VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z ČASOPROSTOROVÝCH DAT

KNOWLEDGE DISCOVERY FROM SPATIO-TEMPORAL DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DAŠA LIPTÁKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

147110

Ústav:	Ústav informačních systémů (UIFS)
Studentka:	**Liptáková Daša, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Vývoj aplikací
Název:	**Získávání znalostí z časoprostorových dat**
Kategorie:	Data mining
Akademický rok:	2022/23

Zadání:

1. Seznamte se s problematikou dolování dat, podrobněji se zaměřte na oblast dolování v časoprostorových datech, zejména datech trajektorií pohybujících se objektů.
2. Vyhledejte několik datasetů s vhodnými daty tohoto typu.
3. Po konzultaci s vedoucím definujte vhodné úlohy a navrhněte ukázkovou aplikaci, která bude zvolenou úlohu provádět a zobrazovat ve vhodné formě její výsledky.
4. Navrženou aplikaci implementujte a ověřte její funkčnost.
5. Proveďte experimenty a vyhodnoťte přínos použitých metod.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování v tomto projektu.

Literatura:

- Yu Zheng:Trajectory Data Mining: An Overview. ACM Transactions on Intelligent Systems and Technology, Vol. 6, No. 3, Article 29, May 2015. Available at https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/TrajectoryDataMining-tist-yuzheng_published.pdf.
- Wang D., Miwa T., Morikawa T.: Big Trajectory Data Mining: A Survey of Methods, Applications, and Services. Sensors (Basel). 2020 Aug 14;20(16):4571. Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7472055/pdf/sensors-20-04571.pdf

Při obhajobě semestrální části projektu je požadováno:
Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	26.10.2022

Abstrakt

Táto diplomová práca sa zaoberá získavaním znalostí z časopriestorových dát. Najprv popisuje všeobecné princípy získavania znalostí a následne získavanie znalostí z časopriestorových dát, kde sa zameriava hlavne na metódy pre detekciu odľahlých trajektórií pohybujúcich sa objektov. V ďalšej časti práca popisuje návrh a implementáciu dolovacej úlohy a ukážkovej aplikácie. Na záver je nad troma rôznymi datasetmi vykonaných niekoľko experimentov.

Abstract

This thesis deals with knowledge discovery from spatio-temporal data. Firstly, it describes the general principles of knowledge discovery and then knowledge discovery from spatio-temporal data, where it mainly focuses on methods for detecting outlying trajectories of moving objects. In the next section, the thesis describes the design and implementation of the mining task and demonstration application. Finally, several experiments are performed over three different datasets.

Kľúčové slová

získavanie znalostí, časopriestorové dáta, dolovanie z dát, trajektória, detekcia odľahlých trajektórií, TRAOD, DBTOD

Keywords

knowledge discovery, spatio-temporal data, data mining, trajectory, trajectory outlier detection, TRAOD, DBTOD

Citácia

LIPTÁKOVÁ, Daša. *Získávání znalostí z časoprostorových dat.* Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Získávání znalostí z časoprostorových dat

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracovala samostatne pod vedením pána Ing. Vladimíra Bartíka, Ph.D. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Daša Liptáková
17. mája 2023

Poďakovanie

Ďakujem môjmu vedúcemu Ing. Vladimírovi Bartíkovi, Ph.D. za jeho rady, pripomienky a čas, ktorý mi venoval na konzultáciách. Ďakujem svojej rodine a priateľovi za možnosť štúdia a hlavne za podporu, ktorú mi počas neho venovali.

Obsah

Kapitola 1

Úvod

Informačné technológie sa stali neoddeliteľnou súčasťou nášho každodenného života, čo spôsobilo produkovanie pomerne veľkého množstva dát, ktoré stále stúpa. Prirodzene teda vzniká potreba analyzovania a získavania znalostí z týchto dát. V posledných rokoch sa tiež rozšírilo využitie lokalizačných systémov, ktoré sa neustále rozvíjajú. Lokalizačné systémy, ako napríklad GPS, sa v dnešnej dobe nachádzajú takmer v každom zariadení, ktoré sa denne využíva, ako napríklad mobilný telefón alebo automobil. Rozvíjajú sa tiež rôzne sledovacie zariadenia, senzory či satelity, vďaka ktorým je možné sledovať nie len ľudskú aktivitu či dopravnú situáciu, ale tiež migráciu zvierat či rôzne prírodné javy. Všetko toto prispieva k nárastu časopriestorových dát v podobe trajektórií pohybujúcich sa objektov.

Časopriestorové databázy sú pomerne aktívnou oblasťou výskumu a oproti iným druhom databáz je ich výskum ešte len v začiatkoch. Tieto dáta sa bežne používajú v oblastiach ako napríklad plánovanie dopravy, geografia či environmentalistika a konkrétne sa jedná napríklad o trajektórie ľudských pohybových aktivít, automobilov, hurikánov či zvierat. Dolovaním časopriestorových dát je možné získať mnoho užitočných informácií, ako napríklad nájdenie rôznych vzorov v trajektóriách alebo zhlukov podobných trajektórií, vďaka čomu sa dá napríklad predikovať ďalší pohyb objektu.

Zameraním tejto práce je oblasť detekcie odľahlých hodnôt v dátach trajektórií, vďaka ktorej je možné odhaliť napríklad podvody taxíkov. Pre detekciu odľahlých trajektórií dnes existuje niekoľko metód, z ktorých boli pre túto prácu vybrané *TRAOD* a *DBTOD*. V práci je navrhnutá a implementovaná ukážková aplikácia, ktorá vhodným spôsobom demonštruje výsledky oboch zvolených metód. Práca sa ďalej venuje experimentom, kde je každá z metód prevedená na troch rôznych datasetoch.

Práca je rozdelená do ôsmych kapitol. V kapitole 2 je zhrnuté a popísané všeobecné získavanie znalostí z dát a konkrétne získavaniu znalostí z časopriestorových dát sa venuje až kapitola 3. Detekcii odľahlých trajektórií sa venuje kapitola 4, kde sú detailne popísané aj vybrané metódy *TRAOD* a *DBTOD*. Tieto tri kapitoly poskytujú teoretické informácie. V nasledujúcej kapitole 5 je popísaný návrh výsledného riešenia – konkrétna dolovacia úloha, výber datasetov a návrh ukážkovej aplikácie. Následne, v kapitole 6, je popísaná jej implementácia. Kapitola 7 obsahuje výsledky experimentov a ich vyhodnotenie. V závere je zhrnutá táto práce a zhodnotenie výsledkov dolovania.

Kapitola 2

Získavanie znalostí z dát

Neustály rozvoj a nárast využitia informačných technológií, ktoré sú dnes neoddeliteľnou súčasťou života, spôsobil, že množstvo dát začalo extrémne rýchlo stúpať a stúpa dodnes. Získavanie znalostí z dát (angl. *Knowledge Discovery in Databases*) je vedný odbor, ktorý vznikol na základe potreby získavania relevantných informácií z tak veľkého množstva dát. Formálne ho môžme definovať ako extrakciu zaujímavých modelov dát a vzorov z veľkého množstva dát, ktoré následne reprezentujú získané znalosti.

Pojem **zaujímavá znalosť** predstavuje znalosť alebo informáciu, ktorá je:

- **netriviálna** – nedá sa získať obyčajným SQL dotazom či logickým odvodením a pre získanie je potrebné využiť sofistikovanejší postup
- **skrytá** – nie je na prvý pohľad vidieť a v dátach je nájdená netriviálnym spôsobom
- **vopred neznáma**
- **potencionálne užitočná**

Pri získavaní znalostí sa stretneme aj s množstvom iných výskumných oblastí ako je napríklad strojové učenie, databázové systémy, rôzne druhy štatistík či algoritmy. V súčasnosti existuje tiež mnoho oblastí praktického využitia získavania znalostí z dát, medzi ktoré patrí napríklad analýza trhu, marketingu, financií a rizík, detekcia podvodov či neobvyklého chovania a tiež rôzne špecializácie informatiky, napríklad bioinformatika, geoinformatika a zdravotníctvo či vzdelávanie.

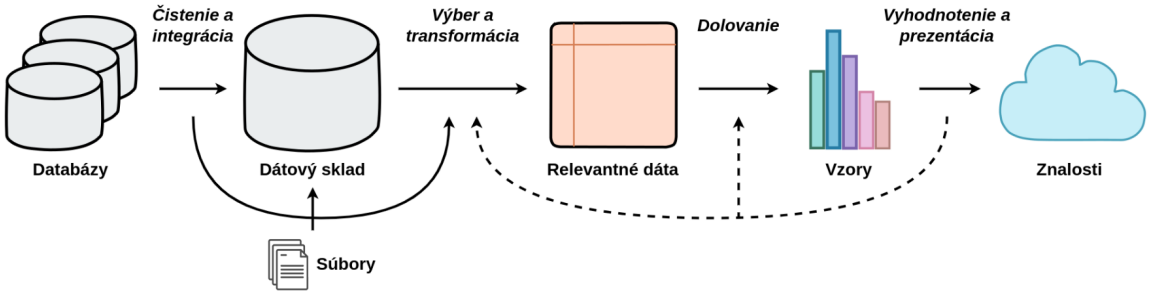
Získavanie znalostí z dát má viacero alternatívnych pomenovaní a jedným z nich je napríklad dolovanie dát (angl. *data mining*), ktoré sa v súčasnosti dokonca používa častejšie. Aj keď sú oba pojmy chápané ako synonymum, tak dolovanie dát je v skutočnosti len jedným z krokov v procese získavania znalostí, ktorému sa bude táto kapitola tiež venovať.

Kapitola je zameraná na teoretický úvod do získavania znalostí z dát, čo je proces pozostávajúci z niekoľkých krokov, ktoré sú bližšie popísané v podkapitole 2.1. V podkapitole 2.2 sú spomenuté niektoré z druhov dátových zdrojov, ktoré je možné využiť k dolovaniu dát. A záver kapitoly, teda podkapitola 2.3, je venovaný rôznym typom úloh pre dolovanie z dát. Pokiaľ nebude uvedené inak, tak všetky informácie spomenuté v tejto kapitole boli čerpané z [11] a [13].

2.1 Proces získavania znalostí

Proces získavania znalostí z dát pozostáva z viacerých krokov, ktoré sa v určitých iteráciách opakujú a to za účelom získania čo najlepšieho výsledku. Proces sa skladá z nasledujúcich siedmych krokov, ktoré je tiež možné vidieť na obrázku 2.1.

1. **Čistenie dát** – čistenie dát zahŕňa odstránenie chýbajúcich hodnôt, odstránenie šumu, identifikáciu odľahlých hodnôt a vyriešenie nekonzistencie dát.
2. **Integrácia dát** – v tomto kroku sa jedná o integráciu alebo inak povedané zlúčenie dát pochádzajúcich z viacerých dátových zdrojov. Keďže práve integrácia dát je jedným zo zdrojov nekonzistencie, tak je tento krok často spojený s krokom čistenia dát.
3. **Výber dát** – v tomto kroku sa vyberajú dáta, ktoré sú pre danú dolovaciu úlohu relevantné.
4. **Transformácia dát** – v tomto kroku sú dáta transformované do konsolidovanej podoby vhodnej pre následné dolovanie z dát. Vykonávajú sa operácie ako napríklad agregácia či sumarizácia.
5. **Dolovanie z dát** – dolovanie z dát je jadrom celého procesu získavania znalostí. V tomto kroku sú na dáta aplikované vhodné metódy a konkrétne algoritmy, ktorých výsledkom sú extrahované vzory, resp. modely dát.
6. **Hodnotenie modelov a vzorov** – v tomto kroku je cieľom identifikovať skutočne zaujímavé modely a vzory na základe miery užitočnosti.
7. **Prezentácia znalostí** – posledným krokom je prezentácia získaných znalostí užívateľovi. Znalosti môžu byť prezentované pomocou rôznych vizualizačných techník a reprezentácií znalostí ako napríklad tabuľky či grafy.



Obr. 2.1: Proces získavania znalostí z dát (upravené a prevzaté z [11]).

Kroky čistenie, integrácia, výber a transformácia dát sú rôznou formou *predspracovania dát* pre následné dolovanie. Dolovanie z dát, ako už bolo spomenuté, je hlavným krokom procesu získavania znalostí a môže v ňom dochádzať k potrebe interakcii s užívateľom či databázou znalostí. Naopak výsledok kroku dolovania dát môže databázu znalostí doplniť o nové znalosti, ktoré môžu byť využité či už v ďalších iteráciách alebo pri riešení ďalších dolovacích úloh.

2.2 Druhy dát k dolovaniu

Dolovanie dát je v podstate možné v akomkoľvek druhu zmysluplných dát. Jedná sa o perzistentne uložené dáta v rôznych úložiskách alebo dáta, ktoré sú tranzistentné, teda prúdy dát. V súčasnosti sú pravdepodobne najčastejším zdrojom dát relačné databáze. Medzi typické druhy dát patria tiež dátové sklady, transakčné databázy a ďalšie. V tejto podkapitole sú spomenuté niektoré z najznámejších druhov dát.

- **Relačné databázy** – jedná sa o kolekciu tabuliek, kde všetky stĺpce v každej z tabuliek obsahujú len atomické hodnoty, teda musia splňovať pravidlo prvej normálnej formy. Tabuľka pozostáva zo stĺpcov, ktoré obsahujú informácie o atribútoch záznamov a riadkov, ktoré reprezentujú jednotlivé záznamy. Každý záznam musí mať jednoznačný identifikátor nazývaný tiež ako primárny kľúč, ktorý môže byť zložený z viacerých atribútov.

  K dátam v relačnej databáze je možné pristupovať pomocou databázových dotazov, ktoré sú typicky napísané v jazyku SQL. Výsledok je možné využiť k rozhodovaniu, ale nejedná sa o dolovanie dát. Pri procese dolovania dát je z relačných databázy možné získať omnoho komplexnejšie znalosti.

- **Dátové sklady** – jedná sa o druh komplexného úložiska, v ktorom sú uchovávané dáta z niekoľkých rôznych dátových zdrojov. Vznikajú procesom čistenia dát, integrácie dát, transformácie dát, načítania dát a tiež periodického aktualizovania dát. Dáta sú v dátovom sklade uchovávané z historickej perspektívy a typicky sú sumarizované.

  Obvykle je dátový sklad modelovaný pomocou tzv. **multidimenzionálnej dátovej kocky**. Jedná sa o dátovú štruktúru, kde každá dimenzia predstavuje jeden alebo skupinu atribútov v schéme a každá bunka má agregované údaje. Z tohto dôvodu sú dátové sklady vhodné pre vykonávanie OLAP (Online Analytical Processing) operácií, vďaka ktorým poskytujú lepšiu podporu pre analýzu dát, ako napríklad relačné databázy. [22]

- **Transakčné databázy** – táto databáza je vo všeobecnosti obyčajný súbor, ktorý obsahuje záznamy. Tieto záznamy predstavujú jednotlivé transakcie, ktoré sú typicky tvorené jednoznačným identifikátorom transakcie a zoznamom položiek. Z pohľadu relačného modelu môže byť táto databáza chápaná ako nenormalizovaná tabuľka.

- **Objektovo-relačné databázy** - tieto databázy umožňujú prácu s objektami uloženými v tabuľkách a nenormalizovanými tabuľkami. Vďaka tomu je možné v tabuľkách ukladať zložitejšie štruktúry ako pri relačných databázach.

- **Temporálne databázy** – jedná sa o databázy, ktoré podporujú prácu s časom. Sú vhodné pre uchovávanie hodnôt atribútov, ktoré sa v čase menia a je potrebné mať uloženú ich históriu.

- **Sekvenčné databázy** – tieto databázy uchovávajú sekvencie usporiadaných udalostí. Rozhodujúce je usporiadanie jednotlivých udalostí a teda nie je nutné zaznamenávať čas vykonania danej udalosti.

- **Databázy časových radov** – tu sa uchovávajú postupnosti hodnôt či udalosti zaznamenané chronologicky v daných časových okamihoch.

- **Priestorové databázy** – v týchto databázach sú uchovávané údaje vzťahujúce sa k priestorovému usporiadaniu. Typicky sa jedná napríklad o geografické databázy obsahujúce mapy.
- **Časopriestorové databázy** – jedná sa o databázu, ktorá súčasne uchováva časovú aj priestorovú zložku. Typicky sa jedná o pohybujúce sa objekty, ktoré uchováva v podobe ich trajektórií. Podrobnejšiemu popisu časopriestorových dát a získavania znalostí z nich sa venuje nasledujúca kapitola 3.
- **Textové databázy** – v tomto druhu databázy sa uchovávajú neštrukturované, čiastočne štrukturované alebo štrukturované dokumenty. Dolovaním v tomto druhu dát je možné zistiť napríklad to, v akom jazyku je dokument napísaný.
- **Multimediálne databázy** – tento druh databázy uchováva hlavne obrazové, video a audio dáta, ale niekedy sa sem radia aj dáta textovej formy.
- **Prúdy dát** – jedná sa o súvisle produkované dáta, ktorých objem je obrovský, potenciálne nekonečný a dynamicky sa meniaci. Dostupnosť dát v prúde je obmedzená, dáta sa neskladujú a teda je dostupný len obmedzený počet predchádzajúcich dát. Typicky sa jedná napríklad o údaje pochádzajúce zo senzorov.
- **Web** – obrovská a rýchlo rastúca heterogénna databáza, ktorá je jednoducho dostupná cez internetovú sieť. Obsahuje veľké množstvo dát rôznych formátov a preto má veľký potenciál pre získavanie znalostí. Medzi typické dolovacie úlohy patrí napríklad automatické zhlukovanie či klasifikácia webových stránok.

2.3 Typy dolovacích úloh

Dolovanie z dát, ako už bolo spomenuté, je jadrom celého procesu získavania znalostí z dát. V tejto podkapitole sú vysvetlené základné typy dolovacích úloh. Inak povedané, aké rôzne druhy modelov či vzorov môžeme získať dolovaním zo zdrojových dát.

Medzi základné typy dolovacích úloh patrí napríklad popis konceptu/triedy, dolovanie frekventovaných vzorov, klasifikácia a predikácia, zhluková analýza a tiež analýza odľahlých hodnôt, ktoré sú detailnejšie popísané v tejto podkapitole. Okrem nich existujú aj iné, ale menej obvyklé úlohy. Vo všeobecnosti je možné typy dolovacích úloh rozdeliť do nasledujúcich dvoch základných skupín:

1. **Deskriptívne** – jedná sa o dolovacie úlohy, ktoré chrakterizujú všeobecné vlastnosti analyzovaných dát. Do tejto skupiny patrí napríklad zhluková analýza alebo hľadanie asociačných pravidiel.
2. **Prediktívne** – jedná sa o dolovacie úlohy, ktoré na základe analýzy dostupných dát určujú predpoveď budúceho chovania alebo hodnôt. Do tejto skupiny patrí napríklad klasifikácia alebo regresia.

Systémy pre dolovanie z dát sú schopné generovať veľké množstvo modelov, vzorov alebo pravidiel, kde väčšina z nich nie je v konečnom dôsledku pre koncového užívateľa zaujímavá. Typicky platí, že je preňho zaujímavá len malá časť z nich. Miera zaujímavosti vzoru môže byť braná z objektívneho aj subjektívneho hľadiska, ale je možné vymedziť niekoľko vlastností, ktoré charakterizujú **zaujímavosť vzoru**:

- **jednoduchá zrozumiteľnosť** pre užívateľa
- **platnosť** pre nové alebo testovacie údaje, ktoré majú istý stupeň **určitosti**
- **potenciálna užitočnosť**
- **novosť**

Zaujímavé vzory, modely s pravidlá predstavujú získanú znalosť.

2.3.1 Popis konceptu/triedy

Popis konceptu/triedy alebo inak povedané zovšeobecňovanie je typ dolovacej úlohy, ktorý radíme do skupiny deskriptívnych úloh. Jej cieľom je popísať koncepty alebo triedy nejakým stručným, súhrnným a zároveň dostatočne presným spôsobom. **Popis konceptu/triedy** je možné získať jedným z nasledujúcich spôsobov:

- **Charakterizácia dát** – predstavuje sumarizáciu všeobecných charakteristík alebo vlastností cieľovej triedy. Dáta zodpovedajúce triede je typicky možné získať jednoduchým databázovým dotazom.
- **Diskriminácia dát** – predstavuje vymedzenie triedy a to porovnávaním s triedou alebo množinou rozdielnych tried. V tomto prípade teda hľadáme atribúty a ich hodnoty, v ktorých sú triedy najviac rozdielne.

2.3.2 Frekventované vzory a asociačné pravidlá

Ďalším typom dolovacích úloh je odhalenie **frekventovaných vzorov**, teda vzorov, ktoré sa v dátach vyskytujú často. Frekventované vzory sa môžu objaviť v rôznych podobách a to napríklad ako frekventované množiny, frekventované postupnosti, podgrafy a mnoho ďalších. Frekventovaná množina typicky označuje množinu položiek, ktoré sa spolu často vyskytujú, napríklad môžeme mať frekventovanú množinu u transakčných databázy.

Dolovaním frekventovaných vzorov nachádzame v dátach zaujímavé **asociácie** a **korelácie**. Pre nájdenie zaujímavých asociácií sa využíva asociačná analýza, ktorej výsledkom sú asociačné pravidlá. Pomerne často sa asociačná analýza doplňuje o hľadanie zaujímavých korelácií medzi dvojicami asociačných pravidiel.

Definícia 2.1. *Nech $I = \{i_1, i_2, i_3, ...\}$ je množina položiek a nech D je množina transakcií, kde každá transakcia T je neprázdna množina položiek taká, že $T \subseteq I$. Ku každej transakcii T prináleží unikátny identifikátor, ktorý sa nazýva TID.* ***Asociačné pravidlo*** *je implikácia tvaru $A \Rightarrow B$, kde $A \subset I$, $B \subset I$ a $A \cap B = \emptyset$. Pravidlo $A \Rightarrow B$ má podporu (angl. support) a spoľahlivosť (angl. confidence) v množine transakcii D, ktoré sú s využitím pravdepodobnosti definované následovne:*

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B \mid A)$$

V prípade percentuálneho vyjadrenia podpory a spoľahlivosti má pravidlo $A \Rightarrow B$ podporu s, ak $s\%$ transakcií v databáze D obsahuje množinu položiek $A \cup B$. Spoľahlivosť c potom vyjadruje, že $c\%$ transakcií, ktoré obsahujú množinu A, obsahujú tiež množinu B. Nie všetky frekventované množiny a asociačné pravidlá sú pre nás zaujímavé a pre ich rozoznanie sa využívajú hodnoty minimálnej podpory (angl. *minimal support*) a minimálnej spoľahlivosti (angl. *minimal confidence*). Tieto hodnoty sa spravidla zadávajú v % z celkového počtu transakcií. Dolovanie asociačných pravidiel prebieha v nasledujúcich krokoch:

1. Nájdenie všetkých frekventovaných množín, ktoré spĺňajú vopred stanovanú podmienku minimálnej podpory.

2. Generovanie silných asociačných pravidiel z frekventovaných množín tak, aby vygenerované pravidlá splňovali nielen podmienku minimálnej podpory, ale aj minimálnej spoľahlivosti.

Jedným z vhodných príkladov je generovanie asociačných pravidiel z množiny nákupných transakcií, kde jedným z frekventovaných vzorov v nákupoch je mobilný telefón a ochranný obal. Vďaka tomuto zisteniu môžeme vytvoriť asociačné pravidlo medzi týmito položkami, ktoré bude vyzerať následovne:

$$mobil \Rightarrow obal \quad [support = 15\%, confidence = 85\%]$$

Podpora teda znamená počet nákupov, v ktorých sa spolu objavili mobilný telefón a ochranný obal. Spoľahlivosť označuje v koľkých % prípadov pri nákupe mobilného telefónu bol zakúpený súčasne aj ochranný obal.

2.3.3 Klasifikácia a regresia

Klasifikácia a regresia sú ďalšie z dolovacích úloh, ktoré radíme do skupiny prediktívnych úloh. Cieľom **klasifikácie** je hľadanie modelu, ktorý na základe vlastností priraďuje dáta do niektorej z konečnej množiny tried, teda popisuje a vymedzuje jednotlivé triedy dát. Nájdený model je následne využitý pre predikciu triedy objektu, ktorého zaradenie je neznáme. U klasifikácie predikujeme kategorické hodnoty, teda hodnoty, ktoré sú diskrétne a nezoradené. Pre predikciu numerických hodnôt spojitého charakteru sa využíva **regresia** a to najčastejšie metóda nazývaná regresná analýza.

Proces klasifikácie sa skladá z troch fáz – trénovanie/učenie, testovanie a následná aplikácia modelu. Proces pri regresnej analýze je mu obdobný, teda rovnako ako u klasifikácie prebiehajú tu nasledujúce tri fáze:

1. **Trénovanie/učenie** – fáza trénovania prebieha na trénovacej množine dát, pre ktoré platí, že poznáme ich triedy. Na základe tejto množiny je možné vytvoriť príslušný klasifikačný model, ktorý dokáže predikovať triedy neznámych objektov.

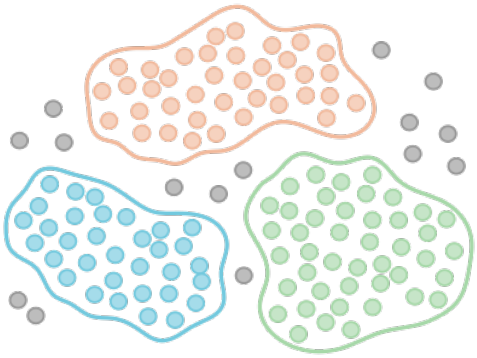
2. **Testovanie** – fáza testovania klasifikačného modelu prebieha na testovacej množine dát, pre ktoré taktiež platí, že poznáme ich triedy. Prebieha tu testovanie presnosti klasifikačného modela, ktorú dokážeme určiť vďaka porovnaniu referenčných tried s výsledkami klasifikátora. Na základe výsledkov testovania je následne rozhodnuté, či môže byť daný klasifikačný model ďalej použitý alebo je potrebné ho upraviť, prípadne vytvoriť nový.

3. **Aplikácia** – fáza, v ktorej je využitý klasifikačný model pre klasifikáciu objektu, ktorého trieda je neznáma. Jedná sa o klasifikačný model, o ktorom bolo v predošlom kroku rozhodnuté, že je dostatočne presný.

Klasifikačný model nemusí mať vždy rovnakú podobu. Môže nadobudnúť podobu napríklad rozhodovacieho stromu, klasifikačných pravidiel tvaru IF-THEN, matematických formúl či neurónových sieti.

2.3.4 Zhluková analýza

Ďalší typ dolovacej úlohy je zhluková analýza. Zatiaľ čo pri klasifikácii a regresii sa priraďujú objekty do vopred známych tried, tak pri zhlukovej analýze sú cieľove triedy vopred neznáme. Zhlukovanie je proces rozdelenia objektov do tried (zhlukov) na základe vopred definovaného kritéria podobnosti. Cieľom je vytvoriť zhluky tak, aby si objekty v rámci jedného zhluku boli v ideálnom prípade maximálne podobné a zároveň aby sa čo najmenej podobali objektom patriacim do iných zhlukov. Pre lepšiu predstavu je uvedený obrázok 2.2, na ktorom je znázornené vytvorenie zhlukov z 2D dát. Pre určenie podobnosti sa najčastejšie využíva vzdialenostná funkcia.



Obr. 2.2: Príklad zhlukovej analýzy, ktorý ukazuje tri zhluky (upravené a prevzaté z [11]).

Metód na zhlukovú analýzu existuje viacero a je možné ich rozdeliť do niekoľkých základných kategórií:

- **Metódy založené na rozdeľovaní** – tieto metódy rozdeľujú n objektov do vopred stanoveného počtu k tried, kde platí, že $k \leq n$. Tiež platí, že každá trieda musí obsahovať aspoň jeden objekt a každý objekt patrí len do jednej triedy. Najprv sa náhodne vyberie k objektov, ktoré reprezentujú práve jeden zhluk a ostatné objekty sú priradené do týchto zhlukov na základe podobnosti. Následne sa iteratívne hľadajú objekty, ktoré čo najlepšie reprezentujú jednotlivé zhluky a potom nasleduje presúvanie objektov medzi zhlukmi, tak aby bol splnení cieľ zhlukovania. Medzi najznámejšie a najčastejšie používané metódy patrí metóda založená na centrálnom bode (k-means) alebo metóda založená na reprezentujúcom objekte (k-medoids).

- **Hierarchické metódy** – tieto metódy vytvárajú hierarchický rozklad množiny objektov, ktorý môže prebiehať dvoma prístupmi. Prvý prístup je zdola-nahor (zhlukujúce hierarchické metódy), kde sú všetky objekty najprv priradené do vlastnej triedy a potom dochádza k zlúčeniu jednotlivý tried na základe podobnosti, až kým nevznikne jedna trieda alebo sa nedosiahne ukončovacej podmienky. Nie je tu nutné stanoviť počet tried, ako pri metódach založených na rozdeľovaní, ale je vhodné uviesť ukončujúcu podmienku. Druhým, opačným prístupom je zhora-nadol (rozdeľujúce hierarchické metódy), kde na začiatku patria všetky objekty do jednej triedy, ktorá je postupne rozdeľovaná na menšie zhluky. Konkrétne metódy sú napríklad AGNES alebo DIANA.

- **Metódy založené na hustote** – tieto metódy rozdeľujú množiny objektov na základe hustoty. Za zhluky sú považované oblasti s veľkou hustotou objektov v priestore, ktoré sú od seba oddelené oblasťami s malou hustotou objektov. Metódy založené

na hustote sú schopné nájsť zhluky rôznych tvarov a nemajú problém so šumom a odľahlými hodnotami. Medzi tieto metódy patrí napríklad metóda DBSCAN alebo DENCLUE.

- **Metódy založené na mriežke** – tieto metódy využívajú viacúrovňovú mriežkovanú dátovú štruktúru. Priestor objektov je rozdelený do buniek, ktoré vytvoria túto mriežku a ich počet je konečný. Zhlukovanie nie je realizované nad danými objektami, ale nad bunkami mriežky. Taktiež existuje viacero metód, napríklad sem patrí metóda WaveCluster.

- **Metódy založené na modeloch** – tieto metódy sa snažia optimalizovať zhody medzi dátami a vhodným matematickým modelom. Teda sa snažia nájsť také zhluky, ktoré maximálne zodpovedajú danému matematickému modelu. Patrí sem napríklad metóda Expectation-Maximization alebo COBWEB (jedna z metód konceptuálneho zhlukovania).

2.3.5 Analýza odľahlých hodnôt

Vždy je možné naraziť na dáta, ktoré sa výrazne odlišujú od väčšiny ostatných dát v dátovej sade. Jedná sa o odľahlé objekty (angl. *outliers*), inak povedané anomálie, ktoré sa nezhodujú s všeobecným chovaním alebo modelom dát. Väčšina dolovacích úloh nepovažuje odľahlé hodnoty za zaujímavé, ale práve naopak za nežiadúci šum, ktorý vo fáze predspracovania býva odstránený. Existujú ale tiež úlohy, ktoré považujú odľahlé hodnoty práve za tie zaujímavé. Medzi také úlohy patrí napríklad detekcia neobvyklého chovania alebo podvodu, ktoré využívajú analýzu odľahlých hodnôt vo fáze dolovania. Všeobecne môžme odľahlé hodnoty rozdeliť do troch nasledujúcich kategórií:

1. **Globálne odľahlé hodnoty** – jedná sa o izolované objekty, ktoré sú od ostatných objektov vzdialené a zodpovedajú ojedinelému chovaniu.

2. **Kontextové (okrajové) odľahlé hodnoty** – jedná sa o objekty s riedkym výskytom v oblasti zhlukov normálnych objektov, ktoré zvyčajne zodpovedajú hranici distribúcie.

3. **Kolektívne odľahlé hodnoty** – jedná sa o odľahlé zhluky objektov, ktoré sa spoločne odlišujú od ostatných, normálnych objektov. V rámci jedného odľahlého zhluku môžu, ale nemusia, byť jednotlivé objekty samy o sebe odľahlé.

Kapitola 3

Získavanie znalostí z časopriestorových dát

Časopriestorové dáta sú dáta, ktoré okrem iných vlastností uchovávajú aj *časovú* a súčasne *priestorovú* zložku. Kombinácia týchto zložiek umožňuje analýzu a porozumenie zložitejších javov, ktoré sa v čase a priestore menia. Podľa [10] je možné časopriestorové dáta kategorizovať do niekoľkých skupín, ktoré je možné zjednodušiť na udalosti v priestore a čase, postupnosti udalostí a trajektórie pohybujúcich sa objektov. Typicky sa u časopriestorových dát jedná práve o dáta pohybujúcich sa objektov v podobe ich trajektórií, ktorým konkrétne sa táto práca ďalej venuje. Trajektória je definovaná v podkapitole 3.1, ktorá sa okrem jej definície zaoberá tiež kategorizáciou a dátovými zdrojmi trajektórií a tiež klasifikáciou mobility objektov.

V súčasnosti sú lokalizačné a sledovacie systémy takmer v každom zariadení. Prirodzene tak dochádza aj k nárastu objemu časopriestorových dát. Aj z tohto dôvodu sú časopriestorové databázy pomerne aktívnou oblasťou výskumu a dá sa skonštatovať, že oproti iným druhom databáz je ich výskum ešte len v začiatkoch. Tieto dáta sa bežne používajú v oblastiach ako napríklad geografia, plánovanie dopravy či environmentalistika a konkrétnym príkladom sú trajektórie hurikánov, automobilov, zvierat a ďalšie. Vzhľadom k tomu, že sa v súčasnosti generuje podstatne veľký objem dát trajektórií a nie vždy sú dáta namerané správne, tak je potrebné ich pred samotným dolovaním predspracovať. Metódy predspracovania dát trajektórií sú popísané v podkapitole 3.2. V závere kapitoly, teda v podkapitole 3.3, sú popísane rôzne typy úloh pre dolovanie z dát trajektórií.

3.1 Trajektória pohybujúcich sa objektov

Trajektória je stopa generovaná pohybujúcim sa objektom v geografickom priestore, ktorá je zvyčajne reprezentovaná chronologicky usporiadanými bodmi. Aby bolo možné jednotlivé body chronologicky usporiadať a vytvoriť tak trajektóriu, musí každý bod obsahovať nielen priestorové súradnice, ale aj časové razítko. Striktná definícia trajektórie sa naprieč rôznymi zdrojmi mierne odlišuje a z tohto dôvodu som pre túto prácu na základe niekoľkých zdrojov odvodila nasledujúcu definíciu 3.1 [10, 33].

Definícia 3.1. *Trajektória TR je lomená čiara v trojrozmernom priestore, ktorá je reprezentovaná ako postupnosť bodov $TR = \{(x_i, y_i, t_i) \mid t_1 < ... < t_n,\ i \in \mathbb{N}\}$, kde x_i, y_i predstavujú geografické súradnice pohybujúceho sa objektu v čase t_i a n je celkový počet bodov v postupnosti.*

Ako vyplýva z definície, tak na vytvorenie trajektórie musí pohybujúci sa objekt získavať svoje súradnice x, y v čase t. Zdrojov dát technológií je v súčasnosti hneď niekoľko, napríklad globálny polohovací systém (GPS), globálny systém pre mobilnú komunikáciu (GSM), rádiofrekvenčná identifikácia (RFID) alebo WiFi. Podľa [16] a [23] je možné na základe vyššie uvedených zdrojov približne kategorizovať dáta trajektórií do nasledujúcich dvoch skupín, ktoré sú tiež zobrazené na obrázku 3.1.

1. **Explicitné dáta trajektórií** – sú definované ako dobre štruktúrované dáta, ktoré poskytujú informácie o polohe a čase. Majú pomerne silnú časopriestorovú kontinuitu. Medzi explicitné dáta trajektórií patrí napríklad GPS, ktoré zaznamenáva polohy objektu nepretržite v určitých časových intervaloch. Každý GPS záznam ale okrem informácii o polohe a čase môže uchovávať aj ďalšie informácie ako napríklad rýchlosť, ktorou sa objekt pohybuje.

2. **Implicitné dáta trajektórií** – sú, na rozdiel od explicitných dát trajektórií, rôznorodé a neštruktúrované dáta ako napríklad text, zvuk, obrázok či video. Ďalším zásadným rozdielom je to, že majú slabú časopriestorovou kontinuitu. Dáta sú zaznamenávané pasívne spustením nejakej udalosti ako napríklad prihlásením sa do sociálnej siete alebo príjmu signálu z veže. To znamená, že body s časopriestorovou informáciou sa zaznamenajú len vtedy, keď nastane príslušná udalosť. Distribúcia zaznamenaných bodov je teda dosť náhodná a časová granularita pomerne veľká. Aj keď sa nejedná o dáta trajektórií, ktoré si bežne pod týmto pojmom predstavíme, tak na základe ich spracovaniu je možné také dáta získať.

   Na základe dátových zdrojov je možné implicitné dáta trajektórií klasifikovať do nasledujúcich troch skupín:

   - **Senzorové dáta** – dáta zo senzorov sa zaznamenávajú vtedy, keď objekt prechádza okolo daného senzoru. Vzhľadom k tomu, že senzory majú pevnú polohu a sú aktívne len na krátke vzdialenosti, tak rozsah získania dát je pomerne malý, ale presnosť polohy je vysoká.
   - **Signálové dáta** – aby bolo možné získavať dáta na základe signálu, je potrebné aby boli zdroje signálu (GSM, WiFi, Bluethooth, RFID atď.) vopred distribuované na rôznych miestach a zároveň sa vyžadujú aj prijímacie zariadenia ako napríklad mobilný telefón. Rozsah získania signálových dát je pomerne široký, ale presnosť polohy je, žiaľ, nízka.
   - **Sieťové/webové dáta** – v súčasnosti sú sociálne siete (Facebook, Twitter, Instagram atď.) pomerne rozšírené a vybavené funkciami geotagov. Sieťové dáta však okrem časopriestorových dát poskytujú mnoho ďalších sémantických informácií o rôznych udalostiach, ľudských aktivitách atď. Preto sa v dátach nachádza pomerne dosť veľa šumu a ich predspracovanie býva zložité.

Vďaka neustálemu pokroku v technológiách je dnes k dispozícii nespočetné množstvo trajektórií predstavujúcich rôzne pohybujúce sa objekty. Podla [27] môžeme mobilitu klasifikovať do nasledujúcich štyroch hlavných kategórií:

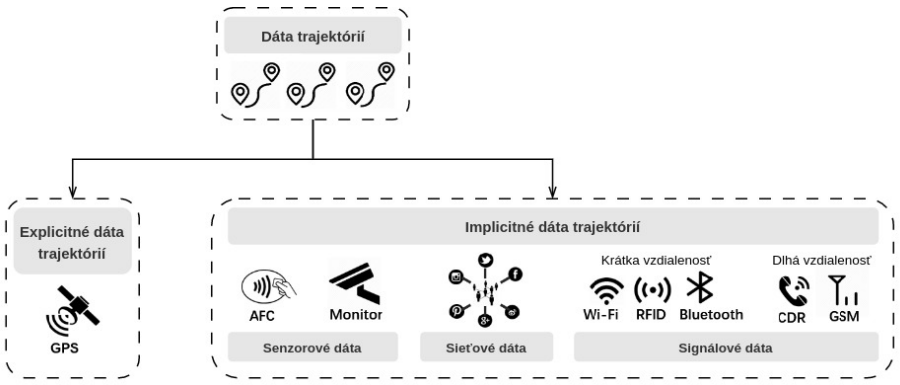
1. **Mobilita ľudí** – v súčasnosti väčšina populácie vlastní mobilný telefón, ktorý je významným zdrojom dát trajektórii, keďže obsahuje rôzne technológie pripojenia – WiFi, Bluetooth a hlavne už spomenuté GPS. Trajektórie môžu byť zaznamenávané

aktívnou alebo pasívnou formou. Medzi aktívne formy patrí napríklad užívateľom zaznamenaná športová aktivita vrátane trasy za účelom neskoršej analýzy. Toto umožňuje napríklad aplikácia Strava[1], kde si užívateľ môže zaznamenávať trajektórie behu, cyklistiky a iných športových aktivít. Užívateľ ale tiež generuje mnoho trajektórií pasívne už len tým, že so sebou nosí a využíva svoj mobilný telefón. Napríklad spoločnosť Google pasívne zbiera informácie o užívateľoch. V prípade, že s tým užívateľ súhlasí, mu spoločnosť zašle každý mesiac súhrn toho, kde všade sa pohyboval, aké mestá navštívil či dokonca koľko kilometrov prešiel a ďalšie podobné informácie.

2. **Mobilita dopravných prostriedkov** – v dnešnej dobe existuje veľké množstvo vozidiel vybavené senzorom GPS a ich počet stále narastá. Takto získané údaje môžu pomôcť napríklad pri predikcii či analýze dopravnej situácie.

3. **Mobilita zvierat** – vďaka rôznym sledovacím zariadeniam je dnes možné tiež sledovať mobilitu zvierat a zhromažďovať ich trajektórie napríklad za účelom štúdia ich migrácie alebo chovania.

4. **Mobilita prírodných javov** – medzi trajektórie prírodných javov patria napríklad trajektórie tornád, hurikánov či morských prúdov. Tento druh trajektórií teda zachytáva zmenu prostredia a klímy, na základe ktorej sú vedci schopní lepšie sa vysporiadať s prírodnými katastrofami a celkovo chrániť prírodné prostredie.



Obr. 3.1: Kategórie dát trajektórií (upravené a prevzaté z [16]).

3.2 Predspracovanie

V súčasnosti sa generuje pomerne veľké množstvo trajektórií, ale nie vždy sú dáta presné. Zaznamenané body môžu byť aj niekoľko metrov nepresné alebo môže dôjsť k rôznym iným odchýlkam v meraní. Z tohto dôvodu je zvyčajne nutné dáta pred vykonaním dolovacej úlohy predspracovať. Spadajú sem techniky ako napríklad eliminácia šumu, redukcia bodov, segmentácia alebo mapovanie, ktorým sa bude táto podkapitola venovať.

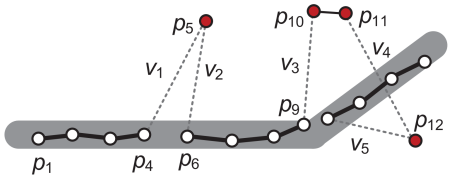
3.2.1 Eliminácia šumu

Tak ako všetky druhy dát, tak aj dáta trajektórií nie sú nikdy úplne presné a obsahujú šum alebo inak povedané obsahujú body, ktoré vybočujú. Šum môže byť spôsobený napríklad

[1] https://www.strava.com/mobile

slabým signálom alebo nesprávnym meraním hodnôt zo senzoru. V niektorých prípadoch nie je filtrovanie šumu potrebné, pretože sa chybne nameraná hodnota bodu opraví napríklad pri mapovaní, ktorému sa bude venovať sekcia 3.2.3. V prípade, že je eliminácia šumu potrebná, tak existuje viacero metód, ktoré spadajú do nasledujúcich troch hlavných kategórií:

1. **Priemer a medián** – využitie priemeru alebo mediánu prechádzajúcich bodov je jedným z najjednoduchších spôsobov ako vyhladiť šum. Priemerový/mediánový filter je možné si predstaviť ako posuvné okno, ktorého veľkosť sa určuje podľa potrieb aplikácie. Veľkosť okna zároveň určuje počet predchádzajúcich bodov, ktoré budú následne využité na výpočet priemeru/mediánu. Využitie priemeru je praktickejšie pri manipulácii s jednotlivými chybne nameranými bodmi, ako je napríklad bod p_5 na obrázku 3.2. Medián ale vychádza lepšie v prípade, že sa jedná o body, ktoré ležia príliš ďaleko od trajektórie. V prípade, že sa pracuje s viacerými chybne nameranými bodmi idúcimi po sebe, je potrebné väčšie posuvné okno. Tento prípad predstavujú body p_{10}, p_{11} a p_{12} na obrázku 3.2. Väčší počet bodov použitých pri výpočte priemeru/mediánu znamená nielen náročnejší výpočet, ale aj vyššiu chybovosť medzi vypočítanou hodnotu a skutočnou polohou bodu.



Obr. 3.2: Chybne namerané body trajektórie (upravené a prevzaté z [27]).

2. **Kalmanov a časticový filter** – oba sa skladajú z dvoch modelov – model merania šumu a dynamický model trajektórie. Kalmanov filter eliminuje chybne namerané hodnoty bodov využitím odhadu nasledujúcej hodnoty. K danému odhadu môže využívať niekoľko hodnôt ako napríklad vzdialenosť či rýchlosť, ale hlavne zohľadňuje zákony fyziky ako je napríklad gravitácia. Časticový filter funguje podobne, ale je viac všeobecnejší a teda je náročnejší na výpočet.

3. **Detekcia odľahlých hodnôt** – vyššie uvedené filtre nahrádzajú chybne nameranú hodnotu bodu nejakou odhadnutou hodnotou. Metódy tejto kategórie hodnoty nenahrádzajú, ale chybne namerané body z trajektórie úplne odstránia. Algoritmy detekcie odľahlých hodnôt fungujú na podobnom princípe. Spočítavajú sa rôzne hodnoty ako napríklad vzdialenosť alebo rýchlosť medzi bodmi a následne sa hodnoty porovnajú s určitým prahom. V prípade, že hodnoty presahujú prah, sú detegované ako odľahlé hodnoty a odstránené z trajektórie. Odlišujú sa hlavne v tom, aké hodnoty a ako sa počítajú, napríklad existuje viacero algoritmov, ktoré pracujú s vzdialenosťou, ale každá z nich využíva inú vzdialenostnú funkciu.

   Ako už ale bolo spomenuté v sekcii 2.3.5, tak niekedy sú práve odľahlé hodnoty zaujímavé a z detekcie odľahlých hodnôt sa stáva typ dolovacej úlohy a nielen časť predspracovania dát. Detekcii odľahlých hodnôt v dátach trajektórií, ako typu dolovacej úlohy, sa detailne venuje nasledujúca kapitola 4.

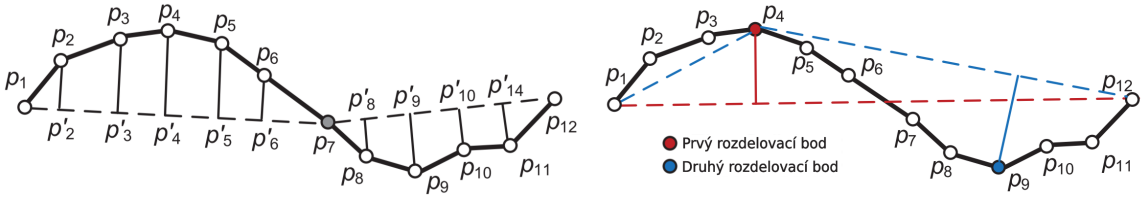
Detailnejší popis jednotlivých metód z vyššie uvedených kategórií je možné nájsť v [33].

3.2.2 Redukcia bodov

Vzhľadom k tomu, že v dnešnej dobe sa vo svete generuje obrovské množstvo trajektórií, je občas nutné dáta redukovať, alebo inak povedané, komprimovať. Komprimovať trajektóriu znamená obmedziť počet bodov trajektórie za účelom zníženia réžie pri spracovaní a ukladaní dát a to tak, aby sa zachovala dátová hodnota trajektórie alebo zmenila len minimálne. V prípade, že by sa spracovávali a ukladali úplne všetky body trajektórie, tak by také množstvo dát nemusel výpočtový systém zvládnuť. Podľa [27] je možné kompresné metódy založené na tvare trajektórií kategorizovať do dvoch nasledujúcich skupín:

1. **Offline/dávková kompresia dát** – pracuje s celou trajektóriou až po jej úplnom vygenerovaní. Cieľom týchto algoritmov je vygenerovať približnú trajektóriu a to tak, že vyradia niektoré body so zanedbateľnou zmenou v pôvodnej trajektórii.

   Patrí sem napríklad algoritmus *Douglas-Peucker* [6], ktorého myšlienkou je nahradiť pôvodné trajektórie približnými segmentami. V prípade, že náhradný segment nespĺňa prah chyby, tak algoritmus rekurzívne delí trajektóriu ďalej a to v bode, ktorý prispieval najväčšou chybou. Tento proces sa opakuje až do chvíle, kým nie je chyba medzi približnou a pôvodnou trajektóriou pod vopred špecifikovaným prahom chyby. Algoritmus je znázornený na obrázku 3.3, kde je ako prah použitá kolmá euklidovská vzdialenosť. Prvý rozdeľovací bod p_4 nesplnil prah chyby a preto bol pridaný ďalší rozdeľovací bod p_9.

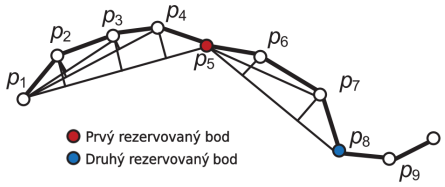


Obr. 3.3: Kolmá euklidovská vzdialenosť a Douglas-Peucker (upravené a prevzaté z [27]).

   Ďalším algoritmom, ktorý spadá do tejto kategórie je *Bellmanov algoritmus* [2], ktorý je vlastne vylepšením algoritmu Douglas-Peucker. Zaisťuje, že delenie je optimálne, teda najlepšie možné a to pomocou techniky dynamického programovania.

2. **Online redukcia dát** – pracuje s trajektóriou okamžite v momente, keď sa objekt pohybuje. Keďže v súčasnosti sa nepretržite generuje veľké množstvo trajektórií v reálnom čase, tak prirodzene vznikla aj potreba techník, ktoré redukujú dáta trajektórií už v priebehu ich generovania. Patrí sem napríklad algoritmus *Sliding Window* [14], ktorého myšlienkou je postupné pridávanie bodov trajektórie do zväčšujúceho sa posuvného okna so segmentom trajektórie. A ďalej zväčšovať toto okno a jeho segment až kým chyba nepresiahne určitý prah. Po presiahnutí prahu sa prvý bod z posuvného okna zapíše do výslednej trajektórie a posledný bod sa stáva prvým bodom nového okna. Algoritmus je možné vidieť na obrázku 3.4. Medzi ďalšie algoritmy tejto kategórie patria *Before Open Window*, *Normal Opening Window* alebo *Reservoir sampling*, ktoré sú detailnejšie popísané v [33].

Algoritmy spomenuté vo vyššie uvedených kategóriach sú všetky založené na vzdialenosti, ale existujú aj ďalšie spôsoby redukcie dát, ktoré sú napríklad závislé na rýchlosti alebo na smere pohybujúceho sa objektu. Podľa [27] ich môžme zaradiť do kategórie online redukcií a v [33] sa nachádza ich detailnejší popis.

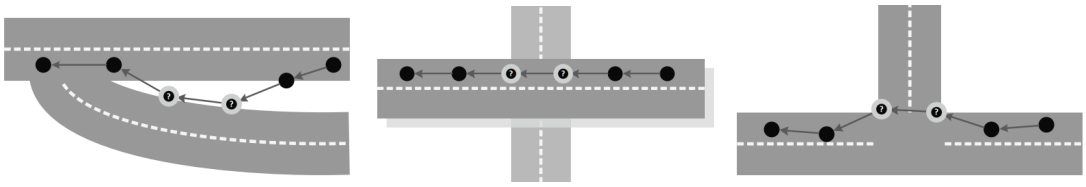


Obr. 3.4: Sliding Window algoritmus (upravené a prevzaté z [27]).

3.2.3 Map matching/mapovanie

Ako už bolo spomenuté, tak dáta pravdepodobne nikdy nebudú dokonalé a vždy budú obsahovať rôzne chyby a odchýlky v meraní. V súčasnosti je ale pre mnohé aplikácie potrebné poznať body, po ktorých sa objekt pohyboval respektíve stále pohybuje, čo najpresnejšie. Vhodným príkladom je navigačná aplikácia, tá bez znalosti toho, na ktorej ceste sa nachádza vozidlo nebude schopná nasmerovať vodiča tak, aby sa vyhol zápcham či iným problémom na trase. Body môžu byť aj niekoľko metrov nepresné a teda ich súradnice sa nemusia zhodovať so súradnicami reálnej cestnej siete. Z tohto dôvodu začali vznikať techniky mapovania bodov na reálnu cestu.

Mapovanie je teda proces, pri ktorom sú body trajektórie priradené k súradniciam reálnej cestnej siete, inak povedané k mape. Mapovanie je pomerne zložitý proces a existuje hneď niekoľko problémov pri priraďovaní bodov, ktoré môžu spôsobiť nejednoznačnosť. Niektoré z nich sú znázornené na obrázku 3.5, kde problém nastáva práve pri bodoch so sivými obrysmi.



(a) Paralelne cesty (b) Nadjazd (c) Ostrovček

Obr. 3.5: Niektoré z problémov, ktoré môžu nastať pri mapovaní (prevzaté z [33]).

Aj pri mapovaní je možné dané algoritmy kategorizovať do dvoch režimov a to **online** a **offline** režimu. Pri online režime je mapovanie vykonávané nepretržite a spracováva sa streamingovým spôsobom. Pri offline režime sa mapovanie vykonáva až po získaní celej trajektórie. Online algoritmy sú však žiadanejšie, pretože väčšina inteligentných dopravných služieb a aplikácií vyžaduje práve spracovanie v reálnom čase. Naprieč rôznymi zdrojmi existuje niekoľko kategorizácií techník pre mapovanie. Podľa [27] a [33] je možné ich kategorizovať do nasledujúcich štyroch skupín:

1. **Geometrické** – jedná sa o jednoduchú techniku, kde je bod pridaný k najbližšej ceste. Vzhľadom k tomu, že body sú chybné aj niekoľko desiatok metrov a existuje viacero ciest pomerne blízko seba, tak táto technika často zlyháva.

2. **Topologické** – tieto algoritmy pri priradení bodu berú do úvahy aj prepojenosť a štruktúru cestnej siete. Z popisu vyplýva, že topologické techniky by mohli vyriešiť problémy znázornené na 3.5, čo sa o geometrických povedať nedá.

3. **Pravdepodobnostné** – jedná sa o algoritmy, ktoré zvažujú niekoľko možných ciest a na základe pravdepodobnosti vyberú tú najlepšiu.

4. **Pokročilé** – časom sa začali objavovať aj pokročilejšie techniky, ktoré napríklad zohľadňujú topológiu cestnej siete, šum a zároveň odľahlé hodnoty v dátach trajektórií.

Prehľad existujúcich techník z týchto kategórií je možné nájsť v niektorom zo spomenutých zdrojov. Vzhľadom k tomu, že informačné technológie neustále napredujú, tak aj pre mapovanie vznikajú stále nové riešenia a preto podľa [4] už vyššie uvedená kategorizácia nie je úplne aktuálna a navrhujú novú, ktorá je založená na základe mapovacích modelov a pracovných scenárov danej techniky. Jedná sa o nasledujúce štyri modely, ktorých detailnejší popis je uvedený v [4]:

- **podobnostný model** (angl. *similarity model*)
- **model prechodov medzi stavmi** (angl. *state-transition model*)
- **model vývoja kandidátov** (angl. *candidate-evolving model*)
- **model založený na skóre** (angl. *scoring model*)

3.2.4 Segmentácia

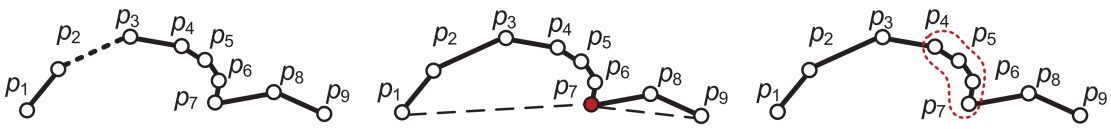
Ďalšou technikou využívanou pri predspracovaní dát je segmentácia, ktorá spočíva v rozdelení trajektórie na sub-trajektórie. Každú jednu sub-trajektóriu je možné nazvať aj segmentom, oddielom alebo rámcom trajektórie, ale v tejto práci bude vždy označovaná ako segment. Segmentácia má niekoľko pozitívnych prínosov. Jedným z nich je zníženie výpočtovej zložitosti algoritmu. Ďalším je možnosť získania znalostí o trajektórií, ktoré nie je možné vidieť z pohľadu celej trajektórie, ako napríklad nájdenie vzorov sub-trajektórií.

Segmentačných metód je pomerne dosť a vlastne každá metóda dolovania z dát, v závislosti na dolovacej úlohe, využíva iný druh segmentácie alebo ich upravené formy. Podľa [27] je vo všeobecnosti možné rozdeliť segmentačné metódy na nasledujúce tri typy:

1. **Založené na časovom intervale** – trajektóriu je možné rozdeliť na segmenty s rovnakou časovou dĺžkou alebo je možné stanoviť prah časového intervalu medzi dvoma po sebe idúcimi bodmi a trajektória sa rozdelí v prípade, že bude daný prah prekročený. Na obrázku 3.6a je znázornený príklad, kde medzi bodmi p_2 a p_3 došlo k prekročeniu prahu a preto sa trajektória rozdelí na dve segmenty a to segment obsahujúci body p_1 a p_2 a na segment obsahujúci body p_3 až p_9.

2. **Založené na tvare trajektórie** – prvým spôsobom je trajektóriu rozdeliť v takzvaných otočných bodoch, v ktorých sa očividne zmení smer trajektórie a to, či sa jedná o zmenu smeru, sa určí na základe daného prahu. Tento príklad je uvedený na obrázku 3.6b. Ďalej sem spadajú metódy, ktoré identifikujú kľúčové body, v ktorých bude trajektória rozdelená na segmenty. Na identifikáciu kľúčových bodov je napríklad možné využiť aj algoritmus na redukciu bodov *Douglas-Peucker*, ktorý bol bližšie popísaný v 3.2.2. Algoritmus popísaný v [18], ktorý pracuje na princípe rozdelenia trajektórie, využíva minimálnu dĺžku popisu (anlg. *Minimum Description Length*, skr. *MDL*), nájde zoznam charakteristických alebo, inak povedané, kľúčových bodov, ktorými je trajektória následne rozdelená na jednotlivé segmenty.

3. **Založené na sémantickom význame** – trajektóriu je možné rozdeliť na základe sémantického významu bodov ako napríklad na základe bodov zotrvania. Tento príklad je zobrazený na obrázku 3.6c, kde je vidieť že body p_4 až p_7 sú identifikované ako body zotrvania a preto sa trajektória rozdelí na segment bodov p_1 až p_3 a segment bodov p_8 a p_9. To, či budú body zotrvania úplne odstránené, alebo sa s nimi bude nejako pracovať, už záleží od konkrétnej metódy a aplikácie. Ďalším sémantickým významom, podľa ktorého je trajektória rozdelená na segmenty, môže byť napríklad spôsob dopravy. Viac informácií nielen o tomto type segmentačných metód je možné nájsť v [27].



(a) Časový interval (b) Otočný bod (c) Body zotrvania

Obr. 3.6: Rôzne typy segmentačných metód trajektórií (prevzaté z [27]).

3.3 Typy dolovacích uloh

Dáta trajektórií môžu obsahovať zložitejšie údaje. Z toho dôvodu všeobecné metódy dolovania z dát nemusia byť efektívne. Na začiatku dolovania časopriestorových dát sa skúsili využiť už existujúce metódy pre dolovanie v priestorových alebo temporálnych dátach. Tieto metódy síce môžu odhaliť zaujímavé vzory či modely, ale pozerajú sa na časovú a priestorovú zložku zvlášť a teda nie sú schopné odhaliť zložitejšie vzťahy, ktoré je možné dolovaním časopriestorových dát získať. Z tohto dôvodu začali a dodnes vznikajú nové metódy dolovania v dátach trajektórie.

Typy dolovacích úloh sa od všeobecných veľmi neodlišujú a teda existuje napríklad časopriestorové zhlukovanie, časopriestorová klasifikácia a predikcia, dolovanie asociačných vzorov, dolovanie frekventovaných pohybov objektov či detekcia odľahlých hodnôt. Niektoré z nich sú popísané v tejto podkapitole. Detekcii odľahlých hodnôt, teda zameraniu tejto diplomovej práce, je venovaná celá nasledujúca kapitola 4.

3.3.1 Dolovanie vzorov

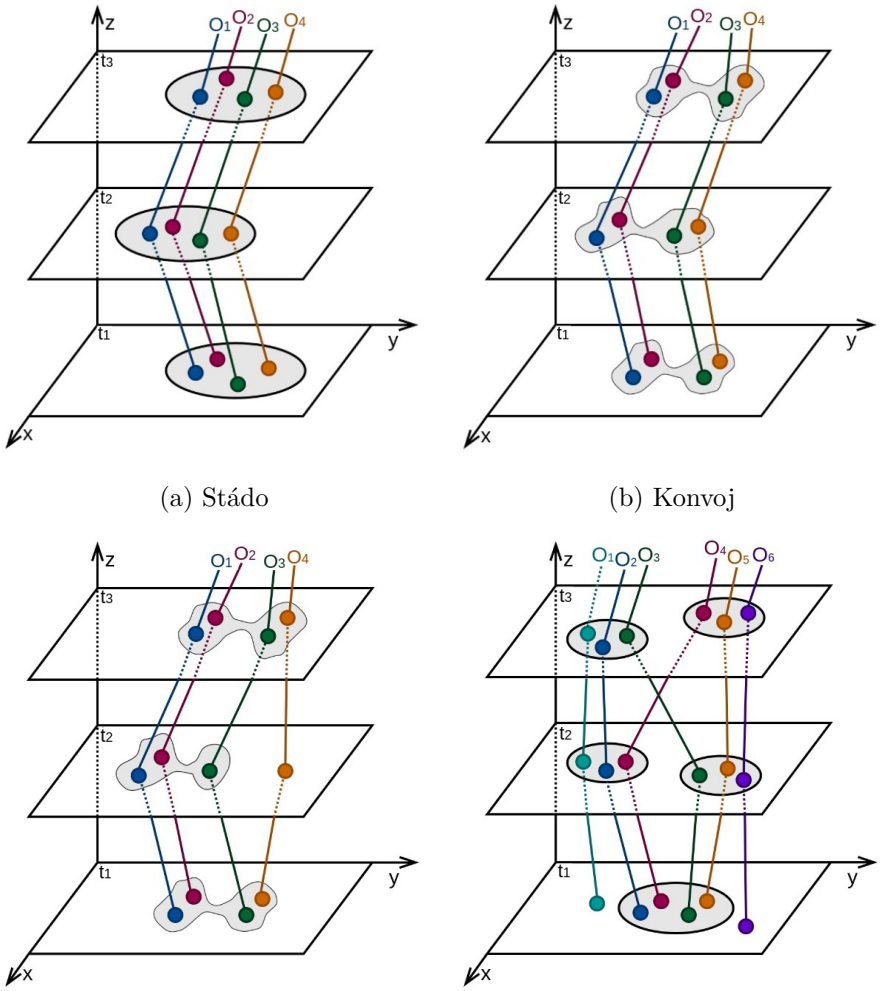
Dolovanie vzorov je jednou z najzákladnejších úloh dolovania dát, pri ktorej je možné objaviť pomerne veľké množstvo zaujímavých, významných a tiež neočakávaných vzorov. Je možné dolovať pomerne veľkú škálu rôznych vzorov, ako napríklad frekventované, sekvenčné, topologické či periodické vzory, podgrafy alebo asociácie. Dolovanie vzorov môže slúžiť pre analýzu mobility pohybujúceho sa objektu ale tiež viacerých objektov spolu. Rôzne druhy vzorov zodpovedajú rôznym druhom algoritmov, ktoré sa podľa [23] delia do nasledujúcich troch skupín:

1. **Dolovanie frekventovaných vzorov** – sa zameriava na špecifické trajektórie, po ktorých sa často pohybovalo viacero objektov. Tieto vzory je možné vydolovať pomerne jednoduchým spôsobom a to za pomoci priestorových vlastností trajektórie. Je možné využiť napríklad zhlukovanie alebo štatistickú analýzu. Pri niektorých úlohách teda postačuje tento jednoduchý spôsob, kde sa sleduje len priestorová sekvencia.

Existujú ale aj úlohy, kde je potrebné sa zamerať na objekty, ktoré sa pohybujú cez rovnaké miesta v približne rovnakom čase, teda nie je možné zanedbať časovú zložku trajektórie. Dolovanie vzorov z časopriestorovej sekvencie už nie je také jednoduché a zvyčajne je potrebné využiť viackrokový prístup.

2. **Dolovanie periodických vzorov** – sa zameriava na trajektórie, ktoré pohybujúci sa objekt periodicky vykonáva. Pohybujúce sa objekty majú zvyčajne pomerne dosť periodické aktivity, ako napríklad každoročná migrácia zvierat z jedného miesta na druhé alebo ľudia, ktorí chodia nakupovať každú sobotu. Vďaka vydolovaným periodickým vzorom je možné napríklad predpovedať budúci pohyb objektu.

3. **Dolovanie kolektívnych vzorov** – sa zameriava na skupiny objektov, ktoré sa pohybujú spolu počas určitého časového obdobia. Zohľadňuje teda priestorovú aj časovú zložku a je vlastne kombináciou predchádzajúcich dvoch skupín. Existuje niekoľko rôznych typov pohybujúcich sa zhlukov objektov, ako napríklad stádo, konvoj, roj a zhromažďovanie, ktoré sú znázornené na obrázku 3.7.



(a) Stádo (b) Konvoj

(c) Roj (d) Zhromažďovanie

Obr. 3.7: Rôzne typy kolektívnych vzorov (upravené a prevzaté z [23, 27]).

- Stádo (angl. *flock*) – jedná sa o skupinu objektov, ktoré sa spolu pohybujú minimálne dané časové obdobie. Stádo je možné vidieť na obrázku 3.7a.
- Konvoj (angl. *convoy*) – je podobný stádu, ale každý objekt, ktorý doň spadá, musí byť schopný sa spojiť s iným cez ostatné. Vzdialenosť medzi dvoma najbližšími objektami nemôže byť menšia než vopred určený prah. Konvoj je znázornený na obrázku 3.7b.
- Roj (angl. *swarm*) – je podobný konvoju, ale dovoľuje, aby sa objekt na nejaký čas odklonil od roja. Naopak, pri stáde a konvoji musia byť všetky objekty po celú dobu cesty spolu. Roj je zobrazený na obrázku 3.7c.
- Zhromažďovanie (angl. *gathering*) – detekuje zhluky objektov, ktoré majú danú veľkosť a sú si priestorovo blízke. Tieto zhluky objektov sa môžu neustále meniť. Zhromažďovanie je možné vidieť na obrázku 3.7d.

Podľa [27] existuje ešte štvrtá skupina a to zhlukovanie trajektórií, ale v tejto práci, tak ako vo väčšine využitých zdrojov, sa zhlukovaniu venuje samostatná sekcia 3.3.3. Detailnejší, ale stále pomerne stručný popis alebo prehľad konkrétnych prác týkajúcich sa dolovania vzorov sa nachádza v [9, 23, 27]. Pomerne dosť detailný popis dolovania vzorov v časopriestorových dátach je uvedený v [12].

3.3.2 Časopriestorová klasifikácia a predikcia

Časopriestorová klasifikácia a predikcia spadajú do skupiny prediktívnych dolovacích úloh. Ako už bolo spomenuté v sekcii 2.3.3, proces klasifikácie a predikcie je podobný, ale odlišujú sa v tom, aké hodnoty predikujú. Zatiaľ čo pri klasifikácii sa predikujú hodnoty diskrétne a nezoradené, tak pri predikcii ide o hodnoty spojitého charakteru.

Klasifikácia trajektórií

Klasifikácia trajektórií sa zameriava na rôzne odlišnosti medzi trajektóriami alebo ich segmentmi ako napríklad spôsob dopravy či rôzne druhy ľudských aktivít spojených s pohybom. Podľa [27] klasifikácia trajektórií pozostáva z nasledujúcich troch krokov:

1. **Rozdelenie trajektórie** na segmenty. Segment môže obsahovať len jeden alebo niekoľko po sebe idúcich bodov.
2. **Extrakcia vlastností** z každého segmentu.
3. **Vytvorenie modelu** na klasifikáciu segmentov.

Trajektória je v podstate sekvencia a preto je možné využiť pri vytváraní modelu už známe techniky. V [27] sú napríklad uvedené skryté Markovové modely alebo dynamické Bayesovské siete. Podľa [9] je možné pri klasifikácií trajektórií využiť algoritmy založené na najbližšom susedstve a to za predpokladu, že je poskytnutá vhodná vzdialenostná funkcia. Vzdialenostná funkcia závisí od danej klasifikačnej úlohy a nie je jednoduché ju určiť.

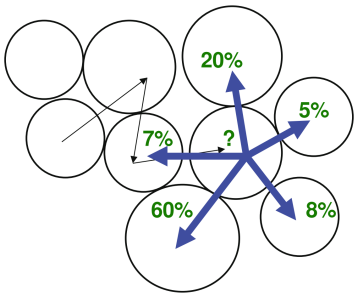
Klasifikácia trajektórií alebo jej sub-trajektórií umožňuje získať a pridávať k trajektóriám pomerne širokú škálu metadát. V [28] a [30] autori klasifikujú používateľa na základe spôsobu dopravy – jazda autom, autobusom, bicyklovanie a chôdza. V [34] je zas cieľom zistiť stav taxíka – obsadený, neobsadený a zaparkovaný. Ďalšie informácie o klasifikácií trajektórií a príklady sú uvedené v [9, 23, 27].

Predikcia

Časopriestorové dáta ponúkajú pomerne širokú škálu prediktívnych úloh a okrem už spomenutej klasifikácie trajektórií sem patria aj rôzne druhy predikcie. Je možné predikovať niekoľko vlastností časopriestorových dát ako napríklad polohu, hustotu, udalosti alebo trajektórie. Podľa [9] sa predikcia u časopriestorových dát delí na nasledujúce skupiny:

1. **Predikcia polohy a trajektórie** – na základe dát trajektórií je možné predpovedať budúcu polohu pohybujúceho sa objektu alebo predpovedať celú jeho trasu v kontexte cestnej siete. Existuje niekoľko prístupov k predikcii polohy. Jedným z nich je predikcia budúcej polohy pohybujúceho sa objektu na základe jeho súčasnej pozície a vektora rýchlosti pohybu [9]. Ďalším prístupom je predikcia polohy na základe analýzy historických dát trajektórií pohybujúcich sa objektov a odvodenie vzorov z nich. Existujú ešte ďalšie prístupy ako napríklad model Markovovho reťazca [23]. Na obrázku 3.8 je znázornený príklad predikcie budúcej polohy pohybujúceho sa objektu na základe pravdepodobnosti pohybu daných zhlukov.

   K predikcii celej trasy alebo cieľovej pozície objektov je možné využiť niektoré obdobné prístupy, ako napríklad predikcia založená na pozorovaní trajektórií a odvodení vzorov alebo prístup založený na Markovovom modeli. Pri týchto prístupoch predikcie polohy a tiež trajektórie sa predpokladá, že sa objekt pohybuje po niektorej z častých trás ako napríklad cesta do práce.



Obr. 3.8: Predikcia budúcej polohy (prevzaté z [9]).

   Detailnejší popis a prehľad štúdii v oblasti predikcie polohy a trajektórie sa nachádza v [9, 23].

2. **Predikcia hustoty** – hustota objektov v danej oblasti je definovaná ako pomer počtu objektov vo vnútri oblasti k veľkosti oblasti v danom časovom bode. Predikciu hustoty je možné využiť v rôznych oblastiach. Jednou z nich sú dopravné systémy, kde je vďaka nej možné odhaliť napríklad blížiacu sa prekážku.

3. **Predikcia dosahu** – dosah je časovo závislá miera, ktorá vyjadruje publicitu miesta v rámci určitej populácie. Nemusí byť obmedzený len na jedno miesto, ale môže sa jednať o dosah pre sieť miest. V takom prípade je dosah definovaný ako podiel populácie, ktorá prejde aspoň jedným z miest siete v danom časovom období.

4. **Predikcia udalostí** – ako už sám názov napovedá jedná sa o predikciu toho, že v danej oblasti a časovom období nastane nejaká udalosť. Predikuje sa napríklad na základe zodpovedajúcich historických dát. Napríklad v [3] autori predpovedajú pravdepodobnosť, že bude v danej oblasti a časovom období spáchaný trestný čin a to na základe miesta, času a sociálno-ekonomických charakteristík minulých incidentov.

3.3.3 Časopriestorové zhlukovanie

Časopriestorové zhlukovanie je proces rozdelenia trajektórií do zhlukov na základe vopred definovaného kritéria ich podobnosti. Inak povedané, trajektórie by mali byť v rámci jedného zhluku čo najviac podobné, zatiaľ čo medzi jednotlivými zhlukmi by sa mali odlišovať. Napríklad, vďaka zhlukovaniu trajektórií, je možné zistiť trasy, po ktorých sa pohybuje pomerne veľké množstvo ľudí, ale nie sú dostatočne pokryté verejnou dopravou. Na obrázku 3.9 je znázornené zhlukovanie trajektórií, kde každý zhluk je reprezentovaný jednou hlavnou trajektóriou. Podľa [9] je pre zhlukovanie trajektórí možné využiť nasledujúce dva prístupy:

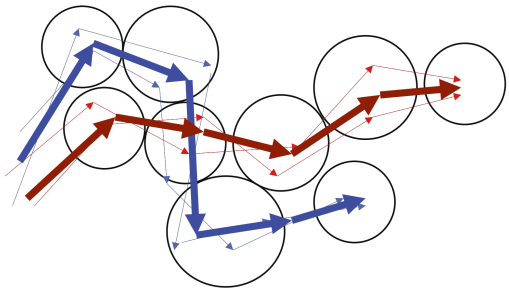
1. **Zhlukovanie trajektórií založené na vzdialenosti**

   Trajektórie pohybujúcich sa objektov je možné zhlukovať pomocou už existujúcich všeobecných zhlukovacích algoritmov. Pre určenie, ktoré objekty majú spadať do rovnakého zhluku, sa využívajú vzdialenostné funkcie. Je nutné definovať vhodnú vzdialenostnú funkciu pre danú úlohu, môže sa jednať napríklad o Euklidovskú, Manhattanskú alebo Hausdorffovu vzdialenostnú funkciu, ktorú využívajú napríklad autori v [18]. Sémantika dát trajektórii je tu úplne zapuzdrená vo vzdialenostnej funkcii.

   Podľa [23] je všeobecným prístupom k zhlukovaniu vytvorenie vektora vlastností trajektórie a meranie vzdialeností medzi týmito vektormi. Vytvoriť taký vektor, ktorý bude dobre reprezentovať trajektóriu a zároveň bude porovnateľný s ostatnými vektormi, je pomerne zložité, pretože trajektórie môžu mať rôzny tvar, dĺžku, frekvenciu bodov a podobné premenlivé vlastnosti.

2. **Zhlukovanie prispôsobené trajektóriám**

   V tomto prípade sa využívajú na zhlukovanie nové algoritmy, ktoré sú prispôsobené konkrétnemu typu dát. Podľa [23] sú mnohé z týchto algoritmov prispôsobené štatistickým alebo pravdepodobnostným modelom na meranie charakteristík trajektórií, ako napríklad skrytý Markovov model, ktorý bol využitý v [1]. Na rozdiel od predchádzajúceho prístupu, v tomto prípade je sémantika dát trajektórií využitá v celom zhlukovacom algoritme. Prehľad konkrétnych techník je uvedený v [9, 23, 27]



Obr. 3.9: Zhlukovanie trajektórií (prevzaté z [9]).

Definícia zhlukovania vyvoláva otázku, čo ak je objavený taký objekt, ktorý dobre nezapadá ani do jedného z nájdených zhlukov. Takýto objekt je možné nazvať odľahlým a buď predstavuje nechcený šum, ktorý je možné odstrániť pri predspracovaní dát alebo sa stáva zaujímavým pre dolovanie. Detekcii odľahlých hodnôt v dátach trajektórií, ako typu dolovacej úlohy, sa detailne venuje nasledujúca kapitola 4.

Kapitola 4

Detekcia odľahlých trajektórií

Detekcia odľahlých hodnôt a neobvyklého či podozrivého chovania pohybujúceho sa objektu je veľmi dôležitá a používa sa v mnohých oblastiach, ako je napríklad monitorovanie zvierat, prírodných javov alebo riadenie dopravy. Metódy detekcie odľahlých hodnôt v závislosti na druhu aplikácie by mali pracovať s rôznymi súbormi dát, ako sú napríklad video dáta, streamové dáta alebo aj dáta trajektórie.

Detekcia odľahlých hodnôt v dátach trajektórií má za úlohu odhaliť také trajektórie alebo sub-trajektórie, ktoré sa odlišujú od väčšiny ostatných trajektórií z hľadiska určitej miery podobnosti. Trajektórie sa môžu líšiť nielen v tom, že majú značne odlišnú trasu, ale tiež napríklad v rýchlosti alebo smere pohybujúceho sa objektu. Podľa [21] je možné odľahlú hodnotu trajektórie rozdeliť do nasledujúcich skupín:

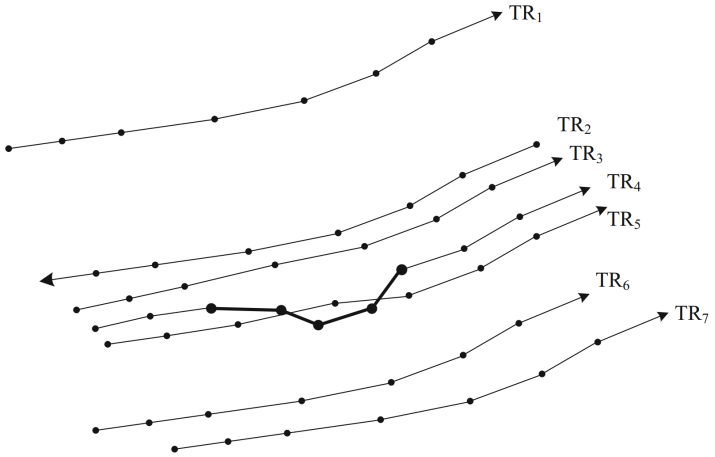
1. **Odľahlá trajektória založená na vzdialenosti** je taká trajektória, od ktorej je vzdialených p trajektórií a to vo väčšej vzdialenosti než je D od trajektórie. Parameter p je počet trajektórií a D je parameter prahu vzdialenosti, ktoré sú zvyčajne vstupom do konkrétnej metódy detekcie. Na obrázku 4.1 je vidieť, že TR_1 je pomerne ďaleko od TR_6 a TR_7 a v prípade ak by parameter p bol nastavený na 2, tak by bola TR_1 detegovaná ako odľahlá trajektória.

2. **Odľahlá trajektória založená na hustote** je taká trajektória, ktorej hustota je menšia ako daný prah *threshold*. Pri predchádzajúcom type odľahlej trajektórie je vždy potrebné zvoliť globálny prah vzdialenosti. V tomto prípade sú odľahlé hodnoty korelované s hustotou susedov, teda je možné ich považovať za lokálne odľahlé hodnoty. Trajektórie TR_6 a TR_7, znázornená na obrázku 4.1, sa z hľadiska vzdialenosti za odľahlé nepovažujú, ale pokiaľ sa jedná o hustotu tak sú obe považované za odľahlé.

3. **Odľahlá trajektória založená na charakteristike** je taká trajektória, ktorá sa výrazne líši od ostatných trajektórii v niektorej z ich charakteristík, ako je napríklad rýchlosť alebo smer. Na obrázku 4.1 je takou trajektóriu TR_2, ktorá sa pohybuje opačným smerom než ostatné trajektórie.

4. **Odľahlá sub-trajektória** je taká sub-trajektória, ktorá je abnormálna z hľadiska čiastočného porovnania trajektórie. Na obrázku 4.1 je vidieť, že časť trajektórie TR_4 je vyznačená hrubšie – jedná sa o odľahlú sub-trajektóriu.

5. **Odľahlá trajektória založená na aktivitách** je taká trajektória, ktorú je možné odhaliť analýzou aktivít pohybujúceho sa objektu vo fyzickom svete.



Obr. 4.1: Rôzne typy odľahlých trajektórií (prevzaté z [21]).

Existuje niekoľko prístupov k detekcii odľahlých trajektórií. Táto kapitola popisuje tri hlavné prístupy, medzi ktoré patria metódy založené na vzdialenosti, ktorým sa venuje podkapitola 4.1. Ďalej sa sem radia metódy založené na hustote, popísané v podkapitole 4.2. Posledným prístupom, na ktorého popis sa zameriava podkapitola 4.3, sú metódy založené na klasifikácii. V závere kapitoly, teda v podkapitole 4.4, je krátky prehľad ďalších metód detekcie odľahlých trajektórií.

4.1 Metódy založené na vzdialenosti

Metódy založené na vzdialenosti sú pomerne známe pri zhlukovaní, kde vedľajším účinkom môže byť práve aj detegovanie odľahlých hodnôt. Jedná sa o objekty, teda trajektórie, ktoré nezapadajú do žiadneho zhluku. Neformálne je odľahlá hodnota založená na vzdialenosti definovaná už v úvode tejto kapitoly. V štúdii [15], ktorá je považovaná za jeden z najskorších výskumov detekcie odľahlých trajektórií, autori zavádzajú koncept odľahlej hodnoty založenej na vzdialenosti. Podľa tejto štúdie je odľahlá hodnota založená na vzdialenosti (*distance-based*) definovaná následovne:

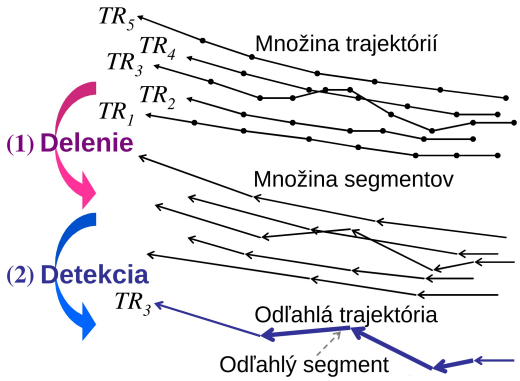
Definícia 4.1. *Objekt O v dátovej sade T je $DB(p, D)$-odľahlý, ak aspoň p objektov z T leží vo väčšej vzdialenosti než D od objektu O.*

Parameter p určuje počet objektov a D je globálny prah vzdialenosti. Vďaka ním je do procesu zapojený aj užívateľ, ktorý tieto parametre poskytuje. Oba parametre sú bližšie popísané pri metóde *TRAOD*, ktorá je významným zástupcom metód založených na vzdialenosti. Jej popis je náplňou tejto podkapitoly. Existujú aj ďalšie metódy a hlavný rozdiel medzi nimi spočíva v tom, akú vzdialenostnú funkciu využívajú.

4.1.1 TRAOD

Metóda *TRAOD* bola vyvinutá na základe nového rámca delenia a detekcia, ktorý navrhli autori v článku *„Trajectory Outlier Detection: A Partition-and-Detect Framework“* [17]. Ako už z názvu vyplýva, tak metóda *TRAOD* sa skladá z dvoch fáz – delenie a detekcia, ktoré sú znázornené na obrázku 4.2. Vo fáze delenia alebo inak povedané fáze segmentácie, sa trajektória delí na segmenty. Následne, vo fáze detekcie, sa detekujú segmenty, ktoré sú

odľahlými sub-trajektóriami. Detekcia odľahlých sub-trajektórii prebieha hlavne pomocou vzdialenosti, ale na zlepšenie kvality sa berie do úvahy aj hustota.



Obr. 4.2: Detekcia odľahlých trajektórií metódou TRAOD (upravené a prevzaté z [17]).

Hlavnou výhodou tejto metódy je práve to, že dokáže detegovať odľahlé sub-trajektórie. Ostatné metódy, ktoré boli navrhnuté pred ňou, mali obmedzenú schopnosť ich detekcie. Napríklad v [15] sa porovnávajú trajektórie ako celok a preto nemusia byť všetky odľahlé trajektórie detegované. Na obrázku 4.1, z úvodu kapitoly, je trajektória TR_4, ktorej hrubá časť zobrazuje odľahlú sub-trajektóriu. Za predpokladu, že trajektórie na obrázku sú dlhšie, metóda z [15] by neobvyklé chovanie trajektórie TR_4 pravdepodobne nezistila. Z dôvodu, že rozdiely sa spriemerujú v rámci celej trajektórie a celkové správanie trajektórie je podobné ako u susedných trajektórii.

Segmentácia

Metóda *TRAOD* má dve verzie – základnú a optimalizovanú. V základnej verzii, aby sa vyhlo prehliadnutiu možných odľahlých hodnôt, sa trajektória rozdelí na základné jednotky, čo sú najmenšie jednotky trajektórie podľa typu dát. Základnou jednotkou môže byť bod, ale môže ich zahŕňať aj viac. Tento pomerne jednoduchý spôsob síce vedie ku kvalitným výsledkom, pretože vyhľadávací priestor má jemnú granularitu, ale čím je tento priestor väčší, tým slabší je výkon algoritmu.

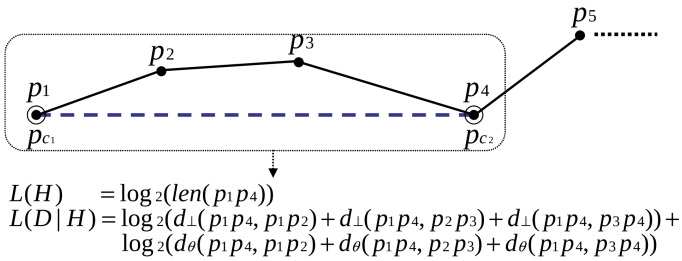
Pre optimalizovanú verziu je nutné využiť sofistikovanejší prístup k segmentácii. Vylepšená verzia *TRAOD* využíva dvoj-úrovňový segmentačný algoritmus, ktorý je súčasťou predchádzajúcej práce [18] autorov metódy *TRAOD*. Na prvej úrovni delenia je každá trajektória rozdelená na hrubé segmenty, z ktorých sa vyberú tie, ktoré sú pravdepodobne odľahlé. Následne na druhej úrovni delenia je každý z vybraných hrubých segmentov rozdelený na jemné segmenty. Vďaka tomu sa zúži vyhľadávací priestor, ktorý je potrebné kontrolovať s jemnou granularitou a tým sa zvýši výkon algoritmu.

1. **Delenie na hrubé segmenty**

   Je založené na minimálnej dĺžke popisu (*MDL*). Cena *MDL* sa skladá z dvoch zložiek a to z $L(H)$ a $L(D|H)$, kde H je hypotéza a D sú dáta. Formálne definície oboch zložiek sú uvedené v [17, 18]. Najlepšia hypotéza H na vysvetlenie D je potom tá, ktorá minimalizuje súčet $L(H)$ a $L(D|H)$.

   V prípade metódy *TRAOD* predstavuje H množinu hrubých segmentov a D zodpovedá trajektórií. Optimálne rozdelenie hrubej granularity, teda znamená nájdenie

najlepšej hypotézy pomocou princípu MDL. Z niekoľkých jemných segmentov sa vytvorí jeden hrubý, tak ako to je znázornené na obrázku 4.3. Pri výpočte, či bude do hrubého vložený ďalší jemný segment je potrebné vypočítať MDL medzi hrubým segmentom a všetkými už pridanými jemnými segmentami a novým pridávaným jemným segmentom. Pri výpočte $L(D|H)$ je na obrázku 4.3 vidieť $d_\perp$ a d_θ. Jedná sa o kolmú a uhlovú vzdialenosť, ktoré budú bližšie popísané v časti detekcie odľahlých hodnôt.



Obr. 4.3: Tvorba hrubého segmentu a výpočet ceny MDL (prevzaté z [18]).

Vzhľadom k tomu, že výpočet podľa ideálnej MDL ceny je výpočtovo náročný, tak je využité aproximačné riešenie, ktoré počíta dve ceny MDL. Jednu normálnu, tzv. MDL_{par}, ktorá sa vypočíta tak, ako je uvedené vyššie. Druhá, tzv. MDL_{nopar}, sa vypočíta len na základe počiatočného a koncového bodu segmentu. V podstate pri výpočte MDL_{nopar} je $L(D|H)$ rovné nule. Ak platí $MDL_{par} \leq MDL_{nopar}$, tak je jemný segment pridaný do hrubého a následne sa pridávajú ďalšie jemné segmenty až kým podmienka neprestane platiť.

2. Delenie na jemné segmenty

V predchádzajúcej úrovni delenia sa o hrubých segmentoch ukladali niektoré informácie. Medzi ktoré patrí minimálna aj maximálna dĺžka jemných segmentov v hrubom segmente a maximálna kolmá a maximálna uhlová vzdialenosť medzi hrubým segmentom a jeho jemnými segmentami. Následne pri porovnaní dvoch hrubých segmentov sa ešte vypočítajú niektoré ďalšie potrebné informácie ako napríklad uhol medzi nimi.

Všetky tieto informácie sú následne využité k výpočtu hornej hranice (*lower bound*) a dolnej hranice (*upper bound*). Hranice slúžia na rozhodnutie, či budú využité jemné segmenty, nebudú alebo bude potreba ich podrobiť výpočtu vzdialenosti. Rozhoduje sa na základe nasledujúcich pravidiel:

- Ak platí $lb(L_i, L_j, distance) > D$, tak sa rozdelenie na jemné segmenty pri porovnaní hrubých segmentov L_i a L_j nevyžaduje.
- Ak platí $ub(L_i, L_j, distance) \leq D$, tak sa rozdelenie na jemné segmenty vyžaduje, ale nie je potrebné počítať vzdialenosť medzi jemnými segmentami v hrubých segmentoch L_i a L_j.

Algoritmus *TRAOD* pri odhalení či v rámci detekcii budú využité jemné segmenty teda využíva informácie, ktoré sa získavajú pri segmentácii. Z toho dôvodu by sa mal na segmentáciu využiť tento dvoj-úrovňový segmentačný algoritmus. Ale v podstate je možné využiť akýkoľvek segmentačný algoritmus vhodný pre trajektórie, ktorý bude ale potrebné upraviť.

Detekcia odľahlých trajektórií

Ako už bolo spomenuté, tak existujú parametre, vďaka ktorým je užívateľ zapojený do procesu detekcie odľahlých trajektórií. V metóde *TRAOD* sa užívateľ stretne s nasledujúcimi parametrami:

- Parameter D je prah vzdialenosti, ktorý určuje, do akej vzdialenosti medzi segmentmi sú dané segmenty považované za blízke.
- Parameter p určuje koľko blízkych trajektórii je potrebné, aby nebol segment považovaný za odľahlý.
- Parameter F je prah odľahlosti segmentov v danej trajektórii, ktorý určí, či je samotná trajektória odľahlá.

Pred vysvetlením samotného algoritmu je potrebné definovať niekoľko pojmov. Na začiatku je definované, ktoré segmenty trajektórie sú považované za blízke voči inej trajektórii.

Definícia 4.2. *Trajektória TR_i je blízka segmentu $L_j \in P(TR_j)$, ak $(TR_i \neq TR_j)$ a $\sum_{L_i \in CP(TR_i, L_j, D)} len(L_i) \geq len(L_j)$, kde D je parameter daný užívateľom, $P(TR_j)$ je množina všetkých segmentov TR_j a $CP(TR_i, L_j, D)$ je množina všetkých segmentov TR_i, ktoré sú vzdialené od L_j menej než je D.*

Ďalej sú uvedené definície odľahlého segmentu a trajektórie.

Definícia 4.3. *Segment L_i je odľahlým, ak je rovnica 4.1 pravdivá.*

$$|CTR(L_i, D)| \leq \lceil (1-p)|\mathcal{I}| \rceil \tag{4.1}$$

kde $\mathcal{I}$ je počet všetkých trajektórií, parameter p je daný užívateľom a $CTR(L_i, D)$ je množina trajektórií blízkych k L_i podľa parametru D zadaného užívateľom.

Definícia 4.4. *Trajektória TR_i je odľahlá, ak je rovnica 4.2 pravdivá.*

$$Ofrac(TR_i) = \frac{\sum_{L_i \in OP(TR_i, D, p)} len(L_i)}{\sum_{M_i \in P(TR_i)} len(M_i)} \geq F \tag{4.2}$$

kde $OP(TR_i, D, p)$ je množina všetkých odľahlých segmentov trajektórie TR_i, $P(TR_i)$ je množina všetkých segmentov trajektórie TR_i a parameter F je daný užívateľom.

V prípade, že dáta obsahujú husté a riedke oblasti, tak vyššie uvedené definície môžu viesť k problémom, pretože neberú hustotu do úvahy. Aby sa problémom predišlo je do riešenia zakomponovaná aj hustota. Následujúce definície popisujú pojmy hustota segmentu a korekčný koeficient.

Definícia 4.5. *Hustota segmentu L_i je definovaná ako rovnica 4.3, ktorá predstavuje počet segmentov vo vzdialenosti σ od segmentu L_i.*

$$density(L_i) = |\bigcup_{TR_j \in \mathcal{I}} CP(TR_j, L_i, \sigma)| \tag{4.3}$$

kde σ je smerodajná odchýlka vzdialenosti medzi dvoma segmentmi, ktorá sa počíta pre všetky dvojice segmentov.

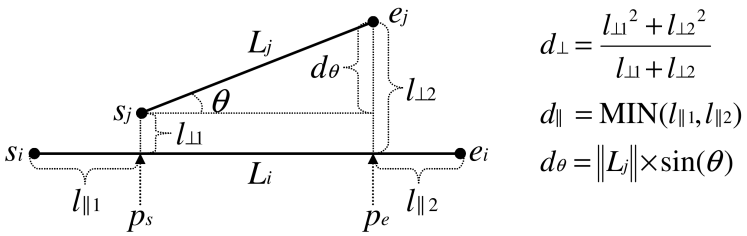
Definícia 4.6. *Korekčný koeficient segmentu L_i je definovaný rovnicou 4.4, ktorá predstavuje pomer priemernej hustoty k hustote segmentu L_i.*

$$adj(L_i) = \frac{\sum_{L_j \in \mathcal{L}} density(L_j)/|\mathcal{L}|}{density(L_i)}, \quad kde \quad \mathcal{L} = \bigcup_{TR_k \in \mathcal{I}} P(TR_k) \tag{4.4}$$

Korekčným koeficientom $adj(L_i)$ násobíme počet blízkych trajektórii $|CTR(L_i, D)|$, čím dosiahneme, že $|CTR(L_i, D)|$ je v hustých oblastiach znížená a naopak v riedkych oblastiach zvýšená. Vďaka tomu je zaistené nie len to, že hustota nemá na výsledok vplyv, ale aj to, že užívateľom zvolené parametre nemajú vplyv na hustotu segmentu.

Pri detekcii odľahlých hodnôt trajektórie je dôležitá vzdialenosť medzi segmentami, ktorá je definovaná rovnicou 4.5. Skladá sa z troch zložiek – kolmá vzdialenosť $d_\perp$, paralelná vzdialenosť $d_\parallel$ a uhlová vzdialenosť d_θ. Jednotlivé zložky sú formálne definované v [17], ale ich výpočet je možné vidieť na obrázku 4.4.

$$dist(L_i, L_j) = w_\perp \cdot d_\perp(L_i, L_j) + w_\parallel \cdot d_\parallel(L_i, L_j) + w_\theta \cdot d_\theta(L_i, L_j) \tag{4.5}$$



Obr. 4.4: Výpočet zložiek vzdialenostnej funkcie (prevzaté z [17]).

V rovnici 4.5 sú jednotlivé vzdialenosti násobené váhami $w_\perp$, $w_\parallel$ a w_θ. Tieto váhy sa určujú v závislosti od aplikácie. Keďže každý má odlišnú predstavu o tom, ktoré trajektórie sú odľahlé, tak algoritmus by mal byť natoľko neutrálny, aby pokryl rôzne druhy aplikácií. Metóda *TRAOD* dokáže odhaliť dva typy odľahlých hodnôt trajektórie, ktoré sú detegované na základe nastavenia jednotlivých váh. Prvým typom sú pozičné, ktoré sú odlišné na základe polohy a ich detekcia sa dosiahne zvýšením $w_\perp$ alebo $w_\parallel$. Druhým typom sú uhlové, ktoré sú odlišné na základe smeru a ich detekcia sa dosiahne zvýšením w_θ.

Popis algoritmu

Keďže základná verzia *TRAOD* je v podstate rovnaká, len bez sofistikovanejšej segmentácie, tak bude popísaný len algoritmus optimalizovanej verzie. Najprv sa prevedie fáza dvojúrovňovej segmentácie tak, ako bola popísaná vyššie. Teda každá trajektória sa rozdelí na hrubé segmenty a následne sa vypočíta horná a dolná hranica, podľa ktorých bude rozhodnuté, či sú jemné segmenty daného hrubého segmentu L_i vložené do zoznamu $C(L_i)$, nad ktorým sa ďalej prevedie detekcia odľahlých hodnôt.

V druhej fáze, teda vo fáze detekcie, je prvým krokom pre každý segment vypočítať všetky blízke trajektórie a to podľa definície 4.2. V druhom kroku sa zistí, či je segment odľahlý podľa definície 4.3, kde počet blízkych trajektórii $|CTR(L_i, D)|$ je upravený vynásobením korekčným koeficientom $adj(L_i)$ a počíta sa pomocou zoznamu $C(L_i)$. V základnej

verzii sa porovnáva každá dvojica jemných segmentov, čo je výpočtovo náročné. Napokon posledným krokom je samotné detegovanie odľahlej trajektórie podľa definície 4.4.

Ako už bolo spomenuté, tak hlavnou výhodou tohto algoritmu je, že dokáže detegovať odľahlé sub-trajektórie alebo inak segmenty. Tiež je výhodou, že je do istej miery neutrálny a poskytuje detekciu rôznych druhov odľahlých hodnôt.

Medzi slabé stránky tohto algoritmu patrí detekcia len globálne odľahlých trajektórií a to na základe globálneho prahu vzdialenosti, teda parametru D. Nielen tento parameter, ale aj parametre p a F, je ťažké nastaviť a sú pomerne citlivé, obzvlášť parameter D. Nevýhodou je do istej miery aj to, že algoritmus využíva len priestorové informácie o trajektórii a teda nie je možné získať iné zaujímavé poznatky, ktoré by priniesla časová zložka.

4.2 Metódy založené na hustote

Metódy založené na vzdialenosti vyžadujú užívateľom špecifikovaný globálny prah vzdialenosti. Odľahlé trajektórie zistené pomocou globálneho prahu vzdialenosti je možné považovať za globálne odľahlé hodnoty. V prípade metód založených na hustote je možné odľahlé hodnoty považovať za lokálne, pretože sú korelované s hustotou ich susedov.

Pri vyššie uvedenej metóde *TRAOD*, ktorá využíva hybridný prístup založený na vzdialenosti a hustote, sú spomenuté aj jej slabé stránky. Jednou z nich je to, že za pomoci globálneho prahu vzdialenosti deteguje len globálne odľahlé trajektórie. Navyše, ako už bolo spomenuté, tak nastaviť tento prah je pomerne zložité. Tieto nedostatky rieši napríklad metóda *DBTOD*, ktorá na rozdiel od *TRAOD* používa len výpočet hustoty. Jej popis je náplňou tejto podkapitoly.

4.2.1 DBTOD

Metódu *DBTOD* (*Density-based trajectory outlier detection algorithm*) [20] navrhli autori z dôvodu prekonania slabých stránok metódy *TRAOD*. Metóda *DBTOD* je postavená na metóde *TRAOD* a plne využíva rámec delenia a detekcie. Vo fáze delenia je každá trajektória rozdelená na segmenty. Metóda *DBTOD* využíva rovnaký dvoj-úrovňový segmentačný algoritmus ako metóda *TRAOD*. Keďže detailný popis tejto dvoj-úrovňovej segmentácie je súčasťou popisu metódy *TRAOD* v sekcii 4.1.1, tak už nebude súčasťou tejto podkapitoly.

Vo fázy detekcie sa využíva algoritmus metódy *DBTOD*, ktorý je založený na hustote a nie na vzdialenosti. Na rozdiel od metódy *TRAOD* je *DBTOD*, ako algoritmus založený na hustote, schopný efektívne detegovať lokálne odľahlé hodnoty. Koncept hustoty berie do úvahy rozloženie susedných trajektórií a obsahuje dve zložky – vzdialenosť medzi sub-trajektóriami a počet sub-trajektórií v daných rozsahoch.

Detekcia odľahlých trajektórií

Metóda *DBTOD* má taktiež tri vstupné parametre, ktoré ale nie sú také citlivé ako parametre pri metóde *TRAOD*. V metóde *DBTOD* sa užívateľ stretne s nasledujúcimi parametrami:

- Parameter k je kladné celé číslo určujúce počet najbližších susedných segmentov trajektórií, s ktorými sa bude v algoritme pracovať.
- Parameter $threlof$ je prah lokálnej anomálnosti segmentov, ktorý určí či je segment odľahlý.

- Parameter F je prah odľahlosti segmentov v danej trajektórii, ktorý určí, či je samotná trajektória odľahlá.

Detekcia odľahlej trajektórie *DBTOD* vychádza z algoritmu k-najbližších susedov, ktorý má veľký význam aj pri klasifikácii a regresii. Pred popisom algoritmu je potrebné uviesť niekoľko definícii, ktoré priblížia koncept detekcie založenej na hustote a k-NN algoritmu. Na meranie vzdialenosti medzi dvoma segmentami sa využíva funkcia $dist(L_i, L_j)$ z metódy *TRAOD*, ktorá je definovaná rovnicou 4.5.

Definícia 4.7. *Nech k je ľubovoľné kladné celé číslo, tak potom k-distance(L_i) je vzdialenosť k-tého najbližšieho segmentu od L_i.*

Definícia 4.8. *Pre každý segment L_i existuje množina k-najbližších susedou, označovaná ako $kNB(L_i)$, ktorá má veľkosť k. Množina $kNB(L_i)$ obsahuje všetky segmenty L_j, pre ktoré platí $dist(L_i, L_j) \leq$ k-distance(L_i)*

Definícia 4.9. *Hustota lokálnej dosiahnuteľnosti segmentu L_i je definovaná rovnicou 4.6.*

$$llrd_k = 1/\left[\frac{\sum_{L_j \in kNB(L_i)} reach\text{-}dist_k(L_i, L_j)}{|kNB(L_i)|}\right], \quad (4.6)$$
$$kde \;\; reach\text{-}dist_k(L_i, L_j) = max\{k\text{-}distance(L_j), dist(L_i, L_j)\}$$

Definícia 4.10. *Faktor lokálnej anomálnosti segmentu L_i je definovaný rovnicou 4.7.*

$$llof_k = \frac{\sum_{L_j \in kNB(L_i)} \frac{llrd_k(L_j)}{llrd_k(L_i)}}{|kNB(L_i)|} \quad (4.7)$$

Definícia 4.11. *Trajektória TR_i je odľahlá, ak je rovnica 4.8 pravdivá.*

$$Ofrac(TR_i) = \frac{\sum_{L_i \in OP(TR_i)} len(L_i)}{\sum_{M_i \in P(TR_i)} len(M_i)} \geq F \quad (4.8)$$

kde $OP(TR_i)$ je množina všetkých odľahlých segmentov trajektórie TR_i, $P(TR_i)$ je množina všetkých segmentov trajektórie TR_i a parameter F je daný užívateľom.

Popis algoritmu

V prvej fáze algoritmu sa využije dvoj-úrovňový segmentačný algoritmus, ktorý je súčasťou metódy *TRAOD*. Každá trajektória sa teda rozdelí na hrubé segmenty a následne sa vypočíta horná a dolná hranica, podľa ktorých bude rozhodnuté, či sa jemné segmenty daného hrubého segmentu využijú alebo nevyužijú pri následnej detekcii. Detailný popis tejto segmentácie je uvedený v sekcii 4.1.1.

Druhú fázu, teda fázu detekcie je možné popísať v šiestich krokoch. Prvým krokom je vypočítať *k-distance*(L) pre všetky segmenty L. Najprv sa za pomoci vzdialenostnej funkcie, definovanej rovnicou 4.5, vypočítajú vzdialenosti všetkých segmentov od segmentu L. Následne sa z nich vyberie k minimálnych vzdialeností, z ktorých sa vyberie maximálna hodnota a tá sa stane *k-distance*(L). V druhom kroku sa vytvorí množina $kNB(L)$ podľa definície 4.8. Množina $kNB(L)$ obsahuje susedné segmenty k segmentu L. Tretím krokom je výpočet vzdialenosti dosiahnuteľnosti (*reachability distance*), ktorej účelom je uistenie, že všetky segmenty v susedstve sú si veľmi blízke. Vzorec na výpočet $reach\text{-}dist_k(L_i, L_j)$ je

uvedený pri rovnici 4.9, ktorá slúži na výpočet hustoty lokálnej dosiahnuteľnosti, čo je štvrtým krokom algoritmu. V piatom kroku sa vypočíta faktor lokálnej anomálnosti segmentu L podľa rovnice 4.7, ktorý je následne porovnaný s parametrom $threlof$. V prípade, že je vypočítaná hodnota vyššia ako parameter $threlof$, tak je segment označený za odľahlý. Posledným, šiestym krokom, je samotná detekcia odľahlej trajektórie podľa definície 4.11.

Výhodou tohto algoritmu je, že dokáže detegovať nielen odľahlé sub-trajektórie, ale aj lokálne odľahlé trajektórie. Ďalšou výhodou oproti metóde *TRAOD* je to, že rieši problém citlivých a ťažko nastaviteľných parametrov.

4.3 Metódy založené na klasifikácii

Detekcia odľahlých trajektórii prebieha na základe modelu vstupných dát, ktorý je vytvorený na základe trénovacej množiny dát. Jedná sa teda o metódy s učením, čo prináša nevýhodu silnej závislosti výsledkov od kvality trénovacej množiny dát. Výhodou ale je to, že oproti metódam bez učenia je detekcia časovo efektívnejšia. Jedným z najznámejších zástupcom týchto metód je rámec *ROAM*, ktorý bude stručne popísaný v tejto podkapitole.

4.3.1 Rámec ROAM

Rámec *ROAM* [19], celým názvom *Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets*, je založený na motívoch trajektórií a pravidlách. Metóda extrahuje vlastnosti z motívov, ktoré následne klasifikuje podľa hierarchického kvalifikátoru založeného na pravidlách. Na obrázku 4.5 je znázornený príklad dvoch trajektórií, ktoré sú reprezentované motívmi m_i. Tento rámec pozostáva z nasledujúcich troch krokov:

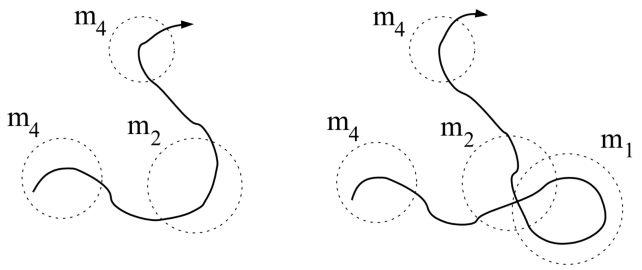
1. Pre extrahovanie motívov zo vstupných trajektórií sa využíva kombinácia posuvného okna a zhlukovania. Pomocou posuvného okna sa extrahuje vektor od jeho prvého bodu po posledný. Následne sú pomocou zhlukovania zistené podobnosti týchto vektorov. Tie reprezentujú jednotlivé motívy, inak povedané motív je nejaký typický vzor pohybu.

   Po extrakcii motívov sa prejde trajektória znova, posuvným oknom rovnakej veľkosti, a v prípade, že je splnená daná podmienka, tak je možné túto časť trajektórie reprezentovať daným motívom. Každá trajektória je potom reprezentovaná ako sekvencia rôznych motívov. Pre každý motív je extrahovaných aj niekoľko atribútov, ktoré sú potrebné pre danú aplikáciu, ako napríklad priemerná, maximálne a minimálna rýchlosť či doba trvania.

2. Po extrakcii motívov je zvyčajne nutná sémantická analýza. Z dôvodu, aby klasifikátor zvládol fungovať je nutné mu predať len správne dáta. Využíva sa tu generátor vlastností, ktorý každému motívu priradí množinu zodpovedajúcich vlastností. Ďalej sa tu využíva generalizačná metóda pre zmenšenie priestoru. Napríklad časové rozsahy sú nahradené časťou dňa – ráno, obed, večer a podobne. Následne množina všetkých motívov tvorí hierarchický priestor rysov. Z trajektórie sa tak stáva vektor týchto rysov.

3. Posledným krokom je už samotná detekcia odľahlých trajektórií. Jednotlivé trajektórie sú klasifikované pomocou klasifikátora CHIP (*Classification using Hierarchical*

Prediction Rules), ktorý je založený na pravidlách. Klasifikátor pracuje nad vytvoreným hierarchickým vektorom rysov a to iteratívnym a nenásytným spôsobom, teda hľadá najlepšie dostupné pravidlo, kým nie sú pokryté všetky prípady.



Obr. 4.5: Trajektórie po extrahovaní motívu m_i (prevzaté z [19]).

4.4 Ostatné metódy

V tejto podkapitole sa nachádza stručný popis ďalších prístupov k detekcii odľahlých hodnôt trajektórií. Sú tu popísané metódy *TOD-SS*, *iBAT* a *TOP-EYE*, okrem ktorých existujú ešte ďalšie ako napríklad detekcia odľahlých zhlukov trajektórií [24], detekcia odľahlej trajektórie medzi regiónmi [7] atď., ktorých popis sa v tejto práci už nenachádza.

Metóda TOD-SS

Prístup k detekcii odľahlých trajektórií, ktorý navrhli v [25], sa nazýva *TOD-SS* (*Trajectory Outlier Detection Algorithm Based on Structural Features*) a je založený na štrukturálnej podobnosti. Algoritmus *TOD-SS* najprv rozdelí trajektórie na segmenty podľa určitého prahu. Následne vyhodnotí štrukturálnu podobnosť z vlastností trajektórie. Zvažuje viac vlastností trajektórie a dôležitosť každej z vlastností je možné upravovať ich váhami. Zistiť odľahlé segmenty je možné vďaka matici SSIM (*structural similarity matrix*) a to porovnávaním štruktúry medzi jednotlivými pármi segmentov trajektórie. Následne je možné detegovať odľahlé trajektórie na základe podielu odľahlých hodnôt segmentov.

Metóda iBAT a metóda iBOAT

Obe metódy sú založené na izolovaní odľahlých trajektórií. Metóda *iBAT* (*Detecting Anomalous Taxi Trajectories from GPS Traces*) [26] bola navrhnutá pre trajektórie taxíkov a pozostáva z nasledujúcich troch krokov:

1. Mapa mesta je rozdelená pomocou mriežky, kde každá bunka má rovnakú veľkosť. Následne sa vytvoria sekvencie buniek reprezentujúce trajektórie, ktoré sú zoskupené na základe rovnakej počiatočnej a cieľovej bunky.

2. Aplikácia algoritmu detekcii odľahlých trajektórií, ktorý využíva vlastnosti *„few and diffrent“*, teda *„niekoľko a odlišných“* a jeho detailnejší popis je uvedený v štúdii [26]. Základný pricíp tejto detekcie spočíva v generovaní náhodného stromu trajektórií. Súbor trajektórií sa rekurzívne rozdeľuje, až kým nie sú takmer všetky trajektórie izolované a následne sa získajú kratšie cesty trajektórií. Tieto trajektórie sú považované za odľahlé, pretože sa izolujú rýchlejšie ako bežné trajektórie.

3. Využitie vydolovaných dát v nejakej reálnej aplikácii.

Výhodou tejto metódy je nižšia výpočtová náročnosť, pretože nie je nutné počítať žiadnu vzdialenosť, zhluky a podobné.

Metóda *iBOAT* (*Isolation-Based Online Anomalous Trajectory Detection*) [5] je následným vylepšením metódy *iBAT*. Metóda funguje na podobnom princípe, ale rozširuje ju, okrem iného, o možnosť online detekcie.

Metóda TOP-EYE

Metóda *TOP-EYE* (*Top-k Evolving Trajectory Outlier Detection*), ktorú navrhli v [8], slúži pre detekciu odľahlých trajektórii už v priebehu ich vývoja. Umožňuje teda identifikáciu odľahlých trajektórií v reálnom čase. Pre každú trajektóriu priebežne počíta mieru odľahlosti, ktorú postupne akumuluje voči predchádzajúcim mieram odľahlosti. Predchádzajúce miery odľahlosti sú postupne znižované prostredníctvom exponenciálnej funkcie. Algoritmus TOP-EYE umožňuje detektovať dva typy odľahlých trajektórií – odľahlé trajektórie podľa smeru a odľahlé trajektórie podľa hustoty.

Kapitola 5

Návrh výsledného riešenia

Pred začiatkom samotnej implementácie výsledného riešenia je potrebné sa nad ním zamyslieť a vypracovať návrh. Každá aplikácia, projekt či výskum sa zaoberá nejakým problémom. Základom dobrého návrhu je rozdeliť tento jeden problém na niekoľko menších podproblémov, ktoré je nutné zvoliť zmysluplne. Vďaka tomu vznikne niekoľko ľahšie dosiahnuteľných cieľov a samotná implementácia je tak omnoho jednoduchšia.

Návrh som rozdelila do troch hlavných krokov. Prvým krokom, ktorému sa venuje podkapitola 5.1, je výber vhodných datasetov. Keďže zadanie tejto diplomovej práce je pomerne otvorené, tak dolovacia úloha nebola predom definovaná. Definovaniu dolovacej úlohy, ktorú bude ukážková aplikácia vykonávať, sa venuje podkapitola 5.2. Posledným krokom je návrh samotnej ukážkovej aplikácie a to predovšetkým návrh jej užívateľského rozhrania, ktorý je popísaný v podkapitole 5.3.

5.1 Výber datasetov

Podstatným krokom pred samotným získavaním znalostí z dát je ich zhromažďovanie, teda vytvorenie datasetu, nad ktorým bude dolovacia úloha vykonaná. V súčasnosti je dostupných viacero datasetov obsahujúcich trajektórie pohybujúcich sa objektov. Môže sa jednať napríklad o pohyb ľudí, zvierat, dopravných prostriedkov alebo prírodných javov. Aj keď sa v súčasnosti zbiera pomerne veľké množstvo dát trajektórii, tak verejne dostupných datasetov nie je až tak veľa.

Prvým verejne dostupným datasetom, ktorý ma zaujal, je dataset *Geolife*. Tento dataset obsahuje trajektórie ľudí a je bližšie popísaný v sekcii 5.1.1. Celkovo ma zaujala myšlienka pracovať s datasetmi, ktoré obsahujú trajektórie ľudí a ich aktivít, preto som zvažovala využiť aj podobný neverejný dataset.

Oslovila som už spomenutú aplikáciu Strava, ktorá ponúka dataset Strava Metro[1]. Požiadala som o prístup k nemu. Tento prístup mi ale nebol poskytnutý, pretože z dôvodu pomerne veľkého počtu žiadostí o spoluprácu poskytujú tento dataset len ľuďom alebo skupinám, ktoré sa priamo podieľajú na plánovaní a vylepšovaní dopravnej infraštruktúry.

Druhý dataset som taktiež vybrala z tých verejne dostupných. Jedná sa o dataset obsahujúci trajektórie *taxíkov*, ktorý je bližšie popísaný v 5.1.2. Pôvodne som plánovala pracovať len s dvoma datasetami, ale nakoniec som sa rozhodla do riešenia zakomponovať aj tretí dataset obsahujúci trajektórie *hurikánov*, ktorý je popísaný v 5.1.3

[1] https://metro.strava.com/

5.1.1 Geolife

Dataset *Geolife*[2] bol vytvorený v rámci projektu Geolife [29, 31, 32] od spoločnosti Microsoft Research Asia. Dataset je zozbieraný od 182 užívateľov v časovom období viac ako päť rokov a to konkrétne od apríla 2007 do augusta 2012. Väčšina údajov bola vytvorená v čínskom meste Peking. Dataset obsahuje viac než 17 tisíc trajektórií s celkovou vzdialenosťou okolo 1,2 milióna kilometrov.

Trajektória v tomto datasete predstavuje postupnosť bodov s časovým razítkom, z ktorých každý obsahuje informácie o zemepisnej šírke, dĺžke a nadmorskej výške. Jednotlivé trajektórie boli zaznamenávané pomocou rôznych sledovacích zariadení s GPS, ako napríklad mobilný telefón. V dôsledku toho majú rôznu vzorkovaciu frekvenciu. Väčšina z nich má vysokú hustotu, teda trajektória obsahuje pomerne veľký počet bodov, ktoré boli zaznamenávané často alebo po krátkej vzdialenosti. Trajektórie zahŕňajú pomerne širokú škálu vonkajších aktivít, ako napríklad cesta do práce, nakupovanie, turistika, cyklistika a iné zábavné či športové aktivity.

Formát dát trajektórií

Dataset obsahuje 182 priečinkov, v ktorých sa nachádzajú jednotlivé trajektórie daného užívateľa. Trajektórie sú uložené v súboroch formátu `plt`, kde každý súbor predstavuje jednu trajektóriu. Každý z týchto súborov má na začiatku šesť riadkov, ktoré obsahujú informácie ako napríklad názov projektu a teda je možné ich plne ignorovať. Následne každý riadok prestavuje jeden bod trajektórie a zahŕňa nasledujúcich sedem informácií o ňom, ktoré sú oddelené čiarkou:

- **Zemepisná šírka** – uvedená v desatinných stupňoch
- **Zemepisná dĺžka** – uvedená desatinných stupňoch
- **0** – v tomto datasete je tu vždy nastavená 0
- **Nadmorská výška** – uvedená v stopách
- **Dátum** – uvedený ako počet dní, ktoré uplynuli od 30.12.1899
- **Dátum** – uvedený ako reťazec
- **Čas** – uvedený ako reťazec

5.1.2 Taxíky

Dataset, ktorý v tejto práci ďalej nazývam *Taxíky*[3], bol vytvorený zozbieraním trajektórií 442 taxíkov v portugalskom meste Porto. V priebehu jedného roku a to konkrétne od začiatku júla 2013 do konca júna 2014 bolo zozbieraných cez 1,7 milióna trajektórií.

Formát dát trajektórií

Trajektórie sú uložené v súbore s formátom `csv`, kde každý riadok predstavuje jednu trajektóriu a obsahuje nasledujúce položky:

[2] https://www.kaggle.com/datasets/arashnic/microsoft-geolife-gps-trajectory-dataset
[3] https://www.kaggle.com/datasets/crailtap/taxi-trajectory

- **Jedinečný identifikátor cesty taxíku** – uvedený ako reťazec
- **Spôsob služby** – uvedený ako znak A - C (pre túto prácu nepodstatné)
- **Jedinečný identifikátor pre každé tel. číslo**, z ktorého bola služba vyžiadaná – uvedené ako celé číslo alebo nadobúda hodnotu `NULL`
- **Jedinečný identifikátor pre stanovisko taxíkov** – uvedené ako celé číslo alebo nadobúda hodnotu `NULL`
- **Jedinečný identifikátor taxikára** – uvedený ako celé číslo
- **Čas začiatku cesty** – uvedený ako unix timestamp (celé číslo, čas v sekundách)
- **Typ dňa začiatku cesty** – uvedený ako znak A - C (pre túto prácu nepodstatné)
- **Chýbajúce dáta** – uvedené ako booleanská hodnota
- **Trajektória** – uvedená ako zoznam GPS súradníc s formátom: $[[x_1, y_1], \ldots, [x_n, y_n]]$, kde x, y predstavujú zemepisnú dĺžku a zemepisnú šírku bodu. Prvá položka zoznamu zodpovedá začiatku cesty a posledná je jej cieľom.

Niektoré z vyššie uvedených položiek, ako napríklad spôsob služby, nie sú dostatočne vysvetlené. Z dôvodu, že nebudú v práci využité a teda v rámci predspracovania odstránené. V prípade záujmu o danú položku je jej popis uvedený na stránke, z ktorej dataset pochádza.

5.1.3 Hurikány

Posledný dataset, ktorý v tejto práci ďalej nazývam *Hurikány*[4] som sa rozhodla do finálneho riešenia zakomponovať z dôvodu, že už bol súčasťou existujúcej implementácie TRAOD[5]. Táto implementácia bude v práci využitá a viac o nej bude povedané v kapitole 6, ktorá sa venuje implementácii výsledného riešenia. Dataset obsahuje 111 trajektórií, čo je v porovnaní s prechádzajúcimi datasetmi pomerne málo.

Formát dát

Trajektórie sú uložené v jednom súbore s formátom `tra`. Tento typ súboru má nasledujúcu štruktúru:

- Prvý riadok: **počet dimenzií** – uvedený ako číslo 2
- Druhý riadok: **počet trajektórií** – uvedený ako číslo 111
- Ostatné riadky: **trajektórie** – každý riadok obsahuje jednu trajektóriu, ktorá sa skladá z nasledujúcich položiek, kde všetky sú oddelené len medzerou:
  - **jedinečné ID trajektórie** - uvedené ako celé číslo
  - **počet bodov obsiahnutých v trajektórii** - uvedený ako celé číslo
  - **pozície všetkých bodov** - uvedené vo formáte $x_1\ y_1\ x_2\ y_2 \ldots x_n\ y_n$

[4] https://github.com/hansenrl/trajectory/blob/master/hurricane2000_2006.tra
[5] https://github.com/hansenrl/trajectory

Keďže dataset obsahuje pomerne málo tarjektórii, tak sa domnievam, že sa jedná len o časť pôvodných dát. Autori už spomenutej implementácie TRAOD[5] sa na ne neodkazujú, takže som sa pokúsila ich dohľadať z dôvodu, že ma zaujímalo, aké ďalšie informácie o týchto trajektóriach je možné sa dozvedieť. Mohlo by sa jednať o dataset z článku o metóde TRAOD [17], z ktorého táto implementácia vychádza. V tomto prípade, žiaľ, už dataset nie je na uvedenej adrese k dispozícii. Každopádne je dohľadateľný verejný dataset hurikánov[6], ktorý môže, ale nemusí túto časť dát obsahovať. Je možné sa z nich dozvedieť o hurikáne napríklad jeho meno a typ, centrálny tlak či rôzne informácie o vetre. V práci bude ďalej využitý len popísaný dataset *Hurikány* a to z dôvodu, že je už vo formáte, s ktorým ukážková aplikácia pracuje.

5.2 Dolovacia úloha

V súčasnosti je veľmi dôležitou dolovacou úlohou detekcia odľahlých hodnôt a neobvyklého chovania, ktorá je aj náplňou tejto práce. Detekcia odľahlých hodnôt v dátach trajektórií má za úlohu odhaliť také trajektórie alebo sub-trajektórie, ktoré sa odlišujú od väčšiny ostatných trajektórií z hľadiska určitej miery podobnosti.

Vybranými metódami pre detekciu odľahlých hodnôt v tejto práci sú *TRAOD* a *DBTOD*, ktoré sú detailne popísané v kapitole 4. Metódy sa odlišujú hlavne v prístupe k detekcii odľahlej hodnoty. Zatiaľ čo *TRAOD* využíva hybridný prístup založený na vzdialenosti a hustote, tak prístup metódy *DBTOD* je založený čisto na hustote a mal by riešiť slabé stránky metódy *TRAOD*. Navyše obe umožňujú užívateľovi nastaviť niekoľko parametrov a tým ovplyvniť proces detekcie a jej výsledky.

Cieľom je teda detegovať odľahlé trajektórie pomocou týchto dvoch metód a výsledky vhodným spôsobom prezentovať užívateľovi. V experimentoch, ktoré sú súčasťou práce, sa porovnajú nielen ich výsledky nad rôznymi dátovými sadami, ale aj to ako rýchlo metóda dáta spracovala. Tiež bude zhodnotené to, ako veľmi zložité je nastavenie jednotlivých parametrov.

5.3 Ukážková aplikácia

Výsledné riešenie diplomovej práce zahŕňa aj implementáciu ukážkovej aplikácie. Aplikácia by mala vhodným spôsobom zobrazovať výsledky stanovenej dolovacej úlohy. V návrhu boli najprv špecifikované jednotlivé požiadavky na aplikáciu. Následne, na základe stanovených požiadaviek, bol vytvorený grafický návrh užívateľského rozhrania.

5.3.1 Špecifikácia požiadaviek

Na špecifikáciu požiadaviek je možné sa pozerať z viacerých pohľadov. Z pohľadu funkčnosti by aplikácia mala spĺňať nasledujúce požiadavky – načítanie datasetu, nastavenie vstupných parametrov metódy, vykonanie metódy a zobrazenie jej výsledkov.

Každá aplikácia obsahujúca grafické užívateľské rozhranie, by mala z tohto pohľadu spĺňať požiadavky na intuitívnosť a jednoduchosť ovládania. Pri návrhu mojej aplikácie som sa snažila všetky zmienené požiadavky čo najviac dodržať.

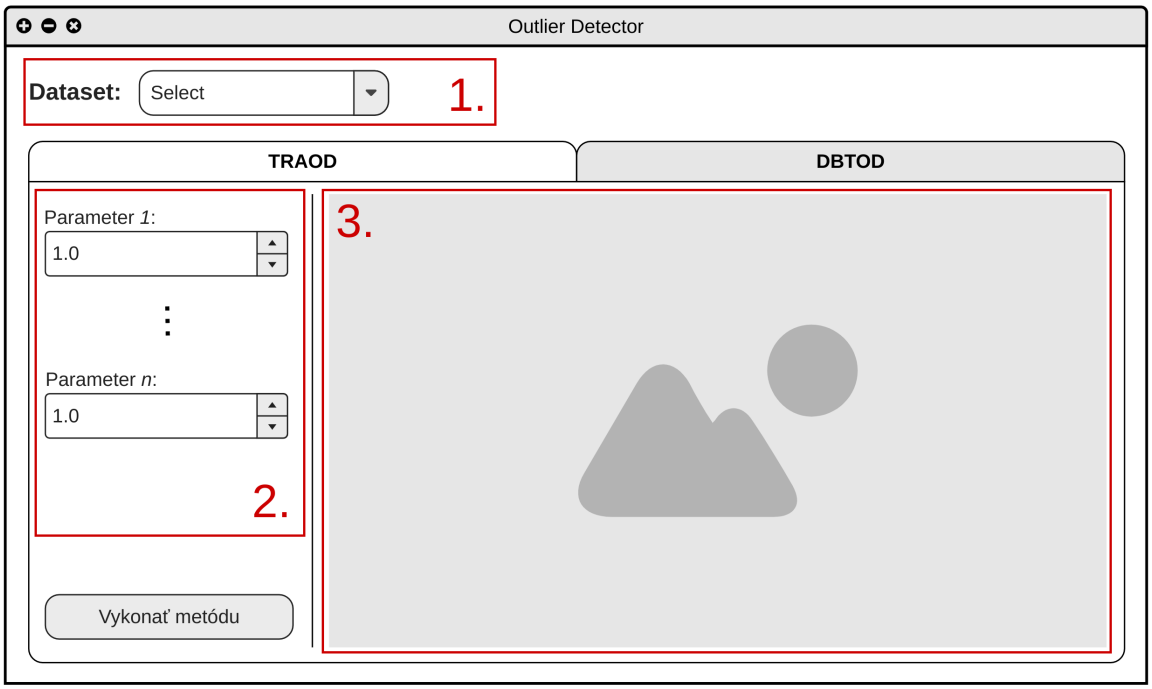
[6] https://www.kaggle.com/datasets/noaa/hurricane-database

5.3.2 Užívateľské rozhranie

Z vyššie uvedených požiadaviek na funkčnosť aplikácie je možné odvodiť tri hlavné časti, z ktorých sa užívateľské rozhranie skladá. Jeho návrh je zobrazený na obrázku 5.1, kde sú jednotlivé časti vyznačené červeným rámom a označené zodpovedajúcim číslo z nasledujúceho zoznamu:

1. **Výber datasetu** – v tejto časti si užívateľ bude môcť vybrať jeden z ponúknutých datasetov, ktorých popis je uvedený v podkapitole 5.1.
2. **Možnosti nastavenia parametrov** – v tejto časti bude môcť užívateľ nastaviť hodnoty jednotlivých parametrov danej metódy. Pri každom parametri bude naviac uvedená krátka nápoveda obsahujúca informácie o parametri a odporúčané nastavenie. Nápoveda bude zobrazená po prejdení kurzorom na názov parametra.
3. **Zobrazenie výsledkov** – v tejto časti užívateľ uvidí výsledky danej metódy. Výsledky budú pravdepodobne zobrazené ako graf spojitých čiar reprezentujúcich jednotlivé trajektórie. Odľahlé trajektórie budú mať od ostatných odlišnú farbu. Konkrétne detaily grafu sa doladia až v priebehu implementácie.

Metódy, ktoré aplikácia vykonáva, majú rôzne parametre. Z tohto dôvodu sú v užívateľskom rozhraní dve záložky – TRAOD a DBTOD, ktoré sú pomenované podľa názvu metódy. Prvá časť užívateľského rozhrania, teda výber datasetu, je spoločná pre obe metódy. Nasledujúce dve časti sú pre metódy rozdielne a nachádzajú sa v jednotlivých záložkách.



Obr. 5.1: Návrh grafického užívateľského rozhrania, ktorý sa skladá z troch hlavných častí: 1. výber datasetu, 2. možnosti nastavenia parametrov a 3. zobrazenie výsledkov.

Kapitola 6

Implementácia

Pri každej implementácii je potrebné dbať na zásady čistého kódu (*clean code*). Kód by mal byť dostatočne zrozumiteľný na to, aby bol pochopiteľný pre ostatných vývojárov. Pri mojej implementácii, ktorej sa táto kapitola venuje, som sa snažila zachovať všetky zásady čistého kódu a objektovo orientovaného programovania.

V podkapitole 6.1, je spomenutý jazyk, v ktorom je aplikácia implementovaná. Tiež je tam uvedený dôvod, prečo som zvolila práve tento jazyk a aj vývojové prostredia využité pri implementácii. Následne sa kapitola venuje open-source knižnici TRAOD v podkapitole 6.2 a mnou implementovanému rozšíreniu o metódu DBTOD v podkapitole 6.3. Podkapitola 6.4 popisuje predspracovanie datasetov. Jedná sa o popis implementácie skriptov, ktoré slúžia na úpravu formátu dát. V závere kapitoly, v podkapitole 6.5 sa nachádza popis implementácie grafického užívateľského rozhrania podľa predchádzajúceho návrhu.

6.1 Programovací jazyk a vývojové prostredie

V úvode implementácie som sa rozhodla pracovať s programovacím jazykom Python, pretože som vedela, že má pomerne veľkú podporu pre dolovanie dát. Tiež som pri výbere jazyka zohľadňovala moje predošlé skúsenosti s daným jazykom. Následne som urobila prieskum, aké rôzne knižnice v rámci dolovania časopriestorových dát ponúka nielen Python, ale aj iné jazyky ako napríklad C a C++.

Pri prieskume som narazila na open-source knižnicu TRAOD, ktorú som sa rozhodla využiť. Jej popis bude náplňou nasledujúcej podkapitoly 6.2. Knižnica je implementovaná v jazyku C++. Zatiaľ čo Python je interpretovaný jazyk, tak C++ je jazyk kompilovaný a teda rýchlejší. Vzhľadom k tomu, že obe vybrané metódy obsahujú pomerne dosť výpočtov, tak je rýchlosť programovacieho jazyka dôležitou vlastnosťou. Z týchto dôvodov som sa nakoniec rozhodla, že aj metódu DBTOD bude vhodné implementovať v jazyku C++, pretože to bude rozšírenie knižnice TRAOD. Keďže táto knižnica vyžaduje špecifický formát vstupných dát, tak je nutné doňho datasety upraviť. Predspracovanie dát nie je na knižnicu nijako naviazané a preto bolo implementované podľa pôvodného plánu v jazyku Python.

Výsledným riešením je aplikácia, ktorej hlavným účelom je prezentovať užívateľovi výsledky detekcie odľahlých trajektórií. Aplikácia teda musí obsahovať jednoduché grafické užívateľské rozhranie. Pri implementácii som sa rozhodla využiť multiplatformové integrované vývojové prostredie Qt Creator, ktoré okrem editoru kódu integruje aj vizuálny grafický editor Qt Designer. Ten slúži na navrhovanie a vytváranie grafických užívateľských

rozhraní z Qt widgetov. Vďaka rámcu Qt je možné nielen vytvoriť jednoduché užívateľské rozhranie, ale navyše je výsledná aplikácia multiplatformová.

6.2 TRAOD

Prvou z vybraných metód na detekciu odľahlých trajektórií je metóda TRAOD, ktorá je detailne popísaná v podkapitole 4.1.1. Profesor Jae-Gil Lee, ktorý je jedným z autorov tejto metódy, sprístupnil originálne zdrojové kódy na svojej webovej stránke[1]. Na základe týchto zdrojových kódov vznikla open-source multiplatformová knižnica TRAOD[2], ktorá bola pri implementácii využitá.

6.2.1 Knižnica TRAOD

Táto knižnica obsahuje implementáciu optimalizovanej verzii metódy aj s dvoj-úrovňovým segmentačným algoritmom. Pôvodná knižnica sa skladá z niekoľkých tried, kde väčšina z nich je deklarovaná v súbore `.h` a definovaná v súbore `.cpp` so zodpovedajúcim menom triedy. Okrem nich sa tu nachádzajú súbory, ktoré obsahujú len výpočty alebo nastavenia parametrov a tiež je súčasťou `csv` parser vstupných dát.

Knižnicu som pri implementácii mierne upravila alebo nevyužila všetky jej časti. Všetky zmeny sú zanedbateľné, ale budem ich v tejto podkapitole priebežne spomínať. Na začiatok popíšem niekoľko najdôležitejších tried a následne formát vstupných dát, ktorý knižnica vyžaduje. Nasledujúce dve triedy patria medzi tie najdôležitejšie a je možné ich považovať za hlavné:

- `TrajData` – trieda uchováva všetky údaje o trajektóriách a ponúka funkcie pre prácu s nimi, ako napríklad načítanie a výstup dát.

  Trieda `TrajData` obsahuje niekoľko funkcií. Jednou z nich je funkcia `readFile`, ktorá slúži na načítanie dát zo vstupného súboru a ich transformáciu do vnútornej reprezentácie tejto triedy. V tejto funkcii došlo len k malej úprave pri načítaní súboru, pretože kvôli využitiu Qt rámca bolo nutné pracovať s `QString` namiesto `string`. Ďalej obsahuje funkcie, ktorých účelom je vytvoriť výstup v podobe grafu vygenerovaného pomocou `gnuplot`. Jednotlivé funkcie sa odlišujú vo formáte výstupného obrázku, ktorý môže byť `eps` alebo `png`. Ani jedna z týchto výstupných funkcií však nie je v mojej práci využitá, pretože som sa rozhodla na vytvorenie grafu využiť modul `QtCharts`, ktorý bude popísaný pri implementácii užívateľského rozhrania v podkapitole 6.5.

  Trajektórie sú uchovávané v zozname `m_trajectoryList`, kde jednotlivé položky zoznamu sú dátového typu `CTrajectory`. Po vykonaní detekcie odľahlých hodnôt sa v tejto triede uchovávajú aj informácie o odľahlých trajektóriách a odľahlých segmentoch a to v zozname s názvom `m_outlierList`, ktorý má štruktúru triedy `COutlier`. Triedy `CTrajectory` a `COutlier` budú stručne popísané ďalej. Trieda `TrajData` ďalej uchováva informácie ako napríklad počet trajektórií, odľahlých trajektórií alebo odľahlých segmentov a tiež o akú dimenziu sa jedná. Dimenzia sa udáva vo vstupnom súbore a jedná sa o číslo 2 alebo 3. Vo väčšine prípadov sa pracuje len s 2D dátami, ale implementácia knižnice by mala zvládať aj 3D dáta.

[1] https://www.kaistdmlab.org/jaegil
[2] https://github.com/hansenrl/trajectory

- `COutlierDetector` – jedná sa o najdôležitejšiu triedu, pretože vedie celý proces detekcie odľahlých hodnôt metódou TRAOD. V konštruktore triedy sa predajú dáta typu `TrajData`, nad ktorými sa bude vykonávať detekcia. Metóda TRAOD pozostáva z dvoch fáz – segmentácia a detekcia. Trieda obsahuje tri hlavné funkcie a to `PartitionTrajectory`, `DetectOutlier` a `ResetOutlierDetector`. Prvé dve funkcie, ako aj z ich názvu vyplýva, zodpovedajú jednotlivým fázam metódy. Funkcia `ResetOutlierDetector` slúži na vyčistenie všetkých zoznamov a vynulovanie počítadiel, ktoré sa využívajú v priebehu detekcie, ale nevymaže dáta trajektórií. Okrem nich trieda `COutlierDetector` obsahuje pomocné funkcie na niektoré výpočty, ale väčšina výpočtov na samotnú detekciu odľahlých hodnôt je v triede `CDistanceOutlier`.

Triedy `TrajData` a `COutlierDetector` teda spracovávajú dataset a vedú proces detekcie odľahlých hodnôt. Jedná sa o jediné dve triedy, ktoré musí vývojár pri využití knižnice použiť. Ostatné triedy sa väčšinou využívajú v hlavných triedach alebo medzi sebou, takže pokiaľ vývojár vyslovene nechce, tak s nimi nemusí pracovať. Z môjho pohľadu je dôležité spomenúť ešte nasledujúce dve triedy:

- `CTrajectory` – jedná sa o hlavné úložisko informácií o trajektórii. Reprezentuje trajektóriu z datasetu v podobe bodov, ktoré sú dátového typu `CMDPoint`, ktorú taktiež definuje táto knižnica. Okrem základných informácií, ako napríklad ID trajektórie, počet bodov trajektórie či samotné body sa v tejto triede uchovávajú aj informácie o segmentoch trajektórie, ako napríklad ich počet, jednotlivé body a ich indexy. Po vykonaní detekcie sa sem tiež uloží pár informácii o odľahlých segmentoch danej trajektórie. Trieda ďalej obsahuje gettery, settery takmer pre každý jej atribút a pár funkcií na manipuláciu so zoznamami.

- `COutlier` – trieda uchováva informácie, ktoré sú užitočné pri práci s odľahlou trajektóriou. Obsahuje ID trajektórie, zoznam odľahlých segmentov, ich počet atď.

V niektorých triedach boli vykonané zanedbateľné zmeny v podobe pridania funkcií, ale hlavne atribútov. Vo väčšine prípadov sa jednalo o atribúty predstavujúce parametre metódy alebo jednotlivé váhy vzdialenostnej funkcie, ktoré boli v pôvodnej knižnici nastavované ako konštanty v súbore `Param.h`. Táto úprava bola prevedená z dôvodu, že bolo potrebné dynamicky meniť hodnoty a to na základe užívateľom zadaných hodnôt v grafickom užívateľskom rozhraní.

Formát vstupných dát

Knižnica pri načítaní dát očakáva jeden súbor so špecifickým formátom `tra`, ktorý obsahuje všetky trajektórie a ďalšie informácie. Tento typ súboru má nasledujúcu štruktúru:

- Prvý riadok: **počet dimenzií** – uvedený ako celé číslo 2 alebo 3
- Druhý riadok: **počet trajektórií** – uvedený ako celé číslo
- Ostatné riadky: **trajektórie** – každý riadok obsahuje jednu trajektóriu, ktorá sa skladá z nasledujúcich položiek, kde všetky sú oddelené len medzerou:
  - **jedinečné ID trajektórie** – uvedené ako celé číslo
  - **počet bodov obsiahnutých v trajektórii** – uvedený ako celé číslo
  - **pozície všetkých bodov** – uvedené vo formáte $x_1\ y_1\ x_2\ y_2\ \ldots x_n\ y_n$ (pre 2D dáta).

6.3 DBTOD

Druhou vybranou metódou na detekciu odľahlých trajektórií je metóda DBTOD, ktorá je detailne popísaná v podkapitole 4.2.1. Keďže je táto metóda založená na metóde TRAOD a taktiež využíva jej dvoj-úrovňový segmentačný algoritmus, tak som sa rozhodla ju implementovať ako rozšírenie, už spomenutej, knižnice TRAOD. Rozšírenie pozostáva z dvoch súborov – `Dbtod.h` a `Dbtod.cpp`, ktoré implementujú triedu `Dbtod`.

Trieda `Dbtod`

Trieda dedí z triedy `COutlierDetector`. Dáta, nad ktorými sa bude vykonávať detekcia, sú taktiež predané v konštruktore tejto triedy. Uchováva všetky informácie o k-najbližších susedoch potrebné pri výpočtoch a funkcie, ktoré vykonávajú samotné výpočty podľa rovníc z podkapitoly 4.2.1. Najdôležitejšou funkciou je `OutlierDetector`, ktorá vedie celý proces detekcie odľahlých trajektórií po ukončení segementačnej fázy. Na segmentáciu sa využíva funkcia `PartitionTrajectory` z rodičovskej triedy `COutlierDetector`.

Funkcia `OutlierDetector` je implementovaná podľa popisu algoritmu uvedeného v podkapitole 4.2.1. Ako prvé sa vypočíta *k-distance* pre všetky segmenty a uložia sa do zoznamu `kDistanceArray`. Súčasne sa vytvorí pre každý segment množina KNB, ktorá je uchovávaná v zozname `kNBsArray`. Keďže sa jednotlivé výsledky používajú opakovane, tak namiesto toho, aby boli počítané priebežne, sú z dôvodu rýchlosti algoritmu vypočítane vopred a uchovávané v podobe zoznamov. Opakovane sa využívajú aj hodnoty $llrd$ (hustota lokálnej dosiahnuteľnosti), ktoré sú taktiež vypočítané vopred a uchovávané v zozname `llrdArray`. Následne sa pre každý segment dopočíta hodnota $llof$ (faktor lokálnej anomálnosti), ktorá je porovnaná s parametrom $threlof$ zadaným užívateľom. Týmto porovnaním sa zistia odľahlé segmenty a následne sa už len detegujú odľahlé trajektórie na základe porovnania s parametrom F, ktorý taktiež zadáva užívateľ.

6.4 Predspracovanie datasetov

Detekcia odľahlých trajektórií bude vykonaná nad troma datasetmi – *Geolife*, *Taxíky* a *Hurikány*, ktoré sú popísané v podkapitole 5.1. Keďže knižnica TRAOD vyžaduje špecifický formát vstupných dát, tak hlavným cieľom predspracovania datasetov je práve ich úprava do tohto formátu.

Dataset *Hurikány* je dataset, ktorý je súčasťou implementácie knižnice TRAOD. Nebolo ho teda potrebné žiadnym spôsobom upraviť, pretože už bol v správnom formáte pripravený rovno na použitie. Pre úpravu datasetov *Geolife* a *Taxíky* boli implementované krátke skripty v jazyku Python – `geolife.py` a `taxi.py`, ktorých cieľom je vygenerovať súbor obsahujúci dáta vo formáte `tra` prijímaným knižnicou TRAOD.

V prípade týchto dvoch datasetov sú pôvodne hodnoty súradníc bodov normalizované pomocou min-max normalizácie. Pri vykonaní prvých experimentov sa totiž ukázalo, že s pôvodnými hodnotami nebola nájdená žiadna odľahlá trajektória alebo naopak boli za odľahlé považované všetky. Bolo to pravdepodobne z dôvodu, že nastaviť jednotlivé parametre metódy je pomerne zložité a pri ich nastavení je potrebné dáta naozaj dobre poznať. Taktiež každá metóda môže fungovať čo najlepšie pri odlišných hodnotách parametrov. Súradnice som normalizovala do podobného rozmedzia ako pri datasete *Hurikány*, vďaka čomu som získala predstavu ako dané parametre a váhy vzdialenostnej funkcie nastaviť.

6.4.1 Geolife

Dataset *Geolife* je upravený pomocou skriptu `geolife.py`. Pôvodne má dataset zložitejšiu štruktúru, kde sa každá trajektória nachádza v samostatnom súbore a obsahuje viacero informácií. V rámci úpravy boli ponechané len súradnice bodov trajektórie.

Dataset ponúka veľké množstvo trajektórii, ktoré obsahujú pomerne veľa bodov. V niektorých prípadoch sa jedná o tisícky bodov na trajektóriu. Z toho dôvodu je v skripte možné obmedziť nie len počet trajektórií, ktoré budú vo vygenerovanom súbore, ale aj počet bodov danej trajektórie. Zároveň je z datasetu vybratá len oblasť dát, ktorú je taktiež možné v skripte zvoliť. Pre výsek oblasti som sa rozhodla z dvoch dôvodov. Prvým dôvodom je, že sa tým obmedzí počet trajektórií. Druhým je, že je možné vybrať oblasť s danou hustotou trajektórií, keďže sa hustota trajektórií naprieč oblasťami odlišuje. Pokúsila som sa vybrať oblasť, keď sú dáta najhustejšie a zároveň sa tam nachádzajú aj časti s menšou hustotou.

6.4.2 Taxíky

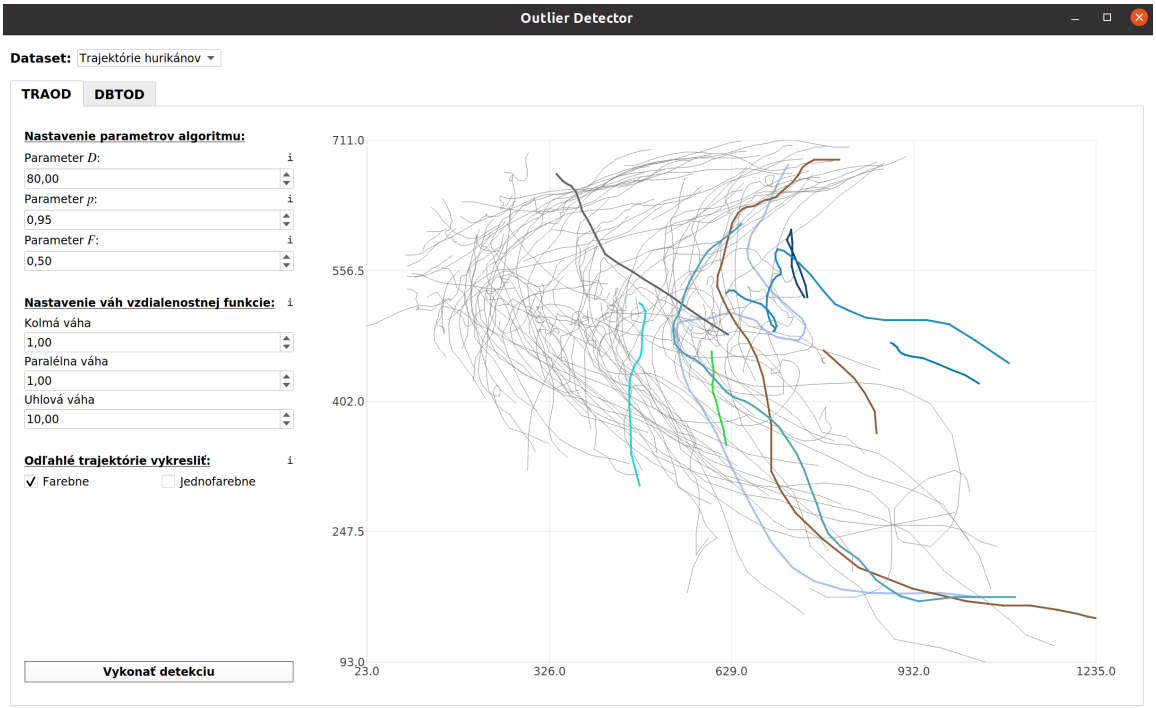
Dataset *Taxíky* je upravený pomocou skriptu `taxi.py`. Pôvodný dataset je jeden súbor s formátom `csv`. Každá položka v súbore predstavuje jednu trajektóriu a obsahuje o nej viacero informácií, z ktorých sú taktiež ponechané len súradnice bodov trajektórie. V skripte sa využíva ešte informácia o chýbajúcich dátach, ktorá ma booleansku hodnotu. V prípade, že je táto hodnota `true`, tak to znamená, že trajektória nemusí byť úplná alebo jej chýbajú niektoré body. Takáto trajektória sa automaticky ďalej neupravuje a nebude obsiahnutá vo výstupnom súbore.

Pôvodný dataset obsahuje viac ako 1,7 milióna trajektórií, takže to, že niektoré trajektórie sú vynechané, nie je vzhľadom k celkovému počtu trajektórií podstatné. Všetky trajektórie by nebolo možné využiť z dôvodu výpočtovej náročnosti algoritmu. Tak ako pri predchádzajúcom datasete *Geolife*, tak aj v tomto prípade je možné obmedziť počet trajektórií vo výslednom súbore. Čo sa týka počtu bodov trajektórie, tak v prípade datasetu *Taxíky* nie je potrebné žiadne obmedzenie, keďže trajektórie neobsahujú veľa bodov. Taktiež som sa ale rozhodla urobiť výsek oblasti z dát a to z rovnakých dôvodov ako pri datasete *Geolife*.

6.5 Užívateľské rozhranie

Výsledná aplikácia by mala obsahovať jednoduché grafické užívateľské rozhranie, vďaka ktorému bude môcť užívateľ nastaviť parametre metód a následne vidieť výsledky. Pri implementácii bol využitý vizuálny grafický editor Qt Designer, kde sa za pomoci `Qt5 widgets` dá vyskladať jednoduché užívateľské rozhranie. Bolo vytvorené na základe návrhu uvedeného v podkapitole 5.3.2. Finálne užívateľské rozhranie je zobrazené na obrázku 6.1 a od pôvodného návrhu sa líši len minimálne.

Hlavné okno aplikácie sa skladá z dvoch častí. Prvou časťou je výber datasetu, nad ktorým bude vykonaná detekcia odľahlých trajektórií. Užívateľ má možnosť vybrať si z troch datasetov popísaných v 5.1. Pre výber datasetu bol využitý widget `QComboBox`, ktorému bola vytvorená funkcionalita klasického select boxu. Druhou časťou je okno s dvoma záložkami vytvorené pomocou `QTabWidget`, kde jedna záložka zodpovedá jednej z metód. Obe záložky obsahujú dve časti – nastavenia a zobrazenie výsledkov. Záložky sa odlišujú iba v tom, aké parametre je možné nastaviť, takže je v práci uvedený len jeden obrázok 6.1, ktorý zobrazuje metódu TRAOD.



Obr. 6.1: Užívateľské rozhranie zobrazujúce metódu TRAOD s datasetom hurikánov.

V časti nastavení sa na získanie hodnôt jednotlivých parametrov využívajú widgety `QDoubleSpinBox` a `QSpinBox`, kde sú na začiatku prednastavené vhodné hodnoty pre daný dataset. Pri názve každého parametru je uvedené písmeno „`i`“, kde sa po prejdení kurzorom zobrazí nápoveda. Nápoveda obsahuje informácie a odporúčané nastavenie parametrov. Keďže aplikácia ponúka dva spôsoby vykreslenia odľahlých trajektórií, tak je v tejto časti možné nastaviť aj ten a to zaškrtnutím jedného z `QCheckBox`.

Výsledky sú zobrazené ako graf pomocou tried `Chart` a `ChartView`, ktoré dedia z tried `QChart` a `QChartView`. Triedy boli vytvorené preto, že bolo potrebné implementovať schopnosť priblíženia, oddialenia a pohybu v grafe, vďaka čomu je možné lepšie analyzovať získané odľahlé trajektórie. Riešenie tejto funkcionality bolo prevzaté z príkladu[3] uvedeného v dokumentácii rámca Qt. Na priblíženie je možné využiť klávesu „+“ a na oddialenie naopak klávesu „−“. Pohybovať sa v grafe je možné pomocou klávesových šípok.

Trajektórie sú v grafe vykreslené sivou farbou a na vykreslenia odľahlých trajektórií aplikácia ponúka dva rôzne spôsoby. Prvým, prednastaveným spôsobom je vykresliť každú trajektóriu inou farbou, čo zvyšuje prehľadnosť v trajektóriách. Vzhľadom k tomu, že tento spôsob môže pre niekoho pôsobiť naopak chaoticky, tak aplikácia ponúka aj druhý spôsob a to vykresliť odľahlé trajektórie len jednou farbou a to konkrétne červenou. V tomto prípade ale nemusí byť jasné, kde začína a končí daná trajektória, keďže sa môžu časti trajektórií prekrývať.

[3] https://doc.qt.io/qt-5/qtcharts-zoomlinechart-example.html

Kapitola 7

Experimenty a vyhodnotenie

V nasledujúcej kapitole sú popísané výsledky experimentov, ktoré boli vykonané na implementovanej ukážkovej aplikácii. Experimenty boli vykonané nad vybranými troma datasetmi, ktoré sú bližšie popísané v podkapitole 5.1. Zhrnutie informácií o trajektóriách z experimentálnych datasetov je uvedené v tabuľke 7.1. Hlavným cieľom týchto experimentov je porovnanie metód *TRAOD* a *DBTOD* určených pre detekciu odľahlých trajektórií. Prvé tri podkapitoly popisujú experimenty spočívajúce v porovnaní výsledkov oboch metód nad daným datasetom, kde každá podkapitola sa venuje práve jednému z vybraných datasetov – *Hurikány* (7.1), *Taxíky* (7.2) a *Geolife* (7.3). V záverečnej podkapitole 7.4 sú zhodnotené výsledky metód všeobecne.

Dataset	**Počet trajektórií**	**Počet bodov v trajektóriách**
Hurikány	111	5 – 88
Taxíky	500	11 – 48
Geolife	500	142 – 15812

Tabuľka 7.1: Zhrnutie informácií o trajektóriách z experimentálnych datasetov.

Z tabuľky 7.1 vyplýva, že datasety boli vytvorené tak, aby pokryli rôzne množstvo trajektórií aj rôzny počet bodov v rámci jednej trajektórie. Vďaka tomu bolo možné zhodnotiť aký vplyv to na metódy má. Ukázalo sa, že jednotlivé počty trajektórií a bodov majú zásadný vplyv na rýchlosť metód. Počet trajektórií v experimentálnych datasetoch je oproti pôvodným dátam pomerne malý. Je to z toho dôvodu, že pri prvých pokusoch o experimenty boli použité dáta obsahujúce viac než tisíc trajektórií a výpočet metód trval aj niekoľko hodín. Vo väčšine prípadov som sa k výsledku ani nedostala, pretože prostriedky, ktorými som disponovala, neboli schopné dokončiť detekciu z dôvodu vysokej výpočtovej náročnosti.

Tak ako pri metóde *TRAOD*, tak aj pri metóde *DBTOD* existujú tri vstupné parametre, ktoré zadáva užívateľ a ovplyvňuje tým výsledok detekcie odľahlých trajektórií. Výsledok je možné ovplyvniť tiež rôznymi hodnotami váh vzdialenostnej funkcie definovanej rovnicou 4.5. Tieto váhy sú zvyčajne pre danú aplikáciu stále rovnaké a nemá ich zadávať užívateľ, ale expert na základe znalostí o dátach a zameraní aplikácie. V tejto práci je ale možné dané váhy dynamicky meniť prostredníctvom užívateľského rozhrania a to z dôvodu, že v rámci experimentov je vhodné vyskúšať aj rôzne hodnoty váh. Zadať hodnoty jednotlivých váh v užívateľskom rozhraní je lepším riešením než zakaždým pristúpiť do zdrojových kódov, upraviť konštantnú hodnotu danej váhy a znovu skompilovať celý program.

Vstupné parametre metódy *TRAOD* sú D, p a F. Parameter D určuje maximálnu vzdialenosť, do akej sú dva segmenty považované za blízke. Parameter p určuje, koľko blízkych trajektórií je potrebných, aby nebol segment považovaný za odľahlý. Parameter F určuje, koľko odľahlých segmentov musí trajektória obsahovať, aby bola považovaná za odľahlú.

Vstupné parametre metódy *DBTOD* sú k, $threlof$ a F. Parameter k určuje počet najbližších susedných segmentov, s ktorými bude algoritmus pracovať. Parameter $threlof$ je prah lokálnej anomálnosti segmentu, ktorý určí, či je segment odľahlý. Parameter F určuje, koľko odľahlých segmentov musí trajektória obsahovať, aby bola sama považovaná za odľahlú, tak ako pri metóde *TRAOD*.

Experimenty nad všetkými datasetami sú podobné. V rámci experimentov nad daným datasetom je pri oboch metódach použitá vždy rovnaká hodnota parametru F a tiež rovnaká hodnota jednotlivých váh vzdialenostnej funkcie. Experimenty spočívajú vo vyskúšaní rôznych kombinácií hodnôt pre zvyšné parametre metód. Keďže každý ma trochu odlišnú predstavu o tom, ako by mal vyzerať výsledok detekcie, koľko by mal obsahovať odľahlých trajektórií atď., tak neexistuje len jedno správne riešenie. Z toho dôvodu je vybratý jeden z výsledkov experimentu, ktorý je tým najlepším podľa mojich kritérií. Následne je nad zvoleným výsledkom vykonaný experiment, ktorý spočíva v zmene jednotlivých váh.

7.1 Hurikány

Dataset *Hurikány* je najmenším experimentálnym datasetom vzhľadom k počtu trajektórií, ale stredne veľkým vzhľadom k maximálnemu počtu bodov v trajektóriách. V prípade oboch metód bolo vykonaných niekoľko experimentov, ktoré vzhľadom k veľkosti datasetu neboli až tak výpočtovo náročné. Pri oboch metódach sa čas výpočtu pohyboval okolo jednej sekundy. Zvyšovaním parametru k pri metóde *DBTOD* tento čas síce narastal, ale v priebehu experimentov nepresiahol tri sekundy. Z tohto dôvodu nebude čas výpočtu jednotlivých experimentov pri popise ich výsledkov uvedený. V rámci experimentov nad týmto datasetom bol zvolený parameter $F = 0,50$. Uhlová váha bola nastavené na 10 a ostatné dve váhy mali hodnotu 1.

Výsledky metódy TRAOD

S metódou *TRAOD* bolo prevedené veľké množstvo experimentov, ktoré spočívali v rôznych kombináciách hodnôt parametrov D a p. Z experimentov je najlepší výsledok s nasledujúcimi parametrami $D = 80$ a $p = 0,95$, ktorý je zobrazený na obrázku 7.1.

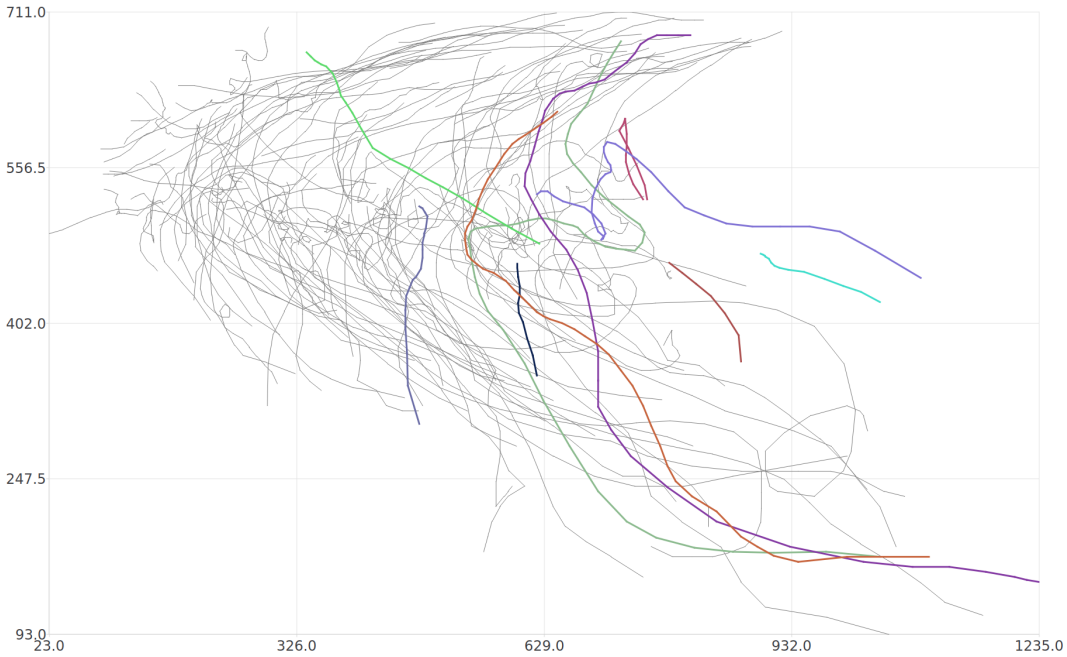
Následne ku každému z týchto parametrov boli vyskúšané rôzne hodnoty druhého parametru, aby som zistila ako ich hodnota ovplyvňuje výsledky. Tabuľky 7.2 a 7.3 zobrazujú počet odľahlých trajektórií v závislosti od jednotlivých hodnôt parametrov. Hodnoty uvedené tučným písmom sú hodnoty uznané za najlepší výsledok, ktorý je zobrazený na obrázku 7.1. Z oboch tabuliek vyplýva, že čím menšiu hodnotu má daný parameter, tak tým viac odľahlých trajektórií algoritmus deteguje.

Parameter D	50	55	60	65	70	75	**80**	85	90	95
Odľahlé trajektórie	41	31	17	15	10	10	**10**	7	6	5

Tabuľka 7.2: Počet odľahlých trajektórií v závislosti od rôznych hodnôt parametru D a parametru $p = 0,95$.

Parameter p	0,90	0,91	0,92	0,93	0,94	**0,95**	0,96	0,97	0,98	0,99
Odľahlé trajektórie	24	12	12	10	10	**10**	8	3	3	2

Tabuľka 7.3: Počet odľahlých trajektórií v závislosti od rôznych hodnôt parametru p a parametru $D = 80$.



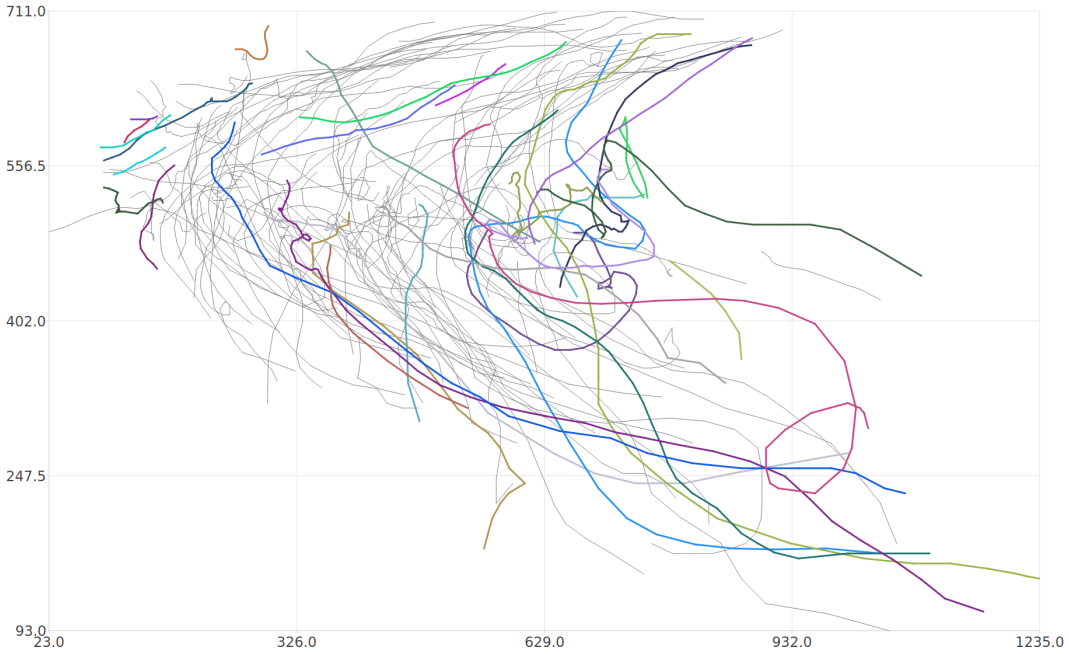
Obr. 7.1: Odľahlé trajektórie hurikánov so zvýšenou uhlovou váhou na hodnotu 10, ostatnými váhami s hodnotou 1 a parametrami $D = 80$, $p = 0,95$ a $F = 0,50$.

Všetky experimenty uvedené v tabuľkách boli vytvorené pri zvýšení uhlovej váhy. Vďaka nej sú odhalené odľahlé trajektórie na základe smeru, ktorý sa zvyčajne odlišuje od väčšiny trajektórií. V rámci ďalších experimentov boli využité rovnaké hodnoty parametrov ako na obrázku 7.1, ale hodnoty jednotlivých váh boli zmenené. Ako prvú som menila paralelnú váhu. Pri zvýšení tejto váhy sa výsledky veľmi nezmenili. Bolo síce nájdených trochu menej trajektórií ako v prípade zobrazenom na obrázku 7.1, ale všetky nájdené boli zhodné s týmto prípadom. Z tohto dôvodu som sa rozhodla sem výsledok neuvádzať.

Následne som menila kolmú váhu, ktorá ovplyvňuje vzdialenosť. Vďaka nej sú odhalené odľahlé trajektórie, ktoré sú vzdialené od hlavných zhlukov. Prvým experimentom bolo zvýšiť hodnotu kolmej váhy na 10 a ostatné váhy znížiť na 1. Výsledkom tohto experimentu bolo označenie takmer všetkých trajektórií za odľahlé, preto som v nasledujúcich experimentoch postupne hodnotu znižovala. Na obrázku 7.2 je uvedený výsledok experimentu obsahujúci 32 odľahlých trajektórií, ktorý som uznala za jeden z najlepších. Kolmá váha v ňom bola zvýšená na hodnotu 4 a ostatné dve váhy znížené na hodnotu 1.

Výsledky metódy DBTOD

S metódou *DBTOD* bolo prevedené približne rovnaké množstvo experimentov ako s metódou *TRAOD*. Experimenty taktiež spočívali v rôznych kombináciách hodnôt parametrov metódy k a $threlof$. Najlepší výsledok nad týmto datasetom dosiahla metódy s nasledu-



Obr. 7.2: Odľahlé trajektórie hurikánov so zvýšenou kolmou váhou na hodnotu 4, ostatnými váhami s hodnotou 1 a parametrami $D = 80$, $p = 0,95$ a $F = 0,50$.

júcimi parametrami $k = 10$ a $threlof = 1,10$. Výsledok je zobrazený na obrázku 7.3. V porovnaní s výsledkom metódy *TRAOD* z obrázku 7.1 je hneď na prvý pohľad vidieť, že bolo detegovaných viac odľahlých trajektórií v hustejších oblastiach a tiež viac okrajových trajektórií. Tiež je možné si všimnúť pár trajektórií, ktoré nie sú na prvý pohľad výrazne odlišné. Jedná sa práve o lokálne odľahlé trajektórie, ktoré získať pomocou metódy *TRAOD* je veľmi obtiažne.

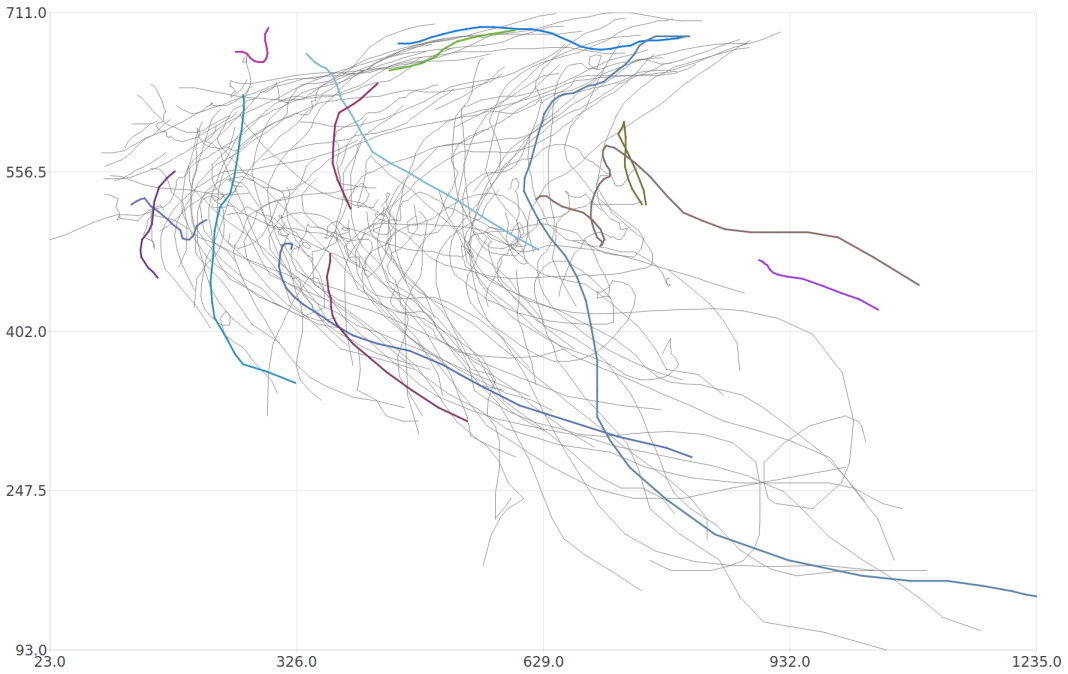
V tabuľkách 7.2 a 7.3 je zhrnutý počet odľahlých trajektórií vzhľadom k rôznym kombináciám hodnôt parametrov, tak ako pri metóde *TRAOD*. Hodnoty uvedené tučným písmom sú hodnoty uznané za najlepší výsledok, ktorý je zobrazený na obrázku 7.3. Z oboch tabuliek taktiež vyplýva, že čím menšiu hodnotu má daný parameter, tak tým viac odľahlých trajektórií algoritmus deteguje.

Parameter k	4	6	8	**10**	12	14	16	18	20
Odľahlé trajektórie	32	20	15	**14**	11	10	9	9	8

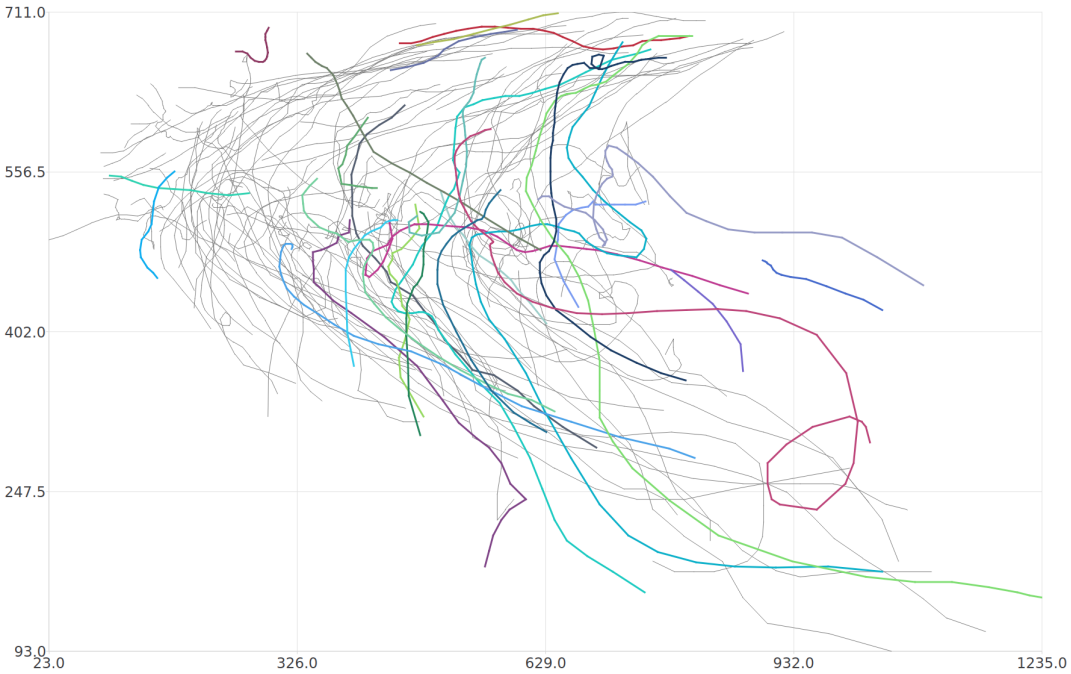
Tabuľka 7.4: Počet odľahlých trajektórií v závislosti od rôznych hodnôt parametru k a parametru $threlof = 1,10$.

Parameter *threlof*	0,95	1,00	1,05	**1,10**	1,15	1,20	1,25	1,30	1,35
Odľahlé trajektórie	101	64	27	**14**	6	5	4	3	2

Tabuľka 7.5: Počet odľahlých trajektórií v závislosti od rôznych hodnôt parametru *threlof* a parametru $k = 10$.



Obr. 7.3: Odľahlé trajektórie hurikánov so zvýšenou uhlovou váhou na hodnotu 10, ostatnými váhami s hodnotou 1 a parametrami $k = 10$, $threlof = 1,10$ a $F = 0,50$.



Obr. 7.4: Odľahlé trajektórie hurikánov, kde všetky váhy majú hodnotu 1 a parametre majú nasledujúce hodnoty $k = 10$, $threlof = 1,10$ a $F = 0,50$.

Ďalšími experimentami bolo zadávanie rôznych hodnôt pre váhy, podobne ako pri metóde *TRAOD*. Aj v tomto prípade sa ukázalo, že výsledky experimentov dopadli podľa očakávania vzhľadom k tomu, čo hodnota danej váhy ovplyvňuje. Zaujímavý výsledok má

experiment, kde boli všetky váhy nastavené na hodnotu 1. Tento výsledok je zobrazený na obrázku 7.4. Je možné si všimnúť, že v porovnaní s obrázkom 7.3 neboli niektoré trajektórie detegované, z čoho vyplýva, že pre ich detekciu je potrebné zvýšiť niektorú z váh.

7.2 Taxíky

Dataset *Taxíky* je najväčším experimentálnym datasetom vzhľadom k počtu trajektórií, ale najmenším vzhľadom k maximálnemu počtu bodov v trajektóriách. Tak, ako pri predchádzajúcom datasete, aj v tomto prípade bolo vykonaných niekoľko experimentov pre obe metódy. Bolo zistené, že väčší počet trajektórií ovplyvňuje rýchlosť výpočtu. Pri metóde *TRAOD* sa jednalo o sekundy, ale pri metóde *DBTOD* to už boli minúty. V rámci experimentov nad týmto datasetom bol zvolený parameter $F = 0,25$. Kolmá váha bola nastavené na 5 a ostatné dve váhy mali hodnotu 1.

Výsledky metódy TRAOD

S metódou *TRAOD* bolo prevedené veľké množstvo experimentov, ktoré taktiež spočívali v rôznych kombináciách hodnôt parametrov D a p. Medzi prvými experimentmi boli vyskúšané hodnoty parametrov, ktoré boli v rámci datasetu *Hurikány* tie najlepšie. Tu sa ukázalo, že každý dataset je iný a vyžaduje taktiež iné hodnoty parametrov, pretože s hodnotami $D = 80$, $p = 0,95$ a $F = 0,50$ neboli nájdené žiadne odľahlé trajektórie. Z tohto dôvodu bola v rámci experimentov nad týmto datasetom zvolená menšia hodnota parametru F.

Najlepší výsledok dosiahla metóda s nasledujúcimi parametrami $D = 50$ a $p = 0,97$. Výsledok je zobrazený na obrázku 7.5. Z obrázku vyplýva, že za odľahlé trajektórie boli označené tie trajektórie, ktorých časti sú výrazne odľahlé. Sú to teda globálne odľahlé trajektórie, pretože sa viditeľne odlišujú od väčšiny trajektórií. Zároveň je pekne ukázané prečo som zvolila vykreslenie trajektórií farebne. Vďaka farbám je vidieť jednotlivé trajektórie a dá sa posúdiť či naozaj obsahujú výrazne odľahlé časti. Keby boli trajektórie zobrazené jednou farbou, tak by u prekrývajúcich sa trajektórií nebolo možné určiť ich začiatok a koniec. Vzhľadom k tomu by bolo ťažké posúdiť, či trajektória vyznačená v hustej oblasti, ktorá vyzerá presne ako väčšina, je naozaj niečím odľahlá alebo niekde nastala chyba.

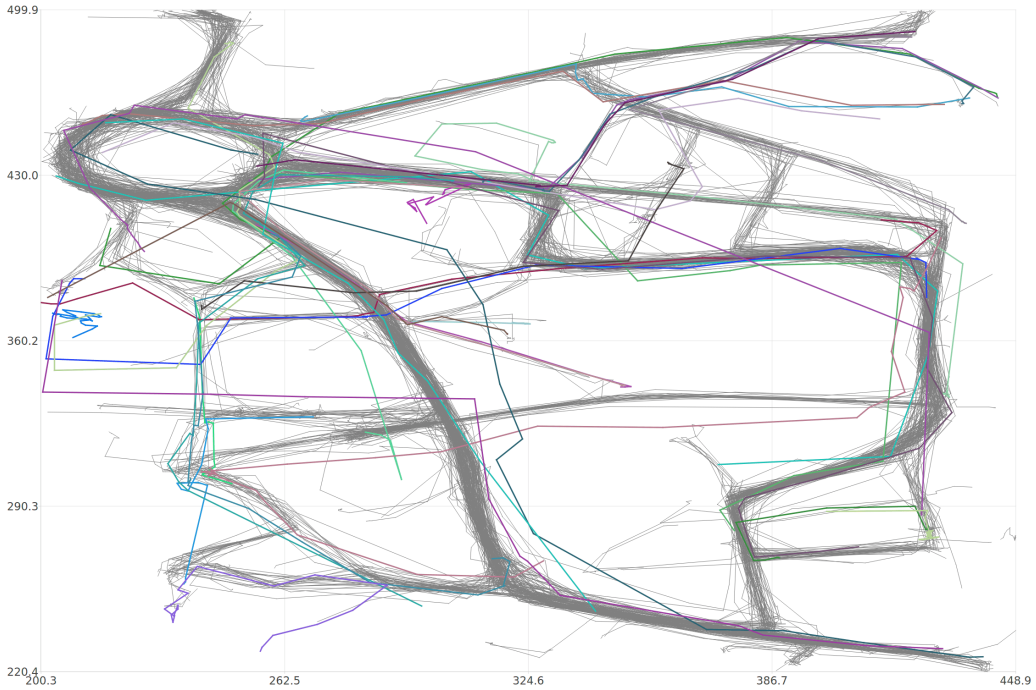
Následne bolo prevedených niekoľko experimentov, kde boli vyskúšané rôzne kombinácie parametrov D a p, tak ako pri predchádzajúcom datasete. Výsledky sú uvedené v tabuľkách 7.6 a 7.7, ktoré okrem počtu odľahlých trajektórií ukazujú aj dĺžku výpočtu. Hodnoty uvedené tučným písmom sú hodnoty uznané za najlepší výsledok, ktorý je zobrazený na obrázku 7.5. Z oboch tabuliek taktiež vyplýva, že čím menšiu hodnotu má daný parameter, tak tým viac odľahlých trajektórií algoritmus deteguje. Ďalej je možné si všimnúť, že veľkosť parametru D ovplyvňuje nie len počet trajektórií, ale aj dĺžku výpočtu. Čím nižšiu hodnotu parameter má, tým kratší je výpočet. Naopak hodnota parametru p dĺžku výpočtu neovplyvňuje a rozdiel v jednotlivých výpočtoch je úplne zanedbateľný.

Parameter D	30	40	45	**50**	55	60	70	80
Odľahlé trajektórie	125	67	48	**35**	26	22	12	5
Dĺžka výpočtu	2s	3s	3s	**3s**	3s	3s	3s	4s

Tabuľka 7.6: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru D a parametru $p = 0,97$.

Parameter p	0,93	0,94	0,95	0,96	**0,97**	0,98	0,99	1,00
Odľahlé trajektórie	116	88	77	50	**35**	25	18	4
Dĺžka výpočtu	3s	3s	3s	3s	**3s**	3s	3s	3s

Tabuľka 7.7: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru p a parametru $D = 50$.



Obr. 7.5: Odľahlé trajektórie taxíkov so zvýšenou kolmou váhou na hodnotu 5, ostatnými váhami s hodnotou 1 a parametrami $D = 50$, $p = 0,97$ a $F = 0,25$.

V rámci posledných experimentov boli vyskúšané rôzne hodnoty váh vzdialenostnej funkcie. Konkrétne sa jednalo o rôzne kombinácie hodnôt 1 a 5. Aj v prípade datasetu *Taxíky* sa ukázalo, že výsledky experimentov dopadli podľa očakávania vzhľadom k tomu čo jednotlivé hodnoty váh ovplyvňujú. Z tohto dôvodu nebude uvedený žiaden konkrétny príklad vo forme obrázku. Výsledky jednotlivých experimentov sú zhrnuté v tabuľke 7.8.

Kolmá váha	**Paralelná váha**	**Uhlová váha**	**Odľahlé trajektórie**	**Dĺžka výpočtu**
5	1	1	35	3s
5	5	1	145	3s
5	1	5	80	3s
1	5	1	20	3s
1	5	5	73	3s
1	1	5	8	3s
5	5	5	199	3s

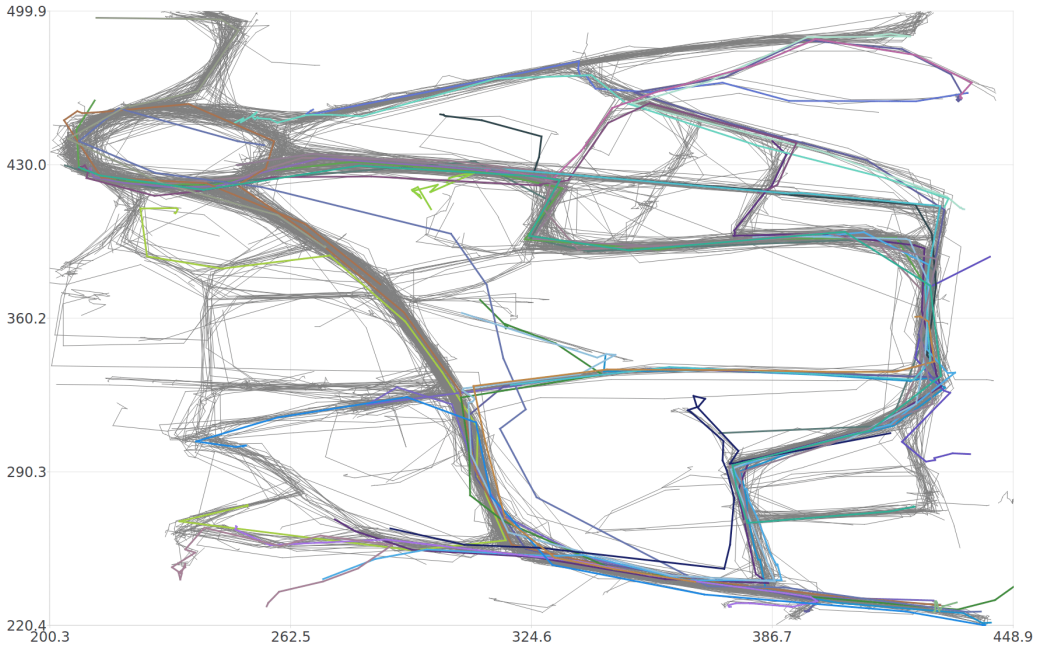
Tabuľka 7.8: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt jednotlivých váh vzdialenostnej funkcie.

Z tabuľky 7.8 vyplýva, že zvýšením viacerých váh naraz bude odhalené väčšie množstvo trajektórií, ale dĺžka výpočtu sa zásadne nezmení. Pri odhalení väčšieho počtu trajektórií začne byť výsledok pomerne neprehľadný a tým pádom sa z neho takmer nič nedozvieme. Preto je lepšie pracovať vždy len s jednou váhou alebo upraviť vstupné parametre tak, aby bolo odhalených menej trajektórií a výsledok tak zostal čo najprehľadnejším.

Výsledky metódy DBTOD

Aj pri tomto datasete bolo s metódou *DBTOD* vykonané približne rovnaké množstvo experimentov ako s metódou *TRAOD*. Najlepší výsledok pri týchto experimentoch dosiahla metóda s parametrami $k = 10$ a $threlof = 1,75$. Výsledok je zobrazený na obrázku 7.6.

Dataset *Taxíky* obsahuje nielen viac trajektórií ako dataset *Hurikány*, ale hlavne hustejšie oblasti. Z toho dôvodu je ťažšie vyznať sa v grafe zobrazujúcom výsledky a rozdiely medzi metódami už nie sú na prvý pohľad tak značne viditeľné. Aj tak je možné si všimnúť, že výsledok metódy *TRAOD* z obrázku 7.5 obsahuje trajektórie, kde každá má výraznú odľahlú časť mimo najhustejšie oblasti. Naopak pri výsledku metódy *DBTOD* z obrázku 7.6 je možné si všimnúť, že bolo odhalených viac lokálne odľahlých trajektórií. Na obrázku 7.7 je znázornená jedna trajektória TR v podobe troch priložených častí, ktoré spôsobili jej odľahlosť.



Obr. 7.6: Odľahlé trajektórie taxíkov so zvýšenou kolmou váhou na hodnotu 5, ostatnými váhami s hodnotou 1 a parametrami $k = 10$ a $threlof = 1,75$ a $F = 0,25$.

Nasledovalo prevedenie experimentov s rôznymi kombináciami hodnôt parametrov k a $threlof$ k hodnotám z najlepšieho výsledku. Výsledky experimentov sú zhrnuté v tabuľkách 7.9 a 7.10. Hodnoty uvedené tučným písmom sú opäť hodnotami najlepšieho výsledku. Z oboch tabuliek tiež vyplýva, že čím menšiu hodnotu má daný parameter, tak tým viac odľahlých trajektórií bolo nájdených. Pri metóde *TRAOD* už bolo spomenuté, že veľkosť parametru môže ovplyvniť aj dĺžku výpočtu. V tabuľke 7.9 je vidieť, že stúpajúcim parametrom k narastá aj dĺžka výpočtu. Kým pri metóde *TRAOD* boli výpočty rýchle a nepresiahli

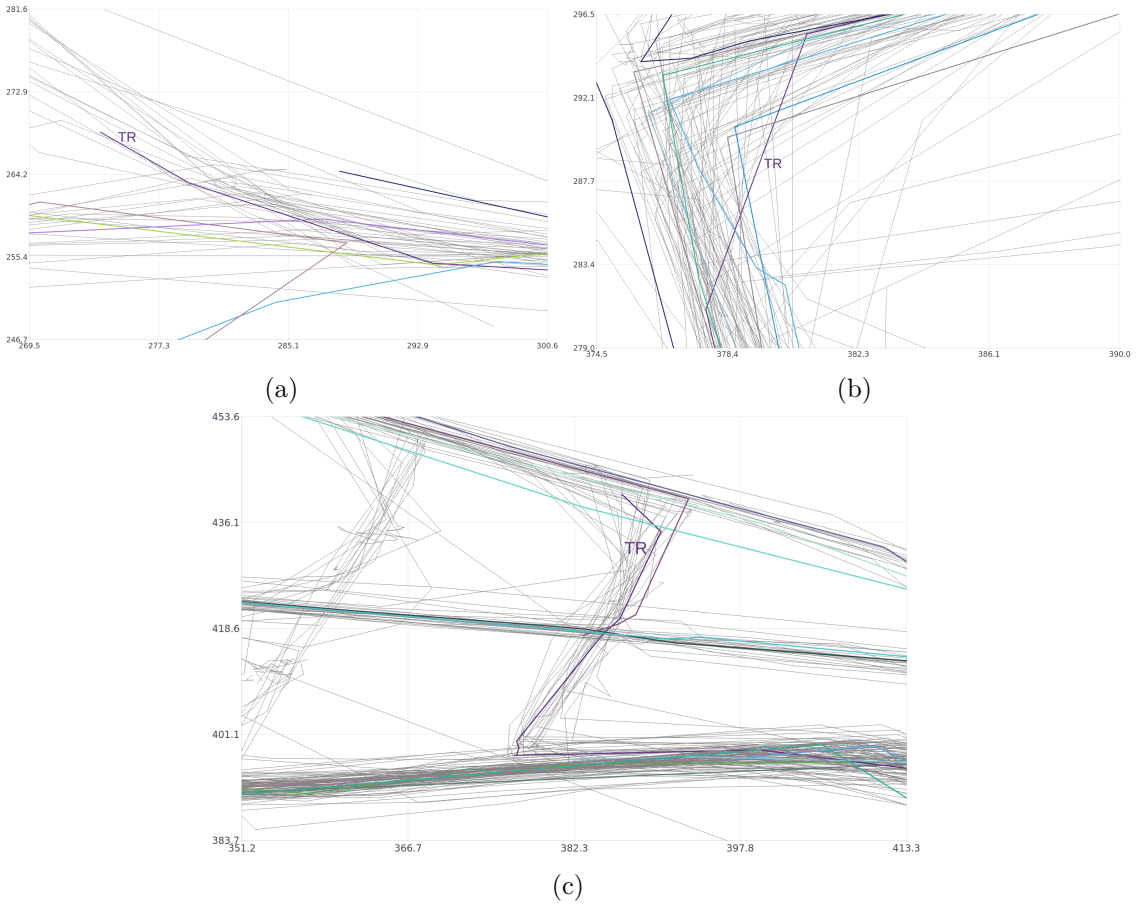
jednotky sekúnd, tak pri metóde *DBTOD* sa už jedná o výpočty trvajúce niekoľko minút. Dôvodom je to, že väčšina vzorcov spočíva v prejdení celej množiny k-najbližších susedov a výpočtu nad každým z nich, čo prirodzene spôsobí vyššiu výpočtovú náročnosť.

Parameter k	6	8	**10**	12	14	16	20
Odľahlé trajektórie	46	36	**32**	26	26	24	21
Dĺžka výpočtu	1m 49s	2m 17s	**2m 46s**	3m 14s	3m 20s	4m 11s	5m 8s

Tabuľka 7.9: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru k a parametru $threlof = 1,75$.

Parameter *threlof*	1,6	1,65	1,7	**1,75**	1,8	1,85	1,9
Odľahlé trajektórie	61	51	36	**32**	28	25	22
Dĺžka výpočtu	2m 46s	2m 46s	2m 47s	**2m 46s**	2m 45s	2m 47s	2m 47s

Tabuľka 7.10: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru *threlof* a parametru $k = 10$.



(a) (b)

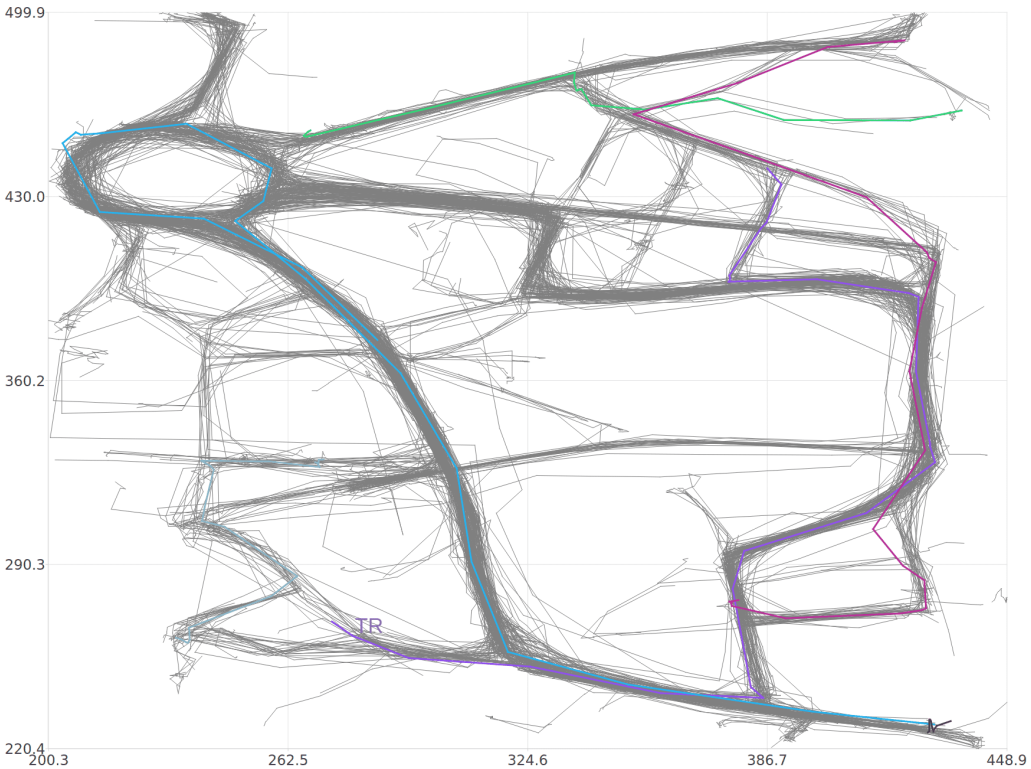
(c)

Obr. 7.7: Rôzne časti trajektórie TR spôsobujúce jej odľahlosť.

Posledným krokom bolo vyskúšať rôzne hodnoty váh vzdialenostnej funkcie. Keďže je cieľom porovnanie oboch metód, tak aj v tomto prípade boli v niekoľkých experimentoch využité rovnaké hodnoty a to konkrétne 1 a 5. Už pri zvýšení jednej z hodnôt sa ukázalo, že výsledky budú podobné ako pri metóde *TRAOD*. Z toho dôvodu som už neskúšala zvýšiť viac váh naraz, počet trajektórií bol totiž v takmer rovnakom pomere. Dĺžka trvania síce nebola rovnaká, ale jednotlivé odchýlky sú pri takýchto vyšších číslach zanedbateľné. Tieto odchýlky môžu byť spôsobené vplyvom iných bežiacich procesov a nie nutne výpočtom metódy *DBTOD*. Výsledky sú zhrnuté v tabuľke 7.11.

Kolmá váha	**Paralelná váha**	**Uhlová váha**	**Odľahlé trajektórie**	**Dĺžka výpočtu**
5	1	1	35	3m 17s
1	5	1	21	3m 15s
1	1	5	13	3m 14s
1	1	10	6	3m 20s

Tabuľka 7.11: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt jednotlivých váh vzdialenostnej funkcie.



Obr. 7.8: Odľahlé trajektórie taxíkov so zvýšenou uhlovou váhou na hodnotu 10, ostatnými váhami s hodnotou 1 a parametrami $k = 10$ a $threlof = 1,75$ a $F = 0,25$.

V rámci posledného experimentu nad týmto datasetom bola vyskúšaná uhlová váha s hodnotu 10 a ostatné váhy s hodnotami 1. Rozhodla som sa to vyskúšať z dôvodu, že pri datasete *Hurikány* bolo práve toto nastavenie najlepšie. Výsledok je, okrem zhrnutia v tabuľke 7.11, uvedený tiež na obrázku 7.8. Na obrázku je vidieť len šesť odľahlých trajektórií,

kde väčšina z nich je na prvý pohľad úplne normálna. Po bližšom preskúmaní je možné si ale všimnúť, že každá z nich má časti, ktoré sa odlišujú od väčšiny susedných častí trajektórií. Na obrázku je tiež označená trajektória TR, ktorej priblížené časti sú znázornené na obrázku 7.7.

7.3 Geolife

Dataset *Geolife* je rovnako veľký vzhľadom k počtu trajektórií ako dataset *Taxíky*. Vzhľadom k maximálnemu počtu bodov v trajektórií je tento dataset najväčší. Tak, ako pri oboch predchádzajúcich datasetoch, aj v tomto prípade bolo vykonaných niekoľko experimentov pre obe metódy. To, že väčší počet trajektórií ovplyvňuje rýchlosť výpočtu, už bolo zistené v predchádzajúcich experimentoch. V rámci týchto experimentov sa tiež ukázalo, že väčší počet bodov trajektórie ovplyvní rýchlosť výpočtu. V tomto prípade sa čas výpočtu oproti datasetu *Taxíky* zväčšil približne trojnásobne. V rámci experimentov nad týmto datasetom bol zvolený parameter $F = 0.40$. Uhlová váha bola nastavená na 10 a ostatné dve váhy mali hodnotu 1, tak ako pri datasete *Hurikány*.

Celkovo sa v rámci experimentov ukázalo, že obe metódy fungujú podľa očakávania, ale čo sa týka datasetu je možné si všimnúť, že sa nejedná o úplne kvalitné dáta a obsahujú šum. Ten môže byť spôsobený nepresnosťou GPS zariadenia alebo rozdielnou frekvenciou snímania dát o polohe. Je možné, že by experimenty dopadli inak, keby bol dataset viac upravený, napríklad výberom len dát s rovnakou vzorkovacou frekvenciou. V práci som sa ale rozhodla ponechať pôvodné dáta, pretože ma zaujímalo, ako sa metódy vysporiadajú s týmto problémom.

Výsledky metódy TRAOD

S metódou *TRAOD* bolo prevedených niekoľko experimentov, ktoré spočívali v rôznych kombináciách hodnôt parametrov D a p, tak ako aj v predchádzajúcich prípadoch. V tomto prípade sa nezdá byť najlepším výsledkom ani jeden. Na obrázku 7.9 je znázornený výsledok pre parameter $D = 50$ a $p = 0,93$.

Z obrázku vyplýva, že metóda *TRAOD* funguje správne a odhalila niektoré globálne odľahlé trajektórie, teda trajektórie, ktoré sa pomerne dosť odlišujú od ostatných. V tomto prípade by som to ale nepovažovala za správne riešenie, keďže sa vo väčšine odľahlých trajektórií pravdepodobne jedná o šum. Šum sa určite vyskytoval aj v predchádzajúcich datasetoch, ale neovplyvňoval výsledky detekcie až natoľko ako v tomto prípade. Pre lepšie výsledky by bolo pravdepodobne potrebné dôkladnejšie predspracovanie tohto datasetu.

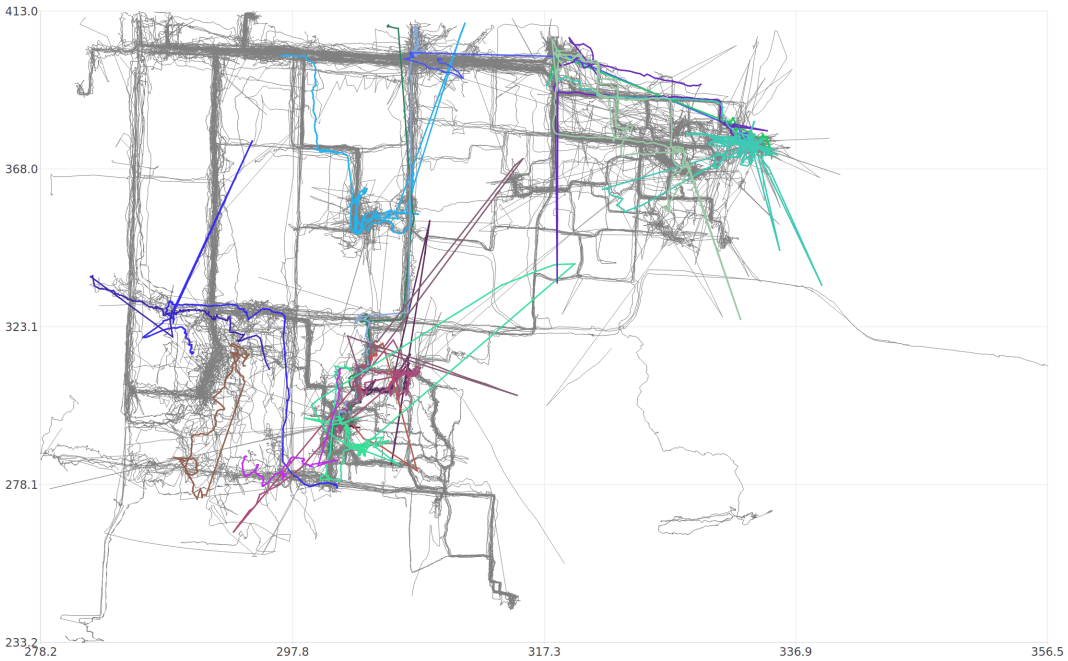
Nasledovalo prevedenie experimentov s rôznymi kombináciami hodnôt parametrov D a p k hodnotám z obrázku 7.9. Zhrnutie výsledkov je uvedené v tabuľkách 7.12 a 7.13, z ktorých vyplýva, tak ako pri predchádzajúcich datasetoch, že čím menšiu hodnotu má daný parameter, tak tým viac odľahlých trajektórií algoritmus deteguje.

Parameter D	20	30	40	**50**	60	70	80	90
Odľahlé trajektórie	53	35	22	**17**	13	11	6	4
Dĺžka výpočtu	9s	10s	11s	**12s**	12s	13s	13s	14s

Tabuľka 7.12: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru D a parametru $p = 0,97$.

Parameter p	0,90	0,91	0,92	**0,93**	0,94	0,95	0,96	0,97
Odľahlé trajektórie	22	21	18	**17**	15	11	8	3
Dĺžka výpočtu	12s	12s	12s	**12s**	12s	12s	12s	12s

Tabuľka 7.13: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru p a parametru $D = 50$.



Obr. 7.9: Odľahlé trajektórie ľudí so zvýšenou uhlovou váhou na hodnotu 10, ostatnými váhami s hodnotou 1 a parametrami $D = 50$, $p = 0,93$ a $F = 0,40$.

Na záver bolo vykonaných ešte niekoľko experimentov na rôzne hodnoty váh vzdialenostnej funkcie. Tak ako v prechádzajúcich experimentoch tohto typu, tak aj v tomto prípade bol počet detegovaných trajektórií mierne odlišný, ale stále sa vo výsledku vyskytovali hlavne spomínané trajektórie, ktoré obsahujú pravdepodobne šum. Z výsledkov teda nebolo veľmi poznať nejaké rozdiely v nájdených trajektóriách a jediným poznatkom bol počet odhalených trajektórií. Z tohto dôvodu nebudem uvádzať žiadne konkrétne výsledky.

Výsledky metódy DBTOD

Aj s metódou *DBTOD* bolo prevedených niekoľko experimentov, ktoré spočívali v rôznych kombináciách hodnôt parametrov k a $threlof$. Najlepší výsledok pri týchto experimentoch dosiahla metóda s parametrami $k = 7$ a $threlof = 1,70$. Výsledok je zobrazený na obrázku 7.10, z ktorého vyplýva, že metóda *DBTOD*, na rozdiel od metódy *TRAOD*, sa so šumom vysporiadala omnoho lepšie. Odľahlé trajektórie boli nájdené zase aj v oblastiach s väčšou hustotou, čo robí metóde *TRAOD* problémy.

Tabuľky 7.14 a 7.15 zobrazujú zhrnutie výsledkov experimentov s rôznymi kombináciami parametrov k a $threlof$. Z oboch tabuliek vyplýva to isté, ako pri predchádzajúcich datasetoch, že čím je nižšia hodnota parametra, tým je vyšší počet nájdených odľahlých

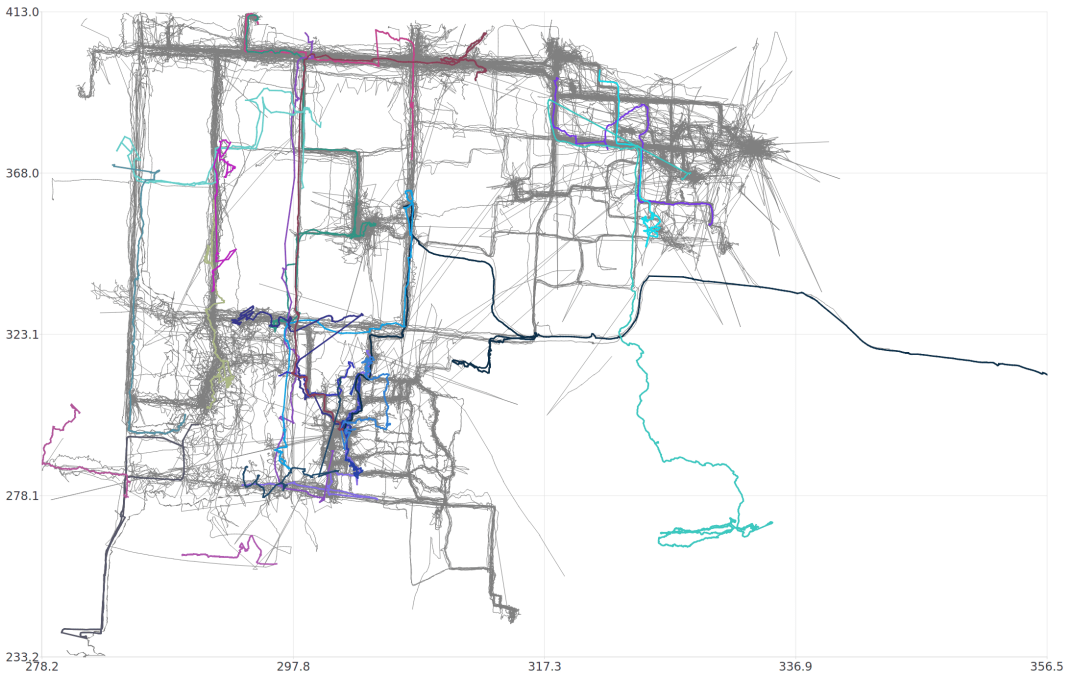
trajektórií. Zmena parametra *threlof* zásadne neovplyvní dĺžku výpočtu a jedná sa znovu len o zanedbateľné odchýlky, ktoré môžu byť spôsobené inými bežiacimi procesmi. Naopak zmena parametra k dĺžku výpočtu ovplyvňuje. Je to preto, že väčšina vzorcov spočíva v prejdení celej množiny k-najbližších susedov a výpočtu nad každým z nich, čo predĺži dobu výpočtu.

Parameter k	3	5	**7**	9	11	13	15
Odľahlé trajektórie	44	23	**22**	22	20	17	17
Dĺžka výpočtu	5m 1s	7m 14s	**9m 34s**	12m 1s	14m 29s	16m 57s	19m 15s

Tabuľka 7.14: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru k a parametru $threlof = 1,70$.

Parameter *threlof*	1,50	1,65	**1,70**	1,75	1,90
Odľahlé trajektórie	40	23	**22**	19	14
Dĺžka výpočtu	9m 38s	9m 36s	**9m 34s**	9m 38s	9m 40s

Tabuľka 7.15: Počet odľahlých trajektórií a dĺžka jednotlivých výpočtov v závislosti od rôznych hodnôt parametru *threlof* a parametru $k = 7$.



Obr. 7.10: Odľahlé trajektórie ľudí so zvýšenou uhlovou váhou na hodnotu 10, ostatnými váhami s hodnotou 1 a parametrami $k = 7$ a $threlof = 1,70$ a $F = 0,40$.

7.4 Všeobecné zhodnotenie metód

V tejto podkapitole som zhrnula výsledky experimentov pre každú metódu a porovnám pre aký typ dát je metóda najvhodnejšia, aké zložité bolo nastavenie jednotlivých parametrov a rýchlosť výpočtu.

Pri experimentoch sa ukázalo, že metóda *TRAOD* je vhodná pre dataset *Hurikány*, ale menej vhodná až nevhodná je pre ostatné dva datasety. Napríklad pri datasete *Geolife* je možné si všimnúť, že metóda neodhalila ani niektoré viditeľne odľahlé trajektórie. To je spôsobené jej korekčným koeficientom, ktorý koriguje hustotu okolia. Nastavenie jednotlivých parametrov je pomerne zložité a je teda nutné zdĺhavé experimentovanie s hodnotami alebo je potrebné, aby ich nastavil expert, ktorý dátam a zameraniu aplikácie dobre rozumie.

Rýchlosť metódy *TRAOD* síce nie je jej silnou stránkou, ale v porovnaní s metódou *DBTOD* je rýchlejšia. V rámci týchto experimentov bola metóda *TRAOD* pomerne rýchla a časovo nepresiahla sekundy, ale to len preto, že sa jednalo o dosť malé datasety. Rýchlosť výpočtu negatívne ovplyvňuje to, že je potrebné vypočítať smerodajnú odchýlku, hustotu každého segmentu a napokon ešte samotnú detekciu.

Pri experimentoch s metódou *DBTOD* sa ukázalo, že pre datasety *Taxíky* a *Geolife* je vhodnejšia ako metóda *TRAOD*. Detekcia založená na hustote je pri týchto datasetoch vhodejšia, pretože obsahujú husté oblasti, po ktorých sa väčšinou objekt pohybuje. Pri metóde *DBTOD* boli odhalené nie len výrazne odľahlé trajektórie, ktoré metóda *TRAOD* neodhalila, ale aj lokálne odľahlé trajektórie. Metóda sa lepšie vysporiadala aj so šumom v datasete *Geolife*. Nastavenie vstupných parametrov bolo jednoduchšie, pretože jednotlivé parametre nie sú také citlivé. Napriek tomu bolo nutné vyskúšať viac hodnôt. V porovnaní s metódou *TRAOD* to ale nevyžadovalo až také zdĺhavé experimentovanie.

Výpočet pri metóde *DBTOD* je časovo náročnejší ako pri metóde *TRAOD*. Ako sa pri experimentoch ukázalo, tak vplyv má na rýchlosť nie len počet trajektórií, ale aj počet bodov jednotlivých trajektórií. Čím je počet bodov trajektórií vyšší, tým je vyšší je aj počet segmentov. Väčšina vzorcov spočíva v prejdení celej množiny k-najbližších susedov segmentu a v následnom výpočte nad každým z nich.

Každá metóda je vhodná na iný typ dát. Vo všeobecnosti je efektívnejšia metóda *DBTOD*, pretože sa ukázalo, že dokáže odhaliť nie len globálne odľahlé trajektórie, ale aj tie lokálne odľahlé. Každá z metód má ale svoje silné aj slabé stránky a v niektorých prípadoch je vhodnejšie využiť metódu *TRAOD*. Napríklad v prípade, kedy by užívateľa zaujímali len globálne odľahlé hodnoty a dataset by obsahoval dáta podobné datasetu *Hurikány*.

Kapitola 8

Záver

Cieľom tejto diplomovej práce bolo zoznámiť sa s problematikou získavania znalostí z časopriestorových dát so zameraním predovšetkým na detekciu odľahlých hodnôt v dátach trajektórií pohybujúcich sa objektov a vhodným spôsobom prezentovať tieto znalosti. Cieľ bol rozdelený na niekoľko menších podcieľov. Prvým z nich bolo zoznámenie sa s problematikou po teoretickej stránke. Nasledoval výber vhodných datasetov, definovanie dolovacej úlohy, implementácia ukážkovej aplikácie a sada experimentov nad vybratými datasetmi.

V rámci prvého podcieľa bolo popísané všeobecné získavanie znalostí z dát. Následne sa práca venovala popisu získavania znalostí z časopriestorových dát. Pomerne detailne bola popísaná detekcia odľahlých trajektórií a vybraté metódy, ktoré bude ukážková aplikácia vykonávať. Okrem teoretického popisu práca obsahuje aj prehľad rôznych existujúcich metód a techník, ktoré sa využívajú pri predspracovaní a dolovaní z časopriestorových dát.

Ďalej nasledovala praktická časť diplomovej práce. Pre detekciu odľahlých trajektórií boli vybraté dve metódy – *TRAOD* a *DBTOD*. V návrhu výsledného riešenia diplomovej práce bola špecifikovaná dolovacia úloha a popísané vybrané datasety. Boli vybraté tri reálne datasety obsahujúce pohyb ľudí, taxíkov a hurikánov. Pri implementácii bola pre metódu *TRAOD* využitá už existujúcej knižnica, ktorá je implementovaná v jazyku C++. Metóda *DBTOD* bola implementovaná ako rozšírenie tejto knižnice. Ukážková aplikácia je implementovaná taktiež v jazyku C++ s využitím rámca Qt5 a vývojového prostredia Qt Creator. Výsledným riešením diplomovej práce je ukážková aplikácia s grafickým užívateľským rozhraním, ktoré vhodným spôsobom demonštruje výsledky metód.

Posledným podcieľom boli experimenty s vybranými metódami nad všetkými troma datasetmi. Pri každom datasete sú vyhodnotené výsledky oboch metód a to, aký vplyv má nastavenie jednotlivých parametrov na detekciu či rýchlosť výpočtu. Následne je uvedené zhodnotenie metód všeobecne. Obe metódy fungujú dobre a majú svoje silné aj slabé stránky. Lepšie výsledky v hustejších oblastiach mala metóda *DBTOD* a tiež odhalila aj lokálne odľahlé trajektórie, ale výpočet bol časovo náročnejší. Takže v prípade, keď užívateľa zaujímajú hlavne globálne odľahlé trajektórie, je vhodnejšie využiť metódu *TRAOD*.

V budúcnosti je možné aplikáciu rozšíriť o ďalšie metódy a vykonať ďalšiu sadu experimentov pre porovnanie jednotlivých metód. Začala by som pridaním metódy *TOP-EYE*, ktorá je univerzálna, dokáže detegovať odľahlé trajektórie na základe ich hustoty, smeru a mala by byť rýchlejšia. Bolo by zaujímavé porovnať ju práve s metódou *DBTOD*, pretože obe detegujú odľahlé trajektórie na základe hustoty. V tejto práci boli ale vybraté metódy *TRAOD* a *DBTOD*, pretože obe využívajú rovnaký prístup k segmentácii.

Literatúra

[1] Alon, J., Sclaroff, S., Kollios, G. a Pavlovic, V. Discovering clusters in motion time-series data. In: IEEE. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* 2003, sv. 1. ISBN 0-7695-1900-8.

[2] Bellman, R. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM.* ACM New York, NY, USA. 1961, zv. 4, č. 6, s. 284. DOI: 10.1145/366573.366611.

[3] Brown, D. E., Liu, H. a Xue, Y. Mining preferences from spatial-temporal data. In: SIAM. *Proceedings of the 2001 SIAM International Conference on Data Mining.* 2001, s. 1–17. DOI: 10.1137/1.9781611972719.26.

[4] Chao, P., Xu, Y., Hua, W. a Zhou, X. A survey on map-matching algorithms. In: Springer. *Databases Theory and Applications: 31st Australasian Database Conference, ADC 2020, Melbourne, VIC, Australia, February 3–7, 2020, Proceedings 31.* 2020, s. 121–133. ISBN 978-3-030-39468-4.

[5] Chen, C., Zhang, D., Castro, P. S., Li, N., Sun, L. et al. IBOAT: Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems.* IEEE. 2013, zv. 14, č. 2, s. 806–818. DOI: 10.1109/TITS.2013.2238531.

[6] Douglas, D. H. a Peucker, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization.* University of Toronto Press. 1973, zv. 10, č. 2, s. 112–122. DOI: 10.3138/FM57-6770-U75U-7727.

[7] Fontes, V. C., Alencar, L. A. de, Renso, C., Bogorny, V. a Pisa, I. Discovering Trajectory Outliers between Regions of Interest. In: *GeoInfo.* 2013, s. 49–60.

[8] Ge, Y., Xiong, H., Zhou, Z.-h., Ozdemir, H., Yu, J. et al. Top-eye: Top-k evolving trajectory outlier detection. In: *Proceedings of the 19th ACM international conference on Information and knowledge management.* 2010, s. 1733–1736. DOI: 10.1145/1871437.1871716.

[9] Giannotti, F. a Pedreschi, D. *Mobility, data mining and privacy: Geographic knowledge discovery.* Springer Science & Business Media, 2008. ISBN 978-3-540-75177-9.

[10] Güting, R. H. a Schneider, M. *Moving objects databases.* Elsevier, 2005. ISBN 978-0-12-088799-6.

[11] Han, J., Pei, J. a Tong, H. *Data mining: concepts and techniques*. 4. vyd. Morgan Kaufmann, 2022. ISBN 978-0-12-811760-6.

[12] Hsu, W., Lee, M. L. a Wang, J. *Temporal and spatio-temporal data mining*. Igi Global, 2007. ISBN 978-1-59904-387-6.

[13] Kantardzic, M. *Data mining: concepts, models, methods, and algorithms*. 2. vyd. John Wiley & Sons, 2011. ISBN 978-0-470-89045-5.

[14] Keogh, E., Chu, S., Hart, D. a Pazzani, M. An online algorithm for segmenting time series. In: *Proceedings 2001 IEEE International Conference on Data Mining*. 2001, s. 289–296. DOI: 10.1109/ICDM.2001.989531.

[15] Knorr, E. M., Ng, R. T. a Tucakov, V. Distance-based outliers: algorithms and applications. *The VLDB Journal*. Springer. 2000, zv. 8, č. 3, s. 237–253. DOI: 10.1007/s007780050006.

[16] Kong, X., Li, M., Ma, K., Tian, K., Wang, M. et al. Big Trajectory Data: A Survey of Applications and Services. *IEEE Access*. 2018, zv. 6, s. 58295–58306. DOI: 10.1109/ACCESS.2018.2873779.

[17] Lee, J.-G., Han, J. a Li, X. Trajectory outlier detection: A partition-and-detect framework. In: IEEE. *2008 IEEE 24th International Conference on Data Engineering*. 2008, s. 140–149. DOI: 10.1109/ICDE.2008.4497422.

[18] Lee, J.-G., Han, J. a Whang, K.-Y. Trajectory clustering: a partition-and-group framework. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, s. 593–604. DOI: 10.1145/1247480.1247546.

[19] Li, X., Han, J., Kim, S. a Gonzalez, H. Roam: Rule-and motif-based anomaly detection in massive moving object data sets. In: SIAM. *Proceedings of the 2007 SIAM International Conference on Data Mining*. 2007, s. 273–284. DOI: 10.1137/1.9781611972771.25.

[20] Liu, Z., Pi, D. a Jiang, J. Density-based trajectory outlier detection algorithm. *Journal of Systems Engineering and Electronics*. BIAI. 2013, zv. 24, č. 2, s. 335–340. DOI: 10.1109/JSEE.2013.00042.

[21] Meng, F., Yuan, G., Lv, S., Wang, Z. a Xia, S. An overview on trajectory outlier detection. *Artificial Intelligence Review*. Springer. 2019, zv. 52, s. 2437–2456. DOI: 10.1007/s10462-018-9619-1.

[22] Ramakrishnan, R. a Gehrke, J. *Database management systems*. 3. vyd. New York: McGraw Hill, 2003. ISBN 0-07-246563-8.

[23] Wang, D., Miwa, T. a Morikawa, T. Big trajectory data mining: a survey of methods, applications, and services. *Sensors*. MDPI. 2020, zv. 20, č. 16, s. 4571. DOI: 10.3390/s20164571.

[24] Ying, X., Xu, Z. a Yin, W. G. Cluster-based congestion outlier detection method on trajectory data. In: IEEE. *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. 2009, sv. 5, s. 243–247. DOI: 10.1109/FSKD.2009.504.

[25] Yuan, G., Xia, S., Zhang, L., Zhou, Y. a Ji, C. Trajectory outlier detection algorithm based on structural features. *Journal of Computational Information Systems*. 2011, zv. 7, č. 11, s. 4137–4144.

[26] Zhang, D., Li, N., Zhou, Z.-H., Chen, C., Sun, L. et al. IBAT: detecting anomalous taxi trajectories from GPS traces. In: *Proceedings of the 13th international conference on Ubiquitous computing*. 2011, s. 99–108. DOI: 10.1145/2030112.2030127.

[27] Zheng, Y. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*. ACM New York, NY, USA. 2015, zv. 6, č. 3, s. 1–41. DOI: 10.1145/2743025.

[28] Zheng, Y., Li, Q., Chen, Y., Xie, X. a Ma, W.-Y. Understanding mobility based on GPS data. In: *Proceedings of the 10th international conference on Ubiquitous computing*. 2008, s. 312–321. DOI: 10.1145/1409635.1409677.

[29] Zheng, Y., Li, Q., Chen, Y., Xie, X. a Ma, W.-Y. Understanding mobility based on GPS data. In: *Proceedings of the 10th international conference on Ubiquitous computing*. 2008, s. 312–321. DOI: 10.1145/1409635.1409677.

[30] Zheng, Y., Liu, L., Wang, L. a Xie, X. Learning transportation mode from raw gps data for geographic applications on the web. In: *Proceedings of the 17th international conference on World Wide Web*. 2008, s. 247–256. DOI: 10.1145/1367497.1367532.

[31] Zheng, Y., Xie, X., Ma, W.-Y. et al. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* Citeseer. 2010, zv. 33, č. 2, s. 32–39.

[32] Zheng, Y., Zhang, L., Xie, X. a Ma, W.-Y. Mining interesting locations and travel sequences from GPS trajectories. In: *Proceedings of the 18th international conference on World wide web*. 2009, s. 791–800. DOI: 10.1145/1526709.1526816.

[33] Zheng, Y. a Zhou, X. *Computing with spatial trajectories*. Springer Science & Business Media, 2011. ISBN 978-1-4614-1628-9.

[34] Zhu, Y., Zheng, Y., Zhang, L., Santani, D., Xie, X. et al. Inferring taxi status using gps trajectories. *ArXiv preprint arXiv:1205.4378*. 2012.