VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

Anomalijų aptikimas interneto prieigos stebėsenos sistemos įrašuose

Detection of anomalies in internet access monitoring system data

Bakalauro darbas

Atliko:	Jokūbas Rusakevičius	(parašas)
Darbo vadovas:	asist. dr. Vytautas Valaitis	(parašas)
Darbo recenzentas:	j. asist. Linas Petkevičius	(parašas)

Vilnius – 2019

Santrauka

TODO: Santrauka

Raktiniai žodžiai: raktinis žodis 1, raktinis žodis 2, raktinis žodis 3, raktinis žodis 4, raktinis žodis 5

Summary

TODO: summary

Keywords: keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

TURINYS

Įvadas

Šis darbas yra Programų Sistemų studijų baigiamasis bakalauro darbas apie anomalijų aptikimą interneto prieigos stebėsenos sistemos įrašuose naudojant „MacroBase“ atviro kodo analitinį įrankį bei „SQL“ paketą.

Problematika

Informacijos rinkimas fizine forma, juos užrašant ant popieriaus lapo, pasak MIT Media Lab direktoriaus Joi Ito, dar visai neseniai buvo vienintelis būdas rinkti, klasifikuoti ir saugoti duomenis. Žmogus mintis ir idėjas užrašydamas ant popieriaus lapo – paversdavo žiniomis (angl. *knowledge*). Tačiau laikai keičiasi, ir dabartinė didžiųjų duomenų (angl. *big data*) situacija yra kitokia. Dėl milžiniškų surenkamų duomenų kiekio, nėra trivialu atskirti naudingus ir nenaudingus duomenis, todėl, natūraliai, negalima visų surenkamų duomenų automatiškai priskirti prie ir vadinti žiniomis. Surinkti duomenys nėra žinios, kol jų nepradedama nagrinėti, analizuoti ir filtruoti, ir tik pradėjus šį procesą galima pastebėti, kad gaunama informacija yra įdomi ir netgi svarbi [Smo14].

Duomenų, kuriais yra operuojama, juos yra renkama ir saugoma, kiekiai nuolatos didėja [EMC14]. Iš tiesų, dėl daugelio priežasčių net greitis, kuriuo šie duomenys yra generuojami eksponentiškai kyla [Mak17]. Jau 2015–2016 metais didžiosios socialinių tinklų kompanijos Twitter, Facebook ir LinkedIn pranešė fiksuojančios iki 12 milijonų įvykių (angl. *events*) per sekundę [Ast16; PFT+15; Woo15]. Šie duomenų kiekiai gerokai lenkia žmogaus gebėjimą juos apdoroti bei žymiai apsunkina darbą tiek analitikams, tiek analitiniams įrankiams. Net aukščiausios kvalifikacijos (angl. *best-of-class*) šių dienų taikomųjų programų operatoriai bei analitikai praneša panaudojantys anekdotiškai mažą jų surenkamų duomenų kiekį – mažiau nei 6% [BGR+17].

Renkant tokius didelius kiekius duomenų natūraliai, užfiksuojami įrašai, kurie yra nebūdingi įrašų rinkiniui bei išsiskiria. Duomenų vienetai nukrypę nuo kitų duomenų rinkinyje yra vadinami anomalijomis. Anomalijų aptikimas yra procesas, kurio metu yra aptinkama ir identifikuojama anomalija arba išskirtis (angl. *outlier*) duomenų rinkinyje [Tec19]. Šio darbo metu bus tiriamas anomalijų aptikimas remiantis belaidės interneto prieigos duomenų perdavimo spartos kontrolinių matavimų, atliktų 2014–2018 metais, rezultatais, kurie gauti naudojantis Lietuvos Respublikos ryšių reguliavimo tarnybos administruojama IPSS [rrtar].

Fiksuojami esminiai ar sisteminiai duomenys tobulu atveju turėtų būti peržiūrimi reguliariai ir pilnai, tam kad būtų galima užtikrinti tolygią sistemų veiklą, matavimų taisyklingumą ar renkamų duomenų korektiškumą ir laiku identifikuoti bei eliminuoti kylančias klaidas bei užkirsti

kelią kenkėjiškoms atakoms. Tačiau kaip jau minėta anksčiau, didėjantis duomenų kiekis gerokai lenkia žmogaus gebėjimą juos apdoroti ir nors duomenų kiekis vis didėja, žmogaus dėmesys išlieką riboto dydžio. Iškyla iššūkis prioritetizuoti žmogaus dėmesį – akcentuoti, filtruoti, jungti, grupuoti ir kontekstualizuoti svarbiausius analizuojamų duomenų elgsenos ir būsenos pakitimus, taip padedant išskirti svarbiausius duomenų ypatumus ir rodyti žmogui tik apibendrintą, svarbia ir ribotą kiekį informacijos [BGR+17]. Visa nereikalinga informacija, kuri yra rodoma žmogui reikalauja ir eikvoja jo dėmesį [Sim71].

1. Duomenų analizės eksperimentas naudojant „MacroBase SQL“ modulį

1.1. Duomenų rinkinys

Šiame skyriuje aprašyti eksperimentui pritaikyti IPSS duomenų rinkiniai.

1.1.1. Eksperimentui pritaikyti duomenys

Eksperimentui buvo naudojami Lietuvos Respublikos ryšių reguliavimo tarnybos administruojamos Interneto prieigos stebėsenos sistemos (IPSS) atliekami belaidės interneto prieigos duomenų perdavimo spartos kontrolinių matavimų 2014–2018 metų rezultatai. Matavimai atlikti operatorių UAB „Bitė Lietuva“, AB „Telia Lietuva“, UAB „TELE2“ ir AB Lietuvos radijo ir televizijos centro (LRTC) judriojo ryšio tinkluose visoje Lietuvos teritorijoje. Matavimų paskirtis yra stebėti ir įvertinti teikiamų interneto prieigos paslaugų kokybę, kaip to reikalauja Europos Sąjungos direktyvos, taip pat supažindinti visuomenę su tokių matavimų rezultatais. Įrašai pateikiami CSV formato failais. Iš viso gauti 3 failai skirti trims skirtingoms technologijoms: 3G, LTRE ir WiMAX. Viename CSV faile priklausomai nuo failui priskirtos technologijos buvo apytiksliai 136000, 157000 ir 18000 įrašų eilučių, vidutiniškai 104000. Vieno failo dydis vidutiniškai 8,5MB.

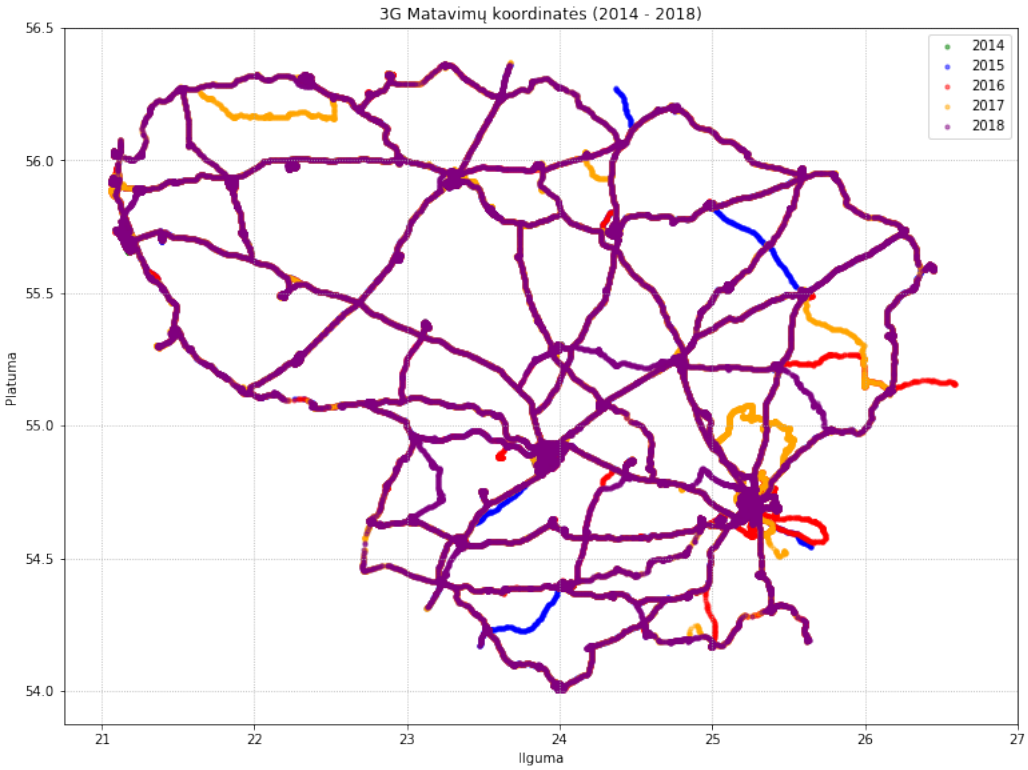
1.1.1.1. Duomenų aprašymas

Kiekvienas IPSS įrašų failas prasideda antraštės (angl. *header*) eilute, kuri susideda iš faile saugomų įrašų sąrašo laukų (angl. *fields*) pavadinimų. IPSS įrašų sarašų laukai ir jų paaiškinimas:

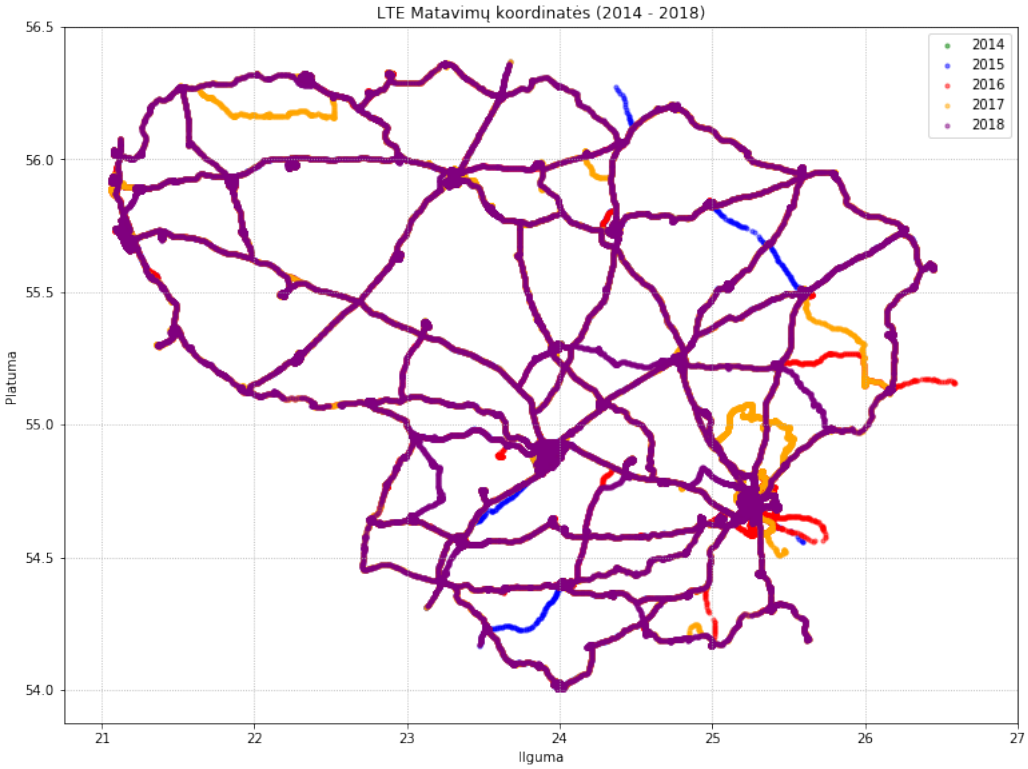
- **Bendri visiems failams laukai**:
- **Data ir laikas** – matavimo įrašo fiksavimo data ir laikas, užrašoma „yyyy-MM-dd hh:mm:ss“ formatu.
- **Platuma** – matavimo įrašo fiksavimo koordinačių platuma užrašoma dešimtainiais laipsniais (angl. *decimal degrees*).
- **Ilguma** – matavimo įrašo fiksavimo koordinačių ilguma užrašoma dešimtainiais laipsniais (angl. *decimal degrees*).
- **Operatorius** – matavimo įrašo operatoriaus skaitinis kodas arba pavadinimas (vieno iš jau minėtų operatorių: „Bitė Lietuva“, „Telia Lietuva“, „TELE2“ ir LRTC).
- **Ryšio tinklas** – tinklo technologija (3G, LTE arba WiMAX).

- **Celės id** – korinio tinklo ląstelės identifikacinis numeris atvaizduotas skaitine šešioliktaine forma.
- **Bazinės stoties id** – bazinės stoties identifikacinis numeris atvaizduotas skaitine šešioliktaine forma, parametras pateikiamas tik LRTC WiMAX tinkle atliktuose matavimuose.
- **3G ryšio technologija** – matavimų metu nustatytas duomenų perdavimo technologijos porūšis (WCDMA, HSPA+, DC-HSPA+).
- **RSSI** – (angl. *Received Signal Strength Indicator*) signalo lygis, dBm (apklausiant modemą gaunamos vertės nuo -51 dBm iki -113 dBm. Mažiausia signalo vertė -113 dBm reiškia labai silpną signalą, kuriam esant praktiškai nebegalima naudotis duomenų perdavimo paslauga).
- **RSRP** – (angl. *Reference Signal Received Power*) pilotinio signalo galia, dBm (apklausiant modemą gaunamos vertės nuo -140 dBm iki -44 dBm. Mažiausia signalo vertė -140 dBm reiškia labai silpną signalą, kuriam esant praktiškai nebegalima naudotis duomenų perdavimo paslauga). Parametras pateikiamas tik LTE tinkle atliktuose matavimuose.
- **RSRQ** – (angl. *Reference Signal Received Quality*) pilotinio signalo kokybė, dB (apklausiant modemą gaunamos vertės nuo -19,5 dB iki -3 dB. Mažiausia vertė -19,5 dB). Parametras pateikiamas tik LTE tinkle atliktuose matavimuose.
- **SINR** – (angl. *Signal to Interference and Noise Ratio*) signalo ir trukdžių plius triukšmo santykis, dB (apklausiant modemą gaunamos vertės nuo -20 dB iki 30 dB. Mažiausia vertė -20 dB). Parametras pateikiamas tik LTE tinkle atliktuose matavimuose.
- **CINR** – (angl. *Carrier to Interference and Noise Ratio*) nešlio ir trukdžių plius triukšmo santykis, dB, parametras pateikiamas tik LRTC WiMAX tinkle atliktuose matavimuose.
- **Sparta kbit/s** – (angl. *Download speed*) išmatuota duomenų gavimo spartos vertė, kbps.

IPSS matavimai atliekami naudojant Ryšių reguliavimo tarnybos turimą įrangą, kuri yra sumontuota matavimams skirtame automobilyje. Matavimai atliekami automobiliui judant (angl. *drive test*) miestų gatvėmis, automagistralėmis arba rajoniniais keliais pagal pasirinktus maršrutus. Atliktų matavimų koordinates galima pamatyti 1, 2 ir 3 pav.



1 pav. 3G Matavimai 2014–2018 metais



2 pav. LTE Matavimai 2014–2018 metais

3 pav. WiMAX Matavimai 2015–2017 metais

Aiškiau išskirtus įrašų koordinačių duomenis galima rasti Priede nr. 1.

1.1.1.2. Duomenų paruošimas

Eksperimentui atlikti buvo atrinkti laukai, kurie turi reikšmes visose įrašų sąrašo eilutėse bei taip pat buvo atmesti pasikartojančios informacijos laukai (ryšio tinklo ir ryšio technologijos laukai, kurie nurodo, koks tinklo tipas iš trijų tiriamųjų yra naudojamas faile). Taip pat buvo nuspręsta datos ir laiko lauką pakeisti tik metų lauku. Taigi, 3G tinklui atrinkti šie pradiniai laukai: metai, ilguma, platuma, operatorius, celės id, rssi, 3G ryšio technologija ir sparta; iš LTE duomenų atrinkti laukai: metai, ilguma, platuma, operatorius, celės id, rssi, rsrp, sparta; iš WiMAX duomenų atrinkti laukai: metai, ilguma, platuma, operatorius, bazinės stoties id, rssi, cirn, sparta.

Eksperimentui atlikti buvo reikalingas CSV formato failai, kuriuose nebūtų lietuviškų raidžių ir būtų naudojami vieno žodžio arba vienos simbolių eilutės antraštės pavadinimai (pavyzdžiui, tarpai gali būti pakeičiami „_“ simboliu), todėl buvo realizuoti „Python“ skriptas, kuris nuskaito duotųjų duomenų CSV failus ir juos apdorojus, grąžina naujus CSV failus. Skriptas kurtas ir leistas „Jupyter Notebook“ aplinkoje. Skripte galima pasirinkti kelią iki duotųjų CSV failų ir kelią iki naujai sukuriamų CSV failų. Šis skriptas skirtas pervadinti antraštes, atskirti metus ir atrinkti nereikalingus laukus. Skriptai atlieka šiuos veiksmus 3 kartus kiekvienai ryšio

technologijai (3G, LTE, WiMAX) (Priedas nr. 2):

1. Nuskaito CSV įrašų failą pateiktą nurodytu keliu.
2. Sukuria naują CSV failą nurodytų keliu ir pavadinimu.
3. Įrašo į naujai sukurtą CSV failą, naują antraštę, su naujais angliškais vieno žodžio (simbolių eilutės) pavadinimais.
4. Vykdo ciklą, kuris skaitant po vieną eilutę iš pradinio failo įrašo tik reikalingus duomenis į naują failą bei iš datos ir laiko paima tik metus. Taip pat kiekviena eilutė yra skaičiuojama.
5. Išvedamas galutinis eilučių skaičius.

Atrinkus duomenis iš pradinių duomenų buvo sukurtas antras skriptas skirtas išgauti daugiau informacijos iš koordinačių. Kadangi ilgumos ir platumos laukai yra realiosios reikšmės, jomis yra nepatogu naudotis. Tam buvo sukurtas dar vienas skriptas, kuris suapvalina koordinačių laukus iki pasirinkto skaičiaus po kableliu taip sumažinant unikalių reikšmių kiekį. Šios reikšmės buvo pridėtos į joms sukurtus naujus laukus. Papildomai originalios koordinačių reikšmės buvo panaudotos išgauti kiekvieno koordinačių taško aukštį virš jūros lygio. Ši bei suapvalinta iki dešimčių metrų reikšmės buvo pridėtos į papildomus joms sukurtus laukus. Taigi naujas „Python“ skriptas nuskaito prieš tai sukurtus CSV failus, gauna aukščio virš jūros lygio reikšmes, suapvalina koordinačių bei aukščio reikšmes ir sukuria naujus CSV failus pasirinktu adresu. Skriptai atlieka šiuos veiksmus 3 kartus kiekvienai ryšio technologijai (3G, LTE, WiMAX) (Priedas nr. 3):

1. Nuskaito CSV įrašų failą pateiktą nurodytu keliu.
2. Iš nuskaitytų duomenų atskiriama antraštė nuo įrašų.
3. Iš antraštės surandami metų, platumos ir ilgumos laukų indeksai.
4. Pasinaudojus gautais indeksais, sukuriami visų ilgumos ir platumos reikšmių masyvai.
5. Vykdomas ciklas, kuris visus įrašus padalina į pasirinkto žingsnio dydžio dalis.
6. Vykdomas ciklas cikle, kuris visas ilgumos ir platumos reikšmes sujungia į „string“ tipo kintamuosius, kuriuose ilguma ir platuma yra atskirtos kableliu „,“.
7. Kiekvienam žingsniui, iš platumos ir ilgumos bendrų kintamųjų sukuriamas vienas „string“ kintamasis, kur kiekvienas bendras koordinačių kintamasis yra atskirtas vertikaliu brūkšniu „|“.
8. Vykdomas ciklas, kuris kiekvienai žingsnio koordinačių kiekio „string“ kintamajam sukuria URL užklausą „Jawg.io“ API, kuri susideda iš koordinačių kintamojo, pagrindinio URL ir vartotojo prieigos žymės (angl. *access token*).
9. Kiekvienam žingsniui yra siunčiama HTTP užklausa, kuri grąžina JSON duomenų struktūrą su koordinatėmis ir jų aukščiais virš jūros lygio.
10. Iš JSON duomenų struktūros atskiriamos aukščio virš jūros lygio reikšmės ir jos yra patal-

pinamos į masyvą atitinkamu indeksu.

11. Vykdomas ciklas, kuris kiekvienam nuskaitytam įrašui sukuria nauja įrašą su suapvalintomis iki pasirinkto skaičiaus po kablelio reikšmėmis bei aukščio virš jūros lygio ir suapvalinta iki dešimčių aukščio virš jūros lygio reikšmėmis. Šie įrašai sujungiami su atitinkamais įrašais ir sukuriamas naujas masyvas su šiais įrašais.
12. Sukuriamas naujas CSV failas nurodytu keliu ir pavadinimu.
13. Įrašoma nauja antraštė su papildytais pavadinimais.
14. Vykdomas ciklas, kuris kiekvieną naujai papildytą įrašą įveda į naują CSV failą.

Eksperimentui atlikti buvo nuskaityti ir apdoroti apie 300000 įrašų, vidutiniškai 100000 per failą. TODO: pridėti citatą

1.2. Eksperimento aplinkos paruošimas

Eksperimentas buvo atliekamas naudojant „Ubuntu (64-bit)“ operacinę sistemą įdiegtą virtualioje mašinoje „Oracle VM VirtualBox“. „MacroBase SQL“ įdiegtas ir paruoštas darbui naudojantis „MacroBase“ dokumentacija.

1.2.1. Virtuali mašina eksperimentui

„MacroBase“ pateikiami pavyzdžiai ir konfigūraciniai nurodymai yra pateikiami „Linux“ operacinėms sistemoms. Dėl šios priežasties, eksperimentui atlikti buvo pasirinkta atviro kodo nemokama operacinė sistema „Ubuntu“. Dėl paprastumo buvo nuspręsta naudoti virtualią mašiną operacinei sistemai įdiegti. Atviro kodo nemokama virtuali mašina „Oracle VM VirtualBox“ buvo pasirinkta šiai užduočiai. Galutinės eksperimentui naudotos sisteminės specifikacijos:

- Reali mašina, kurioje atliekamas eksperimentas – nešiojamasis kompiuteris „Lenovo Y50-70“.

  Realios mašinos Operacinė sistema **„Microsoft Windows 10 Pro“**, versija 10.0.17763.

  Realios mašinos procesorius: **i7-4720HQ**

  Realios mašinos operatyvioji atmintis: **8GB**
- **„Oracle VM VirtualBOX“** virtuali mašina, versija **6.0.8** r130520 (Qt5.6.2).

  Virtualios mašinos operacinės sistemos „Ubuntu“ 64 bitų versiją **„Ubuntu (64-bit)“**, versija **18.04 LTS**.

  Virtualiai mašinai suteiktas virtualus standusis diskas: **40GB**.

  Virtualiai mašinai skirta operatyvioji atmintis: **4GB**.

1.2.1.1. Virtualios mašinos paruošimas

Virtualios mašinos paruošimas eksperimentui:

1. Iš „Oracle VM VirtualBox“ internetinės svetainės atsisiųstas naujausias „Windows 10“ operacinei sistemai skirtas diegimo failas (`https://www.virtualbox.org/wiki/Downloads`).
2. Sekant sąrankos vedlio nurodymus įdiegta „Oracle VM VirtualBox“ virtuali mašina.
3. Iš „Ubuntu“ internetinės svetainės atsisiųstas naujausias „Ubuntu (64-bit)“ operacinės sistemos ISO failas (`https://www.ubuntu.com/download/desktop`).
4. Atidarius „Oracle VM VirtualBox“ programinę įrangą pradėtas naujos virtualios mašinos pridėjimo procesas spaudžiant mygtuką su tekstu „New“.
5. Sekant naujos virtualios mašinos pridėjimo langus pasirinktos šios parinktys:

   Operacinės sistemos tipas: „Linux“.

   Versija: „Ubuntu (64-bit)“.

   Operatyviosios atminties kiekis megabaitais: 4096MB.

   Virtualus standusis diskas: 40GB.
6. „Oracle VM VirtualBox“ pagrindiniame lange pasirinkus naujai sukurtą vitualią mašiną spaustas mygtukas su tekstu „Start“.
7. Atidarytame virtualios mašinos lange pasirinktas anksčiau atsiųstas „Ubuntu (64-bit)“ ISO failas.
8. Pasirinkta „Install“ operacija ir sekant diegimo vedlį, įdiegta „Ubuntu (64-bit)“ operacinė sistema.

1.2.2. „MacroBase“ analitinio įrankio paruošimas

„MacroBase SQL“ diegiamas pagal „MacroBase“ dokumentacijoje nurodytus žingsnius. Tačiau prieš pradedant diegimo procesą reikia įdiegti įrankius bei papildomus paketus, tam kad „MacroBase“ taisyklingai veiktų bei būtų galima atlikti diegimo procesą. Eksperimentui naudotos programinės įrangos specifikacijos:

- „MacroBase“ versija v1.0.
- Java JRE versija 1.8.0_212.
- Java JDK versija 1.8.0_212.
- „Apache Maven“ versija 3.6.0.

1.2.2.1. „MacroBase SQL“ diegimui reikalingi papildomi paketai

Sėkmingam „MacroBase SQL“ diegimo procesui naujai paruoštoje „švarioje“ „Ubuntu“ operacinėje sistemoje, reikalingi papildomi pagalbiniai paketai. Šiuos paketus įdiegti buvo naudojamos šios terminalo komandos:

1. Versijavimo kontrolės sistemos „Git“ diegimas:

   ```
   sudo apt install git
   ```

2. Java projektų valdymo ir diegimo priemonės „Apache Maven“ diegimas:

   ```
   sudo apt install maven
   ```

3. Java JRE diegimas:

   ```
   sudo apt install openjdk-8-jre
   ```

4. Java JDK diegimas:

   ```
   sudo apt install openjdk-8-jdk
   ```

1.2.2.2. „MacroBase SQL“ analitinio įrankio diegimas

Sekant „MacroBase“ dokumentacijoje nurodytus žingsnius įdiegtas „MacroBase SQL“ įrankis:

1. Atidarytas „Ubuntu“ terminalas.
2. Klonuota projekto repozitorija:

   ```
   git clone https://github.com/stanford-futuredata/macrobase.git
   ```

3. Pereita į „MacroBase“ naujai sukurtą direktoriją:

   ```
   cd macrobase
   ```

4. Paleistas „MacroBase SQL“ kompiliavimo ir paruošimo BASH skriptas:

   ```
   ./build.sh sql
   ```

5. Patikrinta ar diegimas atliktas teisingai ir „MacroBase SQL“ įrankis veikia korektiškai pasinaudojus demonstraciniais duomenimis:

   ```
   bin/macrobase-sql -f sql/demo.sql
   ```

1.3. Eksperimento vykdymo eiga

Šiame poskyryje aprašoma eksperimento vykdymo eiga naudojant IPSS pateiktus 2014–2018 metų duomenų rinkinius naudojantis „MacroBase SQL“ analitinį įrankį.

1.3.1. Konfiguracija

„MacroBase SQL“ paleidimui buvo naudojama komandinė eilutė, kuriai paduodami SQL formato failai. Šiuose failuose nurodomi tokia informacija:

-

1.3.2. Eksperimento eiga

Šiame skyriuje aprašyta eksperimento eiga

Rezultatai ir išvados

Rezultatų ir išvadų dalyje išdėstomi pagrindiniai darbo rezultatai (kažkas išanalizuota, kažkas sukurta, kažkas įdiegta), toliau pateikiamos išvados (daromi nagrinėtų problemų sprendimo metodų palyginimai, siūlomos rekomendacijos, akcentuojamos naujovės). Rezultatai ir išvados pateikiami sunumeruotų (gali būti hierarchiniai) sąrašų pavidalu. Darbo rezultatai turi atitikti darbo tikslą.

Literatūra

[Ast16] Anthony Asta. Observability at twitter: technical overview, part i. 2016. URL: https://blog.twitter.com/engineering/en_us/a/2016/observability-at-twitter-technical-overview-part-i.html (tikrinta 2019-03-10).

[BGR+17] Peter Bailis, Edward Gan, Kexin Rong ir Sahaana Suri. Prioritizing attention in fast data: principles and promise. *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research*. Stanford Infolab, 2017.

[EMC14] Dell EMC. The digital universe of opportunities: rich data and the increasing value of the internet of things. 2014. URL: URL:%20http://www.emc.com/leadership/%20digital-universe/.

[Mak17] Aybeniz S. Aliyeva Makrufa Sh. Hajirahimova. About big datameasurement methodologies and indicators. *I.J.Modern Education and Computer Science*, p. 1–9, 2017-10. DOI: 10.5815/ijmecs.2017.10.01.

[PFT+15] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza ir Kaushik Veeraraghavan. Gorilla: a fast, scalable, in-memory time series database. *VLDB Endowment - Proceedings of the 41st International Conference on Very Large Data Bases*, p. 1816–1827, Kohala Coast, Hawaii. Facebook Inc., 2015. (Tikrinta 2019-03-10).

[rrtar] Lietuvos Respublikos ryšių reguliavimo tarnyba. Interneto prieigos stebėsenos sistema 2014-2018 metais. URL: http://matavimai.rrt.lt/about_archive.html (tikrinta 2019-05-20).

[Sim71] Herbert A. Simon. Designing organizations for an information-rich world. in computers, communications, and the public interest. Martin Greenberger, redaktorius, *Computers, Communication, and the Public Interest*, p. 37–72, 1971.

[Smo14] Sandy Smolan. The human face of big data. Trumpametražis dokumentinis filmas, 2014-04-10. URL: https://humanfaceofbigdatafilm.com/ (tikrinta 2019-05-15).

[Tec19] Techopedia. Anomaly detection. 2019. URL: https://www.techopedia.com/definition/30297/anomaly-detection (tikrinta 2019-03-07).

[Woo15] Alex Woodie. Kafka tops 1 trillion messages per day at linkedin. 2015. URL: https://www.datanami.com/2015/09/02/kafka-tops-1-trillion-messages-per-day-at-linkedin/ (tikrinta 2019-03-10).

Santrumpos

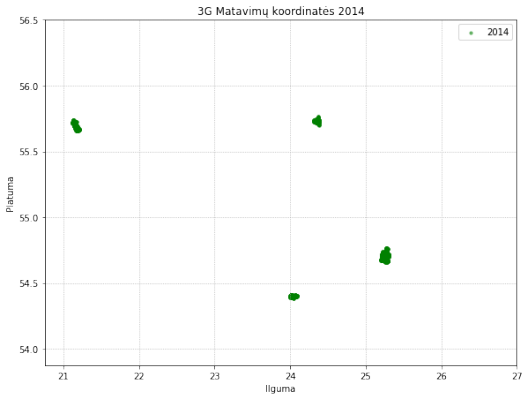
- LRTC
- CSV
- 3G
- LTE
- MAC adresas
- IPSS
- URL
- API
- HTTP
- JSON
- JRE
- JDK
- BASH
- SQL

Sąvokų apibrėžimai ir santrumpų sąrašas sudaromas tada, kai darbo tekste vartojami specialūs paaiškinimo reikalaujantys terminai ir rečiau sutinkamos santrumpos.

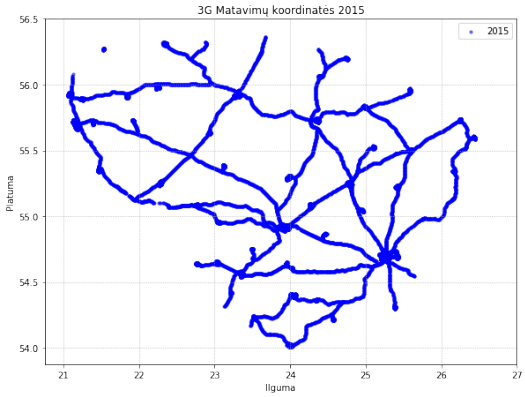
Priedas nr. 1

IPSS matavimų koordinatės 2014–2018 metais

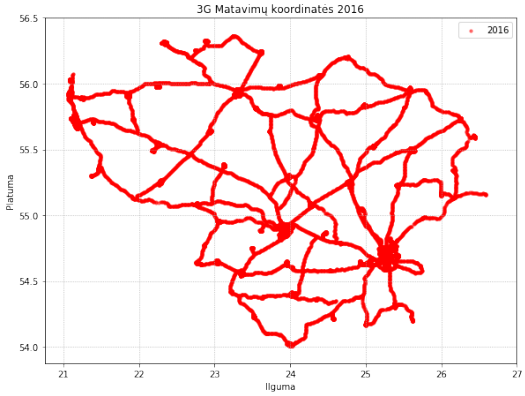
IPSS **3G** ryšio technologijos matavimai 2014–2018 metais. Viso 136003 įrašai.



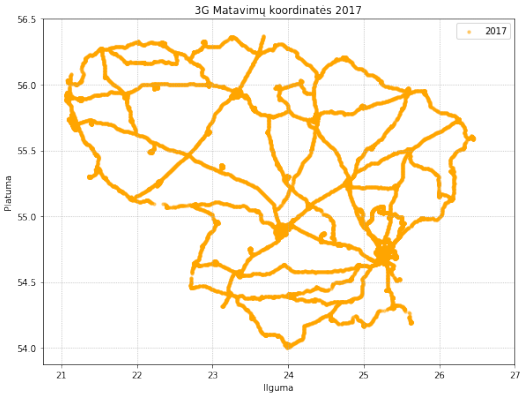
4 pav. 3G Matavimai 2014 metais. Viso – 2245 matavimai



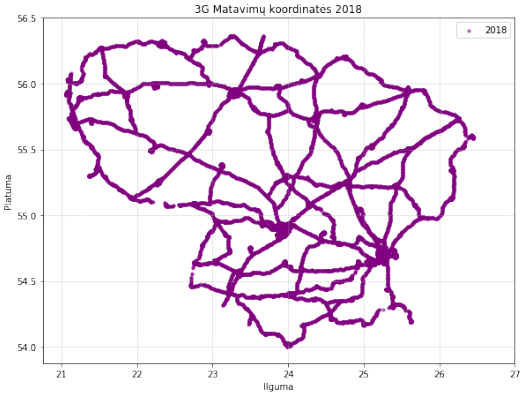
5 pav. 3G Matavimai 2015 metais. Viso – 19427 matavimai



6 pav. 3G Matavimai 2016 metais. Viso – 34769 matavimai

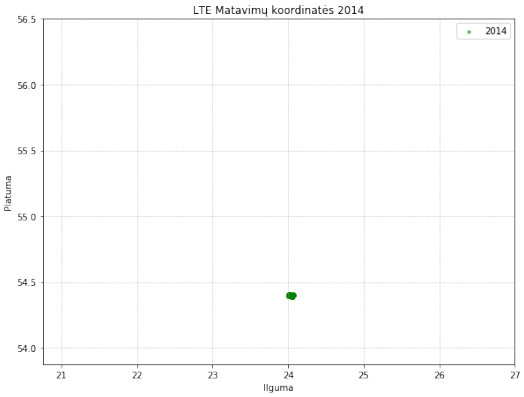


7 pav. 3G Matavimai 2017 metais. Viso – 38789 matavimai

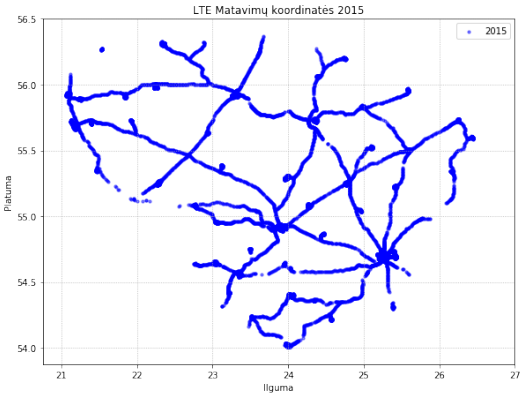


8 pav. 3G Matavimai 2018 metais. Viso – 40773 matavimai

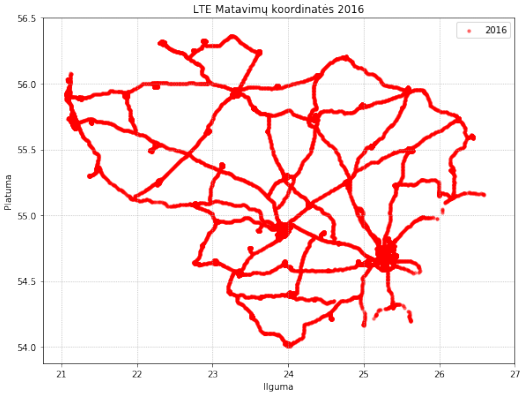
IPSS **LTE** ryšio technologijos matavimai 2014–2018 metais. Viso 157522 įrašai.



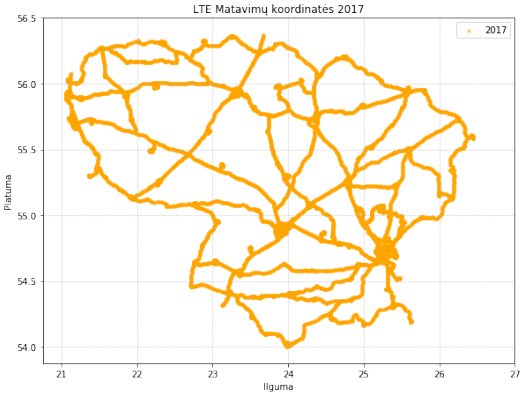
9 pav. LTE Matavimai 2014 metais. Viso – 232 matavimai



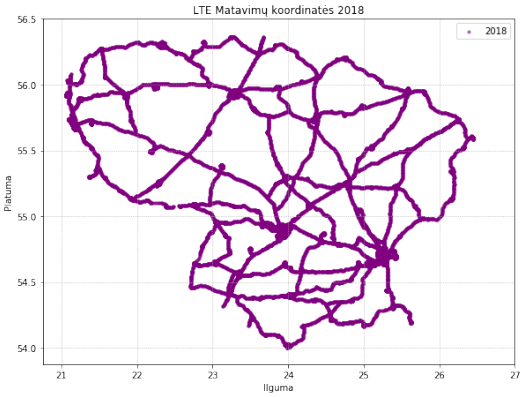
10 pav. LTE Matavimai 2015 metais. Viso – 13962 matavimai



11 pav. LTE Matavimai 2016 metais. Viso – 40738 matavimai

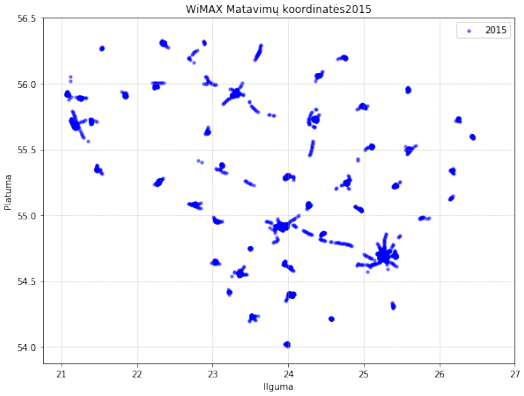


12 pav. LTE Matavimai 2017 metais. Viso – 49672 matavimai

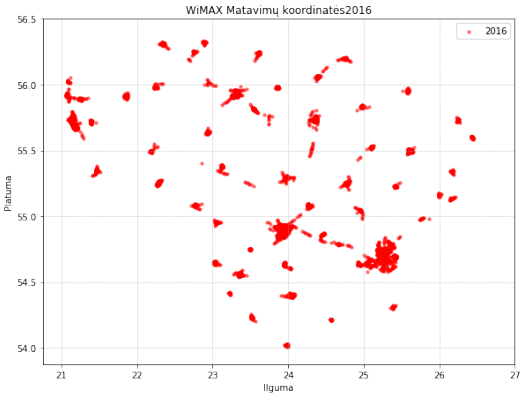


13 pav. LTE Matavimai 2018 metais. Viso – 52918 matavimai

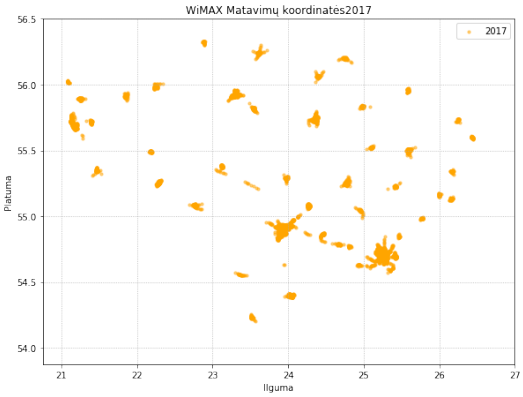
IPSS **WiMAX** ryšio technologijos matavimai 2014–2018 metais. Viso 18166 įrašai.



14 pav. WiMAX Matavimai 2015 metais. Viso – 4307 matavimai



15 pav. WiMAX Matavimai 2016 metais. Viso – 7653 matavimai



16 pav. WiMAX Matavimai 2017 metais. Viso – 6206 matavimai

Priedas nr. 2

Pradinių IPSS CSV failų apdorojimo „Python“ skriptas

„Python“ skriptas skirtas iš duotųjų IPSS duomenų CSV failų atrinkti nereikalingus laukus ir pervadinti lietuviškus antraštės pavadinimus į vieno žodžio (vienos simbolių eilutės) angliškus pavadinimus bei sugeneruoti naujus CSV failus:

```
import csv
file_3G_csv = 'Data/Raw/Matavimai-3G.csv'
file_LTE_csv = 'Data/Raw/Matavimai-LTE.csv'
file_WiMAX_csv = 'Data/Raw/Matavimai-WiMAX.csv'
new_file_3G_csv = 'Data/Processed/measurements-3G.csv'
new_file_LTE_csv = 'Data/Processed/measurements-LTE.csv'
new_file_WiMAX_csv = 'Data/Processed/measurements-WiMAX.csv'

with open(new_file_3G_csv, mode='w') as new_3G_csv_file:
  writer_3G = csv.writer(new_3G_csv_file, delimiter=',', quotechar='"',
   quoting=csv.QUOTE_MINIMAL)
  writer_3G.writerow(['year', 'latitude', 'longitude', 'provider', 'cell_id',
   'rssi', 'technology', 'speed'])
  with open(file_3G_csv , mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    line_count = 0
    for row in csv_reader:
      year = row['data ir laikas'][:4]
      latitude = row['platuma']
      longitude = row['ilguma']
      provider = row['operatorius']
      cell_id = row['ėcels id']
      rssi = row['rssi']
      technology = row['3G šryio technologija']
      speed = row['sparta kbit/s']
      writer_3G.writerow([year, latitude, longitude, provider, cell_id, rssi,
   technology, speed])
      line_count += 1
    print(f'Processed {line_count} 3G CSV file lines.')

with open(new_file_LTE_csv, mode='w') as new_LTE_csv_file:
  writer_LTE = csv.writer(new_LTE_csv_file, delimiter=',', quotechar='"',
   quoting=csv.QUOTE_MINIMAL)
  writer_LTE.writerow(['year', 'latitude', 'longitude', 'provider', 'cell_id',
```

```
    'rssi', 'rsrp', 'speed'])
  with open(file_LTE_csv , mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    line_count = 0
    for row in csv_reader:
      year = row['data ir laikas'][:4]
      latitude = row['platuma']
      longitude = row['ilguma']
      provider = row['operatorius']
      cell_id = row['ėcels id']
      rssi = row['rssi']
      rsrp = row['rsrp']
      speed = row['sparta kbit/s']
      writer_LTE.writerow([year, latitude, longitude, provider, cell_id, rssi,
    rsrp, speed])
      line_count += 1
    print(f'Processed {line_count} LTE CSV file lines.')

with open(new_file_WiMAX_csv, mode='w') as new_WiMAX_csv_file:
  writer_WiMAX = csv.writer(new_WiMAX_csv_file, delimiter=',', quotechar='"',
    quoting=csv.QUOTE_MINIMAL)
  writer_WiMAX.writerow(['year', 'latitude', 'longitude', 'base_station_id', '
    rssi', 'cinr', 'speed'])
  with open(file_WiMAX_csv , mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    line_count = 0
    for row in csv_reader:
      year = row['data ir laikas'][:4]
      latitude = row['platuma']
      longitude = row['ilguma']
      base_station_id = row['ėbazins stoties id']
      rssi = row['rssi']
      cinr = row['cinr']
      speed = row['sparta kbit/s']
      writer_WiMAX.writerow([year, latitude, longitude, base_station_id, rssi,
    cinr, speed])
      line_count += 1
    print(f'Processed {line_count} WiMAX CSV file lines.')
```

Priedas nr. 3

Atrinktų IPSS CSV failų duomenų papildymo „Python“ skriptas

„Python“ skriptas skirtas jau atrinktiems CSV failams pridėti papildomus laukus: aukščio virš jūros lygio (gaunamas iš koordinačių), suapvalintų koordinačių reikšmių bei suapvalintos aukščio reikšmės. Šiems duomenims yra sugeneruojamas nauji CSV failai:

```
import csv
import numpy as np
import pandas as pd
import urllib.parse as up
import urllib.request as ur
import json
new_file_3G_csv = 'Data/Processed/measurements-3G.csv'
new_file_LTE_csv = 'Data/Processed/measurements-LTE.csv'
new_file_WiMAX_csv = 'Data/Processed/measurements-WiMAX.csv'
more_data_file_3G_csv = 'Data/Processed/more-measurements-3G.csv'
more_data_file_LTE_csv = 'Data/Processed/more-measurements-LTE.csv'
more_data_file_WiMAX_csv = 'Data/Processed/more-measurements-WiMAX.csv'
token = "XPlYEbsLssXaMRcPolV9txdQyrZ0LrOTA5Hn9bq6SW7jvnb92rsKQZbx2augAEpL"
main_url = "https://api.jawg.io/elevations?"
request_step = 100
round_digits = 1
new_X_3G = []
new_X_LTE = []
new_X_WiMAX = []

read_3G = pd.read_csv(new_file_3G_csv)
X_3G = np.array(read_3G)
y_3G = np.array(read_3G.columns)
lat_3G = np.where(y_3G == 'latitude')[0][0]
long_3G = np.where(y_3G == 'longitude')[0][0]
lats_3G = X_3G[0:,lat_3G]
longs_3G = X_3G[0:,long_3G]
prev_r = 0
len_3G = len(X_3G)
locations_3G = []
for r in range(request_step, len_3G + request_step, request_step):
  locs_3G = []
  for i in range(prev_r, r):
    if i >= len_3G:
```

```
      break
    locs_3G.append(str(lats_3G[i]) + ',' +  str(longs_3G[i]))
  prev_r = r
  locations_3G.append("|".join(locs_3G))
elevation_3G = np.array([])
for loc in locations_3G:
  params = {'locations': loc, 'access-token' : token}
  full_url = main_url + up.urlencode(params)
  request = ur.urlopen(full_url)
  contents = request.read()
  request.close()
  jdata = json.loads(contents)
  el = np.array([j['elevation'] for j in jdata])
  elevation_3G = np.concatenate((elevation_3G, el))
new_X_3G = []
for i in range(len_3G):
  item = X_3G[i]
  new_item = []
  new_item.append(round(item[lat_3G], round_digits))
  new_item.append(round(item[long_3G], round_digits))
  new_item.append(elevation_3G[i])
  new_item.append(round(elevation_3G[i] / 10, 0) * 10)
  complete_item = np.append(np.array(item), np.array(new_item))
  new_X_3G.append(complete_item)
new_X_3G = np.array(new_X_3G)
with open(more_data_file_3G_csv, mode='w') as new_3G_csv_file:
  writer_3G = csv.writer(new_3G_csv_file, delimiter=',', quotechar='"',
   quoting=csv.QUOTE_MINIMAL)
  writer_3G.writerow(['year', 'latitude', 'longitude', 'altitude', 'lat_1', '
   long_1', 'alt_10', 'provider', 'cell_id', 'rssi', 'technology', 'speed'])
  for item in new_X_3G:
    writer_3G.writerow([item[0], item[1], item[2], item[10], item[8], item[9],
     int(item[11]), item[3], item[4], item[5], item[6], item[7]])

read_LTE = pd.read_csv(new_file_LTE_csv)
X_LTE = np.array(read_LTE)
y_LTE = np.array(read_LTE.columns)
lat_LTE = np.where(y_LTE == 'latitude')[0][0]
long_LTE = np.where(y_LTE == 'longitude')[0][0]
lats_LTE = X_LTE[0:,lat_LTE]
```

```
longs_LTE = X_LTE[0:,long_LTE]
prev_r = 0
len_LTE = len(X_LTE)
locations_LTE = []
for r in range(request_step, len_LTE + request_step, request_step):
  locs_LTE = []
  for i in range(prev_r, r):
    if i >= len_LTE:
      break
    locs_LTE.append(str(lats_LTE[i]) + ',' +  str(longs_LTE[i]))
  prev_r = r
  locations_LTE.append("|".join(locs_LTE))
elevation_LTE = np.array([])
for loc in locations_LTE:
  params = {'locations': loc, 'access-token' : token}
  full_url = main_url + up.urlencode(params)
  request = ur.urlopen(full_url)
  contents = request.read()
  request.close()
  jdata = json.loads(contents)
  el = np.array([j['elevation'] for j in jdata])
  elevation_LTE = np.concatenate((elevation_LTE, el))
new_X_LTE = []
for i in range(len_LTE):
  item = X_LTE[i]
  new_item = []
  new_item.append(round(item[lat_LTE], round_digits))
  new_item.append(round(item[long_LTE], round_digits))
  new_item.append(elevation_LTE[i])
  new_item.append(round(elevation_LTE[i] / 10, 0) * 10)
  complete_item = np.append(np.array(item), np.array(new_item))
  new_X_LTE.append(complete_item)
new_X_LTE = np.array(new_X_LTE)
with open(more_data_file_LTE_csv, mode='w') as new_LTE_csv_file:
  writer_LTE = csv.writer(new_LTE_csv_file, delimiter=',', quotechar='"',
   quoting=csv.QUOTE_MINIMAL)
  writer_LTE.writerow(['year', 'latitude', 'longitude','altitude', 'lat_1', '
   long_1', 'alt_10', 'provider', 'cell_id', 'rssi', 'rsrp', 'speed'])
  for item in new_X_LTE:
    writer_LTE.writerow([item[0], item[1], item[2], item[10], item[8], item
```

```
    [9], int(item[11]), item[3], item[4], item[5], item[6], item[7]])

read_WiMAX = pd.read_csv(new_file_WiMAX_csv)
X_WiMAX = np.array(read_WiMAX)
y_WiMAX = np.array(read_WiMAX.columns)
lat_WiMAX = np.where(y_WiMAX == 'latitude')[0][0]
long_WiMAX = np.where(y_WiMAX == 'longitude')[0][0]
lats_WiMAX = X_WiMAX[0:,lat_WiMAX]
longs_WiMAX = X_WiMAX[0:,long_WiMAX]
prev_r = 0
len_WiMAX = len(X_WiMAX)
locations_WiMAX = []
for r in range(request_step, len_WiMAX + request_step, request_step):
  locs_WiMAX = []
  for i in range(prev_r, r):
    if i >= len_WiMAX:
      break
    locs_WiMAX.append(str(lats_WiMAX[i]) + ',' +  str(longs_WiMAX[i]))
  prev_r = r
  locations_WiMAX.append("|".join(locs_WiMAX))
elevation_WiMAX = np.array([])
for loc in locations_WiMAX:
  params = {'locations': loc, 'access-token' : token}
  full_url = main_url + up.urlencode(params)
  request = ur.urlopen(full_url)
  contents = request.read()
  request.close()
  jdata = json.loads(contents)
  el = np.array([j['elevation'] for j in jdata])
  elevation_WiMAX = np.concatenate((elevation_WiMAX, el))
new_X_WiMAX = []
for i in range(len_WiMAX):
  item = X_WiMAX[i]
  new_item = []
  new_item.append(round(item[lat_WiMAX], round_digits))
  new_item.append(round(item[long_WiMAX], round_digits))
  new_item.append(elevation_WiMAX[i])
  new_item.append(round(elevation_WiMAX[i] / 10, 0) * 10)
  complete_item = np.append(np.array(item), np.array(new_item))
  new_X_WiMAX.append(complete_item)
```

```
149 new_X_WiMAX = np.array(new_X_WiMAX)
150 with open(more_data_file_WiMAX_csv, mode='w') as new_WiMAX_csv_file:
151   writer_WiMAX = csv.writer(new_WiMAX_csv_file, delimiter=',', quotechar='"',
      quoting=csv.QUOTE_MINIMAL)
152   writer_WiMAX.writerow(['year', 'latitude', 'longitude', 'altitude', 'lat_1',
       'long_1', 'alt_10', 'base_station_id', 'rssi', 'cinr', 'speed'])
153   for item in new_X_WiMAX:
154     writer_WiMAX.writerow([item[0], item[1], item[2], item[9], item[7], item
      [8], int(item[10]), item[3], item[4], item[5], item[6]])
```

Priedas nr. 4

Eksperimentinio palyginimo rezultatai

1 lentelė. Lentelės pavyzdys

Algoritmas	$\bar{x}$	σ^2
Algoritmas A	1.6335	0.5584
Algoritmas B	1.7395	0.5647