatívnych a nereprezentatívnych bodov a spočíta sa kvalita zhlukovania pre každú dvojicu. Starý reprezentatívny bod potom nahradíme novým, ktorý najviac prispieva k celkovej zápornej zmene vzdialenosti od ostatných objektov. Objekty sa premiestňujú pokiaľ sa zhluky neustália.

Časová zložitosť algoritmu k-Medoids je $O(k(n-k)^2 i)$, kde k je počet zhlukov, n je počet objektov v dátovej sade a i je počet iterácií. K-Medoids nie je dobre škálovateľný pre veľký počet objektov v dátovej sade [14].

3.1.3 CLARANS

CLARANS (*Clustering Large Applications based upon RANdomized Search*) je algoritmus, ktorý využíva metódu k-Medoids a náhodné vzorkovanie. Na začiatku sa musí určiť počet iterácii a vzdialenosť od medoidu, v ktorej sa budú susedné objekty prehľadávať. Ide o prehľadávanie stromového grafu, kde uzly môžu reprezentovať medoidy. Algoritmus začne na náhodnom uzle a prehľadáva určenú vzdialenosť. Pokiaľ nájde lepší uzol, tak ho vyberie a pokračuje z neho, ak nenájde lepší uzol, tak aktuálny uzol označí ako lokálne minimum. Algoritmus sa zastaví po danom počte iterácií.

Časová zložitosť algoritmu CLARANS je $O(n^2)$, kde n je počet objektov v dátovej sade [11]. Algoritmus pri malej dátovej sade produkuje rovnako kvalitné zhluky ako k-

Medoids, ale je výrazne efektívnejší. Tento algoritmus dobre pracuje aj pri veľkých dátových sadách [19].

3.2 Hierarchické metódy

Hierarchické metódy boli navrhnuté pre zdokonalenie zhlukovania dátových objektov, tak aby zhlukovanie bolo deterministickejšie a flexibilnejšie. Tieto metódy sa delia na zhlukujúce hierarchické metódy a rozdeľujúce hierarchické metódy. Hierarchiu zhlukov môžeme reprezentovať pomocou štandardného binárneho stromu, kde koreň reprezentuje všetky zhluky dátových objektov a tvorí vrchol hierarchie, uzly tvoria podmnožiny celej dátovej sady a zodpovedajú zhlukom. Každá úroveň stromu zodpovedá nejakej sade zhlukov. Základ hierarchie tvoria zhluky, ktoré pozostávajú iba z jedného dátového objektu a sú to listy stromu. Takáto štruktúra stromu sa nazýva dendrogram. Hlavnou výhodou hierarchických metód je to, že umožňujú rozdeliť hierarchiu na ľubovoľnej úrovni a tak získať odpovedajúce zhluky. Nevýhodou je, že pokiaľ niektoré triedy zlúčime alebo rozdelíme, už ich nemôžeme naspäť zlúčiť alebo rozdeliť.

Zhlukujúce hierarchické metódy – berú najjednoduchšie zhluky (zhluky, ktoré sú tvorené iba jedným dátovým objektom) z najnižšej úrovne a pokračujú zhlukovaním dvoch zhlukov naraz a tak vytvárajú hierarchiu zdola-nahor.

Rozdeľujúce hierarchické metódy – začínajú so všetkými dátovými objektmi a postupne rozdeľujú túto makroskupinu a tým vytvárajú hierarchiu zhlukov zhora-nadol [1].

Keď používame tieto metódy, potrebujeme merať vzdialenosť medzi dvomi zhlukmi. Medzi najčastejšie metriky na meranie vzdialenosti $|p - p'|$ medzi dvoma objektmi p a p', m_i je stred zhluku C_i a n_i je počet objektov v C_i, patria:

1. **Minimálna vzdialenosť:**

$$dist_{min}(C_i, C_j) = min_{p \in C_i, p' \in C_j}\{|p - p'|\} \tag{3.2}$$

2. **Maximálna vzdialenosť:**

$$dist_{max}(C_i, C_j) = max_{p \in C_i, p' \in C_j}\{|p - p'|\} \tag{3.3}$$

3. **Stredná vzdialenosť:**

$$dist_{mean}(C_i, C_j) = |m_i - m_j| \tag{3.4}$$

4. **Priemerná vzdialenosť:**

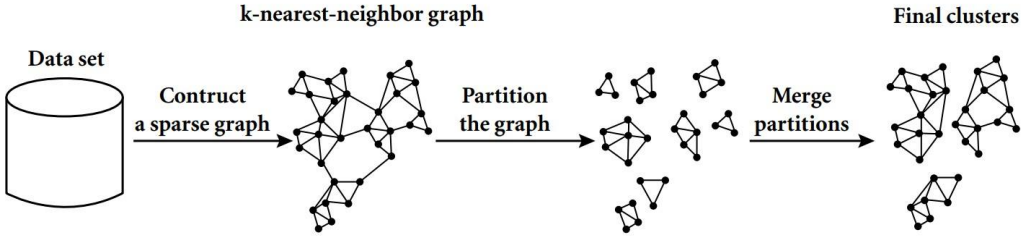
$$dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'|. \tag{3.5}$$

Pokiaľ hierarchický algoritmus používa na meranie vzdialenosti minimálnu vzdialenosť, tak sa nazýva zhlukovací algoritmus najbližšieho suseda (*nearest-neighbor clustering algorithm*), pokiaľ používa maximálnu vzdialenosť, tak sa nazýva zhlukovací algoritmus najvzdialenejšieho suseda (*farthest-neighbor clustering algorithm*). Tieto dve metódy merania predstavujú extrémy v meraní vzdialenosti a sú citlivé na odľahlé hodnoty a zašumené dáta. Na problém citlivosti na odľahlé hodnoty je dobré použiť výpočty strednej a priemernej vzdialenosti. Priemerná vzdialenosť sa používa hlavne pri kategorických dátach, kde stredná vzdialenosť sa dá použiť veľmi ťažko alebo vôbec. Medzi hierarchické metódy patria algoritmy, ako napríklad Chameleon, BIRCH, AGNES, DIANA, pravdepodobnostné hierarchické zhlukovanie a ďalšie [12].

3.2.1 Chameleon

Chameleon je hierarchický zhlukovací algoritmus, ktorý využíva dynamické modelovanie na určenie podobnosti medzi pármi zhlukov. Táto podobnosť sa určuje dvomi podmienkami, a to ako dobre sú prepojené objekty v zhluku (interkonektivita) a ako ďaleko sú zhluky od seba. Napríklad, ak máme dva zhluky, ktorých interkonektivita je vysoká a sú blízko seba, tak tieto dva zhluky sa spoja.

Na obrázku 3.1 môžeme vidieť, ako algoritmus pracuje. Algoritmus využíva k-nearest-neighbor graf [22] k vytvoreniu rozptýleného grafu. Každý vrchol grafu reprezentuje dátový objekt z dátovej sady a hrana reprezentuje podobnosť medzi objektmi, ktorá je určená váhou. Chameleon používa rozdeľujúci grafový algoritmus na rozdelenie k-nearest-neighbor grafu do veľkého počtu relatívne malých podzhlukov. Hrana, ktorá rozdeľuje zhluk na dva podzhluky a minimalizuje váhu hrán, sa nazýva rez hrany (*edge cut (EC)*). Následne používa zhlukujúci hierarchický algoritmus, ktorý iteratívne spája na základe podobnosti podzhluky, až dokým nenájde finálne zhluky.



Obr. 3.1: Hierarchický algoritmus Chameleon. Obrázok prevzatý z [11].

Ako bolo spomenuté vyššie, podobnosť medzi dvomi pármi zhlukov je určená na základe ich vzájomnej relatívnej interkonektivity $RI(C_i, C_j)$ a vzájomnej relatívnej blízkosti $RC(C_i, C_j)$, kde C_i a C_j sú zhluky.

Relatívna interkonektivita sa vypočíta:

$$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)}, \tag{3.6}$$

kde EC_{C_i}, EC_{C_j} minimálny počet rezov hrán, ktorý rozdeľuje C_i, C_j na približne rovnaké časti.

Relatívna blízkosť sa vypočíta:

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}, \tag{3.7}$$

kde $\overline{S}_{EC_{\{C_i,C_j\}}}$ je priemerná váha hrán, ktoré spájajú vrcholy C_i a C_j, $\overline{S}_{EC_{C_i}}$ a $\overline{S}_{EC_{C_j}}$ je priemerná váha hrán, ktorá patrí minimálnemu rezu hrany zhluku C_i a C_j.

Algoritmus má časovú zložitosť pre vysoko-dimenzované dáta $O(n^2)$. Časová zložitosť pre nízko-dimenzované dáta je *O(n log n)*, kde n je počet objektov v dátovej sade. Chameleon je podľa experimentov lepší v získavaní zhlukov ľubovoľného tvaru vysokej kvality ako algoritmus BIRCH [12].

3.3 Metódy založené na hustote

Pre vyhľadávanie zhlukov ľubovoľného tvaru boli navrhnuté metódy založené na hustote. Hustota v ľubovoľnom bode dátového priestoru sa dá určiť počtom objektov v blízkosti daného objektu. Objekty, ktoré sú mimo oblasti veľkej hustoty, sú považované za šum. Medzi tieto metódy patria algoritmy DBSCAN, OPTICS, DENCLUE a ďalšie.

3.3.1 DBSCAN

DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) je algoritmus, ktorý hľadá objekty s veľkou hustotou a spája ich do zhlukov. Veľkosť okolia určuje používateľ, ide o parameter ϵ, pre ktorý platí, že $\epsilon > 0$. ϵ-okolie nejakého objektu je priestor okolo objektu s polomerom ϵ. Používateľ zadáva ešte jeden parameter $MinPts$, ktorý reprezentuje prah hustoty. Ak ϵ-okolie nejakého objektu obsahuje aspoň $MinPts$ objektov, tak tento objekt sa nazýva jadrom.

Ak máme dve jadrá p a q, tak môžeme povedať, že p je priamo dosiahnuteľný na základe hustoty z q, ak p sa nachádza v ϵ-okolí jadra q.

Jadro p je dosiahnuteľné na základe hustoty z jadra q, ak existuje reťazec objektov $p_1, \ldots, p_n$, pre ktoré platí, že $p_1 = q$, $p_n = p$ a p_{i+1} je priamo dosiahnuteľný z p_i, pokiaľ sú zadané parametre ϵ, $MinPts$ a platí $1 \leq i \leq n$ a p_i patrí do dátovej sady.

Dva objekty p_1 a p_2 z dátovej sady sú spojené na základe hustoty, ak existuje objekt q z dátovej sady taký, že objekty p_1 a p_2 sú dosiahnuteľné z tohto objektu.

Všetky objekty sú na začiatku označené ako nenavštívené. Algoritmus náhodne vyberie nejaký nenavštívený objekt p a označí ho ako navštívený, pritom overí, či ϵ-okolie tohto objektu obsahuje aspoň $MinPts$ objektov. Ak neobsahuje, tak objekt p je označený ako šum. Ak obsahuje dostatočný počet objektov, tak sa vytvorí nový zhluk C pre objekt p a všetky body z ϵ-okolia sa pridajú do kandidátnej sady N. Algoritmus iteratívne pridáva objekty z N do zhluku C, ktoré nepatria ešte do žiadneho zhluku. Do zhluku C sa pridávajú objekty, pokiaľ sa v sade kandidátov N nenachádzajú už žiadne objekty a zhluk C sa už nemôže zväčšovať. Ďalší zhluk sa hľadá tým istým spôsobom a ako základný objekt sa vyberie ľubovoľný nenavštívený objekt. Algoritmus pokračuje, pokiaľ nebudú navštívené všetky objekty.

Časová zložitosť algoritmu je *O(n log n)*, ak používame priestorový index (*spatial index* [26]), inak je $O(n^2)$, kde n je počet objektov v dátovej sade. Pri dobre zvolenom ϵ a $MinPts$ dokáže algoritmus efektívne nájsť ľubovoľné zhluky dát [12].

3.4 Metódy založené na modeloch

Tieto metódy sa snažia nájsť také zhluky dát, ktoré najviac odpovedajú nejakému matematickému modelu. Metódy založené na modeloch vychádzajú z toho, že vstupné dáta sú generované na základe nejakej pravdepodobnostnej distribučnej funkcie. Medzi tieto metódy patrí napríklad rozšírenie algoritmu k-Means (Expectation-Maximization) alebo samoorganizujúce siete (SOM) a ďalšie [11].

3.4.1 Expectation-Maximization

Expectation-Maximization algoritmus je rozšírením algoritmu k-Means a pracuje podobne ako k-Means. Algoritmus začína s počiatočnou sadou parametrov a pokračuje pokiaľ sa

zhluky neprestanú zlepšovať. Každá iterácia sa skladá z dvoch krokov: *expectation step* a *maximization step*. V prvom kroku (*expectation step*) sa objekty priraďujú do zhlukov v závislosti na parametroch pravdepodobnosti zhluku. V druhom kroku (*maximization step*) sa tieto parametre pravdepodobnosti používajú na prehodnotenie parametrov modelu.

Algoritmus je jednoduchý a ľahko implementovateľný. Využíva sa na vysporiadanie sa s problémami pri učení v dolovaní dát alebo štatistike. Časová zložitosť algoritmu je lineárna v d (počet vstupov), n (počet objektov) a t (počet iterácií) [11].

3.4.2 SOM

SOM (*Self-Organizing feature Maps*) patrí do kategórie samoorganizujúcich sietí. Tento algoritmus sa obzvlášť využíva pre vizualizáciu a zhlukovú analýzu, pretože dáta, ktoré majú veľký počet dimenzií, môžu byť zredukované do jednej alebo dvoch dimenzií, kde je lepšie vidieť podobnostné oblasti.

Architektúra algoritmu pozostáva z dvoch vrstiev: zo vstupnej a Kohonenovej vrstvy, čo sa považuje ako výstupná vrstva. Počet neurónov vo vstupnej vrstve je daný počtom atribútov objektov. Každý neurón je prepojený s neurónmi vo výstupnej vrstve. Zhlukovanie sa uskutočňuje tak, že neuróny súťažia o aktuálny objekt. Neurón, ktorého váhový vektor je najbližšie k aktuálnemu objektu, sa stáva víťazom. Aby sa neurón ešte viac priblížil, upravia sa váhy.

Na začiatku sa inicializujú váhové vektory na malé hodnoty. Následne sa v cykle opakuje, že sa vytiahne objekt x z dátovej sady s nejakou pravdepodobnosťou a nájde sa víťazný neurón $i(x)$ v časovom kroku s použitím minimálne Euklidovskej vzdialenosti nasledovne:

$$i(x) = arg\ min_{1 \leq j \leq d^*} ||x - w_j^s||, \tag{3.8}$$

kde w_j je váha neurónu a d^* je celkový počet neurónov v Kohonenovej vrstve. Následne sa upravia váhy podľa nasledujúceho vzorca:

$$w_j^{s+1} = w_j^s + \eta(s) h_{j,i(x)}(s)(x - w_j^s), \tag{3.9}$$

kde $\eta(s)$ je parameter učenia (*learning-rate parameter*), ktorý je definovaný nasledovne:

$$\eta(s) = \eta_0 exp\left(-\frac{s}{\tau_2}\right), \tag{3.10}$$

kde τ_2 je časová konštanta. $h_{j,i(x)}(s)$ (*neighborhood function*) je funkcia, ktorá určuje okolie výhercu, v ktorom sa budú meniť váhové vektory a je definovaná nasledovne:

$$h_{j,i(x)}(s) = exp\left(-\frac{d_{i(x),j}^2}{2\sigma^2(s)}\right), \tag{3.11}$$

kde $\sigma(s)$ je definované ako:

$$\sigma(s) = \sigma_0 \left(-\frac{s}{\tau_1}\right), \tag{3.12}$$

a σ_0 je polomer mriežky a τ_1 je časová konštanta. Tieto úpravy váh sa vykonávajú, pokiaľ už nebudú žiadne zmeny váh v sieti. Nakoniec každý neurón reprezentuje jeden zhluk, ktorému náleží skupina objektov [10].

Self-organizing mapa n neurónov, z ktorých každý je q-dimenzionálny, má na vstupe x_j dát, ktoré sú d-dimenzionálne ($p_i \in \mathbb{R}^q$ a $x_j \in \mathbb{R}^d$). Časová zložitosť algoritmu je $O(t_f(dn + pn^2))$, kde t_f je počet trénovacích iterácií [3].

3.5 Ďalšie zhlukovacie metódy

V tejto sekcii sú popísané ďalšie zhlukovacie metódy, ako napríklad metódy založené na mriežke, metódy pre zhlukovanie vysoko-dimenzovaných dát, metódy pre zhlukovanie grafových a sieťových dát a metódy založené na obmedzeniach. Tieto metódy nie sú moc známe a používajú sa len zriedka.

3.5.1 Metódy založené na mriežke

Tieto metódy sa používajú predovšetkým na zhlukovanie multidimenzionálnych dát. Algoritmy, ktoré patria do tohto odvetvia, vytvárajú mriežkovú štruktúru, teda rozdeľujú dátový priestor na konečný počet buniek. Pre každú bunku v mriežke počítajú hustotu objektov a usporiadajú bunky podľa ich hustoty. Z takto usporiadaných buniek nájdu jadrá zhlukov a prechádzajú cez susedné bunky. Do týchto metód patria algoritmy, ako napríklad STING, WaveCluster, CLIQUE a ďalšie [10].

3.5.2 Metódy pre zhlukovanie vysoko-dimenzovaných dát

Vysoko-dimenzované dáta sú dáta s veľkým počtom dimenzií, zvyčajne viac ako 10. Metódy, ktoré sa zaoberajú týmito dátami sa rozdeľujú do dvoch skupín: *podpriestorové zhlukovanie* a *metódy na redukciu dimenzionality* [12].

3.5.3 Metódy pre zhlukovanie grafových a sieťových dát

Grafové a sieťové dáta sú komplikovanejšie ako dáta používané v predchádzajúcich metódach. Grafové dáta môžu obsahovať až niekoľko desiatok miliárd webových stránok, avšak grafy môžu byť aj veľmi riedke. Zhluky v grafoch sa hľadajú tak, že graf sa rozreže na niekoľko častí tak, aby boli vrcholy v zhlukoch dobre prepojené a vrcholy v rozdielnych zhlukoch boli slabšie prepojené. Jedna z metód používaná na zhlukovanie grafových dát sa nazýva SCAN [12].

3.5.4 Metódy založené na obmedzeniach

Pri zhlukovaní dát majú niekedy používatelia znalosti o zhlukovaných dátach, ktoré by chceli do zhlukovacích metód zahrnúť. Tieto znalosti sa nazývajú obmedzenia. Existujú rôzne typy obmedzení, napríklad: obmedzenia na instanciách, obmedzenia na zhlukoch, obmedzenia na meraní podobnosti zhlukov. Algoritmy majú na vstupe nielen dátovú sadu, ale aj množinu obmedzení, ktorá sa do algoritmov aplikuje [12].

Kapitola 4

Dátové sady a implementácie algoritmov

V tejto kapitole sú popísané dátové sady, ktoré boli vybrané pre prácu v ďalšej časti diplomovej práce a taktiež implementácie zhlukovacích algoritmov, ktoré boli použité v aplikácii, ktorá mala za úlohu spúšťať algoritmy na dátových sadách a vyhodnocovať kvalitu zhlukov a časovú náročnosť.

4.1 Dátové sady

Každý zhlukovací algoritmus potrebuje nejaké vstupné dáta. Tieto dáta sú uložené v dátových sadách. Existuje veľmi veľa dátových sád, niektoré boli navrhnuté pre účely porovnávania algoritmov, iné pre vyzdvihnutie niektorej vlastnosti algoritmu. Niektoré dátové sady obsahujú malé množstvo objektov a iné ich môžu obsahovať aj niekoľko miliónov. Niektoré dátové sady majú len dve dimenzie a iné ich môžu mať tisíce. Dáta v dátových sadách môžu vytvárať rôzne obrazce.

Pre účely diplomovej práce bolo vybraných 5 dátových sád: Jain, Compound, Flame, t4.8k a MNIST. Tieto dátové sady boli prebrané z webovej stránky [9].

1. **Dátová sada Jain** – obsahuje 373 objektov, 2 dimenzie a vytvára 2 zhluky. Ukážka dátovej sady je na obrázku 4.2. Dátová sada obsahuje dve vlnovky, v ktorých sa hustota rozmiestnenia objektov značne mení, čo môže spôsobovať pri niektorých algoritmoch, že bude vytvorených viac zhlukov.

2. **Dátová sada Compound** – obsahuje 399 objektov, 2 dimenzie a vytvára 6 zhlukov. Ukážka dátovej sady je na obrázku 4.3. V tejto dátovej sade je možné vidieť niekoľko zhlukov, ktoré vytvárajú rôzne tvary. Taktiež tu nájdeme ťažko oddeliteľné zhluky, ktoré sa nachádzajú v ľavej hornej časti. V pravej časti dátovej sady sa nachádza zhluk, ktorý má okolo seba niekoľko odľahlých hodnôt. Preto by mohla táto sada spôsobovať problém niektorým algoritmom.

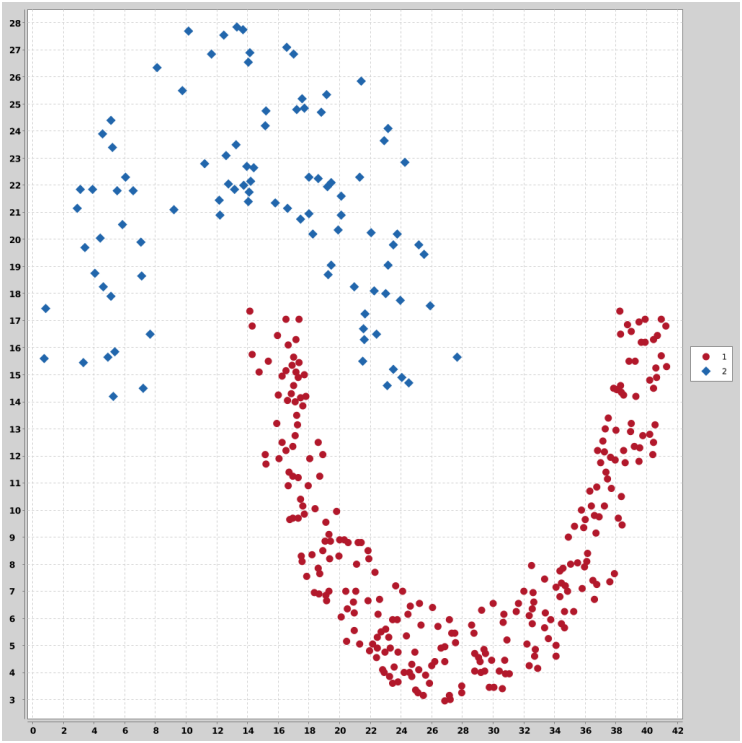
3. **Dátová sada Flame** – obsahuje 240 objektov, 2 dimenzie a vytvára 2 zhluky. Ukážka dátovej sady je na obrázku 4.4. Dátová sada obsahuje nekompaktné zhluky, ktoré sú ťažko odlíšiteľné.

4. **Dátová sada t4.8k** – obsahuje 8000 objektov, 2 dimenzie a vytvára 6 zhlukov. Ukážka dátovej sady je na obrázku 4.5. V tejto dátovej sade sa nachádzajú zhluky,

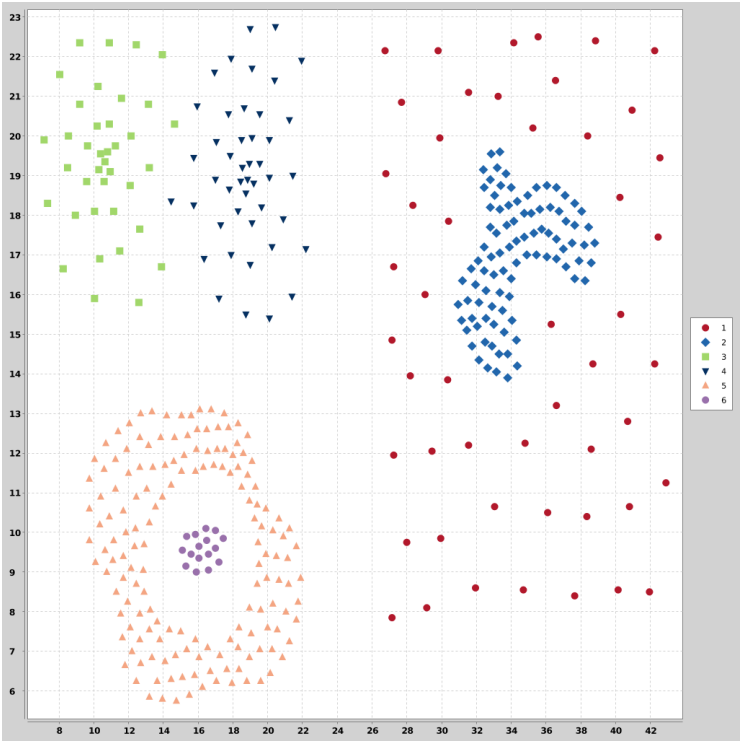
ktoré sú celkom dobre oddeliteľné, avšak okolo týchto zhlukov sa nachádza menšie množstvo objektov, ktoré niektoré zhluky spájajú. Taktiež táto dátová sada obsahuje veľké množstvo odľahlých hodnôt. Z týchto dôvodov je možné, že u niektorých algoritmov môže nastať problém so správnym priradením objektov do zhlukov.

5. **Dátová sada MNIST** – obsahuje 10000 objektov, 748 dimenzií a vytvára 10 zhlukov. Dátová sada obsahuje 10 ručne písaných číslic. Ukážka číslic, ktoré obsahuje dátová sada je na obrázku 4.1.

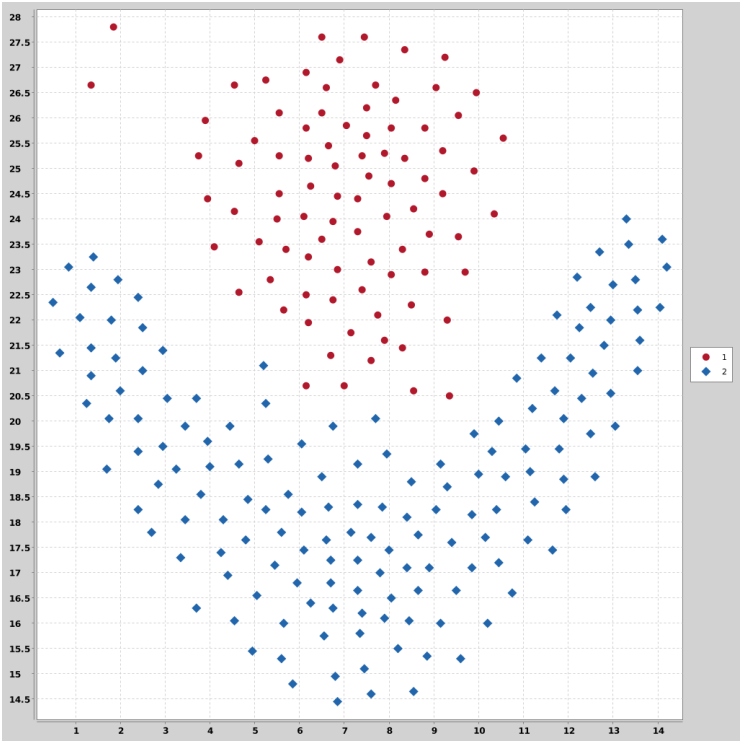
Obr. 4.1: Ukážka jednotlivých číslic z dátovej sady MNIST.



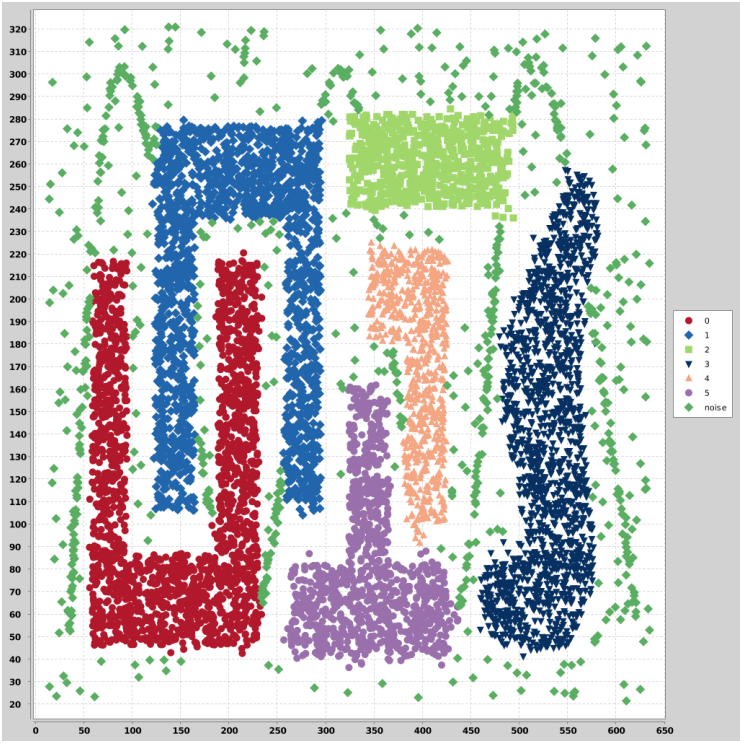
Obr. 4.2: Dátová sada Jain. Obrázok prevzatý z [2].



Obr. 4.3: Dátová sada Compound. Obrázok prevzatý z [2].



Obr. 4.4: Dátová sada Flame. Obrázok prevzatý z [2].



Obr. 4.5: Dátová sada t4.8k. Obrázok prevzatý z [2].

4.2 Implementácie zhlukovacích algoritmov

Spomedzi zhlukovacích algoritmov boli vybrané tri algoritmy, ktoré boli použité v aplikácii a ich vlastnosti boli ďalej analyzované. Algoritmy boli napísané v jazyku Python. Medzi tieto algoritmy patria Chameleon, ktorý reprezentuje hierarchické metódy, DBSCAN, ktorý reprezentuje zhlukovacie metódy založené na hustote a SOM, ktorý reprezentuje zhlukovacie metódy založené na modeloch a využíva neurónové siete. Vybrané algoritmy boli spúšťané na dátových sadách a boli analyzované niektoré ich vlastnosti, ako je časová zložitosť a kvalita zhlukov.

Implementácia algoritmu Chameleon, ktorej autorom je Giovanni Paolo, bola prebraná z [21]. Algoritmus tvoria 3 kroky: vytvorenie k-nearest-neighbor grafu, rozdelenie grafu na menšie podgrafy a zlúčenie do zhlukov. V súbore *graphtools.py* sú implementované operácie s grafmi, v súbore *chameleon.py* sú implementované časti ako: relatívna a vnútorná interkonektivita a blízkosť. Tento algoritmus musel byť vymenený za jeho vylepšenú verziu, pretože neprodukoval akceptovateľné výsledky. Autorom vylepšenej verzie je Amber Lin a algoritmus je prebraný z [17]. Vylepšenie algoritmu je v poslednej časti, kde sa jednotlivé časti grafu (zhluky) spájajú a spočíva v pridaní počtu iterácií spájania a vyhodnocovania skóre pri spájaní.

Implementácia algoritmu DBSCAN, ktorej autorom je Sushant Kafle, bola prebraná z [15]. Algoritmus má na vstupe dva súbory, jeden je dátová sada a druhý je konfiguračný súbor, v ktorom sa nastavujú hodnoty parametrov ϵ, $MinPts$ a počet dimenzií. Samotný algoritmus je implementovaný v súbore *dbscanner.py*

Implementácia algoritmu SOM, bola prebraná z knižnice SimpSOM, ktorej autorom je Federico Comitani a je dostupná na [8]. Táto verzia je vhodná pre zhlukovanie a redukciu

dimenzionality. Implementácia obsahuje kombináciu algoritmu SOM a algoritmu DBSCAN, kde sú dáta najprv zhlukované podľa algoritmu SOM. Vytvorí sa zoznam víťazných neurónov, ktoré sú ďalej zhlukované pomocou algoritmu DBSCAN. Každý neurón predstavuje zhluk, ktorý obsahuje niekoľkých podobných objektov. Pri dátovej sade MNIST je použitá kombinácia algoritmu SOM a Quality Threshold algoritmu [13]. Spôsob zhlukovania je rovnaký ako pri kombinácii s algoritmom DBSCAN.

Kapitola 5

Návrh a implementácia aplikácie

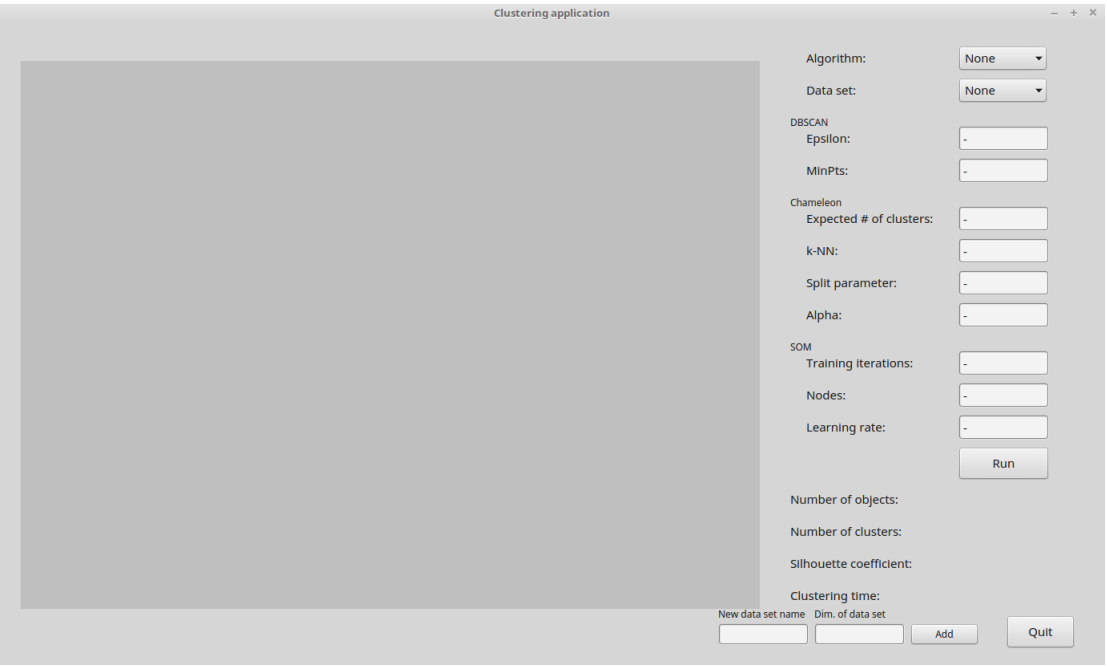
V nasledujúcej kapitole je vysvetlených návrh aplikácie. Kapitola sa tiež venuje implementácii aplikácie a komponentom, ktoré aplikácia používa.

5.1 Návrh aplikácie

Cieľom tejto diplomovej práce bolo vytvoriť aplikáciu, ktorá umožní testovať vlastnosti vybraných zhlukovacích algoritmov a bude vhodným spôsobom zobrazovať výsledky zhlukovania. Preto bol pri návrhu aplikácie kladený dôraz na jednoduchosť ovládania či zobrazovania výsledkov. Návrh aplikácie prešiel niekoľkými fázami, v ktorých bolo do návrhu pridávaných viac a viac prvkov. Pri návrhu grafického užívateľského rozhrania bol použitý QT Designer [23].

Aplikácia používa tri algoritmy a 5 dátových sád. Preto bolo vhodné použiť komponent *Combo Box* pre ich výber. Pre nastavovanie parametrov bola zvolená komponent *Line Edit*, do ktorého je možné zadávať hodnoty. Ďalej bolo nutné zvoliť komponent, do ktorého sa budú vypisovať informácie ako je čas zhlukovania, počet dát v dátovej sade, počet vytvorených zhlukov a pre hodnotu siluetového koeficientu. Pre tieto informácie bol zvolený komponent *Label*. Zhlukovanie bolo nutné spustiť nejakým komponentom, preto pre túto funkciu bol vybraný komponent *Button* s textom *RUN*. Takisto aj pre ukončenie aplikácie bol zvolený komponent *Button* s textom *QUIT*. Priestor pre vykresľovanie výsledkov zhlukovania v podobe grafov je umiestnený v ľavej strane aplikácie. Tento priestor je vytvorený pomocou komponentu *FigureCanvas*. Pre vkladanie novej dátovej sady boli použité komponenty *LineEdit* a pre pridanie komponent *Button*.

Výsledný návrh grafického užívateľského rozhrania aplikácie je možné vidieť na obrázku 5.1. Aplikácia obsahuje pomerne veľa komponentov, preto ich rozdelenie bolo zvolené nasledovne: v pravej hornej časti sa nachádzajú komponenty pre výber algoritmov a dátových sád. Pod nimi je priestor pre nastavovanie parametrov a spúšťacie tlačidlo zhlukovania. Po ukončení zhlukovaní sa výsledky zobrazia v pravej dolnej časti aplikácie a v ľavej strane aplikácie sa vykreslí graf.



Obr. 5.1: Výsledný návrh aplikácie.

5.2 Implementácia aplikácie

Vytvorená aplikácia bola napísaná v jazyku Python. Návrh aplikácie, ktorý bol vytvorený pomocou aplikácie QT Designer, bol prevedený na zdrojový kód v jazyku Python. Pri zvolení konkrétneho zhlukovacieho algoritmu a dátovej sady sa nastavia jednotlivé parametre, s ktorými boli dosiahnuté najlepšie výsledky. Nastavenia parametrov sa dajú ľubovoľne meniť. Funkcionalita jednotlivých tlačidiel je pripojená pomocou funkcie *connect()*.

Zhlukovacie algoritmy sú vyberané pomocou komponentu *ComboBox*. Aplikácia disponuje tromi zhlukovacími algoritmami DBSCAN [15], Chameleon [21] a SOM [8]. Tieto algoritmy sú volané na základe výberu položky v komponente *ComboBox*.

Vstupné dáta aplikácie sú samotné dátové sady (4.1), ktoré sú vo formáte CSV (*Comma-separated values*), čo predstavuje, že dáta v súbore sú oddelené čiarkou. Jednotlivé dátové sady sú vyberané podľa výberu v komponente *ComboBox*. Aplikácia umožňuje spúšťať algoritmy na piatich dátových sadách (Jain, Flame, Compound, t4.8k a MNIST). Alplikácia disponuje funkciou pridania vlastnej dátovej sady, kde sa nastaví názov dátovej sady, ktorý musí byť rovnaký ako názov súboru s dátovou sadou a počet dimenzií dátovej sady. Tlačidlo *Add* pridá dátovú sadu do komponentu *Combobox*. Nová dátová sada musí byť v CSV formáte *(hodnota1,hodnota2,...)* a uložená v priečinku *datasets*.

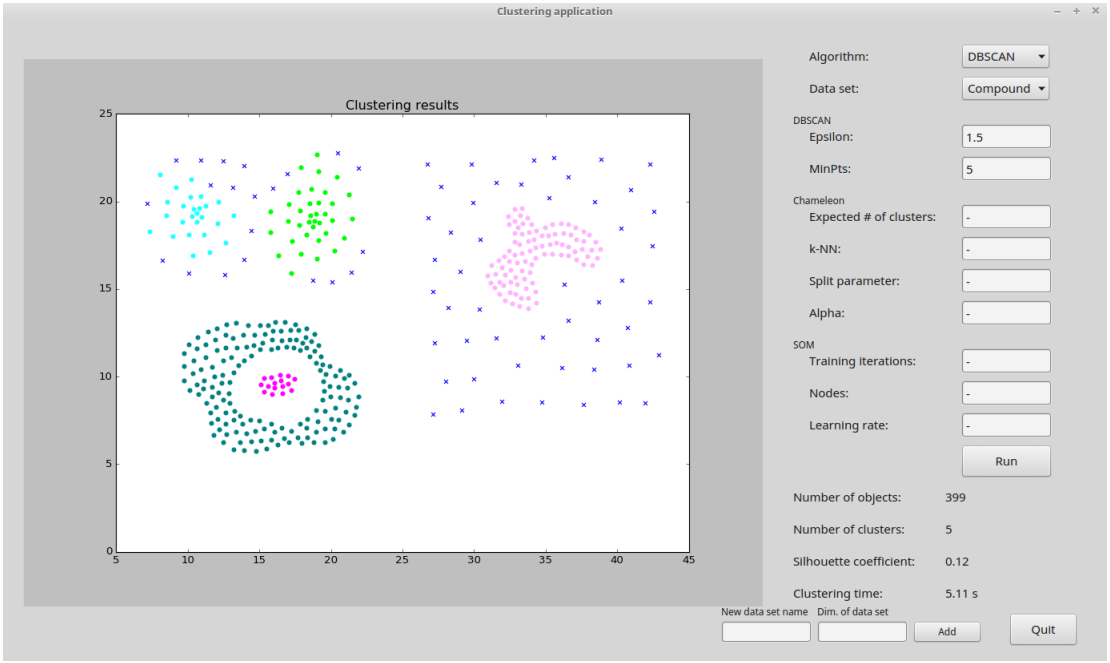
Zoznam parametrov, ktoré sa dajú editovať, pre jednotlivé algoritmy je nasledovný. Pre algoritmus DBSCAN sa nastavujú parametre *Epsilon* a *MinPts*. Parameter *Epsilon* udáva veľkosť okolia okolo objektu a parameter *MinPts* udáva prah hustoty.

Pre algoritmus Chameleon sa nastavujú parametre *Expected # of clusters*, *k-NN*, *Split parameter* a *Alpha*. Parameter *Expected # of clusters* udáva očakávaný počet zhlukov, parameter *k-NN* udáva parameter k pri k-nearest neighbor grafe, *Split parameter* udáva na koľko častí má byť rozdelený k-nearest neighbor graf a parameter *Alpha* udáva, či sa kladie väčší dôraz na relatívnu blízkosť ($Alpha > 1$) alebo na relatívnu interkonektivitu ($Alpha < 1$).

Pre algoritmus SOM sa nastavujú tri parametre: *Training iterations*, *Nodes*, *Learning rate*. Parameter *Training iterations* udáva počet trénovacích iterácií, parameter *Nodes* udáva počet uzlov siete (n x n) a parameter *Learning rate* udáva stupeň učenia.

Výsledky zhlukovania, ktoré jednotlivé algoritmy vrátia, sa zobrazujú do komponentu *Label*. Jedná sa o čas zhlukovania, siluetový koeficient, počet objektov v dátovej sade a počet zhlukov. Na meranie času zhlukovania bola použitá knižnica *time*. Formát času sa mení podľa toho, koľko času zaberie zhlukovanie. Ak zhlukovanie trvá menej ako 60 sekúnd, tak sa čas zhlukovania zobrazí iba vo forme sekúnd. Ak zhlukovanie trvá viac ako 1 minútu, tak sa čas zobrazí vo formáte minút a sekúnd. Pre určovanie siluetového koeficientu bola použitá knižnica *sklearn*, ktorá obsahuje modul *metrics* a funkciu *silhouette_score*, ktorá udáva, ako dobre sú zhluky kompaktné.

Na vykreslenie dátovej sady vo forme zhlukov bola použitá knižnica *matplotlib*. Výsledok zhlukovania dátovej sady sa vykreslí do priestoru komponentu *FigureCanvas*. Jednotlivé zhluky dát sú farebne odlíšené. Pri algoritme SOM sa okrem objektov vykresľujú aj pozície neurónov po zhlukovaní (väčšie zelené bodky). Ukážku grafického užívateľského rozhrania aplikácie je možné vidieť na obrázku 5.2. Výsledky zhlukovania sa automaticky exportujú do CSV súboru, v ktorom sa nachádzajú jednotlivé body a ich príslušnosť do konkrétneho zhluku.



Obr. 5.2: Ukážka GUI aplikácie.

Kapitola 6

Experimenty s hľadaním správnych parametrov

V tejto kapitole sú popísané spôsoby hľadania správnych parametrov algoritmov DBSCAN, Chameleon a SOM pre dátové sady Jain, Flame, Compound, t4.8k a MNIST. Každý algoritmus potrebuje pre rôzne dátové sady rôzne hodnoty parametrov. Tieto hodnoty boli hľadané experimentálne postupným spúšťaním algoritmov na jednotlivých dátových sadách a výsledky boli zaznamenávané a následne vyhodnotené.

6.1 Algoritmus DBSCAN

Pre algoritmus DBSCAN bolo nutné zistiť vhodné hodnoty parametru *Epsilon*, ktorý udáva veľkosť okolia objektu a parametru *MinPts*, ktorý udáva prah hustoty. Pri nevhodne zvolených hodnotách parametrov môžeme dostať neakceptovateľné výsledky. Napríklad pri malej hodnote parametra *Epsilon* a veľkej hodnote parametra *MinPts* sa nemusia nájsť žiadne zhluky a dáta môžu byť označené za šum, naopak pri veľkej hodnote parametra *Epsilon* a malej hodnote parametra *MinPts* sa všetky objekty spoja do jedného zhluku. Odľahlé hodnoty sú na všetkých obrázkoch označené ako $\times$.

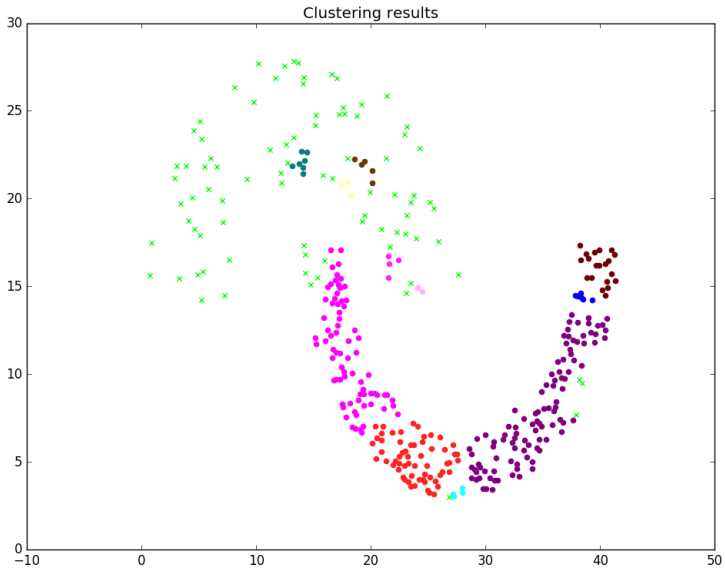
6.1.1 Dátová sada Jain

Dátová sada Jain obsahuje 373 objektov, ktoré vytvárajú 2 zhluky, tieto zhluky je možné vidieť na obrázku 4.2. Celkovo bolo vykonaných 11 experimentov, kde v každom experimente boli použité rôzne hodnoty parametrov, ako je možné vidieť v tabuľke 6.1.

Podľa výstupov z aplikácie bolo zistené, že pri experimentoch, v ktorých sa v parametre *Epsilon* vyskytujú hodnoty 1 a 1.5, tak bolo vytvorených pomerne veľké množstvo zhlukov, avšak veľké množstvo objektov bolo označené za šum, ako je možné vidieť na obrázku 6.1.

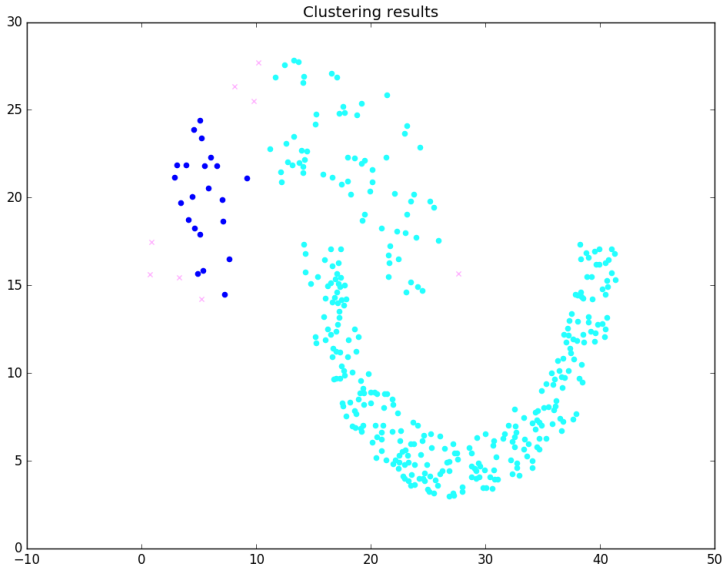
experiment	1	2	3	4	5	6
Epsilon	1	1	1.5	1.5	1.5	2
MinPts	2	3	2	3	6	3
clusters	20	11	11	9	2	6
silhouette coefficient	0.02	0.01	0.04	0.01	0.22	0.22
time [s]	5.05	4.52	4.53	4.57	4.87	4.57
experiment	7	8	9	10	11	
Epsilon	2	2.5	2.5	3	3.5	
MinPts	4	2	5	6	7	
clusters	5	3	3	2	2	
silhouette coefficient	0.21	0.35	0.32	0.33	0.33	
time [s]	4.57	4.84	4.60	4.67	4.81	

Tabuľka 6.1: Prehľad výsledkov jednotlivých experimentov algoritmu DBSCAN pre dátovú sadu Jain.



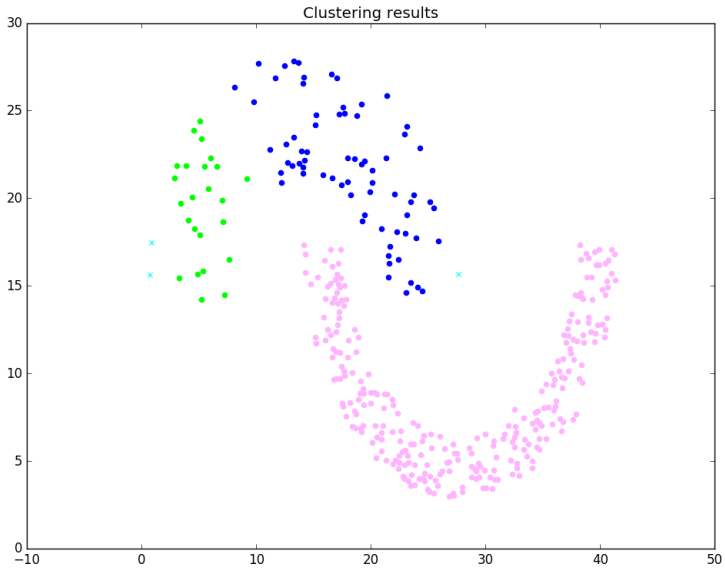
Obr. 6.1: DBSCAN (Epsilon=1, MinPts=3) pre dátovú sadu Jain.

Pri parametre *Epsilon*, ktorého hodnota bola 3 sa časti vrchného a spodného zhluku spojili a celkovo boli vytvorené dva zhluky ako bolo na začiatku požadované. Avšak zhluky, ktoré sa vytvorili, ako je možné vidieť na obrázku 6.2, sú vytvorené nesprávne. Tento výsledok zhlukovania bol vyhodnotený ako nežiaduci.



Obr. 6.2: DBSCAN (Epsilon=3, MinPts=7) pre dátovú sadu Jain.

Najlepší výsledok zhlukovania bol dosiahnutý pri experimente, v ktorom parameter *Epsilon* nadobúdal hodnotu 2.5 a parameter *MinPts* hodnotu 2. V tomto experimente boli vytvorené 3 zhluky, ako je možné vidieť na obrázku 6.3. Požadované dva zhluky sa nepodarilo vytvoriť, pretože pri inom rozdelení hodnôt dostaneme buď spojenú vrchnú a spodnú časť zhluku alebo veľké množstvo šumu.



Obr. 6.3: DBSCAN (Epsilon=2.5, MinPts=2) pre dátovú sadu Jain.

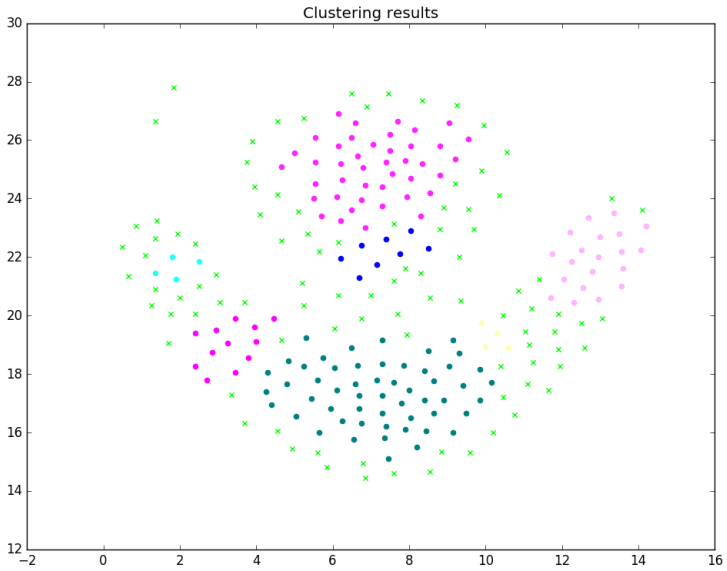
6.1.2 Dátová sada Flame

Táto dátová sada obsahuje 240 objektov a taktiež ako dátová sada Jain vytvára dva zhluky. Na rozdiel od dátovej sady Jain je hranica zhlukov v tejto dátovej sade horšie viditeľná. Celkovo bolo vykonaných 18 experimentov, ale v tabuľke 6.2 je zapísaných len 7, z dôvodu opakovania výsledkov.

experiment	1	2	3	4	5	6	7
Epsilon	1	1	1	1	1	1.5	2
MinPts	2	5	6	7	9	3	5
clusters	1	2	3	7	2	1	1
silhouette coefficient	-	0.30	0.28	-0.07	-0.20	-	-
time [s]	1.91	1.94	2.05	1.94	1.91	1.96	2.04

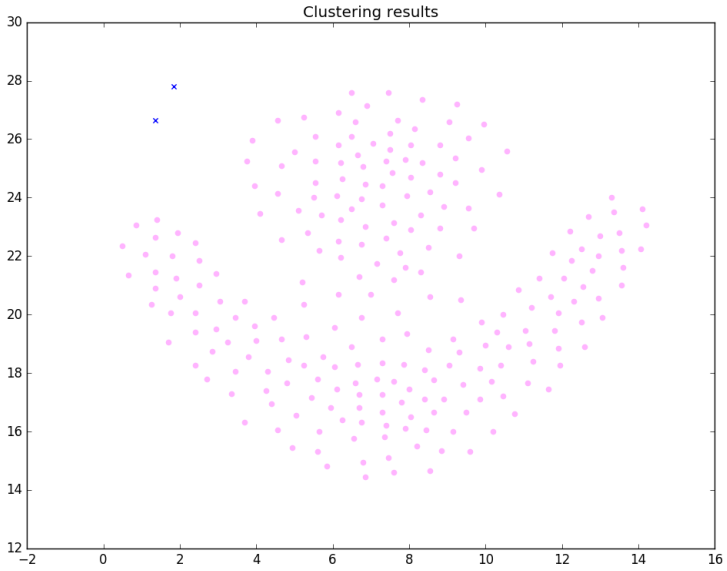
Tabuľka 6.2: Prehľad výsledkov jednotlivých experimentov algoritmu DBSCAN pre dátovú sadu Flame.

Pri parametri *Epsilon* s hodnotou 1 a zvyšujúcou hodnotou parametru *MinPts* sa zvyšovalo aj množstvo dát označených za šum. Napriek tomu, že pri hodnotách 6, 7 a 9 parametru *MinPts* sa vytvorili pomerne dobré a kvalitné zhluky (obrázok 6.4), je množstvo zašumených dát dosť vysoké.



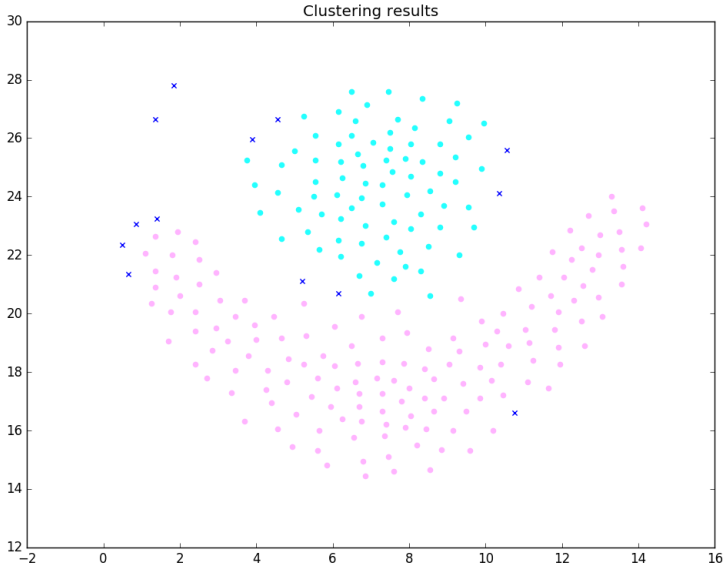
Obr. 6.4: DBSCAN (Epsilon=1, MinPts=7) pre dátovú sadu Flame.

Naopak pri hodnotách 1.5 a 2 parametru *Epsilon* bol vytvorený len jeden zhluk a dve odľahlé hodnoty označené za šum (obrázok 6.5).



Obr. 6.5: DBSCAN (Epsilon=2, MinPts=5) pre dátovú sadu Flame.

Najlepší výsledok bol dosiahnutý s hodnotou 1 pre parameter *Epsilon* a 5 pre parameter *MinPts* (obrázok 6.6). Boli vytvorené 2 zhluky a výsledok obsahoval iba 13 objektov označených za šum. Je ťažké povedať ako majú byť zhluky presne rozdelené pre túto dátovú sadu, pretože hranica zhlukov nie je dobre rozlíšiteľná.



Obr. 6.6: DBSCAN (Epsilon=1, MinPts=5) pre dátovú sadu Flame.

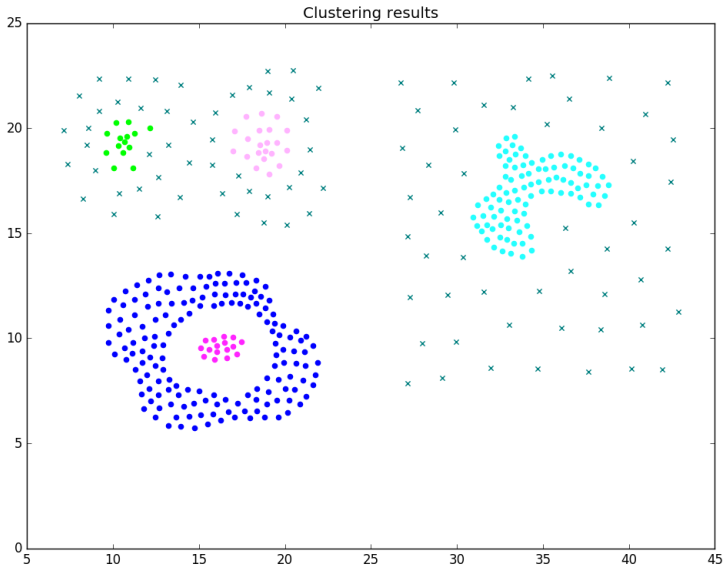
6.1.3 Dátová sada Compound

Dátová sada Compound obsahuje 399 objektov usporiadaných do piatich až šiestich zhlukov (obrázok 4.3). Pre nájdenie vhodných parametrov bolo vykonaných 11 experimentov, ktorých hodnoty parametrov je možné vidieť v tabuľke 6.3.

experiment	1	2	3	4	5	6
Epsilon	1	1	1	1.5	1.5	1.5
MinPts	3	7	9	3	5	7
clusters	5	8	6	5	5	5
silhouette coefficient	0.03	0.02	-0.16	0.14	0.12	0.07
time [s]	5.53	5.35	5.34	5.40	5.39	5.40
experiment	7	8	9	10	11	
Epsilon	2	2	2	3	4	
MinPts	3	5	7	3	3	
clusters	4	3	4	2	2	
silhouette coefficient	0.40	0.45	0.42	0.43	0.64	
time [s]	5.40	5.44	5.39	5.67	5.53	

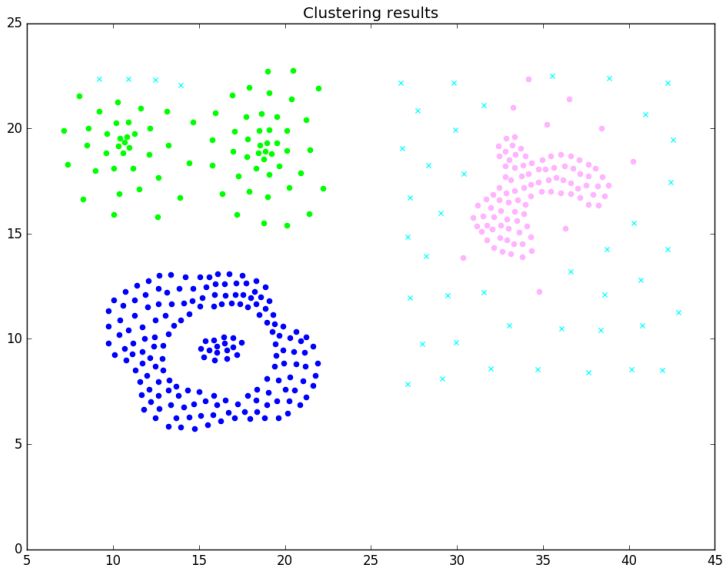
Tabuľka 6.3: Prehľad výsledkov jednotlivých experimentov algoritmu DBSCAN pre dátovú sadu Compound.

V prvom experimente, kde bola hodnota 1 pre parameter *Epsilon* a hodnota 3 pre parameter *MinPts* sa vytvorili pekné kompaktné zhluky oválneho tvaru, avšak pri zhlukoch umiestnených vľavo hore (obrázok 6.7) sa nachádzalo veľa objektov označených za šum. Zvyšovaním hodnoty parametru *MinPts* bolo označených viac a viac za šum.



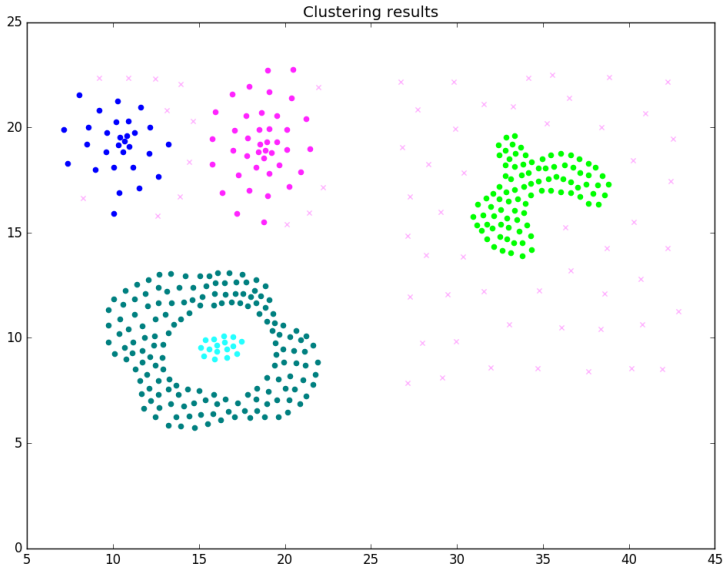
Obr. 6.7: DBSCAN (Epsilon=1, MinPts=3) pre dátovú sadu Compound.

Pri vyších hodnotách ako 1.5 parametra *Epsilon* došlo k spájaniu niektorých zhlukov (obrázok 6.8), čo bolo v tomto prípade považované ako nesprávne riešenie. Išlo hlavne o zhluky dát nachádzajúce sa v ľavej časti dátovej sady. Pri experimente číslo 11 sa vytvorili len dva zhluky, ktoré tvorili objekty len pravej a len ľavej strany dátovej sady.



Obr. 6.8: DBSCAN (Epsilon=2, MinPts=5) pre dátovú sadu Compound.

Pri dátovej sade Compound boli dosiahnuté najlepšie výsledky pri hodnote 1.5 parametru *Epsilon* a hodnotách 3 a 5 parametru *MinPts* (obrázok 6.9). Pri zhlukoch v ľavej hornej časti, taktiež ako pri dátovej sade Flame, nie je úplne jasná hranica, ktorá tieto zhluky oddeľuje, preto sme sa rozhodli ako najlepšie výsledky uviesť hodnoty parametrov experimentu číslo 4 a 5 (tabuľka 6.3). Zhluk v pravej časti dátovej sady je obklopený objektmi označenými ako šum, čo by malo byť správne, pretože objekty sú pomerne ďaleko od seba, aj keď na prvý pohľad vytvárajú zhluk.



Obr. 6.9: DBSCAN (Epsilon=1.5, MinPts=3) pre dátovú sadu Compound.

6.1.4 Dátová sada t4.8k

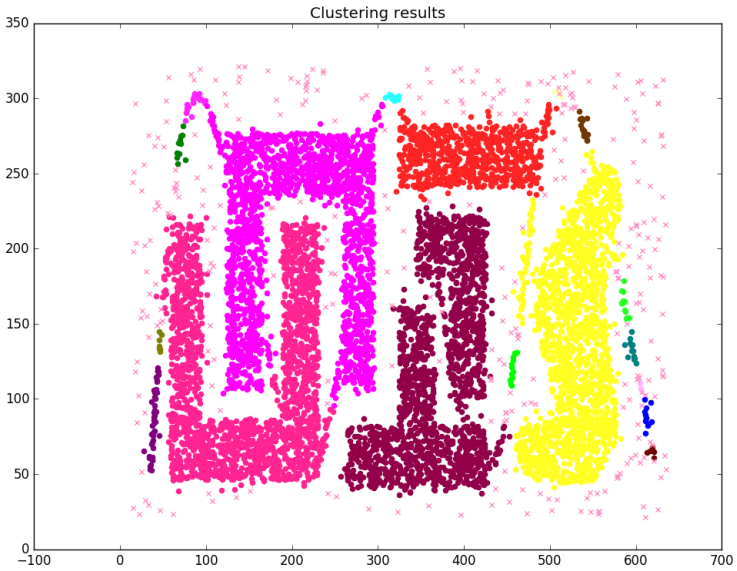
Dátová sada t4.8k obsahuje 8000 objektov a vytvára 6 zhlukov (obrázok 4.5). Táto dátová sada obsahuje veľké množstvo zašumených dát. Celkovo pre nájdenie vhodných parametrov bolo vykonaných 15 experimentov, ktoré sú zaznamenané v tabuľke 6.4.

experiment	1	2	3	4	5	6	7	8
Epsilon	4	8	8	8	8	8	8	9
MinPts	5	6	8	10	13	15	20	6
clusters	302	18	17	13	9	12	37	14
silhouette coefficient	-0.08	-0.10	-0.01	0.01	0.03	0.00	-0.09	-0.28
time [min]	31.0	36.2	36.7	34.5	37.6	37.5	37.6	37.9
experiment	9	10	11	12	13	14	15	
Epsilon	9	9	9	9	9	9	15	
MinPts	8	12	13	14	15	16	12	
clusters	16	11	10	9	7	6	3	
silhouette coefficient	-0.07	0.03	0.03	0.09	0.19	0.24	-0.14	
time [min]	36.5	37.2	36.2	36.5	38.5	35.9	44.5	

Tabuľka 6.4: Prehľad výsledkov jednotlivých experimentov algoritmu DBSCAN pre dátovú sadu t4.8k.

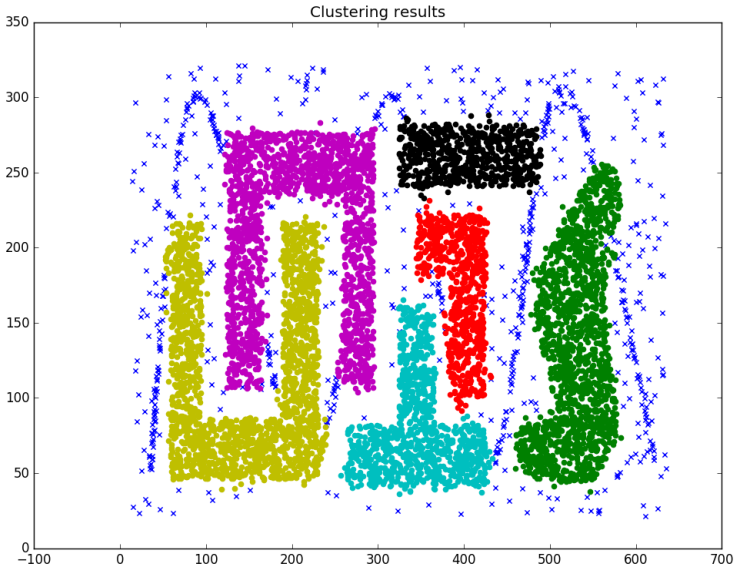
V experimentoch číslo 2, 8, 9 a 15 došlo ku spojeniu niektorých zhlukov, ako je vidieť na obrázku 6.10. Tieto riešenia neboli považované za správne. Toto spojenie je zapríčinené aj vlnovkou, ktorá prechádza cez všetky zhluky a na základe týchto parametrov sa algoritmu nepodarilo zhluky rozdeliť správne. Pri experimente číslo 1, 7 došlo k vytvoreniu veľkého množstva zhlukov. V prvom prípade bolo vytvorených 302 zhlukov a v druhom prípade 37 zhlukov. Výsledok obsahoval väčšie množstvo šumu v častiach, kde mali byť vytvorené

zhluky. Tieto výsledky, rovnako, nebolo možné považovať za správne. Pri experimentoch číslo 10, 11, 12 a 13 boli dosiahnuté takmer správne výsledky, ale vytvorili sa malé zhluky na vlnovke, ktorá spájala hlavné zhluky, preto sme pokračovali so zvyšovaním hodnoty parametra *MinPts*.



Obr. 6.10: DBSCAN (Epsilon=8, MinPts=6) pre dátovú sadu t4.8k.

Za najlepší výsledok sa dá považovať experiment číslo 14, s hodnotou 9 pre parameter *Epsilon* a hodnotou 16 pre parameter *MinPts* (obrázok 6.11). Výsledok zhlukovania obsahoval 6 zhlukov. Všetky objekty vlnovky boli označené ako šum, čo bolo správne.



Obr. 6.11: DBSCAN (Epsilon=9, MinPts=16) pre dátovú sadu t4.8k.

6.1.5 Dátová sada MNIST

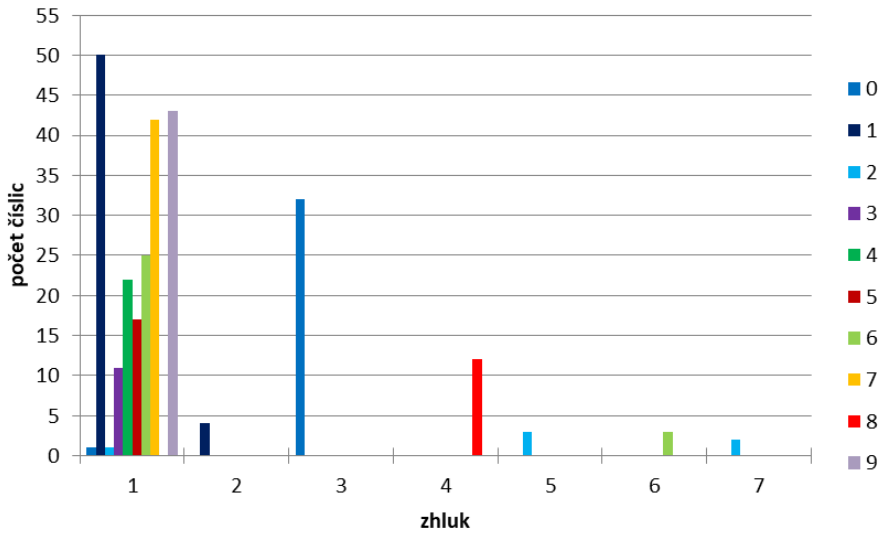
Dátová sada MNIST obsahuje 10000 objektov a vytvára 10 zhlukov, kde každý zhluk predstavuje jednu písanú číslicu (obrázok 4.1). Experimenty prebiehali tak, že po zhlukovaní sa jednotlivé obrázky číslic jednotlivých zhlukov uložili do priečinku a následne boli ručne analyzované. Vzhľadom na čas potrebný k uloženiu obrázkov, boli experimenty vykonané a analyzované pre 500 a 1000 objektov dátovej sady MNIST, ktoré boli vybrané náhodne. Algoritmus bol spustený aj na 5000 objektoch dátovej sady a na celej dátovej sade, ale jednotlivé zhluky neboli analyzované z dôvodu veľkého množstva času potrebného na vykreslenie a uloženie obrázkov. Výsledky zhlukovania sú uložené v súbore, ktorý obsahuje jednotlivé objekty a ich príslušnosť do zhluku. Celkovo bolo vykonaných 5 experimentov pre 500 objektov dátovej sady a 3 experimenty pre 1000 objektov, ktoré je možné vidieť v tabuľke 6.5.

experiment	1	2	3	4	5	6	7	8
number of objects	500	500	500	500	500	1000	1000	1000
Epsilon	1500	1650	1650	1650	1650	1650	1600	1600
MinPts	5	3	1	2	4	3	3	2
clusters	3	7	32	7	3	6	11	16
silhouette coefficient	-0.06	-0.06	-0.08	-0.04	0.02	0.00	-0.10	0.07
time [min]	2.54	2.56	2.60	2.60	2.60	10.3	10.4	10.4

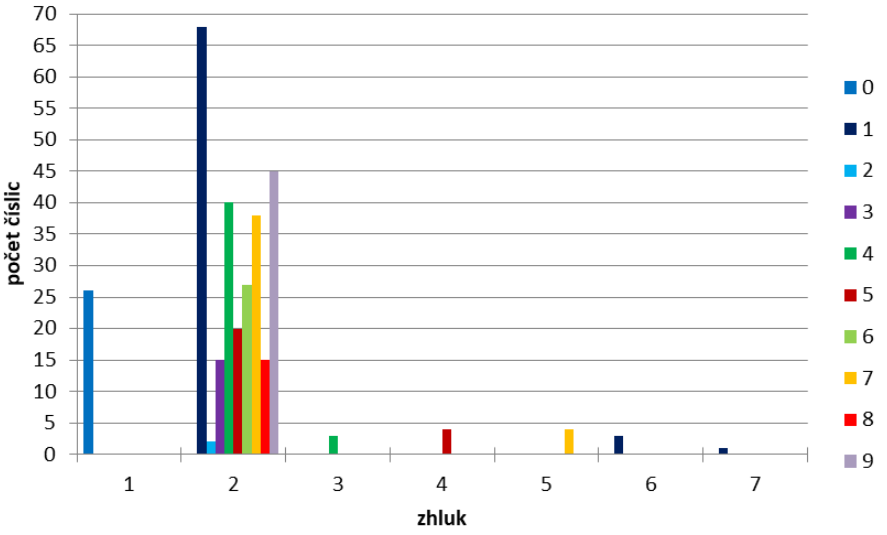
Tabuľka 6.5: Prehľad výsledkov jednotlivých experimentov algoritmu DBSCAN pre dátovú sadu MNIST.

V experimente číslo 1 boli vytvorené iba 3 zhluky, pričom bolo 364 objektov označených za šum, v experimente číslo 5 boli vytvorené takisto len 3 zhluky a veľké množstvo objektov označených za šum. V týchto prípadoch počet zhlukov nebol dostačujúci.

V experimente číslo 2 (obrázok 6.12) a 4 (obrázok 6.13) bolo vytvorených 7 zhlukov, kde zastúpenie jednotlivých číslic je možné vidieť na obrázku 6.12 a 6.13. V týchto prípadoch bolo označených za šum taktiež veľké množstvo objektov (experiment číslo 1 - 232 objektov, experiment číslo 4 - 189 objektov). Taktiež je možné vidieť na obrázkoch, že v jednom zhluku je vždy obsiahnutých viacero číslic.

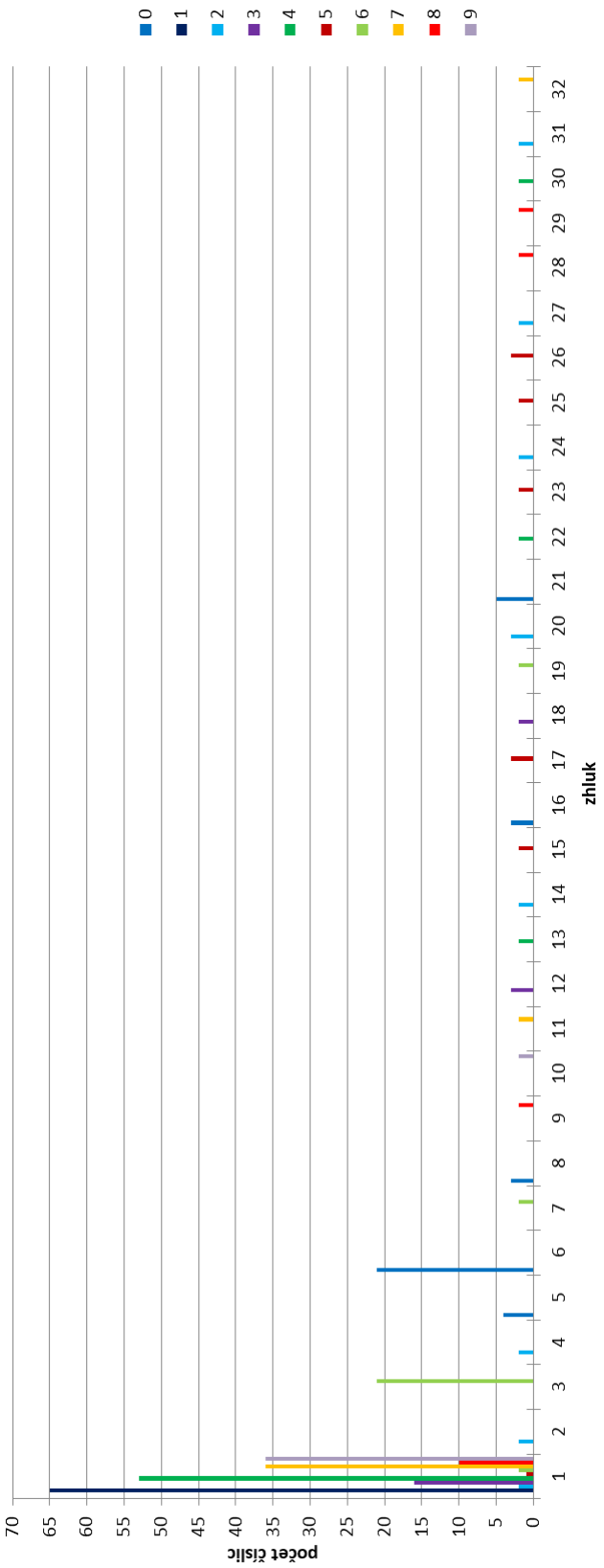


Obr. 6.12: DBSCAN (Epsilon=1650, MinPts=3) pre dátovú sadu MNIST (500 objektov).



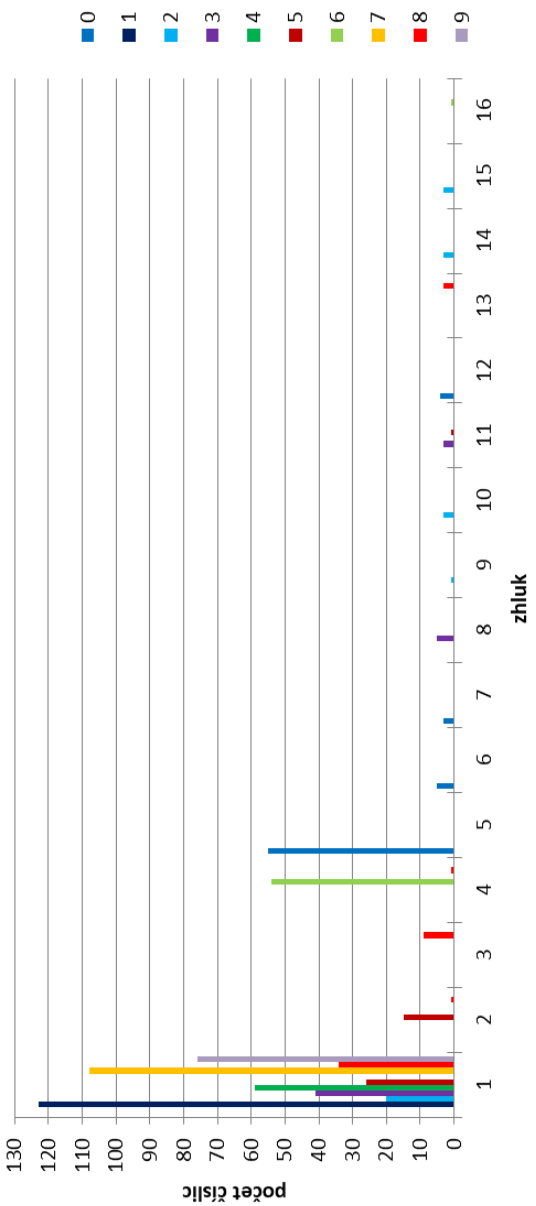
Obr. 6.13: DBSCAN (Epsilon=1650, MinPts=2) pre dátovú sadu MNIST (500 objektov).

V experimente číslo 3 bolo vytvorených až 32 zhlukov, kde zastúpenie jednotlivých číslic v zhlukoch je možné vidieť na obrázku 6.14. V tomto experimente bolo označených za šum 168 objektov, čo je najmenej spomedzi všetkých experimentov. Rovnako, v tomto prípade jeden zhluk obsahuje viacero číslic. Niektoré zhluky obsahovali menšie množstvo objektov, ale väčšinou sa v týchto zhlukoch sa vyskytovala tá istá číslica.



Obr. 6.14: DBSCAN (Epsilon=1650, MinPts=1) pre dátovú sadu MNIST (500 objektov).

V experimentoch číslo 6-8, ktoré boli vykonané pre 1000 objektov dátovej sady MNIST, boli výsledky podobné ako pre 500 objektov. Veľké množstvo objektov bolo zoskupených v jednom zhluku a veľké množstvo objektov bolo označené za šum. Na obrázku 6.15 je možné vidieť výsledok experimentu číslo 8. V tomto experimente bolo vytvorených 16 zhlukov, kde znovu jeden zhluk obsahoval veľké množstvo objektov, ale dva zhluky tvorili celkom presvedčivé rozdelenie, kde v jednom zhluku (zhluk číslo 5) bolo 55 číslic "0", v druhom (zhluk číslo 4) 54 číslic "6". Počet objektov označených za šum bol 342.



Obr. 6.15: DBSCAN (Epsilon=1600, MinPts=2) pre dátovú sadu MNIST (1000 objektov).

Experiment pre 5000 objektov a celú dátovú sadu bol vykonaný s rovnakými parametrami. Parameter *Epsilon* nadobúdal hodnotu 1600 a parameter *MinPts* nadobúdal hodnotu 2. Tieto parametre sme vybrali podľa najlepšieho výsledku, ktorý sme získali pre 500 a 1000

objektov dátovej sady. Pre 5000 objektov bolo vytvorených 22 zhlukov. V jednom zhluku bolo až 4213 objektov a ostatné zhluky tvorili zoskupenia 1-13 objektov. Za šum bolo označených 725 objektov.

Pre celú dátovú sadu bolo vytvorených 25 zhlukov, v jednom zhluku bolo až 9065 objektov a 860 objektov bolo označených za šum. Ostatné zhluky obsahovali 1-10 objektov. Výsledky zhlukovania sú zapísané v súboroch formátu CSV v priečinku *results*, kde každému objektu je pridelený zhluk, do ktorého patrí. Analýza týchto výsledkov nebola vykonaná z dôvodu času potrebného na uloženie každého objektu (číslice).

V experimentoch sme zistili, že DBSCAN nie je vhodný pre túto dátovú sadu, pretože vždy označil veľké množstvo objektov za šum a taktiež väčšiu časť dátovej sady pridelil do jedného zhluku. Pri nižších alebo vyšších hodnotách parametra *Epsilon*, ako sú uvedené v tabuľke 6.5, sa vytvoril menší počet zhlukov, taktiež so zvyšovaním hodnoty parametra *MinPts* došlo k postupnému znižovaniu počtu zhlukov.

6.2 Algoritmus Chameleon

Hľadanie vhodných parametrov pre tento algoritmus bolo trochu zložitejšie ako pri algoritme DBSCAN, pretože je nutné nájsť vhodné hodnoty pre 4 parametre. Pre túto implementáciu algoritmu Chameleon, ktorá bola použitá, sa nastavujú parametre *Expected # of clusters*, *k-NN*, *Split parameter*, *Alpha*, ktoré sú popísané v sekcii 5.2.

6.2.1 Dátová sada Jain

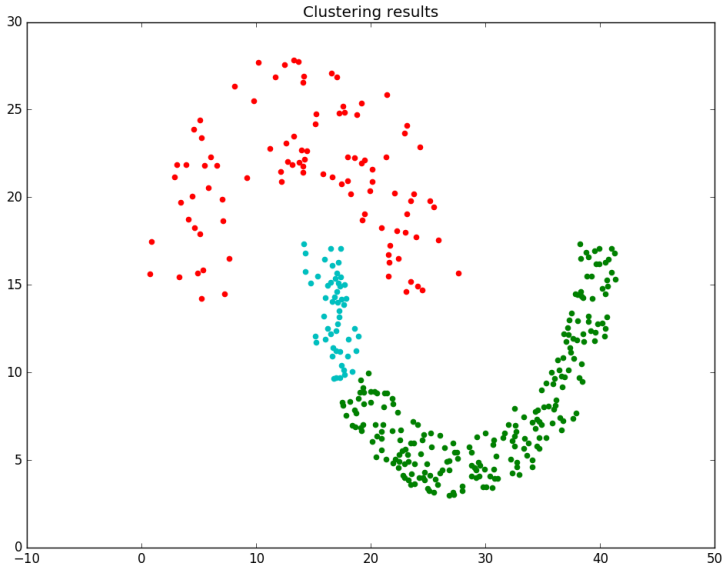
Pre túto dátovú sadu bolo vykonaných 11 experimentov, ktoré je možné vidieť v tabuľke 6.6.

experiment	1	2	3	4	5	6
Expected # of clusters	2	2	2	2	2	3
k-NN	3	3	3	4	4	4
Split	5	10	10	5	5	5
Alpha	2	2	0.5	2	0.5	0.5
clusters	3	3	3	2	2	3
silhouette coefficient	0.45	0.45	0.45	0.40	0.40	0.30
time [s]	3.47	6.39	6.65	4.12	4.22	3.61
experiment	7	8	9	10	11	
Expected # of clusters	3	5	5	8	8	
k-NN	4	4	4	4	4	
Split	5	5	10	10	10	
Alpha	2	2	2	2	0.5	
clusters	3	5	5	8	8	
silhouette coefficient	0.30	0.38	0.35	0.27	0.27	
time [s]	3.55	2.19	5.71	3.75	3.67	

Tabuľka 6.6: Prehľad výsledkov jednotlivých experimentov algoritmu Chameleon pre dátovú sadu Jain.

Na prvý pohľad dátová sada obsahuje dva zhluky. Preto hodnota parametra *Expected # of clusters* bola na začiatku zvolená ako 2. Neskôr sa táto hodnota zvyšovala, z dôvodu presvedčenia sa, či sa nenájde lepšie rozdelenie objektov do zhlukov. Pri experimentoch

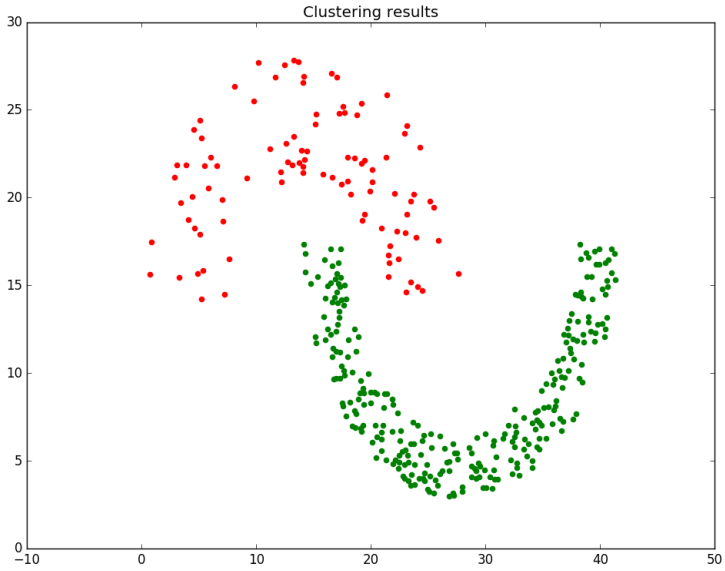
číslo 6-11 bolo zistené, že sa vytvorí väčší počet zhlukov ako 2, pričom sa vždy rozdeľovala spodná vlnovka a horná zostala spojená do jedného zhluku (obrázok 6.16).



Obr. 6.16: Chameleon (Expected # of clusters=3, k-NN=4, Split=5, Alpha=0.5) pre dátovú sadu Jain.

Pri experimentoch číslo 1-3, napriek tomu že boli požadované 2 zhluky, boli vytvorené 3 zhluky, kde sa spodná vlnovka rozdelila na 2 zhluky podobne, ako na obrázku 6.16.

Najlepší výsledok bol dosiahnutý pri experimentoch číslo 4 a 5, kde sa vytvorili dva zhluky (obrázok 6.17) a bolo zistené, že v tomto prípade parameter *Alpha* nemal vplyv na výsledok. V tomto prípade sa na začiatku algoritmu vytváral 4-NN graf (parameter *k-NN*), ktorý sa neskôr rozdelil na 5 častí (parameter *Split*), ktoré boli spojené do 2 zhlukov.



Obr. 6.17: Chameleon (Expected # of clusters=2, k-NN=4, Split=5, Alpha=2) pre dátovú sadu Jain.

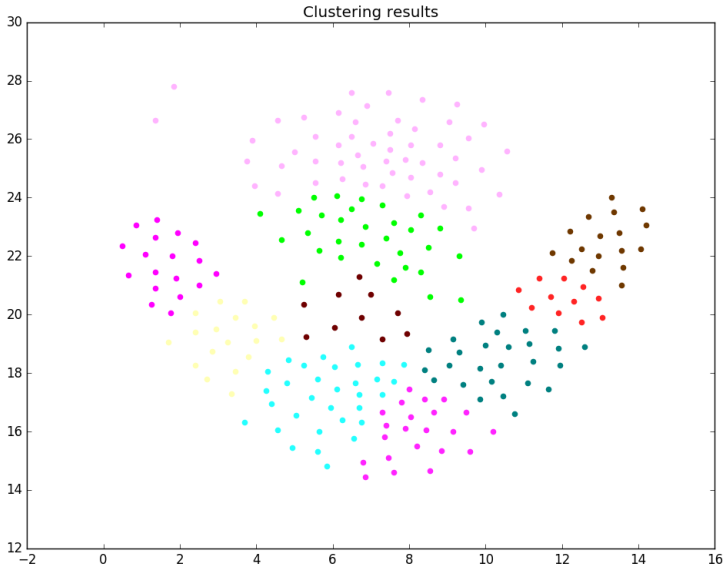
6.2.2 Dátová sada Flame

Pri dátovej sade Flame bolo celkovo vykonaných 12 experimentov (tabuľka 6.7).

experiment	1	2	3	4	5	6
Expected # of clusters	2	2	2	2	2	5
k-NN	4	4	6	10	6	6
Split	10	15	15	15	25	15
Alpha	0.5	2	2	2	2	1
clusters	2	2	2	2	2	5
silhouette coefficient	0.27	0.34	0.30	0.30	0.30	0.33
time [s]	5.45	8.11	9.67	11.7	17.3	7.94
experiment	7	8	9	10	11	12
Expected # of clusters	5	5	5	10	10	10
k-NN	10	6	10	6	10	6
Split	15	25	40	15	15	25
Alpha	2	2	2	2	2	2
clusters	5	5	15	10	10	10
silhouette coefficient	0.29	0.21	-0.38	0.28	0.29	0.21
time [s]	9.22	15.3	62.1	4.73	5.24	11.8

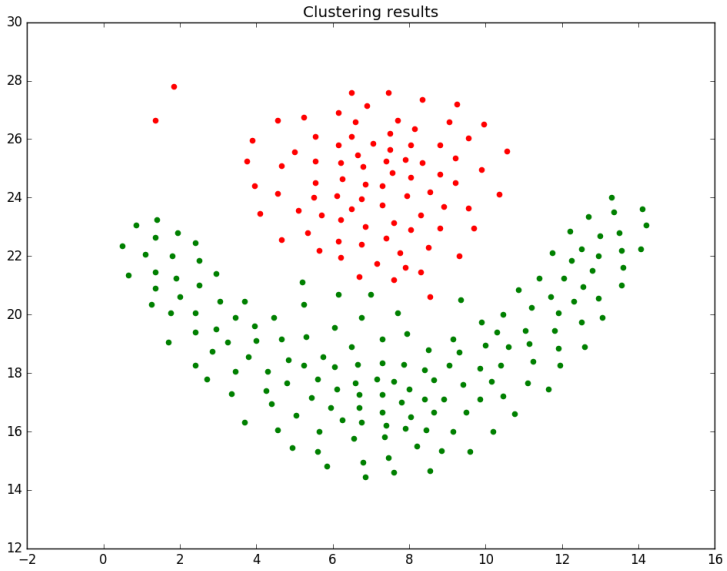
Tabuľka 6.7: Prehľad výsledkov jednotlivých experimentov algoritmu Chameleon pre dátovú sadu Flame.

Pri experimentoch číslo 6-12 bol dosiahnutý vyšší počet zhlukov, čo mal hlavne za následok parameter *Expected # of clusters*. V niektorých prípadoch boli vytvorené malé kompaktné zhluky (obrázok 6.18), v iných prípadoch toto rozdelenie do zhlukov nebolo správne.

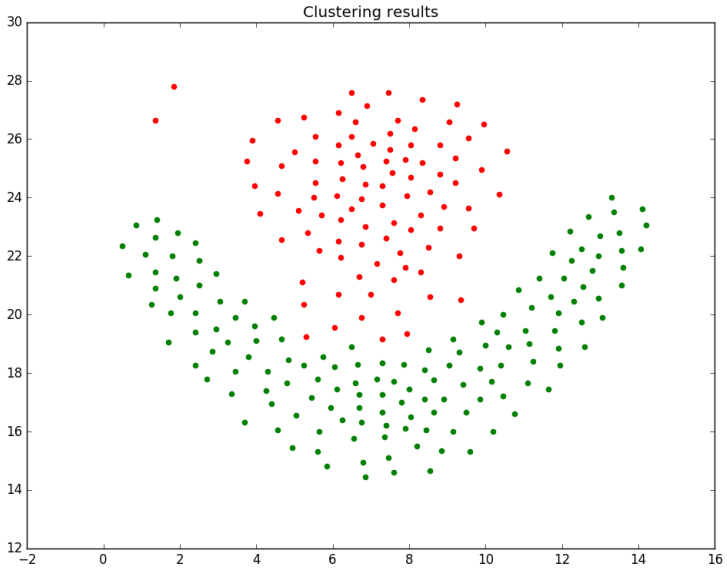


Obr. 6.18: Chameleon (Expected # of clusters=10, k-NN=10, Split=15, Alpha=2) pre dátovú sadu Flame.

Najlepšie výsledky boli dosiahnuté pri experimente číslo 2 (obrázok 6.19) a 3 (obrázok 6.20), kde sa vytvorili dva zhluky. Hranica rozdelenia zhlukov v tomto prípade nie je jednoznačná, preto je ťažké rozhodnúť, ktorý výsledok je správny. Za najlepší výsledok bol označený výsledok experimentu číslo 3, pretože dolný zhluk končí na pomyselnom oblúku, ktorý oddeľuje vrchnú a spodnú časť dátovej sady (obrázok 6.20). V tomto prípade bol vytvorený 4-NN graf, ktorý bol rozdelený na 15 častí a následne spojený do dvoch zhlukov.



Obr. 6.19: Chameleon (Expected # of clusters=2, k-NN=4, Split=15, Alpha=2) pre dátovú sadu Flame.



Obr. 6.20: Chameleon (Expected # of clusters=2, k-NN=6, Split=15, Alpha=2) pre dátovú sadu Flame.

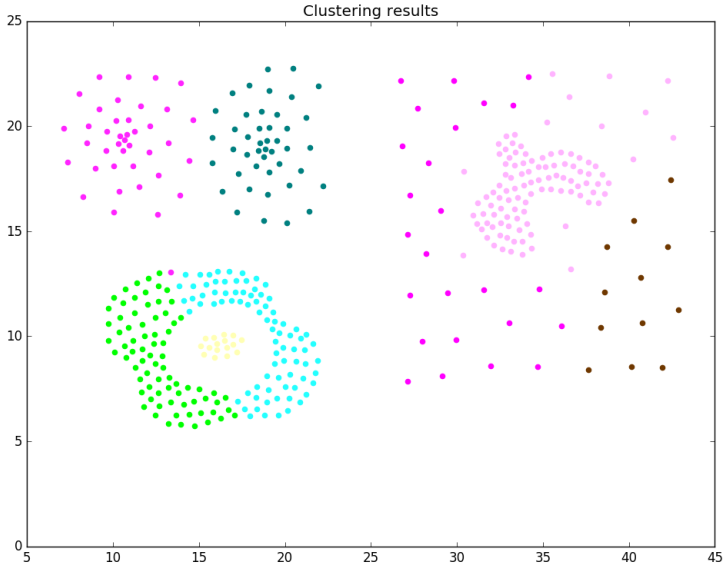
6.2.3 Dátová sada Compound

Pre nájdenie vhodných parametrov bolo pre dátovú sadu Compound vykonaných 9 experimentov (tabuľka 6.8).

experiment	1	2	3	4	5	6	7	8	9
Expected # of clusters	3	3	5	5	5	5	6	8	10
k-NN	10	10	10	6	6	10	10	10	10
Split	20	40	20	20	40	40	40	40	20
Alpha	0.5	2	2	2	0.5	2	2	2	2
clusters	3	3	5	5	5	5	6	8	10
silhouette coefficient	0.44	0.13	0.46	0.27	0.27	0.28	0.19	0.31	0.32
time [s]	30.0	69.6	26.5	19.5	56.1	66.0	64.2	64.3	17.7

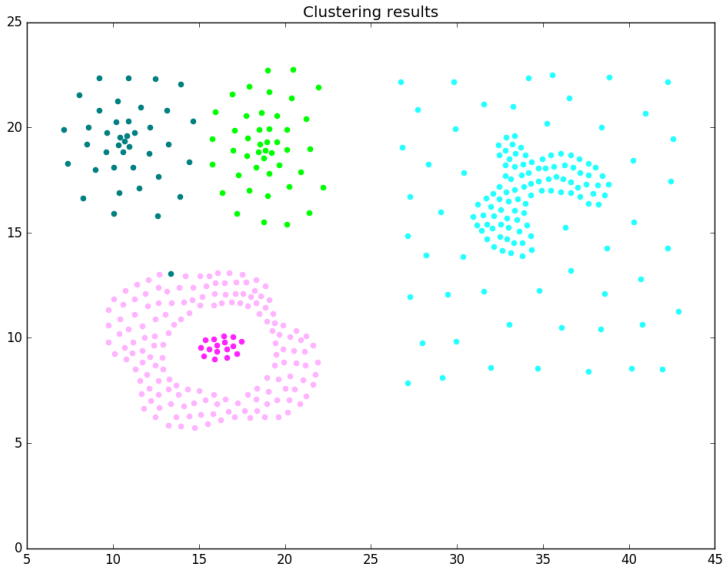
Tabuľka 6.8: Prehľad výsledkov jednotlivých experimentov algoritmu Chameleon pre dátovú sadu Compound.

Pri experimente číslo 1 a 2 boli vytvorené 3 zhluky, čo je pre túto dátovú sadu nedostačujúce. Pri experimentoch číslo 7-9 bol vytvorený väčší počet zhlukov, ako bol požadovaný. Vytvorilo sa viac menších zhlukov v pravej a v ľavej dolnej časti dátovej sady (obrázok 6.21).



Obr. 6.21: Chameleon (Expected # of clusters=8, k-NN=10, Split=40, Alpha=2) pre dátovú sadu Compound.

Najlepší výsledok bol dosiahnutý pre experiment číslo 6, v ktorom bolo vytvorených 5 zhlukov (obrázok 6.22). Nakoľko sa nepodarilo zhluk v pravej časti dátovej sady rozdeliť na dva zhluky, bolo rozhodnuté, že sa tento výsledok označí za najlepší získaný. Pri tomto experimente bol vytvorený 10-NN graf, ktorý bol rozdelený na 40 častí a následne spojený do piatich zhlukov. Spájanie do zhlukov kládlo dôraz hlavne na relatívnu blízkosť, pretože parameter *Alpha* bol nastavený na hodnotu 2.



Obr. 6.22: Chameleon (Expected # of clusters=5, k-NN=10, Split=40, Alpha=2) pre dátovú sadu Compound.

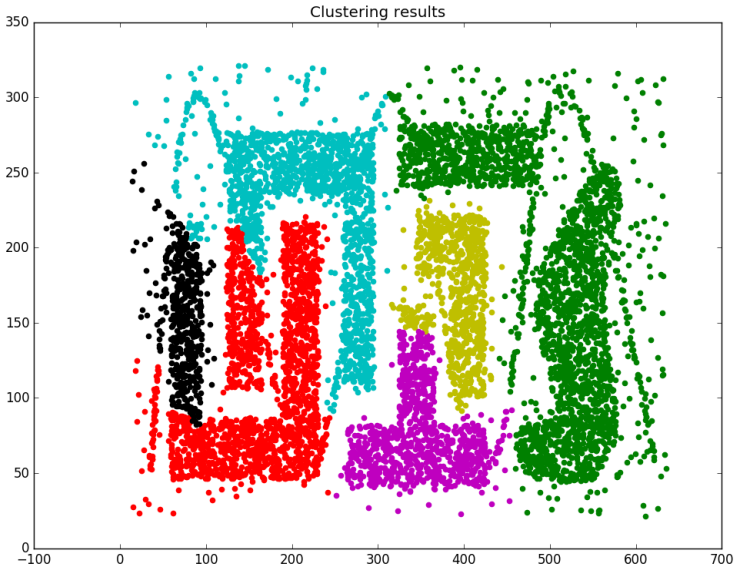
6.2.4 Dátová sada t4.8k

Táto dátová sada obsahuje 8000 prvkov a nájdenie vhodných parametrov zabralo množstvo času, z toho dôvodu, že zhlukovanie trvá dlhšie ako v predchádzajúcich dátových sadách. V tabuľke 6.9 je možné vidieť 11 experimentov, ktoré boli vykonané. Vzhľadom na čas, ktorý je potrebný na zhlukovanie tejto dátovej sady, boli vykonané experimenty len pre hodnotu 6 parametra *Expected # of clusters*.

experiment	1	2	3	4	5	6
Expected # of clusters	6	6	6	6	6	6
k-NN	10	10	10	15	5	6
Split	20	15	10	15	10	10
Alpha	2	2	2	2	2	2
clusters	6	6	6	6	6	6
silhouette coefficient	0.19	0.29	0.16	0.23	0.33	0.26
time [min]	101	70.8	41.9	74.5	39.7	41.5
experiment	7	8	9	10	11	
Expected # of clusters	6	6	6	6	6	
k-NN	6	6	6	8	4	
Split	11	12	13	13	12	
Alpha	2	2	0.5	0.5	2	
clusters	6	6	6	6	6	
silhouette coefficient	0.34	0.34	0.23	0.30	0.29	
time [min]	51.4	58.3	58.7	62.0	53.0	

Tabuľka 6.9: Prehľad výsledkov jednotlivých experimentov algoritmu Chameleon pre dátovú sadu t4.8k.

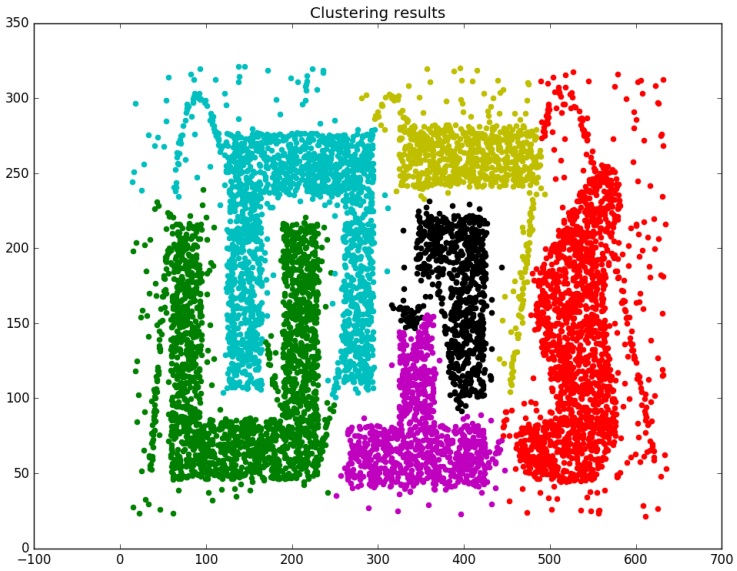
Pri experimentoch číslo 1-9 bolo vytvorených 6 zhlukov, ktorých rozdelenie nebolo celkom správne. Došlo ku spojeniu zhlukov, ktoré nemali byť spojené, čo má za následok vlnovka, ktorá spája všetky hlavné zhluky (obrázok 6.23).



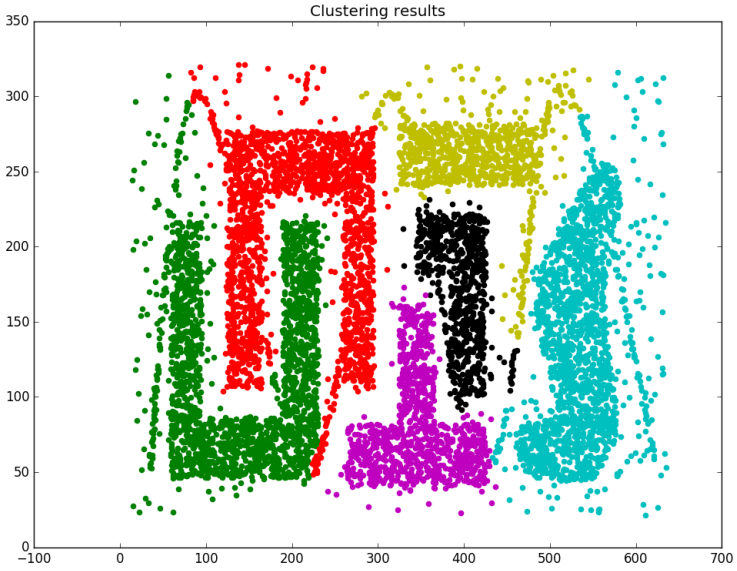
Obr. 6.23: Chameleon (Expected # of clusters=6, k-NN=6, Split=13, Alpha=0.5) pre dátovú sadu t4.8k.

Medzi najlepšie riešenia patria výsledky experimentu číslo 10 (obrázok 6.24) a 11 (obrázok 6.25). V experimente číslo 10 sa jeden zhluk (čierna farba, obrázok 6.24) nevytvoril úplne správne, avšak keď bol porovnaný tento výsledok s ostatnými, bolo potvrdené, že patrí

medzi tie najlepšie získané. Experiment číslo 11 vytvoril celkovo 6 zhlukov, ktoré boli už vytvorené správne. V tomto experimente sa vytvoril 4-NN graf, ktorý sa rozdelil na 12 častí a následne sa z týchto častí vytvorilo 6 zhlukov. Pri tomto experimente bol kladený dôraz na relatívnu blízkosť. Tento výsledok bol považovaný za najlepší získaný.



Obr. 6.24: Chameleon (Expected # of clusters=6, k-NN=8, Split=13, Alpha=0.5) pre dátovú sadu t4.8k.



Obr. 6.25: Chameleon (Expected # of clusters=6, k-NN=4, Split=12, Alpha=2) pre dátovú sadu t4.8k.

6.2.5 Dátová sada MNIST

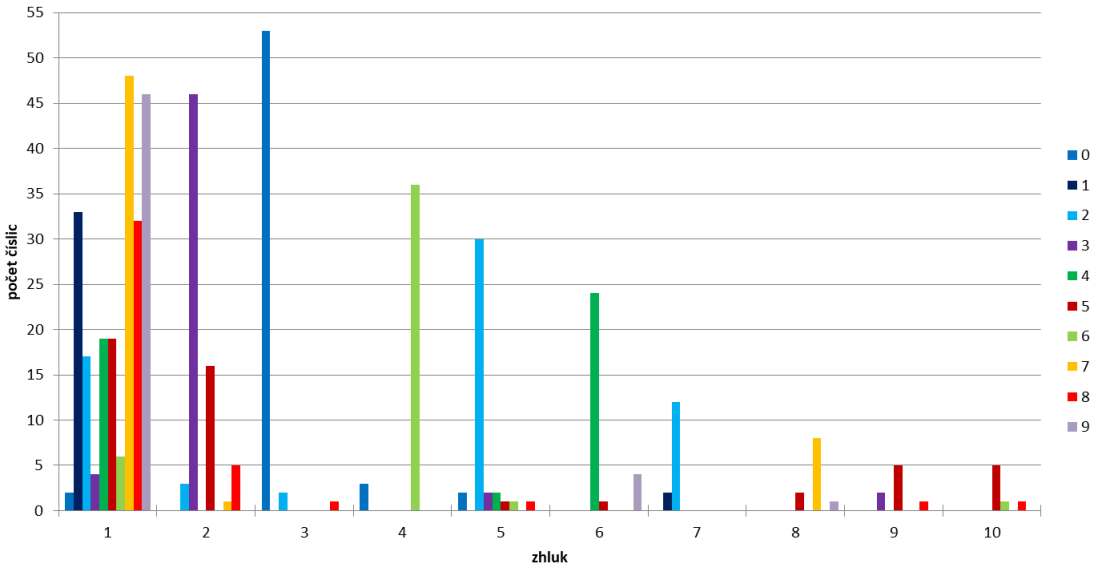
Pre túto dátovú sadu bolo vykonaných 7 experimentov, ktoré sú zobrazené v tabuľke 6.10. Takisto, ako pri algoritme DBSCAN, aj v tomto prípade boli experimenty vykonané len pre 500 a 1000 objektov z dátovej sady MNIST, ktoré boli vybrané náhodne. Algoritmus Chameleon bol spustený aj pre 5000 objektov a pre celú dátovú sadu, ale výsledky neboli analyzované. Výsledky vo forme súboru CSV, v ktorom sú jednotlivým objektom priradené zhluky, sa nachádza v priečinku *results*.

experiment	1	2	3	4	5	6	7
number of objects	500	500	500	1000	1000	1000	1000
Expected # of clusters	10	10	10	10	10	10	15
k-NN	40	30	50	50	50	30	50
Split	40	30	50	50	50	30	50
Alpha	2	2	2	2	0.5	2	2
clusters	10	10	10	10	10	10	15
silhouette coefficient	0.02	0.03	0.05	0.03	0.05	0.06	0.04
time [min]	10.4	11.0	11.7	31.1	32.4	17.2	35.0

Tabuľka 6.10: Prehľad výsledkov jednotlivých experimentov algoritmu Chameleon pre dátovú sadu MNIST.

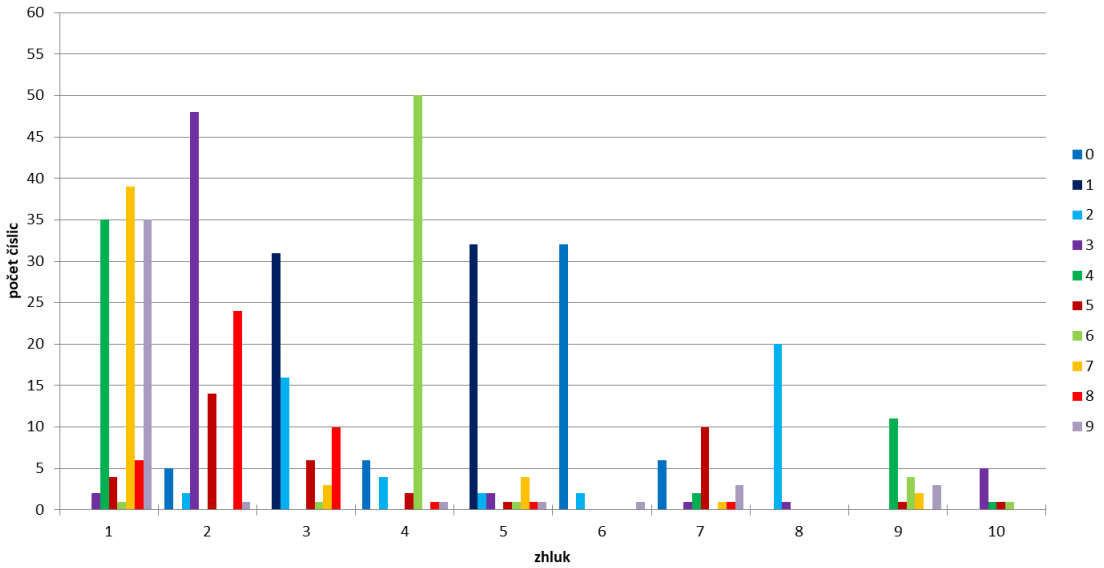
V každom experimente, okrem experimentu čísla 7, bolo podmienkou vytvoriť 10 zhlukov. Pri experimentoch s 500 a 1000 objektmi bolo zistené, že algoritmus Chameleon rozdeľuje číslice do zhlukov celkom dobre. Niektoré zhluky obsahujú viacero číslic, čo môže byť spôsobené štýlom písma a podobnosťou číslic (napríklad číslice "1", "7").

V experimente číslo 1 sa v prvom zhluku zoskupilo viacero číslic, ako je možné vidieť na obrázku 6.26. Ostatné zhluky pozostávali zvyčajne z jednej hlavnej číslice, ktorá dominovala v zhluku.



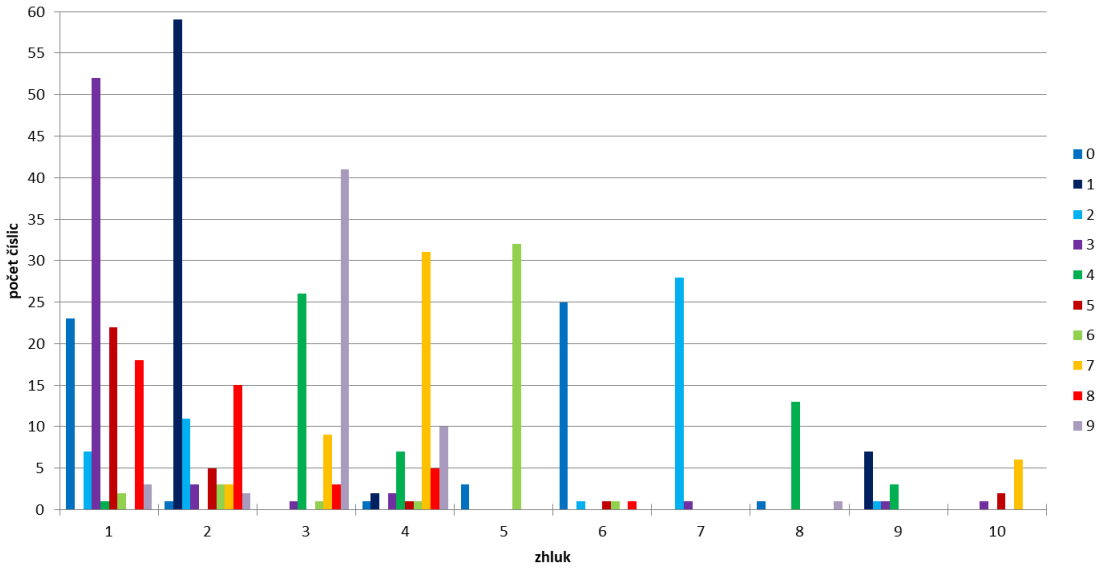
Obr. 6.26: Chameleon (Expected # of clusters=10, k-NN=40, Split=40, Alpha=2) pre dátovú sadu MNIST (500 objektov).

V experimente číslo 3 sa vyznačoval zvýšením hodnoty parametra *k-NN* a *Split*. Tým bolo dosiahnuté to, že prvý zhluk obsahoval menej dominantných číslic, ako je možné vidieť na obrázku 6.27.



Obr. 6.27: Chameleon (Expected # of clusters=10, k-NN=50, Split=50, Alpha=2) pre dátovú sadu MNIST (500 objektov).

Experiment číslo 2 sa vyznačoval znížením hodnoty parametra *k-NN* a *Split*. Týmto znížením hodnôt bolo dosiahnuté zníženie dominantných číslic v zhlukoch, kde sa ich vyskytovalo viac. Na obrázku 6.28 je vidieť, že skoro v každom zhluku je dominantná jedna číslica. V niektorých zhlukoch sa vyskytujú aj iné číslice, čo môže byť spôsobené podobnosťou číslic. V tomto experimente bol vytvorený 30-NN graf, ktorý bol rozdelený na 30 častí a následne spojený do 10 zhlukov. Toto riešenie spomedzi experimentov s 500 objektmi dátovej sady MNIST bolo označené za najlepšie získané.

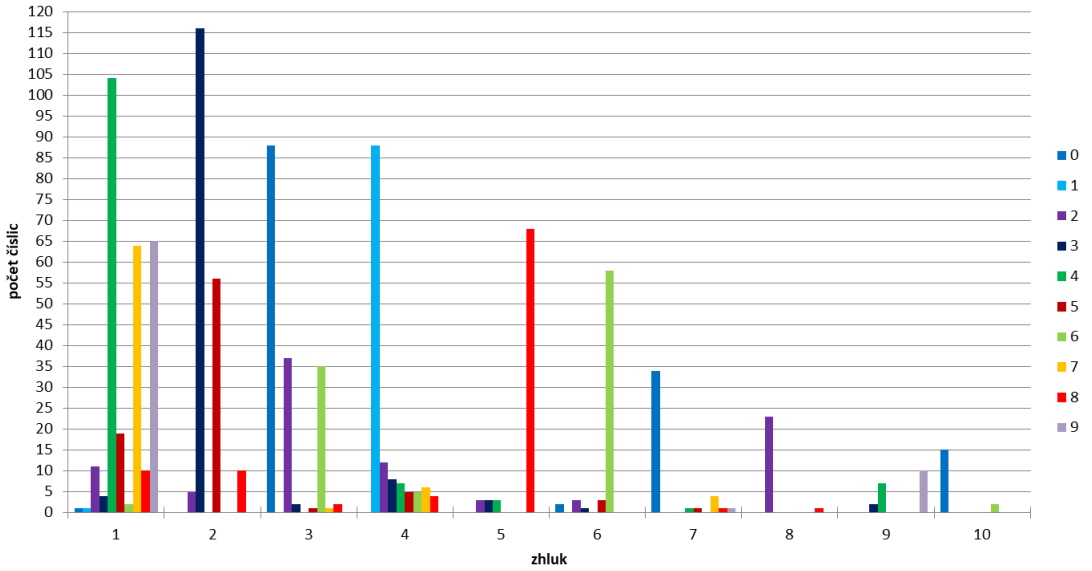


Obr. 6.28: Chameleon (Expected # of clusters=10, k-NN=30, Split=30, Alpha=2) pre dátovú sadu MNIST (500 objektov).

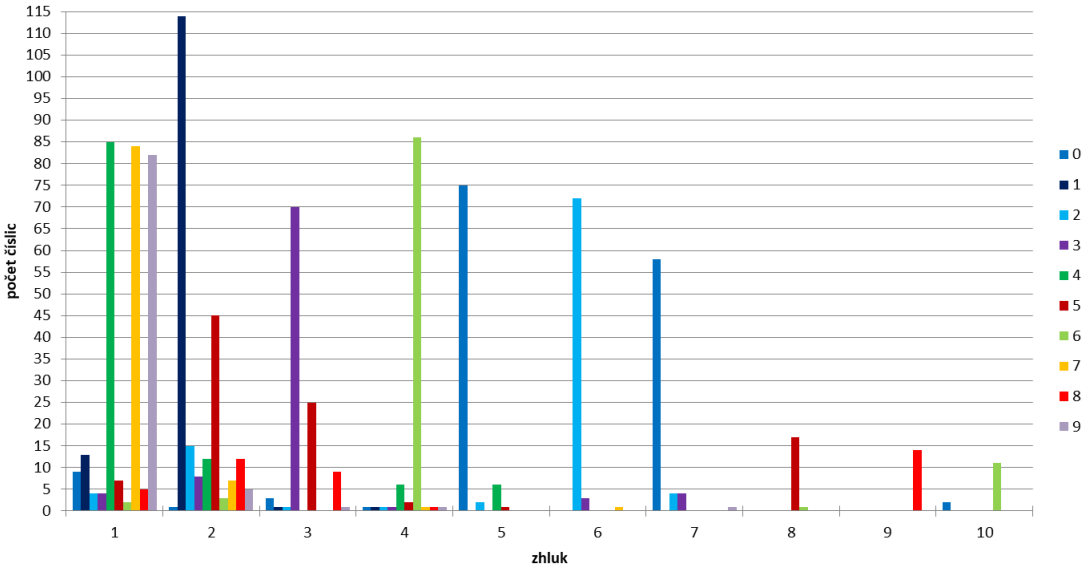
Experimenty číslo 4-7 boli vykonané na 1000 objektoch dátovej sady MNIST. V experimente číslo 7 bolo použité rozdelenie do 15 zhlukov. V tomto prípade došlo k vytvoreniu dvojíc zhlukov, v ktorých bola dominantná tá istá číslica. Túto situáciu bolo treba eliminovať v experimentoch číslo 4-6.

Najlepšie rozdelenie do zhlukov bolo dosiahnuté v experimentoch číslo 4 (obrázok 6.29) a 5 (obrázok 6.30), kde bolo vytvorených 10 zhlukov. V experimente číslo 4 bola v každom zhluku jedna dominantná číslica, zatiaľ čo v experimente číslo 5 bolo v zhluku 1 dominantných viac číslic. Je ťažké rozhodnúť, ktoré riešenie je lepšie, pretože experiment číslo 4 má zhluky, v ktorých je prevažne jedna číslica dominantná, ale zhluky obsahujú menší počet aj iných číslic. V experimente číslo 5 sú zhluky, kde dominantných číslic je viac, ale obsahuje zhluky, v ktorých sa vyskytujú prevažne dominantné číslice. Experiment číslo 4 využíval 50-NN graf a kládol dôraz na relatívnu blízkosť, zatiaľ čo v experimente číslo 5 bol kladený dôraz na relatívnu interkonektivitu. Výsledok experimentu číslo 6 bol podobný výsledku experimentu číslo 5.

Experimenty pre 5000 objektov dátovej sady a pre celú dátovú sadu boli vykonané, ale neboli analyzované. Výsledky z týchto experimentov sú uložené v súbore CSV v priečinku *results*.



Obr. 6.29: Chameleon (Expected # of clusters=10, k-NN=50, Split=50, Alpha=2) pre dátovú sadu MNIST (1000 objektov).



Obr. 6.30: Chameleon (Expected # of clusters=10, k-NN=50, Split=50, Alpha=0.5) pre dátovú sadu MNIST (1000 objektov).

6.3 Algoritmus SOM

Pre tento algoritmus sa nastavovali tri parametre *Training iterations*, *Nodes* a *Learing rate*, ktoré sú popísané v sekcii 5.2. Hľadanie vhodných parametrov pre tento algoritmus zabralo najviac času kvôli natrénovaniu siete. Tento algoritmus vytváral vždy sieť s veľkosťou n x n (n = hodnota parametra *Nodes*). Zhlukovanie prebiehalo na základe zoznamu víťazných

uzlov, ktoré boli aplikované na algoritmus DBSCAN implementovaný v knižnici *sklearn*, okrem dátovej sady MNIST. Hodnoty parametrov pre DBSCAN boli prevzaté z experimentov, v ktorých sme dosiahli najlepšie výsledky, okrem dátovej sady t4.8k, kde sme museli meniť aj tieto hodnoty parametrov. Zhlukovanie dátovej sady MNIST prebiehalo na základe zoznamu víťazných uzlov a Quality Threshold algoritmu [13], ktorého cieľom je nájsť veľké zhluky s dobrou kvalitou.

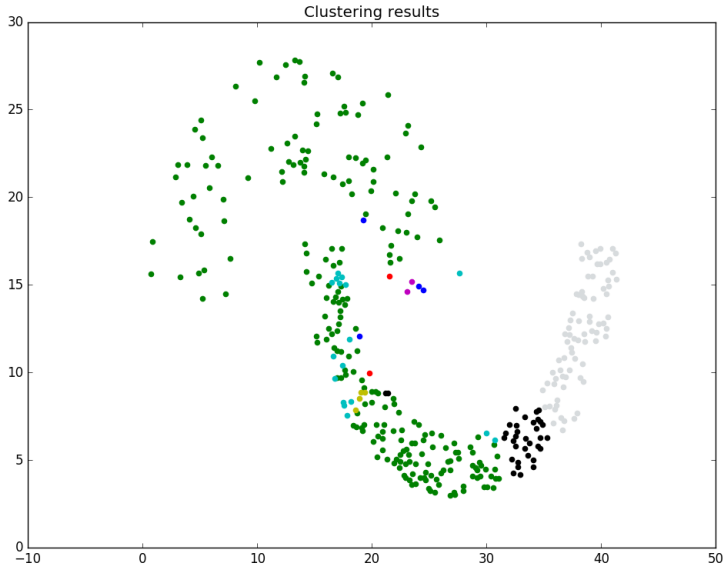
6.3.1 Dátová sada Jain

Pre túto dátovú sadu bolo vykonaných 5 experimentov, ktoré sú zapísané v tabuľke 6.11.

experiment	1	2	3	4	5
Training iterations	2000	2000	3000	3000	4000
Nodes	10	20	20	30	30
Learning rate	0.01	0.01	0.1	0.1	0.1
clusters	3	8	5	2	3
silhouette coefficient	0.06	-0.16	-0.15	0.40	0.06
time [s]	4.80	21.0	34.8	78.0	105

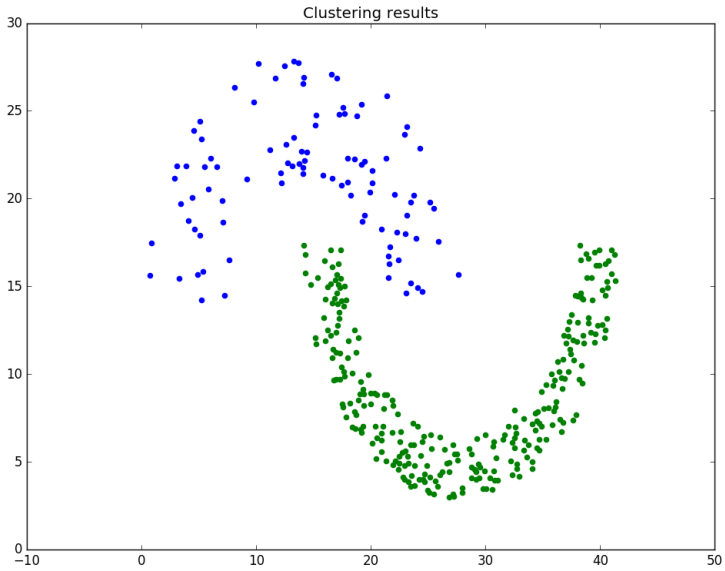
Tabuľka 6.11: Prehľad výsledkov jednotlivých experimentov algoritmu SOM pre dátovú sadu Jain.

V experimente číslo 1 a 2 bola vrchná vlnovka a polovica dolnej vlnovky spojená do jedného zhluku (obrázok 6.31), čo nepredstavovalo správne riešenie, preto sme skúsili zvýšiť počet trénovacích iterácií na 3000 a hodnotu parametru *Learning rate* zmeniť na 0.1. V experimente číslo 3 sme zistili, že pri sieti s rozmermi 20 x 20 uzlov správne riešenie nedostaneme, pretože sme dostali podobný výsledok ako v predchádzajúcich riešeniach.



Obr. 6.31: SOM (Training iterations=2000, Nodes=20, Learning rate=0.01) pre dátovú sadu Jain.

V experimentoch číslo 4 a 5 sme dostali najlepšie riešenia. V experimente číslo 5 boli vytvorené 3 zhluky kde jeden zhluk obsahoval iba jeden objekt. V experimente číslo 4 sme dostali 2 zhluky (obrázok 6.32), ktoré odpovedali spodnej a vrchnej vlnovke. V tomto experimente sme využili natrénovanú sieť s rozmermi 30 x 30 uzlov, kde stupeň učenia bol 0.1. Tento výsledok považujeme za najlepší, ktorý sme získali.



Obr. 6.32: SOM (Training iterations=3000, Nodes=30, Learning rate=0.1) pre dátovú sadu Jain.

6.3.2 Dátová sada Flame

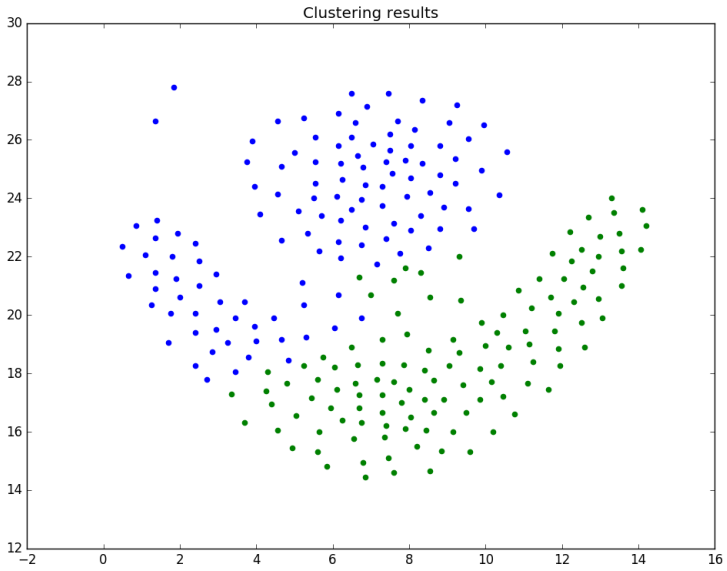
Pri tejto dátovej sade bolo vykonaných 9 experimentov, ktoré sú uvedené v tabuľke 6.12.

experiment	1	2	3	4	5	6	7	8	9
Training iterations	2500	2500	2500	2500	2500	2500	3000	4000	4000
Nodes	10	15	5	3	2	2	3	2	3
Learning rate	0.01	0.01	0.01	0.1	0.1	0.01	0.01	0.01	0.01
clusters	12	15	6	3	2	2	3	2	3
silhouette coefficient	0.00	0.10	0.04	0.21	0.37	0.36	0.03	0.36	0.08
time [s]	7.78	16.0	2.01	0.76	0.44	0.42	0.93	0.58	0.95

Tabuľka 6.12: Prehľad výsledkov jednotlivých experimentov algoritmu SOM pre dátovú sadu Flame.

V experimente číslo 1 a 2 sa nám vytvorilo veľké množstvo zhlukov, preto sme v ďalších experimentoch znížili hodnotu parametra *Nodes*. Počet zhlukov odpovedal číslu, ktoré sme chceli dostať, ale rozdelenie zhlukov nebolo správne (obrázok 6.33). Snažili sme sa pozmeniť hodnotu parametra *Learning rate*, ale výsledok bol taktiež nesprávny. Rovnako sme sa snažili zvýšiť počet trénovacích iterácií, ale výsledok bol opäť nesprávny. Pre túto dátovú sadu sa nám nepodarilo správne nastaviť parametre algoritmu SOM tak, aby sme dostali

požadovaný výsledok. Dôvodom môže byť aj to, že objekty sú umiestnené tak, že na prvý pohľad vytvárajú jeden zhluk. Hranica medzi vrchnou a spodnou časťou nie je jasne daná.



Obr. 6.33: SOM (Training iterations=2500, Nodes=2, Learning rate=0.1) pre dátovú sadu Flame.

6.3.3 Dátová sada Compound

Pri tejto dátovej sade sme strávili veľké množstvo času, kým sme našli parametre, s ktorými bol výsledok zhlukovania dostatočne dobrý. Celkovo bolo vykonaných 15 experimentov, ktoré sú zapísané v tabuľke 6.13.

experiment	1	2	3	4	5	6	7	8
Training iterations	4000	4000	5000	6000	5000	8000	10000	20000
Nodes	20	25	20	20	20	20	19	18
Learning rate	0.01	0.01	0.01	0.01	0.1	0.01	0.01	0.01
clusters	13	14	13	13	6	10	5	7
silhouette coefficient	0.19	0.15	0.07	0.21	0.19	0.23	0.22	0.15
time [min]	0.75	1.30	1.15	1.21	1.10	1.88	2.02	3.23
experiment	9	10	11	12	13	14	15	
Training iterations	20000	30000	30000	40000	40000	50000	60000	
Nodes	19	19	20	19	20	20	20	
Learning rate	0.01	0.01	0.01	0.01	0.01	0.01	0.01	
clusters	10	7	10	5	9	7	9	
silhouette coefficient	0.19	0.10	0.18	0.19	0.14	0.13	0.14	
time [min]	3.41	6.52	7.46	8.61	9.93	10.7	11.0	

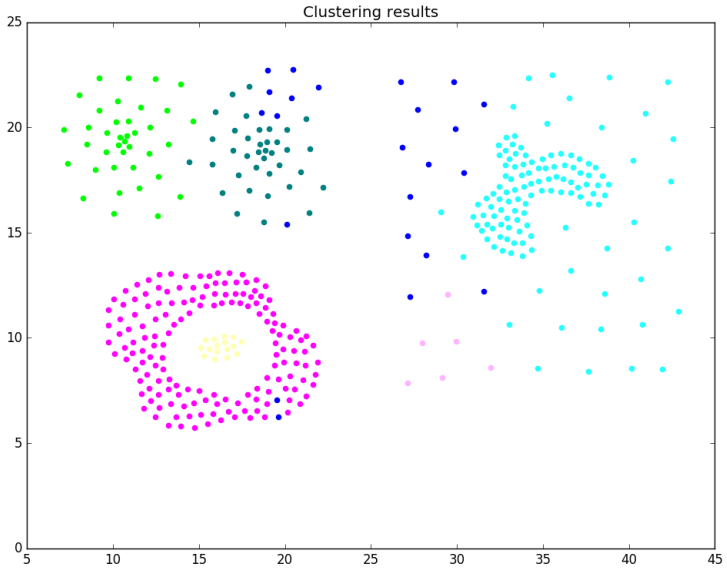
Tabuľka 6.13: Prehľad výsledkov jednotlivých experimentov algoritmu SOM pre dátovú sadu Compound.

V experimente číslo 1 sa vytvorilo 13 zhlukov (obrázok 6.34), kde bolo vidieť ako približne zhluky budú vypadať. V ďalších experimentoch sme skúšali meniť hodnotu parametru *Nodes* a *Learning rate*, ale zistili sme, že asi najlepšie bude využiť sieť s 20 x 20 uzlami a stupňom učenia 0.01.

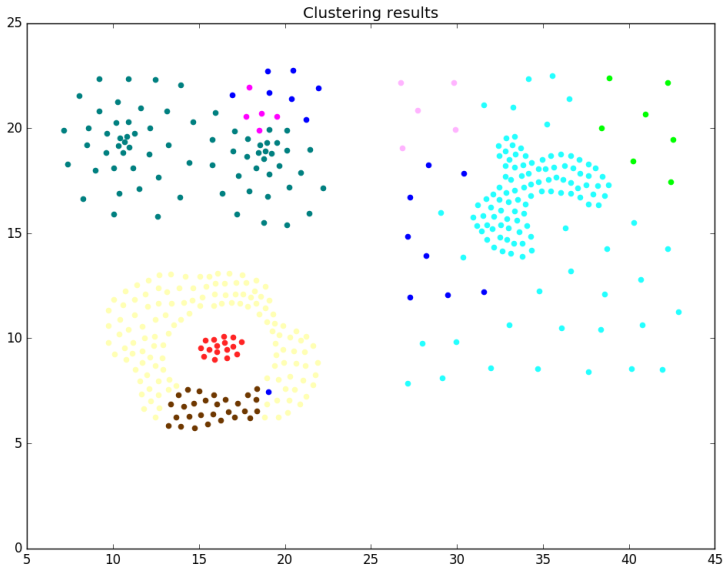
Obr. 6.34: SOM (Training iterations=4000, Nodes=20, Learning rate=0.01) pre dátovú sadu Compound.

V ďalších experimentoch sa zvyšoval počet trénovacích iterácií a výsledok zhlukovania sa postupne zlepšoval. Pri experimente číslo 14 sme dosiahli najlepší výsledok zhlukovania (obrázok 6.35), kde počet trénovacích iterácií bol 50000. Celkovo bolo vytvorených 7 zhlukov, kde do tmavozeleného zhluku zasahoval kúsok modrého zhluku a taktiež v pravej strane dátovej sady bolo vytvorených viac zhlukov, čo môže byť spôsobené tým, že sú objekty ďalej od seba.

Pri poslednom experimente, kde bol počet trénovacích iterácií 60000, sa vrchné dva zhluky spojili do jedného (obrázok 6.36), čo sme považovali za nesprávne, aj keď medzi nimi nie je presná hranica, ktorá by ich oddeľovala. Taktiež spodný okrúhly zhluk sa rozdelil na viacero častí, čo sme považovali za nesprávne.



Obr. 6.35: SOM (Training iterations=50000, Nodes=20, Learning rate=0.01) pre dátovú sadu Compound.



Obr. 6.36: SOM (Training iterations=60000, Nodes=20, Learning rate=0.01) pre dátovú sadu Compound.

6.3.4 Dátová sada t4.8k

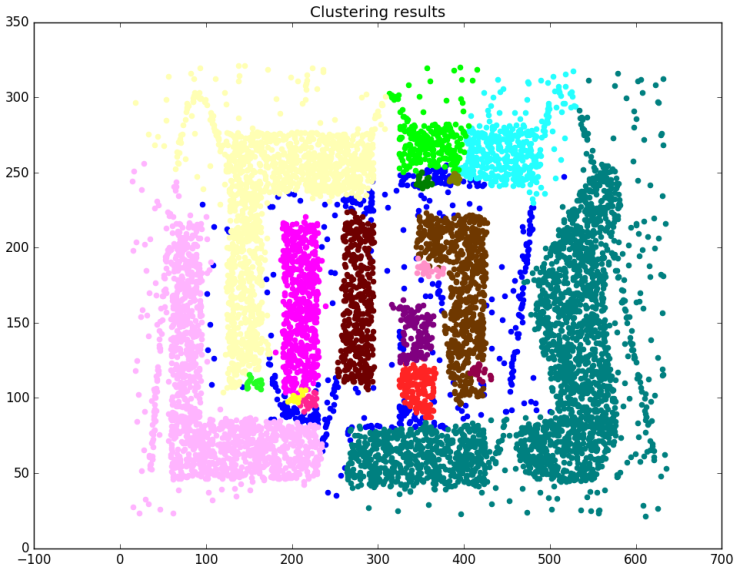
Pre túto dátovú sadu bolo vykonaných najviac experimentov, celkovo 44, pretože sme sa snažili dosiahnuť čo najlepší výsledok. V tabuľke 6.14 je vypísaných 15 experimentov, ktoré sa blížili k správnemu rozdeleniu objektov do zhlukov. Pri tejto dátovej sade sme museli meniť aj hodnoty parametrov pre zhlukovaciu funkciu DBSCAN, aby sme dosiahli čo najlepšieho výsledku.

experiment	1	2	3	4	5
Eps/MinPts	2/8	2/12	2/12	2/12	2/12
Training iterations	10000	14000	15000	17000	15000
Nodes	90	84	84	84	85
Learning rate	0.01	0.01	0.01	0.01	0.01
clusters	40	25	24	18	25
silhouette coefficient	-0.05	0.04	0.02	0.02	0.03
time [min]	58.5	68.2	60.7	67.2	63.6
experiment	6	7	8	9	10
Eps/MinPts	2/12	2/12	2/16	2/16	2.5/12
Training iterations	20000	22000	20000	15000	15000
Nodes	85	83	84	85	84
Learning rate	0.01	0.01	0.01	0.01	0.01
clusters	22	14	19	26	20
silhouette coefficient	-0.03	0.17	-0.01	0.03	0.08
time [min]	89.5	86.4	81.3	57.5	61.9
experiment	11	12	13	14	15
Eps/MinPts	2.5/12	2.5/12	2.5/12	2.5/16	2.5/16
Training iterations	20000	18000	22000	17000	17000
Nodes	83	84	83	83	85
Learning rate	0.01	0.01	0.01	0.01	0.01
clusters	11	14	12	21	18
silhouette coefficient	0.11	0.06	0.12	0.09	-0.08
time [min]	69	63.8	70.4	74.6	77.6

Tabuľka 6.14: Prehľad výsledkov jednotlivých experimentov algoritmu SOM pre dátovú sadu t4.8k.

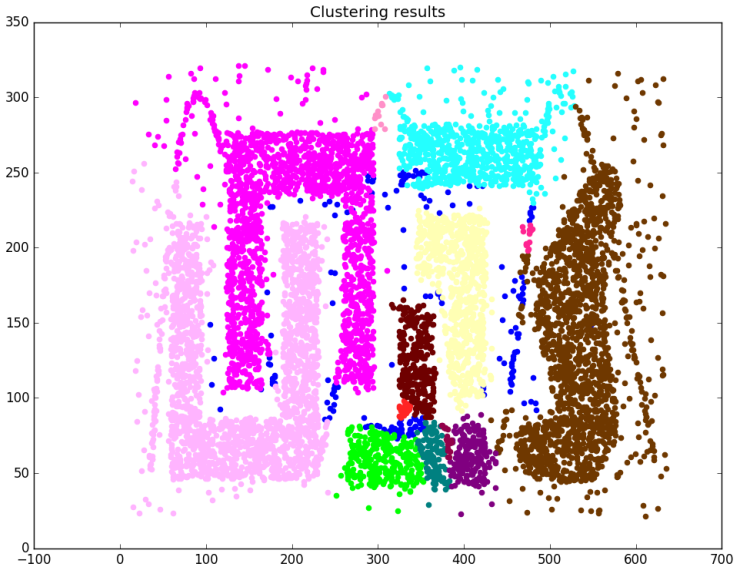
Pri tejto dátovej sade sme skúsili niekoľko kombinácií hodnôt parametrov pre funkciu DBSCAN, medzi ktoré patrili: 2/5, 2/8, 2/12, 2/16, 2.5/12, 2.5/16, 3/8 a 9/16. V tabuľke 6.14 sa nachádzajú len tie, pri ktorých sme mali najlepšie výsledky. Pri tejto dátovej sade sa nám nepodarilo dosiahnuť úplne správneho vytvorenia zhlukov, preto počet experimentov, v porovnaní s ostanými algoritmami a dátovými sadami je vyšší.

Pri niektorých experimentoch sa nám spojili niektoré skupiny objektov, ktoré mali tvoriť niekoľko zhlukov, napríklad experiment číslo 15 (obrázok 6.37). Na tomto obrázku je vidieť aj to, že došlo k vytvoreniu viacero zhlukov na mieste, kde mal byť len jeden väčší zhluk.



Obr. 6.37: SOM (Eps/MinPts=2.5/16, Training iterations=17000, Nodes=85, Learning rate=0.01) pre dátovú sadu t4.8k.

Najlepší výsledok sme dosiahli pri experimente číslo 12, kde sa vytvorilo celkovo 14 zhlukov (obrázok 6.38). Na obrázku je vidieť, že všetky hlavné zhluky, okrem jedného, boli vytvorené správne. V strednej dolnej časti skupinu objektov v tvare ⊥ tvorilo niekoľko zhlukov a algoritmus tieto zhluky nevedel spojiť dohromady. Taktiež došlo k vytvoreniu niekoľkých malých zhlukov, ktoré sa vyskytovali na vlnovke, čo by sa za chybu nepovažovalo, vzhľadom k obrazovej zložitosti dátovej sady.



Obr. 6.38: SOM (Eps/MinPts=2.5/12, Training iterations=18000, Nodes=84, Learning rate=0.01) pre dátovú sadu t4.8k.

6.3.5 Dátová sada MNIST

Pri tejto dátovej sade bol použitý zoznam víťazných uzlov, ktorý bol aplikovaný na Quality Threshold algoritmus a následne prebiehalo zhlukovanie. Pri Quality Threshold algoritme by sa mali hľadať zhluky, ktorých priemer nepresahuje hodnotu threshold parametra, ktorý môže nadobúdať hodnoty od 0 do 2 [13]. Avšak, zámerne bol prekročený tento interval za účelom pozorovania výsledkov pri threshold parametri, ktorého hodnota bola 5.

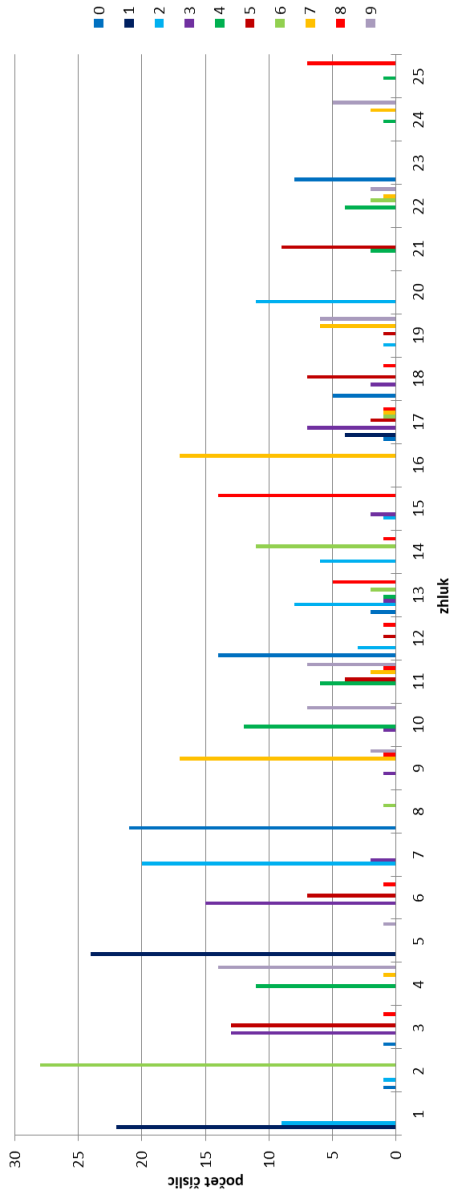
V tabuľke 6.15 sú zapísané experimenty, v ktorých hodnota threshold parametra bola 1 alebo 2. Experimenty boli vykonané pre 500 a 1000 objektov dátovej sady.

experiment	1	2	3	4	5	6	7
threshold parameter	1	2	2	2	2	2	2
number of objects	500	500	500	500	1000	1000	1000
Training iterations	5000	5000	5000	5000	5000	5000	5000
Nodes	30	30	20	50	20	30	10
Learning rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1
clusters	176	74	36	148	37	78	10
silhouette coefficient	-0.03	0.01	0.01	0.00	0.02	0.01	0.02
time [min]	191	192	84.7	520	90.4	202	22.5

Tabuľka 6.15: Prehľad výsledkov jednotlivých experimentov algoritmu SOM (threshold parameter=1 alebo 2) pre dátovú sadu MNIST.

V experimente číslo 1 a 4 sa vytvoril veľký počet zhlukov, kde väčšina zhlukov obsahovala len jeden až dva objekty. V experimente číslo 3 bolo vytvorených 36 zhlukov, kde prvých 25 zhlukov obsahovalo 8 až 31 objektov, ostatné zhluky obsahovali menšie množstvo objektov. Na obrázku 6.39 je možné vidieť, že takmer v každom zhluku je dominantná len jedna

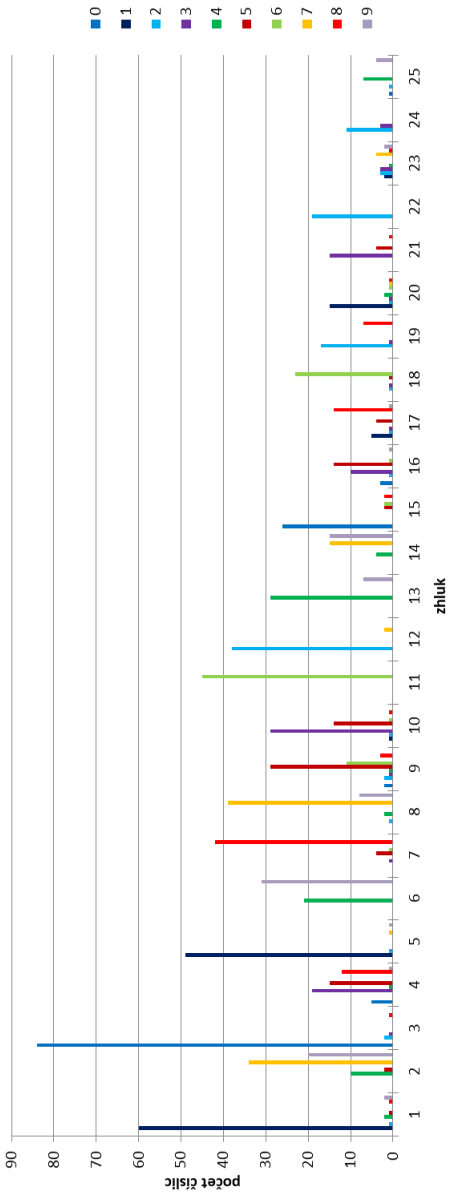
číslica. Avšak v niektorých zhlukoch sa dominantná číslica opakuje, čiže bolo dosiahnutých viacero zhlukov s rovnakou dominantnou číslicou.



Obr. 6.39: SOM (Threshold=2, Training iterations=5000, Nodes=20, Learning rate=0.1) pre dátovú sadu MNIST (500 objektov).

Pri 1000 objektoch dátovej sady boli vykonané tri experimenty. V experimente číslo 6 bolo vytvorených pomerne veľké množstvo zhlukov. V experimente číslo 7 bolo vytvorených 10 zhlukov, v ktorých číslice neboli úplne jednoznačne rozdelené. Zhluky obsahovali veľké množstvo rôznych číslic, čo nebolo správne. Najlepší výsledok bol dosiahnutý pri experimente číslo 5, kde sa vytvorilo 37 zhlukov. Väčšinou je v zhlukoch dominantná jedna číslica. V niektorých zhlukoch sa vyskytuje viac dominantných číslic hlavne sa jedná o číslice 5 a 3,

ktoré sú si podobné. Na obrázku 6.40 je možné vidieť rozloženie jednotlivých číslic v zhlukoch.

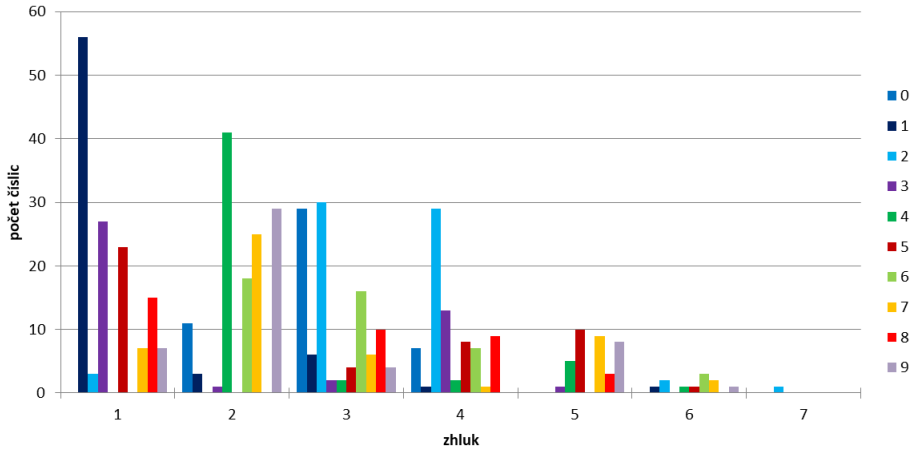


Obr. 6.40: SOM (Threshold=2, Training iterations=5000, Nodes=20, Learning rate=0.1) pre dátovú sadu MNIST (1000 objektov).

Experimenty s threshold parametrom, ktorého hodnota bola 5, sú zapísané v tabuľke 6.16. V experimente číslo 8 bolo použitých 500 objektov z dátovej sady a bolo vytvorených iba 7 zhlukov. Každý zhluk obsahoval niekoľko dominantných číslic (obrázok 6.41), preto rozdelenie nebolo veľmi správne.

experiment	8	9	10	11	12	13
number of objects	500	500	500	500	1000	1000
Training iterations	2500	5000	5000	10000	5000	10000
Nodes	30	30	50	30	30	30
Learning rate	0.1	0.1	0.1	0.1	0.1	0.1
clusters	7	17	39	348	18	531
silhouette coefficient	-0.02	0.01	-0.01	-0.01	-0.02	-0.03
time [min]	49.0	209	572	465	218	482

Tabuľka 6.16: Prehľad výsledkov jednotlivých experimentov algoritmu SOM (threshold parameter=5) pre dátovú sadu MNIST.

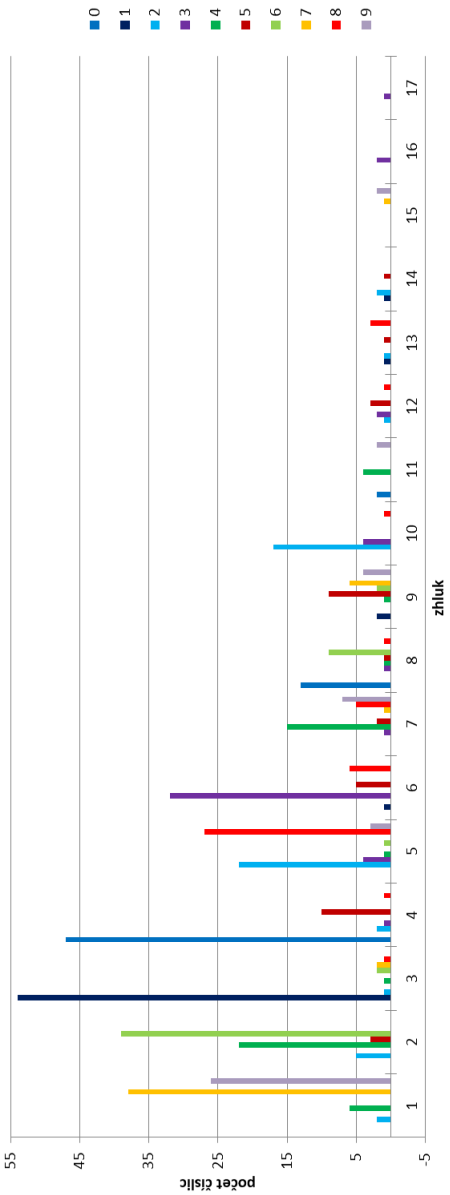


Obr. 6.41: SOM (Threshold=5, Training iterations=2500, Nodes=30, Learning rate=0.1) pre dátovú sadu MNIST (500 objektov).

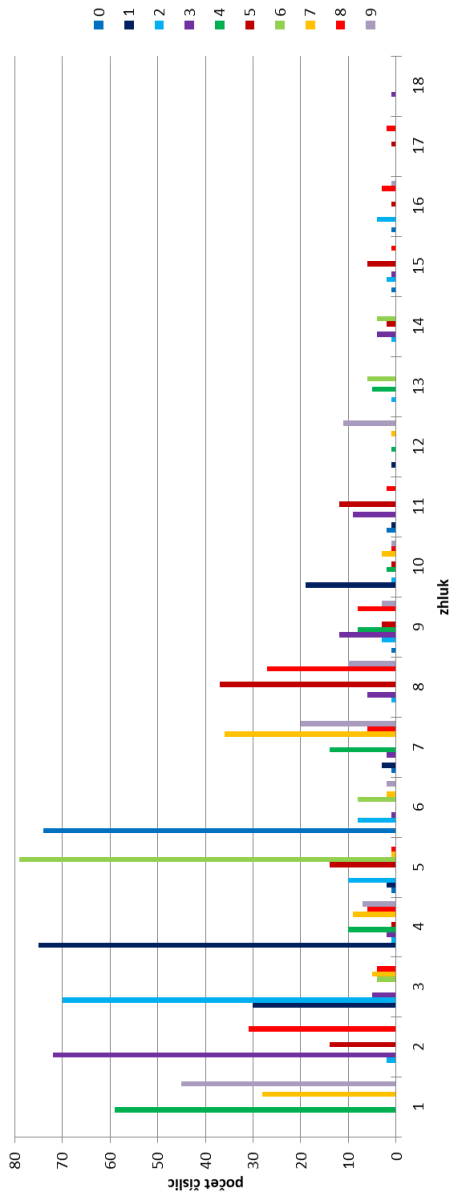
V ďalšom experimente (číslo 9) bol navýšený počet trénovacích iterácií, čo viedlo k lepšiemu výsledku. Celkovo bolo vytvorených 17 zhlukov (obrázok 6.42). Prvých 10 zhlukov obsahovalo jednu až dve dominantné číslice a ostatné zhluky obsahovali zanedbateľné množstvo objektov. V zhlukoch, kde sú dominantné dve číslice mohlo nastať to, že sa číslice na seba podobali, niekde záležalo aj na hrúbke písma.

V experimente číslo 10 bolo vytvorených až 39 zhlukov, v ktorých sa jednotlivé dominantné číslice opakovali. Tento výsledok nebol považovaný za celkom správny. Taktiež výsledky experimentu číslo 11 a 13 neboli považované za správne, kde došlo vytvoreniu veľkého množstva zhlukov, ktoré obsahovali malé množstvo objektov. V experimente číslo 11 obsahovalo 241 zhlukov len 1 objekt, v ostatných zhlukoch väčšinou prevažovala len jedna číslica. Podobná situácia bola pri experimente číslo 6.

Pri experimente číslo 12 bolo použitých 1000 objektov dátovej sady. Celkovo bolo vytvorených 18 zhlukov, z ktorých prvých 12 zhlukov bolo kľúčových. Tieto zhluky obsahovali jednu až dve dominantné číslice (obrázok 6.43). Ostatné zhluky obsahovali malý počet objektov, avšak aj pri týchto zhlukoch bola väčšinou dominantná jedna číslica. Pri tomto experimente bola použitá natrénovaná sieť s 30 x 30 uzlami. Veľkosť tejto siete bola použitá, pretože bolo vychádzané z experimentu číslo 9, kde sa táto veľkosť osvedčila.



Obr. 6.42: SOM (Threshold=5, Training iterations=5000, Nodes=30, Learning rate=0.1) pre dátovú sadu MNIST (500 objektov).



Obr. 6.43: SOM (Threshold=5, Training iterations=5000, Nodes=30, Learning rate=0.1) pre dátovú sadu MNIST (1000 objektov).

Algoritmus bol spustený aj na 5000 objektoch a celej dátovej sade. Výsledky sú zapísané v CSV súbore v priečinku *results*. Tieto výsledky neboli analyzované, ale pri 5000 objektoch sa vytvorilo 40 zhlukov, ktoré budú pravdepodobne obsahovať podobné rozdelenie ako pri 500, či 1000 objektoch. Pri celej dátovej sade bolo vytvorených 38 zhlukov a rozdelenie bude pravdepodobne podobné ako pri 5000 objektoch.

V experimentoch, kde bol použitý threshold parameter, ktorého hodnota bola 2 sa vytvárali kvalitné zhluky, ktoré obsahovali minimálny počet ostatných číslic na rozdiel od experimentov, kde bola hodnota parametra 5. Pri experimentoch, kde bola hodnota threshold parametra 5, bol vytvorený menší počet zhlukov, ktoré však obsahovali väčšie zastúpenie číslic ale neboli tak čisté ako pri experimentoch s hodnotou threshold parametra,

ktorá bola 2. Najvhodnejšie riešenia boli dosiahnuté pri experimente číslo 3 a 5. V týchto experimentoch boli rovnaké parametre, ale experimenty boli pre 500 a 1000 objektov. Hodnota threshold parametra bola 2, počet trénovacích iterácií bol 5000 a využívala sa mapa s veľkosťou 20 x 20 uzlov a koeficient učenia 0.1.

6.4 Zhrnutie

Pri hľadaní vhodných hodnôt parametrov bolo vykonaných celkovo 188 experimentov, ktoré sú zobrazené v tabuľkách pri jednotlivých algoritmoch a dátových sadách. Pre všetky algoritmy a dátové sady sa podarilo nájsť vhodné hodnoty parametrov, okrem dátových sád, do ktorých patrí dátová sada Flame a t4.8k pre algoritmus SOM a dátová sada Jain a MNIST pre algoritmus DBSCAN. V tabuľke 6.17 sú zobrazené hodnoty parametrov pre jednotlivé algoritmy a dátové sady, s ktorými boli dosiahnuté najlepšie rozdelenie objektov do zhlukov. Pre dátovú sadu Flame a t4.8k, ktoré boli zhlukované pomocou algoritmu SOM a dátové sady Jain a MNIST, ktoré boli zhlukované pomocou algoritmu DBSCAN, boli do tabuľky 6.17 zapísané hodnoty parametrov, s ktorými boli dosiahnuté najlepšie výsledky zo všetkých experimentov, avšak tieto výsledky neobsahujú správne rozdelenie objektov do zhlukov (hodnoty parametrov sú v tabuľke označené *).

Data set		**Jain**	**Flame**	**Compound**	**t4.8k**	**MNIST**
DBSCAN	**Epsilon**	2.5*	1	1.5	9	1600*
	MinPts	2*	5	3	16	2*
Chameleon	**Exp. # of clusters**	2	2	5	6	10
	k-NN	4	6	10	4	50
	Split parameter	5	15	40	12	50
	Alpha	2	2	2	2	2
SOM	**Training iterations**	3000	2500*	50000	18000*	5000
	Nodes	30	2*	20	84*	20
	Learning rate	0.1	0.1*	0.01	0.01*	0.1

Tabuľka 6.17: Prehľad hodnôt parametrov jednotlivých algoritmov pre dátové sady.

V experimentoch pri algoritme SOM boli získané obrázky vyzhlukovaných dátových sád ešte pred implementovaním funkcie zobrazenia pozície neurónov, preto sa v týchto obrázkoch pozície neurónov nevyskytujú. Avšak, aplikácia túto funkciu ponúka a výsledky sa už zobrazujú aj s pozíciami výsledných neurónov.

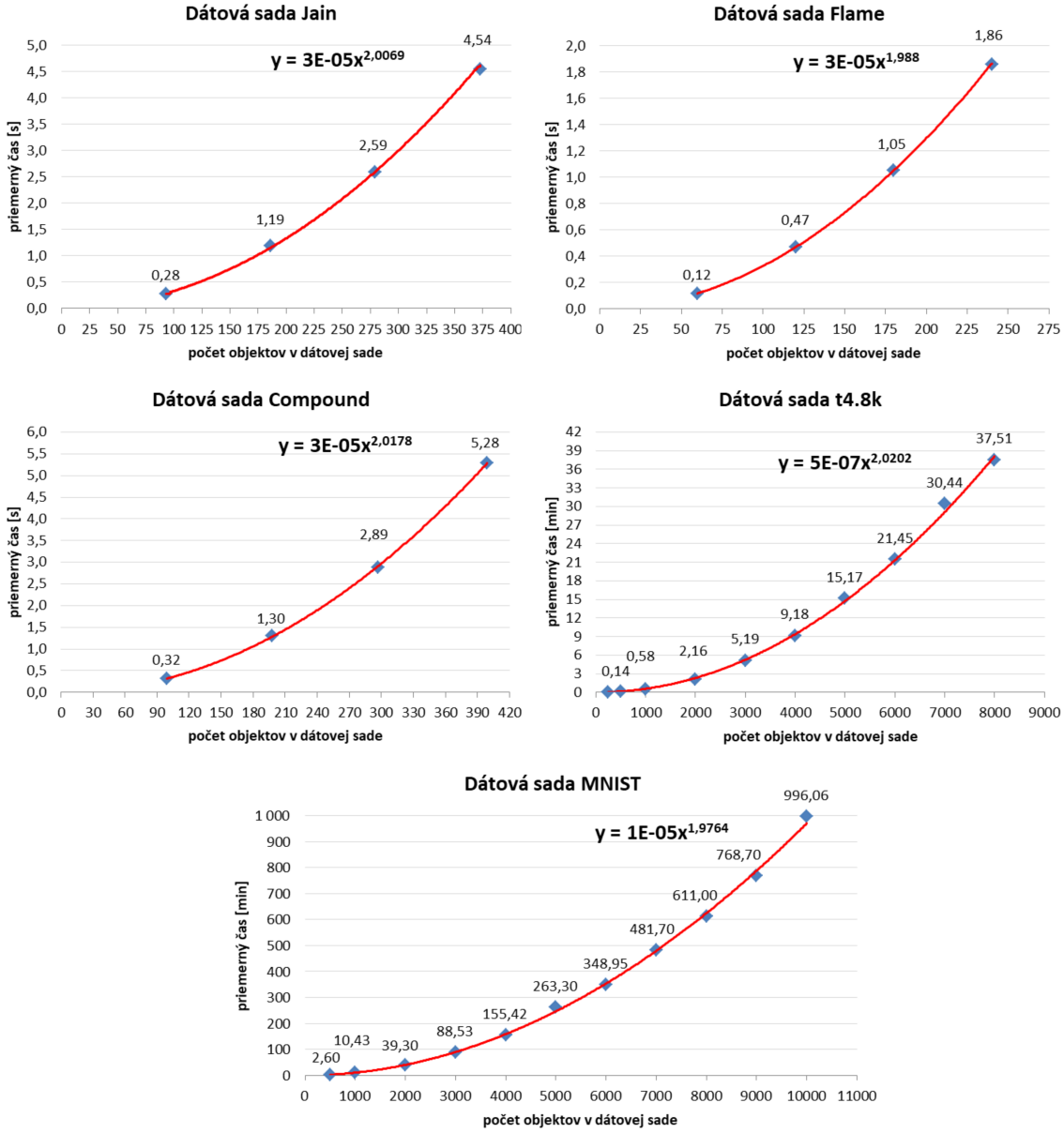
Kapitola 7

Experimenty pre overovanie časovej zložitosti

V tejto kapitole sú popísané experimenty, ktoré boli vykonané pre overenie časovej zložitosti jednotlivých algoritmov na každej dátovej sade. Každá dátová sada bola rozdelená na niekoľko častí a pre každú časť bol algoritmus spustený niekoľkokrát. Prvky do jednotlivých častí boli vyberané náhodne. Následne, získané časy pre každú časť boli spriemerované a vykreslené do grafu. Do grafu bola pridaná trendová spojnica. Čas, ktorý bol meraný, bol iba čisto čas zhlukovania bez načítania dátovej sady alebo vykreslenia výsledku. V experimentoch boli použité hodnoty parametrov jednotlivých algoritmov, pri ktorých boli získané najlepšie výsledky pre danú dátovú sadu.

7.1 Algoritmus DBSCAN

Podľa teoretických predpokladov má DBSCAN linearitmickú až kvadratickú časovú zložitosť [3.3.1], ktorá bola overovaná pre každú dátovú sadu. Dátová sada Jain, Flame a Compound bola rozdelená na 4 časti a algoritmus bol spustený pre každú časť 5-krát. Dátová sada t4.8k bola rozdelená na 10 častí a pre každú časť bol spustený algoritmus taktiež 5-krát. Počet experimentov s dátovou sadou MNIST bol menší, pretože zhlukovanie trvalo niekoľko hodín pri väčšom počte objektov. Táto dátová sada bola rozdelená na 11 častí. Na obrázku 7.1 je možné vidieť výsledky získané z experimentov v podobe grafov. Ako je možné vidieť z trendovej rovnice, v prípade dátovej sady Jain, Compound a t4.8k, sa mierne prekračuje kvadratická zložitosť. V ostatných prípadoch je zložitosť mierne menšia ako kvadratická. Vzhľadom na to, že boli použité dátové sady, ktoré obsahujú malé množstvo objektov, okrem dátovej sady t4.8k a MNIST, nie je dobre vidieť priebeh funkcie. Bolo by potrebné mať dátové sady, ktoré majú väčší počet objektov. Avšak, s ohľadom na odchýlky merania, podľa grafov je možné povedať, že na všetkých dátových sadách bola dosiahnutá kvadratická časová zložitosť a tým potvrdený teoretický predpoklad. Pri dátovej sade MNIST je možné si všimnúť, že pri 8000 objektoch DBSCAN potrebuje približne 10.2 hodiny na zhlukovanie. Táto sada obsahuje 784 dimenzií, preto DBSCAN pri väčšom počte dimenzií potrebuje viac času na zhlukovanie ako pri dvojdimenzionálnych dátových sadách (napríklad dátová sada t4.8k).

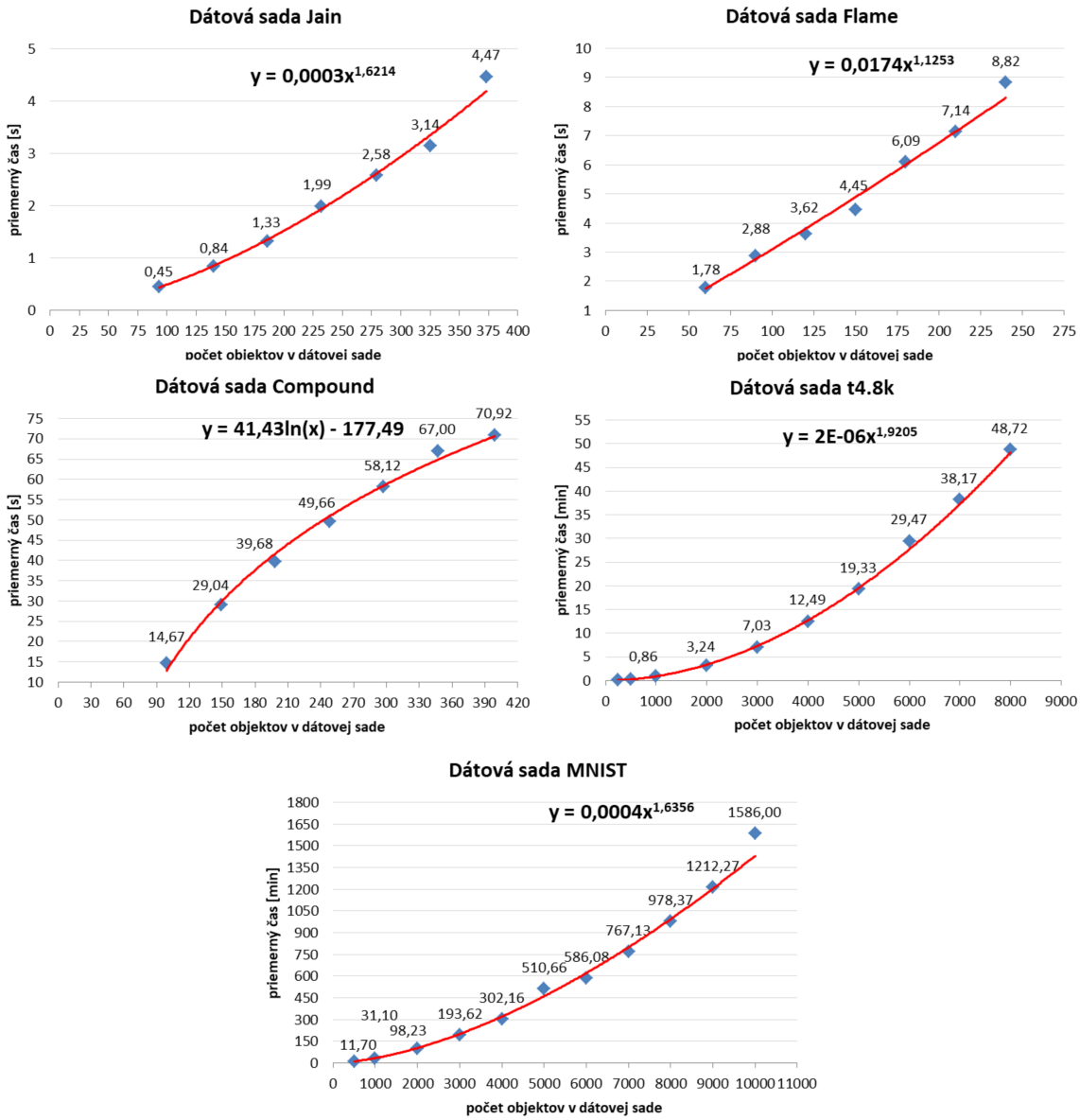


Obr. 7.1: Grafy závislosti času na počte prvkov jednotlivých dátových sád pre algoritmus DBSCAN.

7.2 Algoritmus Chameleon

Pri algoritme Chameleon bola overovaná časová zložitosť, ktorá nemala byť väčšia ako kvadratická [3.2.1]. Experimenty prebiehali rovnako ako pri algoritme DBSCAN, avšak rozdelenie dátových sád na časti bolo iné. Dátové sady Jain, Flame a Compound boli rozdelené na 7 častí, aby bol získaný presnejší časový priebeh. Algoritmus bol spustený pre každú časť 5-krát. Dátová sada t4.8k bola rozdelená na 10 častí a taktiež algoritmus bol spustený pre každú časť 5-krát. Dátová sada MNIST bola rozdelená na 11 častí a počet spustení bol menší. Na obrázku 7.2 je možné vidieť výsledky pre jednotlivé dátové sady. Podľa tren-

dovej rovnice bolo zistené, že pri žiadnej dátovej sade nebola presiahnutá kvadratická časová zložitosť. Taktiež aj pri tomto algoritme bolo treba brať do úvahy menšie odchýlky merania. Pri dátovej sade Jain je možné označiť kvadratickú časovú náročnosť a pri dátovej sade Flame takmer lineárnu. Pri dátovej sade Compound bola dosiahnutá linearitmická časová náročnosť. Pri dátovej sade t4.8k a MNIST bola dosiahnutá kvadratická časová náročnosť. Pri algoritme Chameleon je vidieť, že pri dátových sadách, ktoré majú malý počet objektov, nie je vždy jasné, do ktorej časovej triedy alegoritmus zaradiť. Pri dátovej sade, ktorá obsahuje väčší počet objektov (napríklad t4.8k) a je dvojdimenzionálna, je vidieť kvadratickú časovú náročnosť. Chameleon si vie poradiť aj s viacdimenzionálnymi dátovými sadami v kvadratickej časovej náročnosti, avšak napríklad pri dátovej sade MNIST, ktorá má 784 dimenzií a 10000 objektov trvalo zhlukovanie približne 26.5 hodiny. Pre viacdimenzionálne dáta bola dosiahnutá kvadratická časová zložitosť, čo potvrdzujú aj teoretické znalosti. Pre nízkodimenzionálne dáta bola dosiahnutá najhoršia kvadratická časová zložitosť, čo sa nezhoduje s teoretickými znalosťami, ktoré ukazujú, že pre nízkodimenzionálne dáta má algoritmus Chameleon linearitmickú časovú zložitosť. Táto časová zložitosť bola dosiahnutá iba v dátovej sade Compound.

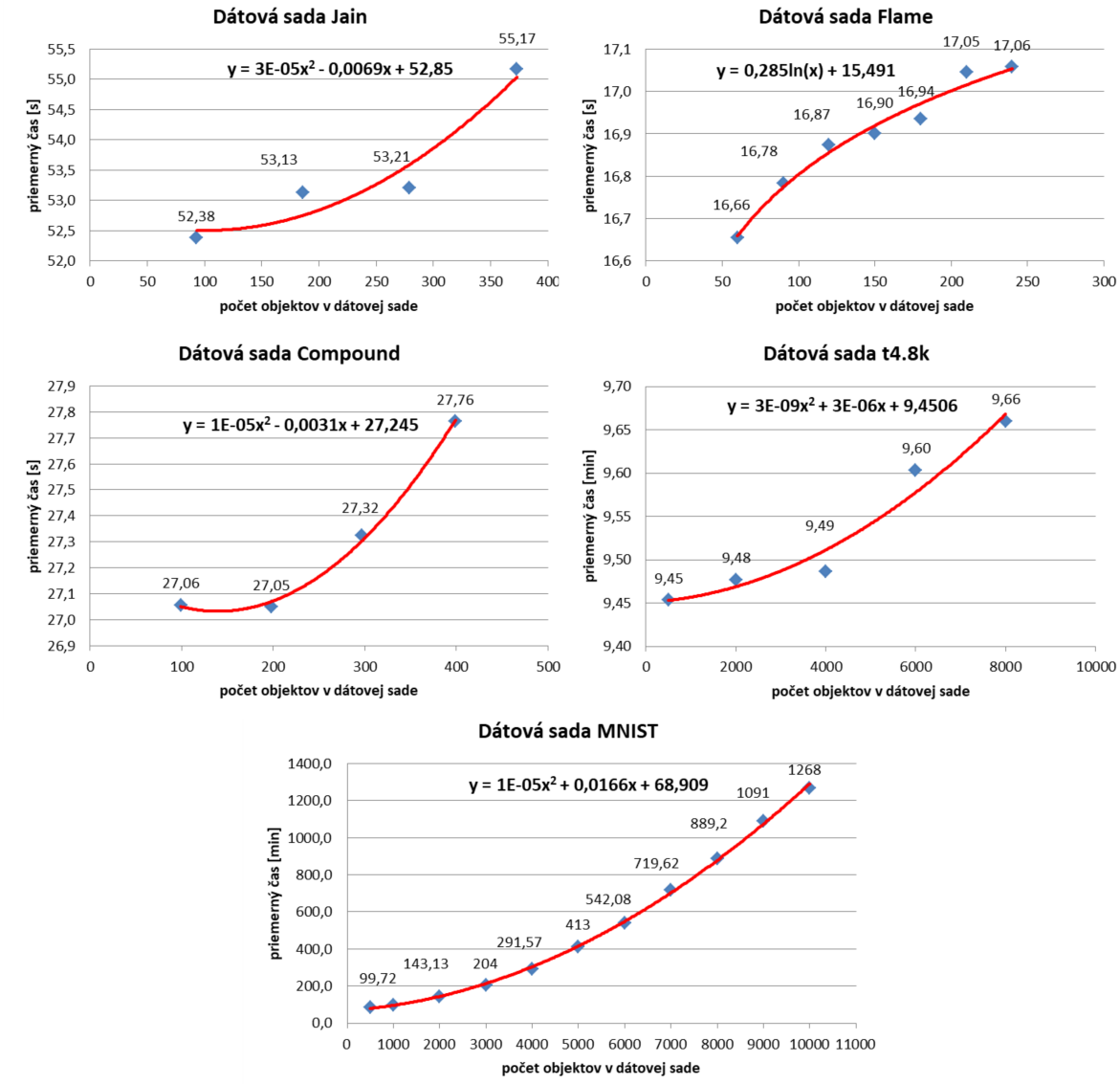


Obr. 7.2: Grafy závislosti času na počte prvkov jednotlivých dátových sád pre algoritmus Chameleon.

7.3 Algoritmus SOM

Pri tomto algoritme sa používa zoznam víťazných uzlov, ktoré sa aplikujú buď na algoritmus DBSCAN (Jain, Flame, Compound, t4.8k) alebo Quality Threshold algoritmus (MNIST). Podľa teoretických znalostí dosahuje algoritmus SOM časovú zložitosť $O(t_f(dn + pn^2))$ [3.4.2]. Avšak pri zhlukovaní sa používa ešte DBSCAN a Quality Threshold algoritmus. Časová zložitosť algoritmu DBSCAN, ktorý je implementovaný v knižnici *sklearn*, je $O(nd)$, kde d je priemerný počet susedov [25]. V prípade použitia Quality Threshold algoritmu je časová zložitosť $O(n^2)$ [18]. Na dátových sadách Jain a Compound a MNIST sa podarilo overiť časovú zložitosť, ktorá zhruba odpovedá kvadratickej. Na dátovej sade bola dosia-

hnutá logaritmická časová zložitosť. Je možné, že pre dátové sady, ktoré majú malý počet objektov ako táto dátová sada, sa kvadratická časová zložitosť neprejaví. Je možné, že práve pre túto dátovú sadu dominuje iná zložka a nie je časová zložitosť závislá čisto len na počte prvkov a preto pre túto dátovú sadu sa nedarí overiť časová zložitosť. Výsledky týchto experimentov sú znázornené na obrázku 7.3.

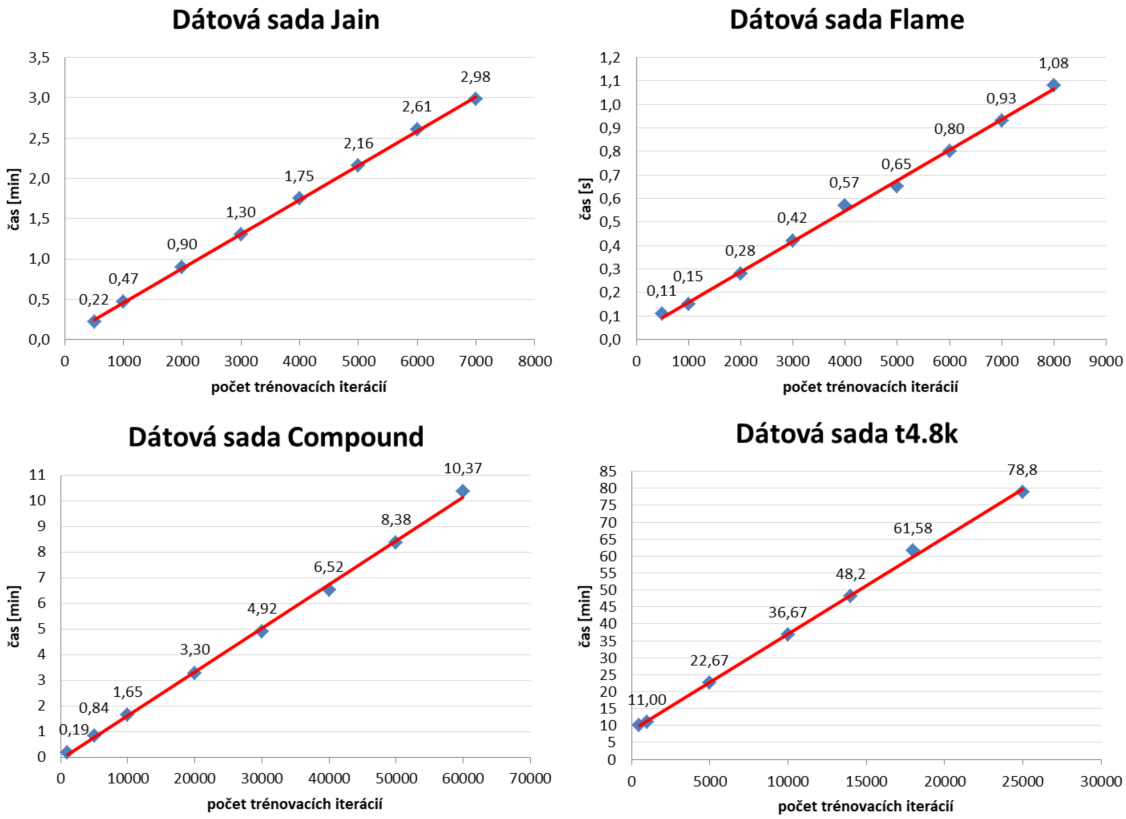


Obr. 7.3: Grafy závislosti času na počte prvkov jednotlivých dátových sád pre algoritmus SOM.

7.3.1 Závislosť času na počte trénovacích iterácií

Pri experimentoch, kde boli hľadané správne parametre algoritmu SOM pre rôzne dátové sady, bolo treba si všimnúť, že s väčším počtom iterácií narastal aj čas celkového zhlukovania. Takže bolo treba zistiť, aký vplyv má počet trénovacích iterácií na čas. Boli vykonané experimenty, kde sa rôzne menil počet trénovacích iterácií, ostatné parametre boli prebrané z riešení, pri ktorých boli získané najlepšie výsledky zhlukovania. Experimenty boli vyko-

nané na 4 dátových sadách: Jain, Flame, Compound a t4.8k. Na dátovej sade MNIST neboli experimenty vykonané z dôvodu časovej náročnosti. Ako je možné vidieť na obrázku 7.4, s narastajúcim počtom trénovacích iterácií narastá aj čas. Pri všetkých dátových sadách bola dosiahnutá lineárna závislosť, čo znamená, že s narastajúcim počtom iterácií bude priamoúmerne narastať aj čas zhlukovania.



Obr. 7.4: Grafy závislosti času na počte trénovacích iterácií algoritmu SOM pre jednotlivé dátové sady.

7.4 Zhrnutie

Pri experimentoch sa vyskytovali menšie odchýlky merania. Tieto odchýlky mohli byť spôsobené niektorými úlohami, ktoré v čase experimentu zaťažili procesor.

Pre algoritmus DBSCAN sa podarilo overiť kvadratickú časovú zložitosť, pri algoritme Chameleon bola dosiahnutá kvadratická a linearitmická časová zložitosť, čo opäť potvrdzujú aj teoretické predpoklady.

Pri algoritme Chameleon je ťažké určiť časovú náročnosť pri dátových sadách, ktoré obsahujú malé množstvo objektov. Kvadratická časová zložitosť bola pri algoritme Chameleon potvrdená na všetkých dátových sadách okrem dátovej sady Compound, kde bola overená logaritmická časová zložitosť.

Časovú zložitosť algoritmu SOM sa podarilo overiť na dátových sadách Jain, Compound, t4.8k a MNIST, kde bola dosiahnutá kvadratická zložitosť. Pri dátovej sade Flame bola dosiahnutá logaritmická zložitosť a preto na tejto dátovej sade sa časovú zložitosť

nepodarilo overiť. Pri algoritme SOM bolo zistené, že s narastajúcim počtom trénovacích iterácií narastá aj čas zhlukovania priamoúmerne.

V tabuľke 7.1 je možné vidieť približný čas jednotlivých algoritmov potrebný na zhlukovanie jednotlivých dátových sád.

Data set	**Jain**	**Flame**	**Compound**	**t4.8k**	**MNIST**
DBSCAN	4.45 s	1.86 s	5.28 s	37 min	16.6 hod
Chameleon	4.47 s	8.82 s	1.18 min	48 min	26.13 hod
SOM	1.23 min	0.39 s	9 min	61 min	21.13 hod

Tabuľka 7.1: Prehľad časov zhlukovania jednotlivých algoritmov pre rôzne dátové sady.

Kapitola 8

Overovanie kvality zhlukovania pomocou siluetového koeficientu

V tejto kapitole sú popísané experimenty, ktoré boli vykonané pri hľadaní vhodných hodnôt parametrov [6], avšak z pohľadu určovania kvality zhlukovania pomocou siluetového koeficientu [2]. V experimentoch boli rôzne menené parametre algoritmov a boli sledované zmeny hodnôt siluetového koeficientu. Hodnoty siluetového koeficientu mohli nadobúdať hodnoty z intervalu <-1,1>, kde hodnota 1 udáva, že zhluky sú kompaktné a ďaleko od seba. Naopak hodnota -1 udáva, že zhluky nie sú kompaktné, prelínajú sa a podobne. Pre každý algoritmus a dátovú sadu bolo vykonaných niekoľko experimentov, ktoré sú popísané v tejto kapitole. Členenie týchto kapitol je iné ako v predchádzajúcich kapitolách, z dôvodu, že siluetový koeficient sa bude chovať skôr podľa dátovej sady ako algoritmu.

8.1 Dátová sada Jain

Dátová sada Jain obsahuje 2 zhluky, ktoré vytvárajú 2 vlnovky. Správne riešenia sa podarilo získať pre algoritmus Chameleon a SOM.

8.1.1 Algoritmus DBSCAN

Pre dátovú sadu Jain a algoritmus DBSCAN bolo vykonaných 11 experimentov, ktorých parametre a výsledky sú zobrazené v tabuľke 6.1. V experimentoch číslo 1-4 je možné vidieť, že bolo vytvorených pomerne veľké množstvo zhlukov, ktoré boli blízko seba (obrázok 6.1). Hodnoty siluetového koeficientu sú skoro rovnaké a blízke číslu 0, čo odpovedá tomu, že zhluky nie sú veľmi kompaktné a sú bližšie pri sebe.

V ostatných experimentoch sú hodnoty siluetového koeficientu vyššie, čo odpovedá tomu, že zhluky sú kompaktnejšie a ďalej od seba. Najkompaktnejšie zhluky boli dosiahnuté pri experimente číslo 8, kde boli vytvorené 3 zhluky (obrázok 6.3). Podľa experimentov, kde boli hľadané hodnoty parametrov algoritmu DBSCAN (tabuľka 6.1), bolo zistené, že najlepší výsledok dosiahneme pri hodnotách parametrov *Epsilon=2.5* a *MinPts=2*. V tomto prípade, pri experimente číslo 8, tento najlepší výsledok potvrdila aj hodnota siluetového koeficienta 0.35, ktorá bola najlepšia zo všetkých hodnôt, síce rozdelenie do zhlukov nebolo celkom správne.

8.1.2 Algoritmus Chameleon

Experimenty vykonané pre dátovú sadu Jain a algoritmus Chameleon je možné vidieť v tabuľke 6.6. V prvých troch experimentoch boli vytvorené celkom kompaktné zhluky s hodnotou siluetového koeficientu 0.45. Výsledok týchto experimentov boli 3 zhluky (obrázok 6.16). Ak by sme mali nájsť najkompaktnejšie zhluky, výsledky týchto experimentov by boli správne riešenia.

Avšak pri experimente číslo 4 a 5 (obrázok 6.17), kde bolo vytvorené správne rozdelenie do 2 zhlukov, bola hodnota siluetového koeficientu o niečo nižšia ako v predchádzajúcich experimentoch. Pri tomto experimente bola dosiahnutá hodnota siluetového koeficientu 0.4, čo odpovedá tomu že, zhluky sú viac menej kompaktné a ďalej od seba. Pre túto dátovú sadu a algoritmus Chameleon sa nepodarilo potvrdiť správny výber hodnôt parametrov pomocou siluetového koeficientu.

8.1.3 Algoritmus SOM

Ako je možné vidieť v tabuľke 6.11, pre dátovú sadu Jain a algoritmus SOM bolo vykonaných 5 experimentov, pri ktorých boli dosiahnuté hodnoty siluetového koeficientu okolo nuly alebo záporné, okrem jednej, ktorá bola 0.4. Táto hodnota vyjadruje fakt, že vytvorené zhluky (obrázok 6.32), nie sú celkom kompaktné. Napriek tomu, že rozdelenie do zhlukov bolo správne, bola dosiahnutá hodnota siluetového koeficientu 0.4 kvôli tomu, že zhluky tvoria vlnovky, kde vzdialenosti medzi objektmi jednej a druhej strany vlnovky sú veľké. Pre túto dátovú sadu a algoritmus SOM bol aj pomocou siluetového koeficientu potvrdený správny výber parametrov.

8.2 Dátová sada Flame

Táto dátová sada obsahuje 2 zhluky, ktoré sú ťažko rozlíšiteľné, preto je ťažké určiť, kde je hranica rozdelenia zhlukov. Najlepšie riešenia sa podarilo získať pri algoritme DBSCAN a Chameleon.

8.2.1 Algoritmus DBSCAN

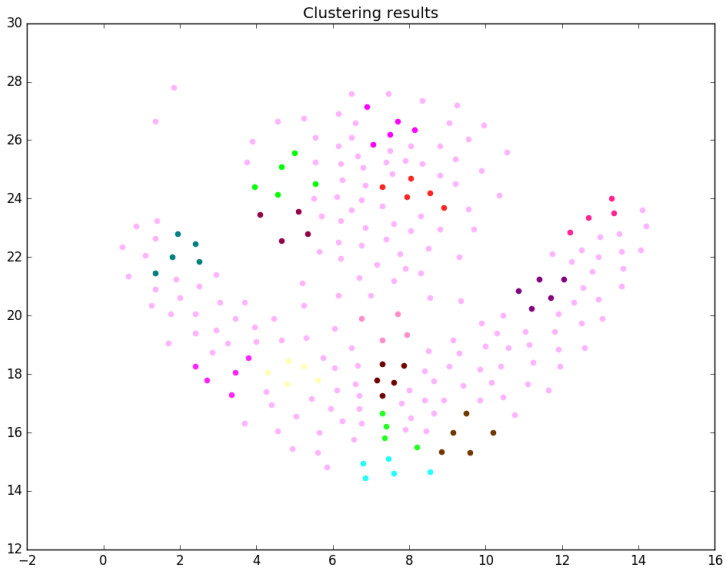
Celkovo bolo vykonaných 7 experimentov, ktoré sú zobrazené v tabuľke 6.2. Pri experimente číslo 1, 6 a 7 sa vytvoril len jeden zhluk a dva objekty, ktoré boli označené za šum (obrázok 6.5). V tomto prípade nebola získaná hodnota siluetového koeficientu, pretože implementácia, ktorú sme použili, potrebuje minimálne 2 zhluky na určenie hodnoty.

Najlepšiu hodnota 0.3 bola dosiahnutá pri experimente číslo 2 (obrázok 6.6), čo opäť potvrdilo, že najkompaktnejšie zhluky je možné dosiahnuť pre túto dátovú sadu pri hodnotách parametrov *Epsilon=1* a *MinPts=5*. Pri tejto dátovej sade sa zhluky nachádzajú hneď vedľa seba, preto hodnota parametra nie je vyšia. Taktiež na túto hodnotu parametra môže vplývať aj tvar spodného zhluku, ktorý je roztiahnutý a pokrýva takmer celú šírku dátovej sady. Napriek tomu, sa podarilo pomocou siluetového koeficientu potvrdiť správnosť výberu parametrov.

8.2.2 Algoritmus Chameleon

Pri tejto dátovej sade a algoritme Chameleon bolo vykonaných 12 experimentov, ktoré sú popísané v tabuľke 6.7. Táto dátová sada má veľmi ťažko rozlíšiteľnú hranicu zhlukov.

Pri experimente číslo 9 bolo vytvorených 15 zhlukov (obrázok 8.1). Na obrázku je vidieť, že jeden zhluk je prepletený celou dátovou sadou a miestami sú vytvorené malé zhluky. Tento fakt spôsobuje to, že zhluky nie sú kompaktné a preto hodnota siluetového koeficientu je -0.38. Na tomto experimente je vidieť nekompaktnosť zhlukov.



Obr. 8.1: Chameleon (Expected # of clusters=5, k-NN=10, Split=40, Alpha=2) pre dátovú sadu Flame.

Pri experimente číslo 2 boli dosiahnuté najkompaktnejšie zhluky podľa hodnoty siluetového koeficientu (obrázok 6.20). Taktiež tento výsledok bol označený ako najlepší pri experimentoch, kde boli hľadané správne nastavenie parametrov. Preto pre túto dátovú sadu a algoritmus Chameleon je možné označiť siluetový koeficient ako meradlo najlepšieho rozdelenia do zhlukov, čiže sa podarilo potvrdiť správny výber hodnôt parametrov pomocou siluetového koeficientu.

8.2.3 Algoritmus SOM

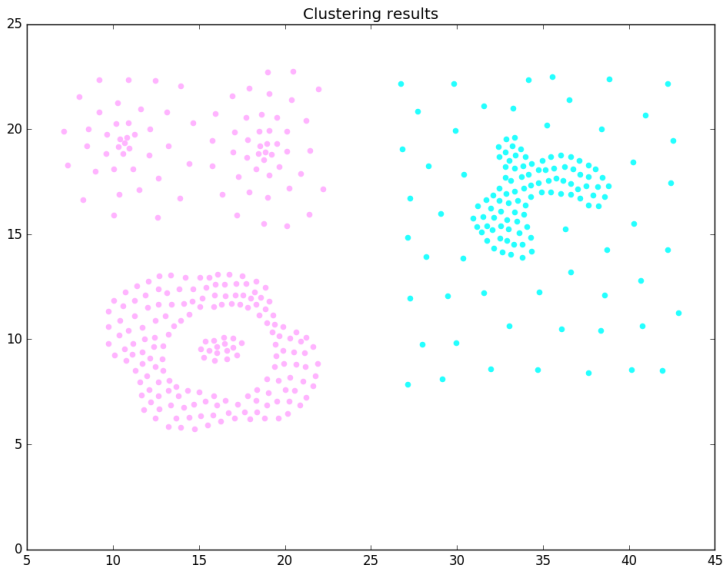
Pre túto sadu sa pri algoritme SOM nepodarilo nájsť správne parametre algoritmu, preto zhluky, ktoré sa vytvorili nie sú také, aké boli na začiatku požadované. Podľa hodnoty siluetového koeficientu 0.37 boli v experimente číslo 5 (tabuľka 6.12) vytvorené najkompaktnejšie zhluky (obrázok 6.33), avšak toto rozdelenie nie je správne.

8.3 Dátová sada Compound

Dátová sada Compound je pomerne ťažko zhlukovateľná, pretože obsahuje rôzne tvary zhlukov, taktiež obsahuje ťažko oddeliteľné zhluky a zhluk, ktorý je obklopený odľahlými hodnotami. Všetky algoritmy si s touto dátovou sadou v rámci možností poradili.

8.3.1 Algoritmus DBSCAN

Pre túto dátovú sadu a algoritmus DBSCAN bolo vykonaných 11 experimentov, ktorých parametre a výsledky je možné vidieť v tabuľke 6.3. Najlepší výsledok, kde boli vytvorené pomerne kompaktné zhluky s hodnotou siluetového koeficientu 0.64, bol dosiahnutý pri experimente číslo 11 (obrázok 8.2), kde boli vytvorené len 2 zhluky. V tomto prípade nejde o správne rozdelenie zhlukov. Preto pre túto dátovú sadu a algoritmus DBSCAN nie je možné použiť siluetový koeficient ako meradlo pre určenie správnych hodnôt parametrov.



Obr. 8.2: DBSCAN (Epsilon=4, MinPts=3) pre dátovú sadu Compound.

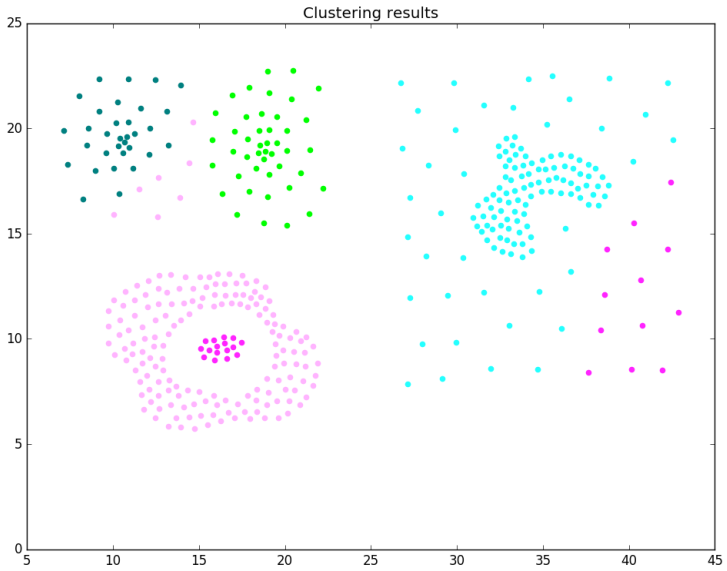
Pri experimente číslo 4 a 5 (obrázok 6.9), kde bolo dosiahnuté správne rozdelenie do zhlukov, bola dosiahnutá hodnota siluetového koeficientu 0.14 a 0.12. Tieto hodnoty ukazujú, že zhluky nie sú veľmi kompaktné a vzdialené od seba. V tomto riešení sa vyskytuje väčšie množstvo šumu, ktorý je označený ako jeden zhluk. Preto tento jeden zhluk, ktorý je roztiahnutý po celej dátovej sade, môže ovplyvňovať hodnotu siluetového koeficientu. Taktiež na hodnotu siluetového koeficientu, môže vplývať zhluk vľavo dolu, ktorý v sebe obsahuje ďalší zhluk.

8.3.2 Algoritmus Chameleon

Podľa experimentov pri hľadaní správnych parametrov bolo zistené, že najlepšie rozdelenie do zhlukov je možné dosiahnuť pri experimente číslo 6 (tabuľka 6.8), kde hodnota siluetového koeficientu je 0.28. Táto hodnota udáva, že zhluky, ktoré vznikli, nie sú veľmi kompaktné, čo je možné vidieť na obrázku 6.22. Takmer pri všetkých zhlukoch sú objekty ďalej od seba, preto hodnota siluetového koeficientu je nižšia.

Najkompaktnejšie zhluky boli vytvorené pri experimente číslo 3, kde bolo vytvorených celkovo 5 zhlukov (obrázok 8.3). Toto rozdelenie do zhlukov nebolo správne, pretože zhluky zasahovali do seba. Na prvý pohľad je vidieť, že zhluky veľmi kompaktne nepôsobia, okrem

zeleného, modrého a tmavozeleného zhluku. Keďže hodnota siluetového koeficientu je priemerná hodnota všetkých zhlukov, tak prípad obsahuje dva kompaktné zhluky (tmavozelený a zelený) a jeden menej kompaktný (modrý) a dva zhluky, ktoré kompaktné neboli. Preto je možné získať priemernú hodnotu koeficientu 0.46, keďže prevládajú kompaktné zhluky. Ako meradlo správneho rozdelenia do zhlukov sa siluetový koeficient pre túto dátovú sadu a algoritmus Chameleon nehodí.



Obr. 8.3: Chameleon (Expected # of clusters=5, k-NN=10, Split=20, Alpha=2) pre dátovú sadu Compound.

8.3.3 Algoritmus SOM

Pri všetkých experimentoch boli dosiahnuté kladné hodnoty siluetového koeficientu (tabuľka 6.13). Taktiež hodnoty sa od seba príliš nelíšili, pohybovali sa v rozmedzí od 0.1 do 0.23. Tieto hodnoty ukazujú, že vzniknuté zhluky nie sú veľmi kompaktné. V niektorých zhlukoch sú objekty ďaleko od seba, čo môže spôsobovať nízku hodnotu siluetového koeficientu. Pre najlepšie rozdelenie objektov do zhlukov (obrázok 6.35) bola dosiahnutá hodnota siluetového koeficientu 0.13, čo odpovedá tomu, že zhluky nie sú veľmi kompaktné, skôr sa prekrývajú (modrý, tyrkysový, tmavozelený a ružový zhluk). Pre túto dátovú sadu a algoritmus SOM sa pomocou siluetového koeficientu nepodarilo overiť správnosť zvolených hodnôt parametrov.

8.4 Dátová sada t4.8k

Dátová sada t4.8k obsahuje dobre odlíšiteľné zhluky, avšak aj veľké množstvo šumu, ktoré môže zhoršiť výsledky zhlukovania. Najlepšie výsledky boli dosiahnuté pre algoritmy DBSCAN a Chameleon.

8.4.1 Algoritmus DBSCAN

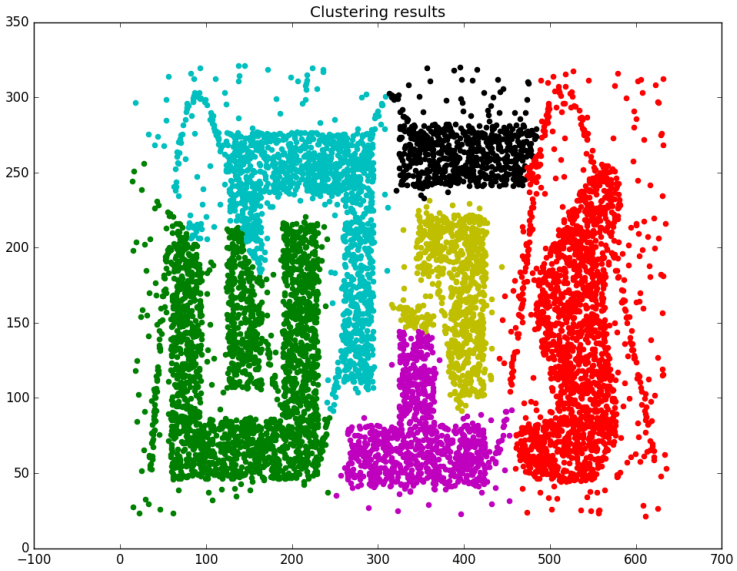
Počet experimentov pre túto dátovú sadu a algoritmus DBSCAN bol celkovo 15, ktoré sú zapísané v tabuľke 6.4. Je možné si všimnúť, že všetky hodnoty sa pohybujú okolo nuly alebo sú záporné okrem jednej, ktorá nadobúda hodnotu 0.24. Pri experimente číslo 2 (tabuľka 6.4) je možné vidieť hodnotu siluetového koeficientu -0.1. V tomto prípade (obrázok 6.10) došlo skoro k správnemu rozdeleniu do zhlukov, avšak v strede dátovej sady sa dva zhluky spojili a na vlnovke sa vytvorili malé zhluky, ktoré mali vplyv na hodnotu siluetového koeficientu.

Na rozdiel od predchádzajúceho experimentu neboli v experimente číslo 14 (obrázok 6.11), ktorý bol označený ako najlepší, vytvorené žiadne malé zhluky na vlnovke a hodnota siluetového koeficientu bola 0.24. Pri tomto experimente boli vytvorené najkompaktnejšie zhluky v rámci dátovej sady. Napriek tomu, že vzdialenosti bočných objektov v rámci zhluku sú pomerne veľké a dátová sada obsahuje pomerne veľké množstvo šumu, bolo pomocou siluetového koeficientu dokázané, že v tomto prípade je možné určiť správnosť rozdelenia do zhlukov, čiže podľa siluetového koeficientu sme neoverili správnosť hodnôt parametrov algoritmu.

8.4.2 Algoritmus Chameleon

Pri tejto dátovej sade a algoritme Chameleon bolo vykonaných 11 experimentov, ktoré sú zapísané v tabuľke 6.9. Pri experimente číslo 11, kde bolo správne rozdelenie objektov do zhlukov neboli vytvorené najkompaktnejšie zhluky zo všetkých experimentov. Napriek tomu bola dosiahnutá hodnota siluetového koeficientu 0.3, čo znázorňuje, že vytvorené zhluky nie sú veľmi kompaktné. Na obrázku 6.25 je vidieť, že zhluky, ktoré sa vytvorili, obsahujú aj časti vlnovky a samostatné objekty mimo hlavnú časť zhluku. Toto môže mať vplyv na hodnotu siluetového koeficientu, že jednotlivé objekty v zhlukoch nie sú blízko seba.

Najkompaktnejšie zhluky z pohľadu siluetového koeficientu boli vytvorené pri experimente číslo 7 a 8 (obrázok 8.4), kde je zelený zhluk obsahuje časť zhluku, ktorá by mala patriť modrému zhluku. Takisto je to pri žltom zhluku. Toto rozdelenie spôsobilo, že sa vytvorili kompaktnejšie zhluky, ktoré ale neboli správne rozdelené. Pre túto dátovú sadu a algoritmus Chameleon nemôžeme použiť siluetový koeficient ako meradlo správneho rozdelenia objektov do zhlukov, čiže pomocou siluetového koeficientu nebol potvrdený správny výber hodnôt parametrov algoritmu.



Obr. 8.4: Chameleon (Expected # of clusters=6, k-NN=6, Split=11, Alpha=2) pre dátovú sadu t4.8k.

8.4.3 Algoritmus SOM

Pri tejto dátovej sade boli získané hodnoty siluetového koeficientu, ktoré sa pohybovali okolo nuly (tabuľka 6.14). Najkompaktnejšie zhluky boli vytvorené pri experimente číslo 7, kde sa vytvorilo 14 zhlukov a hodnota siluetového koeficientu bola 0.17. Táto hodnota udáva, že objekty nie sú veľmi kompaktné, skôr sa zhluky prelínajú. Túto hodnotu mohol spôsobiť aj modrý zhluk (obrázok 6.38), ktorý je roztrúsený takmer po celej dátovej sade. Pomocou siluetového koeficientu nie je možné v tomto prípade určiť najlepšie riešenie.

8.5 Dátová sada MNIST

Dátová sada MNIST obsahuje 784 dimenzií. Kvalita zhlukovania pri tejto dátovej sade pri algoritme Chameleon a SOM bola viac menej rovnaká. Algoritmus DBSCAN si s touto dátovou sadou nevedel poradiť.

8.5.1 Algoritmus DBSCAN

Pri dátovej sade MNIST boli vykonané experimenty pre 500 a 1000 objektov, ktoré je možné vidieť v tabuľke 6.5. Napriek tomu, že sa nepodarilo zistiť vhodné parametre pre tento algoritmus, aby výsledok dátovej sady bol akceptovateľný, hodnoty siluetového koeficientu, ktoré boli získané, by mohli odpovedať rozdeleniu jednotlivých zhlukov (obrázok 6.15). Zhluky nie sú veľmi kompaktné, čo je možné vidieť aj na obrázku 6.15, kde sa v zhlukoch nachádzajú rôzne číslice a niektoré zhluky obsahujú malé množstvo objektov.

Pri experimentoch s 5000 objektmi a celou dátovou sadou boli získané hodnoty siluetového koeficientu -0.1 a -0.12. Rozdelenie číslic v jednotlivých objektoch nebolo analyzované, ale podľa hodnoty siluetového koeficientu, ktorú nadobúdal v experimentoch pri 500 a 1000

objektoch, je možné predpokladať, že rozloženie číslic v zhlukoch bude podobné tým, ktoré boli získané pri 500 a 1000 objektoch.

8.5.2 Algoritmus Chameleon

Pri tejto dátovej sade boli vykonané experimenty pre 500 a 1000 objektov dátovej sady, ktoré boli analyzované (tabuľka 6.10). Pri experimentoch boli dosiahnuté hodnoty siluetového koeficientu okolo nuly. Táto hodnota znázorňuje to, že sa zhluky prekrývajú, čo je v tomto prípade spôsobené tým, že sa v zhlukoch vyskytujú rôzne číslice. Po vyhodnotení jednotlivých zhlukov bolo zistené, že väčšinou obsahujú nejakú dominantnú číslicu a pár iných číslic (obrázok 6.29). Preto pre túto dátovú sadu nie je vhodné použiť siluetový koeficient ako ukazovateľ správneho rozdelenie objektov do zhlukov, keďže sa nevytvárajú kompaktné zhluky.

Algoritmus bol spustený aj pre 5000 objektov, kde hodnota siluetového koeficientu bola 0.03 a pre celú dátovú sadu, kde hodnota bola 0.06. Je možné predpokladať na základe hodnôt zistených pri experimentoch pre 500 a 1000 objektov, že rozdelenie do zhlukov bude podobné ako pri experimentoch pre 500 a 1000 objektov. Zhluky pravdepodobne budú obsahovať prevažne jednu až dve dominantné číslice.

8.5.3 Algoritmus SOM

Pre túto dátovú sadu bolo vykonaných celkovo 13 experimentov, ktoré boli analyzované (tabuľka 6.15 a 6.16). Išlo o experimenty s 500 a 1000 objektmi dátovej sady. Pri týchto experimentoch boli získané hodnoty siluetového koeficientu blízke nule, čo ukazuje na prekrývajúce sa zhluky. V zhlukoch sa vyskytovala zvyčajne jedna až dve, prípadne tri dominantné číslice, okrem ktorých sa v zhlukoch vyskytovali aj ďalšie číslice (obrázok 6.42). Záporné hodnoty siluetového koeficientu, ktoré sa v experimentoch objavili znamenajú, že niektoré objekty mali byť priradené k iným zhlukom. V tomto prípade nie je možné podľa siluetového koeficientu určiť, ktorý experiment je najlepší, pretože hodnoty sú pri všetkých experimentoch skoro rovnaké a blízke nule.

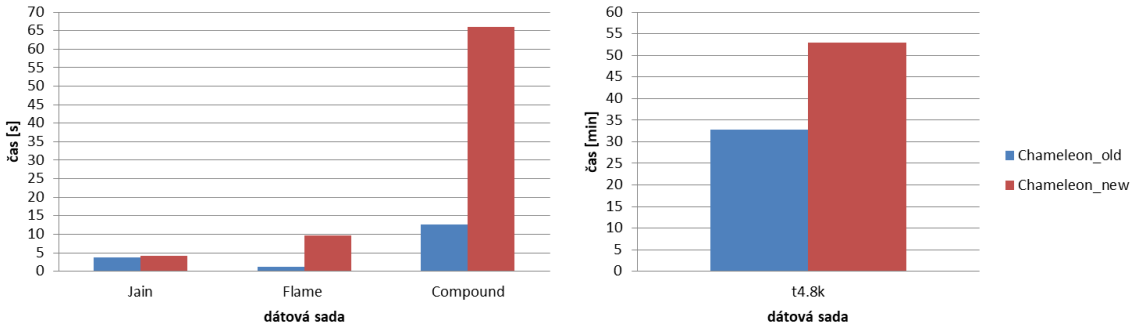
8.6 Zhrnutie

V experimentoch pre overovanie kvality zhlukovania pomocou siluetového koeficientu bolo zistené, že siluetový koeficient nie je vhodný pre hodnotenie kvality zhlukovania pri dátových sadách, v ktorých sa objekty nezhlukujú prirodzene do kompaktných zhlukov, ktoré sú dobre oddeliteľné. Taktiež bolo zistené, že správne riešenia nevytvárajú vždy kompaktné zhluky, ktoré sú ďaleko od seba. V niektorých prípadoch sa podarilo overiť výber správnych parametrov algoritmov podľa siluetového koeficientu tým, že sa vytvorili pomerne kvalitné a kompaktné zhluky. Ďalej bolo zistené, že siluetový koeficient je závislý na tvare zhlukov a vzdialenosti objektov v rámci zhluku. Pri dátovej sade Jain, kde mali zhluky tvar vlnovky bola hodnota siluetového koeficientu okolo 0.4, pretože objekty na jednej a druhej strane vlnovky boli ďaleko od seba. Aby siluetový koeficient označil zhluky za perfektne kompaktné, mali by mať zhluky kruhovitý tvar a mali by byť čo najďalej od seba. Objekty by v rámci jedného zhluku mali byť čo najhustejšie na sebe, vtedy by nám vyšli najlepšie hodnoty pre siluetový koeficient.

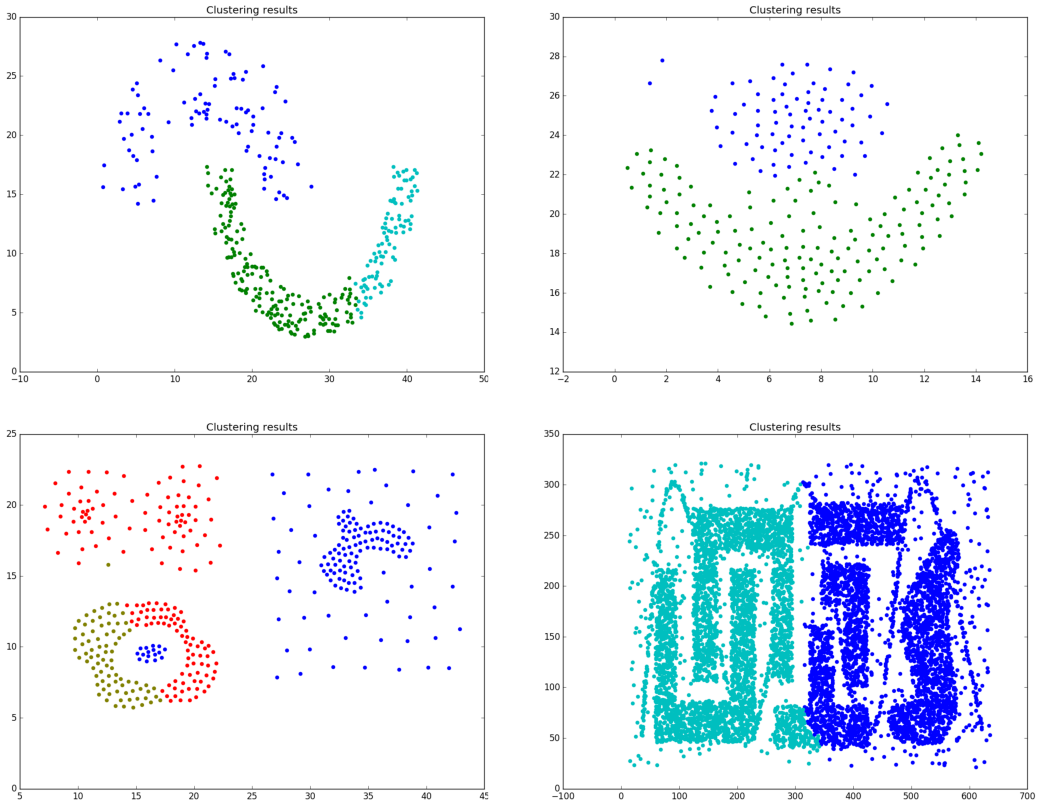
Kapitola 9

Porovnanie implementácií algoritmu Chameleon

V tejto práci boli použité dve implementácie algoritmu Chameleon. Práca začínala s implementáciou, ktorej autorom bol Giovanni Paolo [21], avšak po čase bolo zistené, že táto implementácia neprodukuje akceptovateľné výsledky pre žiadne hodnoty parametrov (obrázok 9.2). Podľa článku [16] ale algoritmus Chameleon by si mal vedieť poradiť aj s dátovou sadou t4.8k, preto bolo potrebné vybrať vylepšenú verziu tejto implementácie, ktorej autorom bol Amber Lin [17]. Táto implementácia pri správnom nastavení parametrov produkovala správne výsledky. Napriek tomu, že prvá implementácia neprodukovala správne výsledky, boli porovnané časy potrebné na zhlukovanie s vylepšenou verziou algoritmu Chameleon. Výsledky, ktoré boli dosiahnuté je možné vidieť na obrázku 9.1. Prvá implementácia potrebovala na zhlukovanie menej času ako vylepšená verzia, ale neprodukovala tak dobré výsledky ako vylepšená verzia. Preto bola použitá len vylepšená verzia algoritmu Chameleon.



Obr. 9.1: Porovnanie časovej náročnosti dvoch implementácií algoritmu Chameleon pre rôzne dátové sady.



Obr. 9.2: Ukážky výsledkov zhlukovania pre prvú implementáciu algoritmu Chameleon pre rôzne dátové sady.

Kapitola 10

Záver

V tejto diplomovej práci bola popísaná zhluková analýza, požiadavky na zhlukovú analýzu, typy dát, ktoré sa v zhlukovej analýze používajú a ako sa zhlukovanie ohodnocuje. Taktiež boli popísané rôzne zhlukovacie metódy a ich teoretické vlastnosti.

Účelom diplomovej práce bolo analyzovať vlastnosti vybraných zhlukovacích algoritmov. Pre účely tejto diplomovej práce boli vybrané tri algoritmy. Prvým algoritmom je DBSCAN, ktorý patrí medzi metódy založené na hustote. Implementáciu tohto algoritmu, ktorá bola použitá napísal autor Sushant Kafle. Druhý algoritmus, ktorý bol vybraný, bol Chameleon, ktorý patrí medzi hierarchické zhlukovacie metódy. Prvá implementácia, ktorá bola vybraná neprodukovala dobré výsledky. Autorom tejto implementácie je Giovanni Paolo. Bolo nutné rozhodnúť sa pre jej vylepšenú verziu, ktorá už produkovala dobré výsledky. Autorom vylepšenej verzie je Amber Lin. Posledný algoritmus, ktorý bol vybraný je SOM, ktorý patrí medzi metódy založené na modeloch. Autorom implementácie, ktorá bola použitá je Federico Comitani.

V ďalšej časti boli vybrané dátové sady, ktoré sú ako vstupné dáta pre algoritmy. Dátové sady sú vo forme CSV súborov. Medzi dátové sady, ktoré boli vybrané patrí: Jain, Flame, Compound, t4.8k a MNIST. Všetky dátové sady, okrem dátovej sady MNIST, sú dvojdimenzionálne. Dátová sada MNIST má 784 dimenzií a predstavuje ručne písané číslice od 0 po 9.

V tejto diplomovej práci bola vytvorená aplikácia pre prácu s algoritmami a dátovými sadami. Návrh grafického užívateľského rozhrania prešiel niekoľkými fázami. Aplikácia je napísaná v jazyku Python a spúšťa vybrané algoritmy na dátových sadách. Aplikácia má jednoduché grafické rozhranie, v ktorom sa dá vybrať konkrétny algoritmus, konktrétna dátová sada a dajú sa nastaviť hodnoty parametrov pre jednotlivé algoritmy. Aplikácia po zhlukovaní vykreslí výsledok v podobe grafu, kde sú farbou rozlíšené jednotlivé zhluky. Taktiež sa zobrazia v pravej dolnej časti informácie ako je počet objektov v dátovej sade, počet vytvorených zhlukov, čas zhlukovania a hodnota siluetového koeficientu. Aplikácia ukladá výsledky v podobe grafov do priečinku *results* a taktiež do CSV súboru, kde je ku každému objektu priradený zhluk, do ktorého objekt patrí. Aplikácia umožňuje pridanie vlastnej dátovej sady, ktorá musí byť vo formáte CSV.

V práci sú popísané experimenty, kde boli hľadané najlepšie hodnoty parametrov pre jednotlivé algoritmy, tak aby produkovali najlepšie rozdelenie do zhlukov pri každej dátovej sade. Zoznam parametrov pre jednotlivé algoritmy a dátové sady je rozpísaný v tabuľke 6.17. Pre všetky algoritmy a dátové sady sa podarilo nájsť vhodné parametre, okrem algoritmu SOM pre dátovú sadu Flame a t4.8k a algoritmu DBSCAN pre dátovú sadu Jain a MNIST.

Pri overovaní a porovnávaní časovej zložitosti algoritmov s teoretickými predpokladmi bolo zistené, že algoritmus DBSCAN mal pri všetkých dátových sadách kvadratickú časovú zložitosť, čo potvrdzujú aj teoretické predpoklady. Pri algoritme Chameleon pri dátovej sade MNIST, ktorá má 784 dimenzií bola dosiahnutá kvadratická časová zložitosť, čo potvrdzujú aj teoretické predpoklady, že pre vysokodimenzionálne dátové sady má kvadratickú časovú zložitosť. Pri dátových sadách, ktoré obsahovali malý počet objektov nebolo zrejmé, do ktorej triedy algoritmus zaradiť, pretože pri rôznych dátových sadách boli dosiahnuté rôzne časové zložitosti. Podľa teoretických predpokladov by mal Chameleon mať pri nízkodimenzionálnych dátových sadách linearitmickú časovú zložitosť, čo sa podarilo overiť pri dátovej sade Compound. Časová zložitosť algoritmu SOM bola overená na dátových sadách Jain, Compound, t4.8k a MNIST, kde bola dosiahnutá kvadratická časová zložitosť, čo potvrdzujú aj teoretické predpoklady. Na dátovej sade Flame bola dosiahnutá logaritmická zložitosť, čo môže byť spôsobené tým, že dátová sada obsahuje málo objektov a že na tejto dátovej sade dominuje iná zložka a časová zložitosť nie je čisto závislá len na počte prvkov. Pri experimentoch, kde bola zisťovaná závislosť času na počte trénovacích iterácií algoritmu SOM bolo zistené, že s narastajúcim počtom trénovacích iterácií narastá aj čas priamoúmerne.

Pri experimentoch, v ktorých bola overovaná kvalita zhlukovania pomocou siluetového koeficientu bolo zistené, že siluetový koeficient nie je vhodný pre určovanie kvality zhlukovania pre dátové sady, v ktorých sa prirodzene vytvárajú nekompaktné zhluky. Ďalej bolo zistené, že správne riešenia nevytvárajú vždy kompaktné zhluky. Taktiež bolo overované, či siluetový koeficient je vhodným ukazovateľom, či sa zhluky vytvorili správne a bolo zistené, že podľa siluetového koeficientu nie je možné vždy určiť, že sa zhluky vytvorili správne, pretože siluetový koeficient je závislý aj na tvare zhlukov a vzdialenosti objektov v rámci zhluku. Najlepšími zhlukmi pre siluetový koeficient sú zhluky kruhového tvaru, v ktorom sú objekty husto pri sebe. Napríklad pri dátovej sade Jain, kde dva zhluky tvoria 2 vlnovky sú objekty blízko pri sebe, ale vzdialenosti medzi objektmi na jednej a druhej strane vlnovky sú veľké, preto siluetový koeficient zobrazil hodnotu 0.4 (hodnota 1 označuje perfektne kompaktné zhluky ďaleko od seba).

V tejto diplomovej práci boli použité dve implementácie algoritmu Chameleon, ktoré boli porovnané. Bolo zistené, že vylepšená verzia produkuje lepšie rozdelenie objektov do zhlukov ako jej predchádzajúca verzia, ale na úkor času. Prvá verzia algoritmu Chameleon potrebovala menej času na produkovanie zhlukov, ako je možné vidieť na obrázku 9.1.

Aplikácia, ktorá bola v práci vytvorená môže byť vhodná na pozorovanie vytvorených zhlukov pri rôznych hodnotách parametrov, pozorovanie počtu vytvorených zhlukov a hodnôt siluetového koeficientu. Aplikácia môže byť v budúcnosti rozšíriteľná o ďalšie algoritmy, avšak to by znamenalo aj väčší zásah do grafického užívateľského rozhrania, kde by sa musel rozšíriť priestor pre hodnoty ďalších parametrov pre ďalšie algoritmy. Takisto by bolo možné pridať vykresľovanie 3-dimenzionálnych dátových sád v 3D grafoch.

Literatúra

[1] Aggarwal, C. C.; Reddy, C. K.: *Data clustering: Algorithms and Applications*. CRC Press, 2014, ISBN 978-1-4665-5822-9.

[2] Barton, T.: *Clustering-benchmark*.
URL https://github.com/deric/clustering-benchmark

[3] Besson, L.: *Self-Organizing Maps and DSOM: From unsupervised clustering algorithms to models of cortical plasticity*.
URL https://perso.crans.org/besson/publis/mva-2016/MVA_2015-16__Neuro-Sciences__Project__Lilian_Besson__Report.en.pdf

[4] Bhatia, P.: *Data Mining and Data Warehousing: Principles and Practical Techniques*. Cambridge University Press, 2019, ISBN 978-1-108-72774-7.
URL http://www.cambridge.org/9781108727747

[5] Burgetová, I.: *Shluková analýza*. online, 2006.
URL https://www.fit.vutbr.cz/study/courses/ZZN/private/prednasky/09_Shlukova_analyza.pdf

[6] Chang, S.-K.: *Data structures and algorithms*. World Scientific Publishing Co. Pte. Ltd., 2003, ISBN 981-238-348-4, doi:https://doi.org/10.1142/5256.

[7] Cios, K. J.; Swiniarski, R. W.; Pedrycz, W.; aj.: *Data Mining: A knowledge discovery approach*. Springer US, 2007, ISBN 978-0-387-36795-8.

[8] Comitani, F.: *SimpSOM*.
URL https://pypi.org/project/SimpSOM/

[9] Fränti, P.; Sieranoja, S.: K-means properties on six clustering benchmark datasets. 2018.
URL http://cs.uef.fi/sipu/datasets/

[10] Gan, G.; Ma, C.; Wu, J.: *Data clustering: Theory, Algorithm and Applications*. American Statistical Association and the Society for Industrial and Applied Mathematics, 2007, ISBN 978-0-898716-23-8.

[11] Han, J.; Kamber, M.: *Data mining: concepts and techniques*. Morgan Kaufmann Publishers, druhé vydanie, 2006, ISBN 978-1-55860-901-3.

[12] Han, J.; Kamber, M.; Pei, J.: *Data mining: concepts and techniques*. Morgan Kaufmann Publishers, tretie vydanie, 2012, ISBN 978-0-12-381479-1.

[13] Heyer, L. J.; Kruglyak, S.; Yooseph, S.: *Exploring Expression Data: Identification and Analysis of Coex-pressed Genes*, ročník 9. Genome Research, 1999, 1106 - 1115 s.
URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC310826/

[14] Jin, X.; Han, J.: *K-Medoids Clustering*. Boston, MA: Springer US, 2010, ISBN 978-0-387-30164-8, 564 - 565 s., doi:10.1007/978-0-387-30164-8_426.
URL https://doi.org/10.1007/978-0-387-30164-8_426

[15] Kafle, S.: *DBSCAN*.
URL https://github.com/SushantKafle/DBSCAN

[16] Karypis, G.; Han, E.-H. S.; Kumar, V.: *Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling*. Technická správa, Department of Computer Science and Engineering University of Minnesota, 1999.

[17] Lin, A.: *CHAMELEON_CLUSTER*.
URL https://github.com/Moonpuck/chameleon_cluster

[18] Loforte Jr., F.: *Efficient Variations of the Quality Threshold Clustering Algorithm*. Doctoral dissertation, Nova Southeastern University, 2015.
URL https://nsuworks.nova.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1042&context=gscis_etd

[19] Ng, R. T.; Han, J.: CLARANS: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, ročník 14, September 2002: s. 1003–1016, ISSN 1041-4347, doi:10.1109/TKDE.2002.1033770.
URL https://ieeexplore.ieee.org/document/1033770

[20] Valente de Oliveira, J.; Pedrycz, W.: *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, Ltd, 2007, ISBN 978-0-470-02760-8.

[21] Paolo, G.: *PyCHAMELEON*.
URL https://github.com/giovannipcarvalho/PyCHAMELEON

[22] Paredes, R.; Chávez, E.: Using the k-Nearest Neighbor Graph for Proximity Searching in Metric Spaces. In *String Processing and Information Retrieval*, editácia M. Consens; G. Navarro, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, ISBN 978-3-540-32241-2, s. 127–138.

[23] The QT Company LTD: *QT Designer*.
URL https://doc.qt.io/qt-5/qtdesigner-manual.html

[24] Saitta, S.; Raphael, B.; Smith, I. F.: *A bounded index for cluster validity*. 2007.
URL https://pdfs.semanticscholar.org/9701/405b0d601e169636a2541940a070087acd5b.pdf

[25] Scikit-learn developers: *sklearn.cluster.DBSCAN*.
URL https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[26] wikipedia: *Spatial database*.
URL https://en.wikipedia.org/wiki/Spatial_database#Spatial_index

Prílohy

Príloha A

Obsah DVD

- Výsledky experimentov pri hľadaní vhodných parametrov (***results***).
- Aplikácia pre analýzu vlastností zhlukovacích algoritmov (***application***).
- Zdrojový text technickej správy pre LaTeX (***technicka_sprava_latex***).
- Technická správa v PDF (***technicka_sprava_pdf***).