Niklas Salonen

Ajotavan analysointi paikannustiedosta

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

1.4.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Niklas Salonen Ajotavan analysointi paikannustiedosta 50 sivua 1.4.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Tuotepäällikkö Tero Laitila Lehtori Vesa Ollikainen

Insinöörityössä oli tavoitteena laatia laskentamenetelmä ajotavan analysoimiseen paikannustiedosta sekä kehittää ohjelma, joka laskentamenetelmän avulla luokittelisi ajoneuvoja. Laskentamenetelmän kuului löytää ajoneuvojen paikkatiedoista tapahtumat, jolloin ajoneuvo on kiihdyttänyt tai jarruttanut liian kovaa. CGI:n asiakkaiden kannalta oli olennaista selvittää, mitkä ajoneuvot ajavat poikkeuksellisesti sekä mahdollistaa ajoneuvojen välinen vertailu ajotavan perusteella.

Työn alkuvaiheessa laskentamenetelmän kehittäminen oli insinöörityön kannalta kriittisintä, koska ohjelmointia ei voinut aloittaa ennen sen laatimista. Laskentamenetelmä laadittiin ajoneuvoseurantajärjestelmän rajoitteet huomioon ottaen mahdollisimman yleiskäyttöiseksi. Tavoitteena oli saada laskentamenetelmä ja sen myötä myös ohjelma, joka toimisi saumattomasti useammassa eri järjestelmässä.

Insinöörityössä tehtiin kaksi paikannustietoa käsittelevää PHP-ohjelmaa. Raskaat laskutyöt suoritettiin yhdessä ja huomattavasti kevyemmät luokittelutyöt suoritettiin toisessa. Kaksijakoisuuden ansiosta pystyttiin keskittymään molempien osien itsenäiseen kehittämiseen ja optimointiin. Työn aikana menetelmiä paranneltiin, ja ohjelmat saivat jatkuvasti parannuksia sekä enemmän ominaisuuksia.

Lopputuloksena syntyi toimiva laskentamenetelmä sekä ohjelmat, jotka tuottavat arvokasta tietoa asiakkaille. Aikaisemmin ajoneuvoseurantajärjestelmissä ei ollut ajoneuvojen välistä vertailua, joka nyt mahdollistettiin.

Projektin kokemukset osoittivat suunnittelun tärkeyden pitkäkestoisissa laskennoissa, jotka virhetilanteissa joutui pahimmassa tapauksessa käynnistämään alusta uudestaan. Insinöörityössä valmistuneet ohjelmat tulevat tarjoamaan asiakkaille täysin uudenlaista tietoa ajoneuvokalustonsa käytöstä.

Avainsanat	ajotapa, paikannustieto, big data, luokittelu

Author(s) Title Number of Pages Date	Niklas Salonen Driving Pattern Analysis Based on Position Data 50 pages 1 April 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Tero Laitila, Product Manager Vesa Ollikainen, Lecturer

The study had two goals. The first goal was to develop a counting method to analyze driving patterns from position data. The second goal was to develop a program which would classify vehicles based on the results of the counting method. The counting method was supposed to find in the position data of the vehicle, all the events when the vehicle had been accelerating or braking too hard. From the perspective of CGI customers, it was critical to find out which vehicles were driving abnormally and to enable comparison between vehicles based on their driving patterns.

In the early stages of the study it was critical to develop the counting method first, because the programming was dependent on the counting method. The counting method was developed with the limitations of the vehicle tracking system in mind, as well to be as widely usable as possible. The aim was to create a counting method and with it a program that would work seamlessly in different systems.

For the study, two separate PHP-programs were developed to handle position data. The heavy calculations were done in one and the considerably less time consuming groupings in the other. Because of this twofold approach, both parts could be developed and optimized independently. During the project, the methods were improved and the programs continually got amendments and new features.

The result of the study was a functional counting method, as well as programs which produce valuable information for the client. Previously it was impossible to do comparisons between vehicles, and now this is possible.

The experience from the project showed the importance of planning when dealing with time consuming calculations, which in the worst case scenario have to be restarted if an error is encountered. The programs created here will provide the clients with completely new kind of information about the usage of their vehicle fleet.

Keywords	driving behavior, position data, big data, classification

Sisällys

1 Johdanto

Tähän työn tarkoitus on perehtyä ajotavan analysoinnin haasteisiin suurissa tietoaineistoissa sekä kehittää ohjelma, joka tekee ajotapa-analyysiä paikannustietoon perustuen. Ohjelman käytettävissä on paikannustietoa yli tuhannesta autosta sekä muutamasta sadasta bussista. Ajoneuvot ovat asiakkaitten omia, ja kyseessä ovat eri asiakkuudet. Ohjelma pyrkii havaitsemaan raja-arvojen ylittäviä nopeuseroja paikannustiedosta, jotka kielivät siitä, että kuski ei aja tasaisesti. Näiden tulosten perusteella ajoneuvoja pitäisi pystyä vertailemaan keskenään. Sovellus käyttää olemassa olevia tietokantoja ja toimii nykyisten ohjelmien rinnalla kirjoittaen kantaan tuloksensa.

Työ tehdään CGI Suomi Oy:lle, joka on Suomessa yli 3000 työntekijää työllistävä IT-alan yritys. CGI on myynyt asiakkailleen ajoneuvonseurantajärjestelmät, ja niiden tuottaman paikkatiedon perusteella työni laskenta suoritetaan. Ajoneuvonseurantajärjestelmät ovat kattavia kokonaisuuksia kaluston seurantaan ja hallintaan. Tästä raportista on jätetty pois ajoneuvoja yksilöivät tiedot sekä muut salassapitovelvollisuuden varassa olevat tiedot. CGI voi hyödyntää työn tulosta myymällä ohjelman tuottamat tulokset ominaisuutena nykyisille asiakkailleen. Vaihtoehtoisesti työn lopputulosta voi laajentaa laskemaan muutakin metriikkaa. Sovelluksen tuotos voidaan myös markkinoida tuleville asiakkaille järjestelmän ominaisuutena. Järjestelmä ei aikaisemmin pystynyt vertailemaan ajoneuvojen ajotapoja keskenään, ja tämä työ tuo siihen ratkaisun.

Työn ensimmäinen tavoite on soveltuvan laskentamenetelmän kehittäminen, joka tuottaisi metriikkaa eli arvoja ajotavan perusteella, minkä avulla ajoneuvoja voi luokitella sekä verrata toisiinsa. Työssä keskitytään siihen, miten kyseistä metriikkaa voi laskea sekä suoritusnopeuden optimointiin valtavan paikannustietomäärän takia. Työn hyöty ilmenee myös nykyisille asiakkaille, mikäli he kaipaavat tapaa verrata ajoneuvoja keskenään. Toinen tavoite työlle on kehittää ohjelma, joka osaa käydä tehokkaasti läpi suuren määrän paikannustietoa ja laskea nopeusvaihteluun perustuen arvon ajotavalle, jonka avulla voidaan ajoneuvoja luokitella ja vertailla keskenään.

Ohjelma tehdään PHP-ohjelmointikielellä ja tietokanta on MySQL-tietokanta. Ohjelmani jakautuu kahteen osaan. Ensimmäisen ohjelman tehtävä on käydä läpi paikannustieto ja tallentaa tietokantaan tapahtumat, joissa ajoneuvo on toiminut liian nopeasti sekä

muita luokitteluun hyödyllisiä laskettuja arvoja. Toisen ohjelman tehtävänä on tallennettujen tapahtumien ja arvojen läpikäyminen ja sen pohjalta analysoida ja tuottaa vertailtavaa metriikkaa.

Työn haasteena ovat suuret tietomäärät ja niiden tehokas käsittely. Kookkaitten tietomäärien läpikäyminen vie metodista riippumatta paljon aikaa, ja mahdollisissa virhetilanteissa tiedon uudestaan laskeminen hidastaa työn edistymistä huomattavasti. Työn luettuaan lukija on paremmin perillä eri paikannustietoon perustuvista ajotavan analysointimenetelmistä. Tämän lisäksi lukija saa tarkan kuvan yhdestä menetelmästä analysoida ajotapaa paikannustiedon perusteella.

2 Ongelman kuvaus

Ajotapa-analyysiä tulen tekemään paikannustietoon perustuen, ja lähteestä riippuen tietoa tulee joko kerran sekunnissa tai viiden sekunnin välein. Paikannustieto sisältää tiedon ajoneuvon koordinaateista, nopeudesta, suunnasta sekä aikaleimasta, jolloin tieto on tuotettu. Autokannan paikantimet lähettävät paikannustietonsa kerran viidessä sekunnissa. Näin ollen työni tulee myös osoittamaan, pystyykö tällä päivitystiheydellä tekemään johtopäätöksiä. Linja-autot päivittävät joka sekunti, joten ohjelmani pystyy paremmin tekemään analyysiä ja arvioimaan niitä.

Kalustojen lukumäärän takia sekä nopean päivitystiheyden takia läpikäytävä tietomäärä on valtava. Tästä syntyy yksi ohjelman keskeisistä haasteista: miten optimoida ohjelma niin, että se suoriutuu laskentatyöstään mahdollisimman nopeasti ja näin ollen lakkaa kuormittamasta palvelimia. Ohjelmalla itsellään ei ole ulkopuolisen tahosta asetettu nopeusvaatimuksia, sillä ajotapametriikka lasketaan ja analysoidaan menneistä tapahtumista eikä ole reaaliaikaista.

Isoissa tietomäärissä käytetään yleensä käsitettä ”Big data”, joka kuvaa valtavaa tietomäärää. Big data on käsite, joka on yleistymässä, ja huomaamme, miten isoja tietomääriä käytetään nykyään eri tavalla kuin aiemmin. Aiemmin emme pystyneet käsittelemään valtavia tietomääriä laskentatehon puutteen takia. Nykyään laskentatehon kasvettua käytämme valtavia tietomääriä aivan eri tavalla hyväksi. Facebook, Google ja muut isot yritykset tallentavat valtavia tietomääriä ja käyttävät niitä markkinointiin, ennustamaan käyttäytymistä ja tarjoamaan yksilöityjä ratkaisuja käyttäjälle. Algoritmit,

jotka nämä yritykset käyttävät, ovat niin raskaita, että niitä ei olisi voinut internetin syntyessä käyttää mutta nykyään ne ovat arkipäivää [1, s.19–21.]

Suuret tietomäärät voivat myös aiheuttaa ongelmia, jos tietoon luottaa sokeasti. Kun tietoa on valtavasti, syyllistytään helposti tiedon ylianalysointiin, ilman että otetaan inhimilliset seikat huomioon. Aina ei tieto kerro koko kuvaa vain, koska sitä on reilusti, sattumat, muutokset ja pienet yksityiskohdat jäävät helposti huomaamatta, jos ainoastaan analysoimme valtavia tietomääriä [1]. Sama koskee työtäni: saavutetut tulokset eivät välttämättä kerro koko kuvaa, mutta sillä pitäisi voida osoittaa eroja ajoneuvojen välillä. Syyt eroihin ei voi täydellä varmuudella osoittaa työni tuloksella.

Ensimmäinen selkeä ongelma, joka aiheutuu suuresta datamäärästä, ilmaantuu hakemalla liian pitkältä aikaväliltä paikannustiedot yhdeltä paikantimelta tietokannasta. Liian isoa tietomäärää kerralla käsiteltäessä saattaa palvelimen PHP-sovelluksille varattu muisti loppua kesken ja ohjelma kaatua. Näin käy, jos ajoneuvo on ollut aktiivisesti ajossa ja sen seurauksena tuottanut paljon paikannustietoa. Ajoneuvo, joka on ajanut paljon, voi tuottaa yli 200 000 riviä paikannustietoa kuukaudessa jo pelkästään autokannassa. Eli kyselyitä tehdessä pitää harkita tarkkaan, kuinka pitkältä aikaväliltä voi kyselyn tehdä. Tekemällä liian lyhyeltä aikaväliltä kyselyjä nousee kyselyjen määrä, ja kun kyselyt tehdään valtavaan tietotauluun, hidastaa se ohjelmaa merkitsevästi. Toisaalta liian pitkä aikaväli taas kaataa ohjelman, joten tasapaino näiden kahden ongelman välille on löydyttävä.

Toinen ongelma, johon työssä törmätään, on se, että ajoneuvoista ei ole saatavilla kiihdytysanturitietoa. Tämä johtaa siihen, että analyysi kohdistetaan ainoastaan ajoneuvojen nopeusvaihteluun. Yleensä analyysiä tehdään kiihtyvyysanturin avulla tai sekunnin päivitystiheydellä kuten Constantinescu [2, s 3] joten olemassa olevaa lähdekirjallisuutta ja tutkimuksia on tehty harvemmin ilman sitä. Tämän takia työni tarjoaa uudenlaisen näkökulman siihen, miten ajotapa-analyysiä voi tehdä rajoitetummalla tietomäärällä. Yhtä selkeitä johtopäätöksiä ei varmaankaan voi tämän takia tehdä autokaluston kohdalla päivitystiheyden takia, mutta työni tulisi pystyä poimimaan esiin ne autot, jotka ovat selkeästi keskijakauman ulkopuolella tietomäärän laajuuden ansiosta. Linja-autokaluston päivitystiheys mahdollistaa sen, että niitä voi luokitella paremmin, analyysi on luotettavampaa ja keskinäiset erot ovat selkeästi nähtävissä.

Nykyisellään ajoneuvojen keskinopeuksia ei lasketa eikä vertailua ajoneuvojen välillä tehdä lainkaan. Tämän ohjelmani pyrkii ratkaisemaan ja tarjoamaan pohjan jatkokehitystä varten. Nykyisessä tietokannassa ei ole valmiiksi sopivia tauluja, joihin ohjelmani voisi kirjoittaa tuloksiaan joten työhöni kuuluu myös näiden taulujen luominen. Ohjelmani voi käyttää valmiita kirjastoja hyväkseen osittain, mutta pääasiassa joudun tekemään omia ratkaisuja tietokantakyselyistä laskentalogiikkaan.

3 Ajotapa-analyysin teko paikannusdatasta

3.1 Tausta

Ajotapa-analyysi raakadatasta on kehittyvä aihealue, ja siihen tullaan jatkuvasti kiinnittämään enemmän huomiota [3]. Seuraamalla ajoneuvokalustoa voidaan saavuttaa merkittäviä säästöjä sekä optimoida reittejä, jos havaitaan tietyn reitin aiheuttavan enemmän äkillisiä jarrutuksia kuin vaihtoehtoinen reitti. Kuljettajien seuranta ja kannustepalkkauksella voi Jolen [4, s. 2] raportin arvion mukaan nostaa energiatehokkuutta 5-15 %.

Ajotapa-analyysin avulla yritykset voivat seurata kalustoaan, ja tulosten perusteella vaikuttaa kuljettajiensa ajotapoihin. Esimerkiksi jos järjestelmä huomaa, että kuljettajalla tulee toistuvasti paljon kiihdytyksiä ja jarrutuksia, voi esimies huomauttaa asiasta. Ajotapa missä kuljettaja kiihdyttää ja jarruttaa paljon, aiheuttaa tutkitusti eniten päästöjä, ja se kuluttaa eniten polttoainetta [5, s.543–554]. Vaihtoehtoinen lähestymistapa on palkita kuljettajia, jotka aiheuttavat vähän tapahtumia järjestelmään. Yritys voisi myös palkita kuljettajaa, joka onnistuu merkittävästi parantamaan ajonsa taloudellisuutta. Riippumatta siitä, miten yritys pyrkisi hyödyntämään järjestelmää, on järjestelmällä rutkasti potentiaalia merkittäviin säästöihin. Toisin sanottuna se nostaa kaluston kustannustehokkuutta vähentämällä polttoaineenkulutusta ja voi myös vähentää onnettomuuksia ja kaluston rikkoutumista kannustamalla turvallisempaan ja taloudellisempaan ajotapaan.

Yritykset ovat kiinnostuneita tästä toiminnallisuudesta useasta muustakin syystä. Kaluston energiatehokkuuden parantaminen on merkittävässä roolissa, kun ajatellaan kestävää kehitystä. Yritys, joka haluaa antaa kuvan siitä, että vihreät arvot ja ekologisuus on tärkeää, arvostaa järjestelmän tuomien säästöjen lisäksi sen tuomaa imagon

kohennusta. Polttoaineenkulutuksen vähentymisen jälkeen voisi yritys markkinoinnissaan käyttää tätä esimerkkinä siitä, miten yritys on sitoutunut vähentämään päästöjä ja kantamaan vastuuta tulevaisuudesta. Ilmastonmuutoksen myötä yritykset ovat entistä halukkaampia tekemään vihreitä päätöksiä ja etenkin tapauksissa, missä vihreät arvot ja kustannustehokkuus kohtaavat, voittavat kaikki osapuolet.

Paikannustieto sisältää vähintään tietoa GPS-sijainnista, nopeudesta sekä aikaleimasta, jonka perusteella analyysi suoritetaan [2, s 3]. Paikannin pystyy myös kertomaan suunnan sekä sen, missä tilassa se on. Saatavilla oleva tieto riippuu ajoneuvossa olevasta paikantimesta, joka lähettää tiedon GPRS-yhteyden yli palvelimelle. Palvelin välittää tiedon eteenpäin tai tallentaa tiedon tietokantaan analyysiä varten. Mitä tiheämmin paikannin välittää paikannustietoa, sitä paremmin ja tarkemmin voimme analysoida ajoa [2]. Alla esimerkki muutamasta rivistä paikannustietoa:

Taulukko 1. Paikannustietoesimerkki. ”paik.id” tarkoittaa paikantimen tunnusta ja ”vast.AL” paikannustiedon vastaanottoaikaleimaa. ”gps.AL” tarkoittaa GPS-laitteen aikaleimaa, eli milloin GPS-järjestelmältä on saatu tieto, minkä pohjalta paikannustieto on luotu.

paik.id	leveysaste	pituusaste	nopeus	suunta	vast.AL	gps.AL
99999	60.152740	24.656110	20	240	1385746580	1385746579
99999	60.152814	24.655845	60	294	1385746584	1385746584
99999	60.152933	24.654721	70	294	1385746590	1385746589

Paikannustiedon (taulukko 1) perusteella voimme laskea ajoneuvon nopeusvaihtelut aikaan nähden ja analysoida tulokset. Paikannustiedosta voimme myös laskea keskinopeuden sekä erilaisten tapahtumien määrän. Esimerkissä (taulukko 1) keskinopeus olisi (20+60+70)/3 = 50 km/h. Tapahtumina voimme pitää nopeat kiihdytykset sekä äkilliset jarrutukset. Paikannustiedon ollessa erittäin tarkkaa ja tiheällä päivityksellä voisimme myös laskea useita muita arvoja ajoneuvolle kuten keskinopeus mutkissa ja päätellä, millä ajotyylillä ajoneuvo lähestyy liikennevaloja. Ajotyylejä on monia, mutta tässä esimerkissä eri ajotyylejä, voisi olla kuljettaja, joka jarruttaa viime hetkessä, tai kuski, joka on päästänyt irti kaasusta reilusti etukäteen ja lähestyy hitaasti valoja antaen ajoneuvon vieriä.

3.2 Luokittelu ja klusterointi

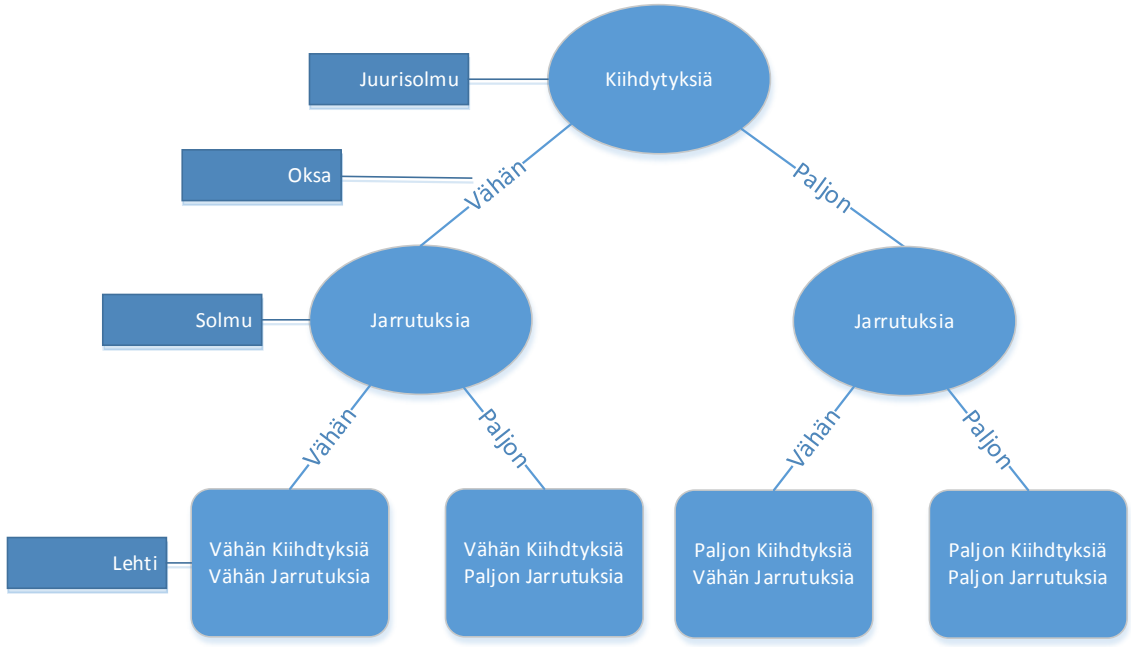
Tulen tiedon analysointivaiheessa käyttämään luokittelu- tai klusterointimenetelmää riippuen siitä, kumpi sopii paremmin tarpeisiini. Mikäli toinen menetelmä osoittautuu toista hyödyllisemmäksi, tulen perustelemaan, miksi ja miten tähän lopputulokseen päätyttiin. Klusteroinnista ei ole paljoakaan hyötyä, mikäli muuttujia on vain pari kappaletta, joten oleellisten muuttujien löytäminen on tärkeää.

Luokittelussa on kyse siitä, että ennakkoon arvioidaan, minkälaisia ja kuinka monta eri ryhmää tarvitaan [6]. Ryhmä voi koostua esimerkiksi ajoneuvoista, jotka kerryttävät paljon kiihdytystapahtumia ja kohtalaisesti jarrutustapahtumia. Toinen ryhmä voi koostua ajoneuvoista, jotka kerryttävät kohtalaisesti kiihdytystapahtumia mutta eivät lainkaan jarrutustapahtumia. Päättelemällä sopivia raja-arvoja näille ryhmille voimme analysointivaiheessa ryhmittää ajoneuvot ja verrata niitä keskenään. Tämän jälkeen voi esittää mielenkiintoisia kysymyksiä, kuten mikä ryhmä on suurin tai pienin ja onko ryhmien keskinopeuksien välillä eroa. Luokittelussa on tärkeää etukäteen analysoida ja määrittää, miten ryhmittelyn aikoo tehdä ja mitä sillä haluaa saavuttaa [6]. Luokittelumenetelmät voi jakaa kahteen ryhmään; erottelevat ja probabilistiset luokittelijat. Erotteleva luokittelija pakottaa kaikki alkiot johonkin luokkaan. Probabilistinen luokittelija määrittää mihin luokkaan alkio todennäköisimmin kuuluu [7, s.25–35].

Päätöspuu (kuva 1) on yksi tunnetuimmista luokittelualgoritmeista ja se on erotteleva luokittelija. Päätöspuussa aineisto pilkotaan pienempiin osajoukkoihin, kunnes osajoukkoihin kuuluu ainoastaan saman luokan jäseniä, tässä tapauksessa ajoneuvoja. Päätöspuu koostuu kolmenlaisista komponenteista: solmuista, oksista ja lehdistä. Solmut ovat piirteitä jotka luokittelevat ajoneuvot. Aineiston osajoukot ovat oksissa ja lehdet on luokkien mukaan nimetty. Juurisolmu on päätöspuun aloituspiste ja siihen valitaan piirre, joka mahdollisimman hyvin jakaa ajoneuvojen piirteet omiin osajoukkoihin [7, s.29–30; 8, s.18.]

Päätöspuuta voisi hyvin käyttää tässä työssä, etenkin jos ainoat huomionarvoiset muuttujat ovat kiihdytys- ja jarrutustapahtumat ajotuntia kohti. Menetelmän hyöty on siinä että luokat voidaan itse määritellä laadullisten kriteerien mukaan [8, s.18]. Vaihtoehtoisesti arvot, joiden mukaan luokitellaan, voidaan myös laskea ennen päätöspuun luontia. Päätöspuun heikkous on siinä, että jos muuttujia on paljon ja oksia on enemmän kuin kaksi, muuttuu puu nopeasti sekavaksi sekä valtavan kokoiseksi. Tämmöisessä

puussa voi myös olla vaikeaa asettaa juurisolmuksi tärkein ajoneuvoja erottava piirre. Päätöspuun arvoja ei voi vaihtaa kesken suorituksen tulosten perusteella, mikä voi olla ongelmallista.



Kuva 1. Esimerkki päätöspuu

Probabilistisiin lajittelumenetelmiin kuuluu esimerkiksi Bayes-verkkoluokittelija, joista naiivi Bayes-luokittelija on suosituin. Kyseessä on yksinkertainen kaksitasoinen puurakenne, jonka juuressa on luokkamuuttuja ja muut muuttujat sen alla. Tämä tarkoittaa sitä, että naiivi Bayes-luokittelijassa oletetaan, että muuttujat eivät ole toisistaan riippuvaisia. Bayes-verkkoluokittelijassa taas oletetaan että muuttujat ovat toisistaan riippuvaisia. Luokkamuuttuja voisi esimerkiksi olla ajoneuvo jonka attribuutit olisivat: vähän kiihdytyksiä, vähän jarrutuksia ja korkea keskinopeus. Muuttujat olisivat kiihdytys, jarrutus ja keskinopeus. Laskemalla Bayesin säännöllä oksien todennäköisyydet muodostuisi luokat. Nämä menetelmät käyttävät siis todennäköisyyksiä laskentaan kun kaikkea tietoa ei ole saatavilla, ja emme voi olla varmoja mihin ryhmään ajoneuvo kuuluu. Menetelmä perustuu tunnettuihin havaintoihin, joita meillä ei ole, joten sitä ei tulla työssä käyttämään. Toisin sanottuna ongelma on, että emme työssä pyri löytämään mihin olemassa olevaan ryhmään ajoneuvo kuuluu ominaisuuksiensa puolesta. Työssä pyritään selvittämään, minkälaisia ryhmiä kannattaa luoda. Tämän jälkeen vasta selvitetään mihin ryhmiin ajoneuvot kuuluvat [7; 8, s.8–12.]

Klusteroinnissa ei tiedetä etukäteen ryhmien luokituksia tai minkälaisia ryhmiä muodostuu. Klusterointia käytetään, kun halutaan ryhmittää ajoneuvoja, jotka näennäisesti ovat lähellä toisiaan arvojen perusteella [9, s.1–4]. Ryhmityksen jälkeen ryhmille päätetään nimet, jotka kuvastavat ryhmien ajoneuvojen arvoja. Etenkin jos muuttujia on paljon, on klusterointi huomattavasti helpompi menetelmä kuin luokittelu. Luokittelu on erittäin hankalaa, jos muuttujia on paljon, koska arvoja ei saa helposti samaan kuvaajaan, jonka perusteella voisi päätellä ryhmien rajat. Toinen ongelma on, että moniulotteisesta kuvaajasta olisi vaikea löytää muuttujien välinen korrelaatio luokittelua käyttäen.

Suosituin klusterointialgoritmi on k means. K means -klusterointialgoritmi perustuu klustereideen luomiseen niin, että yhden klusterin jäsenet muistuttavat enemmän oman klusterinsa jäseniä, kuin toisen klusterin jäseniä. Miten samanlainen, tässä tapauksessa ajoneuvo on toista ajoneuvoa, perustuu miten lähellä ne ovat toisiaan eri muuttujissa tietojoukossa. Menetelmää käytetään kun luokille ei ole vielä nimiä ja olemme ainoastaan päättäneet kuinka moneen ryhmään haluamme luokitella. Tämän jälkeen algoritmi pyrkii löytämään ryhmät tietojoukosta. Jokaisella klusterilla on keskipiste joka vastaa klusterin sisältämien ajoneuvojen keskiarvoa. Algoritmi toimii useammalla muuttujalla kuin kaksi, mutta sen tulosten visualisoiminen on siinä tapauksessa hankalaa. Työssä keskinopeus muuttujalla on reilusti korkeammat arvot kuin ajotapahtumilla, joten sen arvo pitää normalisoida. Tässä tapauksessa olisi järkevintä normalisoida kaikki arvot asteikolle 0-1. Näin ollen ne olisivat kaikki helposti toisiinsa verrattavissa, ilman että yhdellä muuttujalla on toista suurempi painoarvo. Arvon normalisointi tehdään kaavan 1 mukaan [7, s.38; 9.]

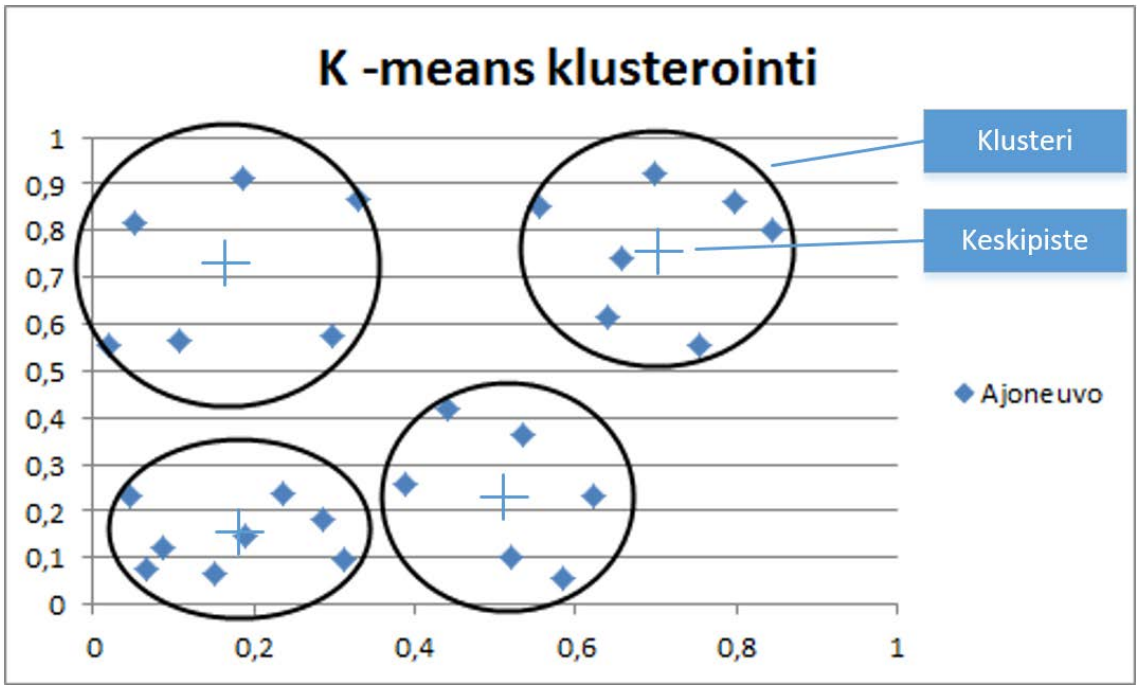
$$nArvo = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1).$$

$nArvo$ on normalisoitu arvo
x on normalisoitava arvo
x_{min} on pienin mahdollinen normalisoitava arvo
x_{max} on suurin mahdollinen normalisoitava arvo.

K means -klusterointialgoritmi toimii niin, että ensiksi valitaan kuinka monta ryhmää halutaan. Tämän jälkeen klusterit saavat satunnaisen keskipisteen. Nyt ajoneuvot lisätään kuvaajaan niin, että ajoneuvo kuuluu siihen klusteriin, jonka keskipiste on ajoneuvon arvoja lähimpänä. Klustereiden keskipisteet lasketaan uudestaan sen perusteella mitkä ajoneuvot siihen kuuluvat. Ajoneuvot määritetään taas uudestaan siihen

klusteriin, jonka keskipiste on lähimpänä ajoneuvoa. Näitä askeleita toistetaan kunnes kaikki ajoneuvot kuuluvat johonkin klusteriin [7, s.38.]

Seuraavana on esimerkki (kuva 2), miltä lopputulos kahdella normalisoidulla arvolla ja neljällä klusterilla voisi näyttää.



Kuva 2. K -means klusterialogritmin kuvaaja.

3.3 Olemassa olevat ratkaisut

Ajotavan analysoinnista on tehty paljon tutkimuksia eri puolilla maailmaa. Ne voi rajata sen perusteella, mitä tietoa on ollut ajoneuvosta saatavilla. Esimerkit vaihtelevat siitä, että ajoneuvoista on ollut vain GPS-tietoa saatavilla [2, s.2], siihen, että ajoneuvoon on asennettu tilapäinen mittalaite [10, s.51], tai jopa siihen, että otetaan talteen ajoneuvon tietokoneen tuottama tieto [11, s.26]. Tieto voidaan myös ottaa talteen joko kaapelia pitkin niin, että paikanninlaite tai mittalaite yhdistetään tietokoneeseen [11, s.24].

Ajotavan seurantajärjestelmiä on jo paljon käytössä ympäri maailmaa. Erityisesti raskaan kaluston seurantajärjestelmät ovat suosiossa niiden mahdollistaessa mittavat säästöt toimialalla, jossa ajotapa suoraan heijastuu kuluihin. Yksinkertaisimmat ja mahdollisesti tehokkaimmat ratkaisut ovat ne, joissa laitteet keräävät suoraan ajoneu-

vosta tiedot. Tämä tarkoittaa, että laitteella on ajoneuvojen ajotietokoneen tuottama tieto käsiteltävissä. Laite voi sitten lähettää tiedot keskustietokoneelle, joka käsittelee ajoneuvojen tietoja. Tämän menetelmän tuottama tieto on täysin ylivoimaisen yksityiskohtaista muihin järjestelmiin verrattuna. Tällä menetelmällä saa ajoneuvoista esimerkiksi seuraavat tiedot [11, s.26]:

- polttoaineenkulutus, sekä hetkellinen, keski- ja kokonaiskulutus.
- nopeus (hetkellinen ja keskinopeus)
- kierrosluku, sekä miten kauan ajoneuvo on ollut eri kierroslukualueilla
- ajomatka ja -aika
- rullaus- ja moottorijarrutusaika.
- tyhjäkäyntiaika ja -kulutus
- jarrujen käytöstä
- kiihtyvyydet (kiihdytys, jarrutus ja sivuttaiskiihtyvyys).

3.4 Menetelmä ohjelmani rakentamiseen

Aloitin perehtymällä kirjallisuuteen, joka liittyi edes jollain tavalla ajotapa-analysointiin, mielellään vielä niin, että paikannustietoa käytettiin analyysin tekoon. Kirjallisuutta oli rajoitetusti tarjolla silloin, kun tietolähteenä on ainoastaan paikannustieto. Löytämäni perusteella pystyin päättelemään, että on poikkeuksellista tehdä analyysiä viiden sekunnin päivitystiheydellä ja yleisempi tapa onkin tehdä se sekunnin välein. [2, s.2; 10.]

Käytännössä kaikki lähteeni viittaavat samaan asiaan: toistuvasti kiihdyttämällä ja jarruttamalla kuluttaa sekä enemmän polttoainetta, aiheuttaa enemmän päästöjä sekä kuluttaa enemmän ajoneuvoa [2; 5; 10; 11, s.26]. Tämän takia saatavalla olevan tiedon perusteella lähtökohtani työhön olikin näiden tapahtumien mittaaminen. Keskittymällä tähän eroaa työni olemassa olevista ja tarjoaakin uuden näkökulman, miten ajotapaa voi analysoida rajoitetummalla tietomäärällä.

Ohjelmani tulee pystyä laskemaan ja luokittelemaan sekä autoista että linja-autoista koostuvat ajoneuvokannat, mutta muutamien keskinäisten erojen takia hieman eri tavoin. Ajoneuvokannat ovat kokonaan erillään toisistaan, kun kerran kyseessä on kaksi

eri asiakkuutta. Lähtökohtaisesti käytän raja-arvona autokannassa, että jos sekunnissa nopeus muuttuu 8 km/h tai yli tallennan siitä tiedon järjestelmään tapahtumana. Vuonna 2010 Yhdysvalloissa myydyillä autoilla kiihdytys nollasta sataan kesti keskimäärin noin yhdeksän sekuntia ja keskiarvoa hitaammilla se kesti 11 sekuntia [12]. Euroopassa ajoneuvot ovat keskimäärin pienempiä, hitaampia ja vähemmän polttoainetta kuluttavia [13], joten päättelin, että Suomen olosuhteissa luku voisi olla noin 12 sekuntia. Kun 100 km/h jakaa tuolla 12 sekunnilla, tulee tulokseksi 8,3 km/h sekuntia kohti, jonka sitten pyöristin alaspäin 8 km/s sekuntia kohti. Tarkoitus on, että raja-arvo on matala, jotta nopeuden muutokset aiheuttavat herkästi tapahtuman, jonka perusteella analysoida. Näitä tapahtumia analysoimalla voin sitten luokitella ajoneuvot eri ryhmiin ja laskea niille ajotapa-arvot. Ajotapa-arvo tulee muodostumaan tapahtumien määrästä ajettuja tunteja kohti erilaiset muuttujat huomioon ottaen. Ajoneuvon keskivauhti tulee mahdollisesti paljastamaan, minkälaisessa ympäristössä ajoneuvo ajaa. Linja-autokannassa tulen käyttämään arvoa 5 km/h autojen vertausluvun perusteella.

Autokannassa viiden sekunnin päivitystiheys pitää ottaa huomioon siinä, että yksittäinen epätarkka arvo voi aiheuttaa kaksi perättäistä tapahtumaa ja mahdolliset virhearvot pitää sivuttaa. Tämä havaitaan, jos esimerkiksi nopeus on noussut 20 km/h:sta 200 km/h:een viidessä sekunnissa. Seuraavan viestin ollessa vaikkapa 40 km/h aiheuttaisi ohjelma kaksi tapahtumaa selkeässä virhetilanteessa. Tämän takia analyysivaiheessa suodatetaan pois sekä selkeästi virheelliset kiihdytykset että jarrutukset. Viestejä ei aina ole luotu viiden sekunnin välein, vaikka ne lähetetään viiden sekunnin välein, joten tämäkin pitää ottaa huomioon ohjelmaa tehdessä. Ohjelma ei voi olettaa, että viestit ovat aina syntyneet viiden sekunnin välein. Autokannassa tiedon määrä tulee mahdollistamaan sen, että ajoneuvoja voi verrata keskenään ajoneuvojen säännöllisesti kerätessä joko runsaasti, kohtalaisesti tai ei ollenkaan tapahtumia. Yksittäiset tapahtumat eivät tule olemaan ohjelmassani ratkaisevia autokannassa, sillä yllättäviä tilanteita ja virheviestejä voi tapahtua. Ison kuvan saaminen ja luokittelu suuren tietomäärän perusteella ovatkin työni olennaisia osia autokannan osalta.

Linja-autoista tulee paikannustietoa kerran sekunnissa, joten sen perusteella voimme tehdä paljon enemmän johtopäätöksiä ajotavasta. Täällä tiedon määrä tulee olemaan reilusti suurempi, koska paikantimet lähettävät tietoa sekunnin välein riippumatta siitä, onko linja-auto ajossa tai ei. Ohjelman tulee pystyä huomaamaan, milloin linja-auto ei ole lainkaan ajossa ja sivuttaa sen silloin lähettämä paikannustieto. Linja-autojen tiiviimmän paikannustiedon takia saamme rutkasti enemmän tapahtumia, joihin voimme

analyysin suorittaa. Suuremman tietomäärän takia virheviestin merkitys on pienempi, sillä voimme olettaa linja-autojen ajaessa samalla seudulla, että todennäköisyys virheviestiin on kaikilla suurin piirtein sama. Tarkoitus onkin, että ajoneuvot voidaan luokitella tapahtumien mukaan, ja kun niitä kertyy toistuvasti joillakin ajoneuvoilla enemmän kuin toisilla, kertoo se kuljettajien välisistä ajotyylien eroavaisuuksista.

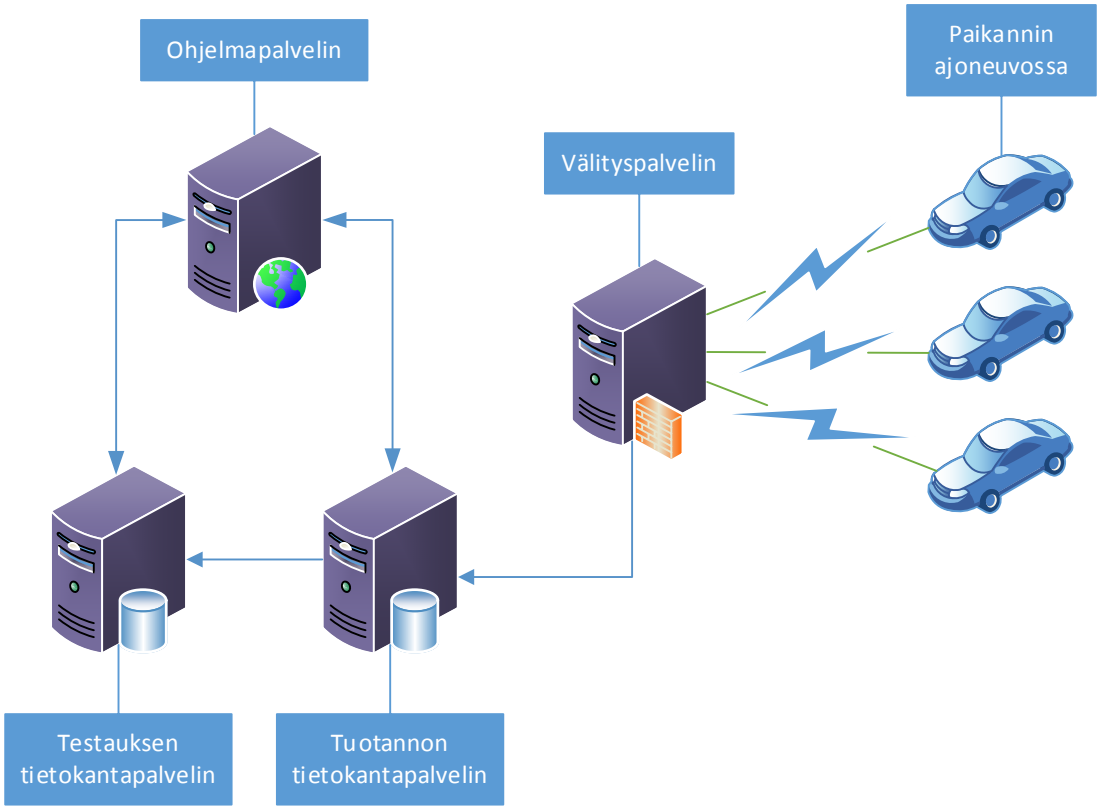
4 Järjestelmäkuvaus

4.1 Yleiskuvaus

Molemmat työssä käsiteltävät järjestelmät ovat ajoneuvonseurantajärjestelmiä, joilla ei ole mitään tekemistä toistensa kanssa. Yleinen toiminta on päällisin puolin samanlainen, mutta eroja on myös. Järjestelmien suurimmat erot ovat paikantimissa ja tiedon päivitystiheydessä. Tietokannat ovat rakenteiltaan samanlaisia mutta asiakkailla on erilaiset tarpeet ajoneuvoseurannassa. Tämän myötä niiden käyttämät taulut, ja miten niitä käytetään, eroavat toisistaan. Koko luku 4 on kirjoitettu käyttäen lähteenä sisäistä verkkodokumenttia [14].

Varsinaisesta järjestelmästä on erillään kehitysympäristö. Kehitysympäristöön viedään tarvittaessa tietokantoja tuotanto- tai testiympäristöistä. Kehitysympäristössä tehdään nimen mukaisesti kehitystyötä, ja uusia ominaisuuksia voi kokeilla ilman pelkoa, että muuttaa tai poistaa tietoja tuotantoympäristöstä. En tule käymään kehitysympäristöä sen tarkemmin läpi, koska se ei kuulu varsinaiseen ajoneuvoseurantajärjestelmään.

Arkkitehtuurikuvassa (kuva 3) on mallinnettu, miten ajoneuvoseurantajärjestelmät ovat rakentuneet molemmissa järjestelmissä. Linja-autojenseurantajärjestelmässä tosin seurattavat ajoneuvot ovat tietenkin linja-autoja, autojen sijaan. Nuolista näkee, mihin suuntaan tieto liikkuu järjestelmän eri osien välillä. Tuotannon tietokantapalvelimen sisältö synkronoidaan testauksen tietokantapalvelimen kanssa säännöllisin väliajoin. Paikantimet lähettävät tietonsa välityspalvelimelle, joka lähettää tiedon tuotannon tietokantapalvelimelle. Ohjelmapalvelin voi kommunikoida molempien tietokantojen kanssa. Kehitysympäristö ei ole arkkitehtuurikuvassa (kuva 3), koska se on täysin irrallaan muusta järjestelmästä.



Kuva 3. Ajoneuvoseurantajärjestelmän arkkitehtuurikuva.

4.2 Paikantimet

Paikantimet on kiinteästi asennettu ajoneuvoihin, eikä kuski pääse niihin vaikuttamaan. Paikantimet on huolella valittu, jotta ne kestävät sen rasituksen, jonka alle ne ajoneuvoihin asennettuna altistuvat. Asennuksen hoitaa ammattilainen, koska paikantimien asennustyössä pitää olla tarkka ja sijaintia pitää myös miettiä, jotta minimoidaan häiriöt. Mahdollisia häiriöitä voi syntyä, jos paikannin on asennettu kuskin tielle ja kuski kolhii paikanninta, tai vaihtoehtoisesti johdot ovat jääneet näkyviin ja ne takertuvat johonkin.

Paikantimet toimivat, GPS-paikannustekniikalla, jonka perusteella se luo paikannustiedon. Paikannustieto lähetetään paikantimelle asetettujen asetusten mukaisesti välityspalvelimelle. Tiedonsiirtoon käytetään mobiiliverkkoteknologioita.

Paikantimien asetuksista voi määrittää, milloin paikantimen kuuluu lähettää viesti. Paikannin voi olla eri tiloissa riippuen sille määritetyistä ehdoista, jotka tekevät paikantimista hyvin mukautuvaisia. Näistä tiloista löytyy esimerkiksi seuraavat tilat: pysähtynyt

ja ajossa. Paikannin voidaan myös määrittää lähettämään viestejä kun tietty ehto täyttyy, kuten vaikkapa liiallinen nopeus tai kiihdytys on tapahtunut. Uusimmat paikantimet huomaavat kiihdytysanturin perusteella, että ajoneuvo on liikenteessä, jolloin paikannin kytkeytyy päälle lähetystilaan.

4.2.1 Autokannassa

Autokannassa paikantimena toimii periaatteessa yksinkertainen paikannuslaite, joka on varta vasten tähän tehtävään suunniteltu. Laitteeseen asetetaan SIM-kortti, jonka avulla paikannin yksilöidään, kun se lähettää paikannustietoa mobiiliverkkoa pitkin GPRS-yhteyden ylitse. Jokaista paikanninta varten avataan erikseen liittymä operaattorille.

Autokannan paikantimien asetukset voi ohjelmoida uudestaan ja muuttaa omien tarpeiden mukaisesti. Säätömahdollisuuksia ovat esimerkiksi minimiarvo sille, kuinka moneen satelliittiin pitää paikantimen olla yhteydessä lähettääkseen viestejä, minkä avulla voimme varmistua että epätarkimmat paikannustiedot jäävät lähettämättä. Paikantimia voi siis räätälöidä tarpeiden mukaan, ja ne voisivatkin olla hyvin erilaisia ajoneuvosta riippuen.

Paikantimet voidaan päivittää etäyhteyden avulla lähettämällä päivitys mobiiliverkon yli paikanninlaitteelle. Päivityksen saatuaan, paikanninlaite osaa päivittää itsensä asentamalla muutokset asetuksiin. Tämä helpottaa muutosten tekemistä paikanninlaitteisiin, kun niihin ei tarvitse päästä fyysisesti käsiksi, mutta jos laite ei ole päällä, se ei saa myöskään päivitystä. Sesonkiajoneuvot saattavat päivittyä vasta reilusti myöhemmin kuin muut ajoneuvot, kun kerran paikanninlaite ei ole päällä ajoneuvon ollessa poissa käytöstä.

Autokannassa paikantimet lähettävät paikannustietoa ainoastaan kerran viidessä sekunnissa. Paikantimet on myös asetettu, olemaan lähettämättä paikannustietoa, jos ne ovat pitkään liikkumatta. Näiden kahden asetuksen vaikutuksesta autokanta ei tuota niin paljon tietoliikennettä, eikä tiedon varastoiminen vie niin paljon tilaa kuin linja-autokannassa. Asetuksiin on päädytty tehokkuussyistä: tiheämpi päivitystahti ei ole tarpeellinen useimmissa seurantatapauksissa ja paikallaan olevan viestejä on turha vastaanottaa tässä tapauksessa.

4.2.2 Linja-autokannassa

Linja-autokannassa paikantimena toimii ajoneuvo-PC, joka suorittaa myös muita tähän työhön liittymättömiä toimintoja. Ajoneuvo-PC:hen on kytketty GPS-vastaanotin, jonka avulla se luo paikkatiedon. Ajoneuvolaitteeseen on myös kytketty 3G-modeemi, johon on asetettu SIM-kortti, jonka avulla se toimittaa välityspalvelimelle mobiiliverkon yli paikannustietoa.

Linja-autokannassa paikantimet lähettävät viestejä kerran sekunnissa. Linja-autojen paikantimet lähettävät myös viestejä riippumatta siitä, onko ajoneuvo ajossa vai ei, joka tarkoittaa, että viestejä tulee aina yhtä paljon riippumatta siitä, onko ajoneuvo käytössä vai ei. Näiden asetusten takia tietoliikennemäärä on moninkertainen autokannan vastaavaan ajoneuvoa kohti, ja tietokantaan välityspalvelimelle tulee huomattavasti enemmän käsiteltävää tietoa.

4.3 Välityspalvelin

Välityspalvelin kuuntelee sille tulevia viestejä ja tallentaa viestiliikenteen raportointia varten. Välityspalvelin tekee sille tulevista viesteistä paikallisen tallennuksen tietokantaan, ja toimittaa sitten tekemänsä paikallisen tietokannan eteenpäin tietokantapalvelimelle säilöttäväksi. Välityspalvelimelle on asetettu asetuksia, joiden perusteella se suodattaa turhia viestejä pois. Näin saadaan vähän hillittyä, kuinka paljon tietoliikennettä syntyy välityspalvelimen ja tietokantapalvelimen välillä, mutta ennen kaikkea se vähentää tarvetta tallentaa tietoa raportointia varten.

Välityspalvelimen tärkein työ on olla jatkuvasti valmis vastaanottamaan tietoa paikantimista, ja tallentaa se, ettei mikään tärkeä paikannustieto huku tai jää säilömättä. Välityspalvelin osaa yksilöidä paikantimet, joten se tietää, mistä paikantimesta tieto on tullut ja se tallentaakin paikalliseen tietokantaansa tiedon siitä, mistä paikantimesta paikannustieto on tullut.

Käyttämällä välityspalvelinta järjestelmä ei kuormita samaa palvelinta tiedonhakuun. Järjestelmä haluaa päästä tietoon käsiksi esitellessään sen asiakkaalle graafisessa käyttöliittymässä tai luodessaan raportin paikannustiedosta. Välityspalvelin saakin keskittyä rauhassa ainoastaan paikantimista tulevaan tietoliikenteeseen, eikä siihen kohdistu tietokantakyselyitä.

4.4 Tietokantapalvelin

Tietokantapalvelin tallentaa välityspalvelimen lähettämän tiedon MySQL-tietokantaansa. Tallennettu data varmuuskopioidaan myös muille palvelimille, mutta se ei ole työni kannalta olennaista. Tietokantapalvelimeen kohdistuvat kaikki ne tietokantakyselyt jotka graafiset käyttöliittymät eli tässä tapauksessa selaimessa toimivat ohjelmat suorittavat. Käyttäjän käyttäessä ohjelmaa se tekee taustalla tietokantakyselyt esittääkseen tietokannasta löytyvän tiedon. Tietokantapalvelimia on kaksi, yksi testaukseen ja toinen tuotantokäyttöön.

Kyselyitä tietokantaan tulee myös erilaisista selaimen ulkopuolelta toimivista itsenäisistä ohjelmista, jotka yleensä joko suorittavat raskasta laskentaa tai luovat kuukausittaisia raportteja. Suurin osa itsenäisistä ohjelmista ajetaan ajastetusti säännöllisin väliajoin. Ohjelmani tulee kuulumaan tähän osaan järjestelmää, eli itsenäisiin ohjelmiin jotka käyttöliittymän ulkopuolella suorittavat raskasta laskentaa ilman vasteaikavaatimusta.

Tietokantapalvelimen tietokanta koostuu vajaasta sadasta taulusta, jotka relaatiotietokantojen suunnitteluperiaatteiden mukaisesti liittyvät toisiinsa. Tauluja on näinkin monta, koska ajoneuvoseurantajärjestelmät ovat ominaisuuksiltaan kattavia. Ajoneuvoille on oma taulu, kuten myös paikantimille, sekä taulu, joka yhdistää paikantimen ajoneuvoon. Paikannustietotaulu on työni kannalta mielenkiintoisin, mutta tarvitsen tietysti myös tietoa paikantimista, ja siitä, mihin ajoneuvoon ne kulloinkin ovat olleet liitettyinä. Taulut on indeksoitu tunnusnumerojen mukaan, ja esimerkiksi paikannustietoa sisältävä taulu on myös ositettu suorituskyvyn parantamiseksi.

4.5 Ohjelmapalvelin

Ohjelmapalvelin ylläpitää järjestelmän graafista käyttöliittymää, eli se sisältää ohjelman lähdekoodit, jotka on tehty PHP-, CSS- ja JavaScript-teknologioilla. Tähän palvelimeen kohdistuvat siis kyselyt, kun asiakas käyttää ohjelmaa selaimen kautta. Muihin palvelimiin tai niissä ajettaviin palveluihin ei asiakkaalla ole minkäänlaista pääsyä. Ohjelmapalvelimella on toteutettu sekä testaus- että tuotantokäytössä olevat graafiset käyttöliittymät.

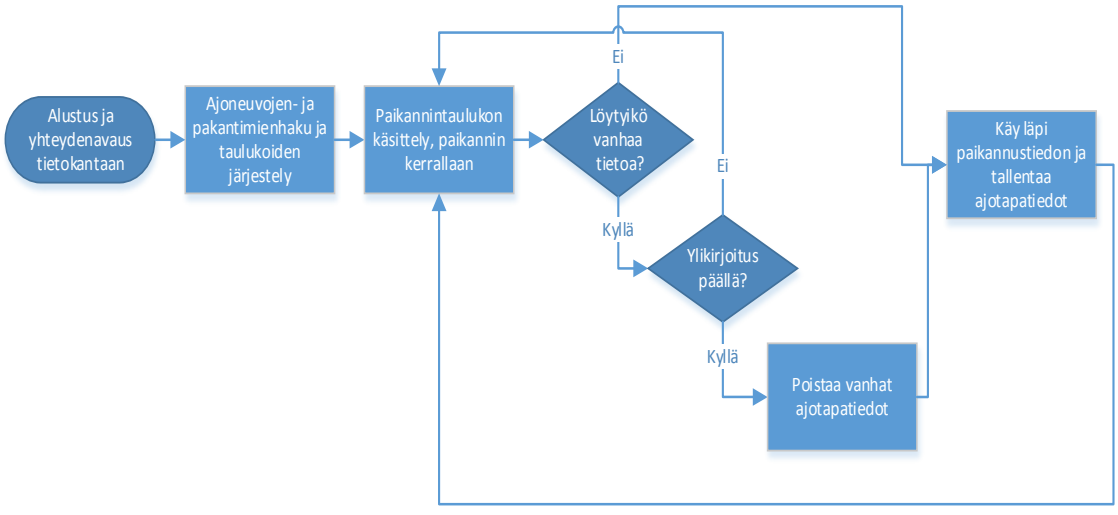
Ohjelmapalvelimella sijaitsee myös aiemmin mainitut itsenäiset ohjelmat, jotka eivät ole ohjelman käyttöliittymästä käytettävissä. Ainoastaan tieto, jonka itsenäiset ohjelmat ovat luoneet tietokantaan, on käyttöliittymän kautta nähtävissä. Ohjelmani itsenäiset ohjelmat tulevat sijaitsemaan ohjelmapalvelimella valmistuessaan.

5 Laskentaohjelman kuvaus

5.1 Tiedonhaku ja alustus

Molempia tietokantoja varten tehdään samat yleiset tiedon alustukset sekä apuohjelmien lataaminen. Ohjelmilla on useita eroavaisuuksia keskenään tiedonhaussa ja suodatuksessa, joten niistä on kerrottu erikseen. Autokantaa varten tiedonhaku voidaan kohdistaa suuremmalle aikavälille, ja paikantimet ovat hieman eri tavalla liitettynä ajoneuvoihin tietokannassa kun linja-autokannassa mainitakseni muutaman eron. Tulen esittämään tämän luvun jokaisessa kappaleessa, minkälaista tietoa käsitellään.

Prosessikaaviosta (kuva 4) näkee, miten ohjelman toiminta etenee. Prosessikaavio on vähän yksinkertaistettu ymmärtävyyden parantamiseksi. Ohjelman suoritus lakkaa, kun ohjelma on käynyt läpi jokaisen läpikäytävän paikantimen.



Kuva 4. Prosessikaavio laskentaohjelmalle, joka esittää, miten ohjelma toimii, kunnes kaikki paikantimet on käyty läpi.

Ensi töikseen ohjelma tarkistaa, että vaaditut apuohjelmat löytyvät määritetyistä paikoista. Ilman apuohjelmia ei ohjelma toimi, joten tarkistus on tarpeellista suorittaa heti ohjelman käynnistyessä. Apuohjelmista mainittakoon niistä tärkein, tietokantayhteydestä huolehtiva DbHandler-ohjelma. Tätä käytetään yhteyden avaamiseen, hakujen sekä lisäyksen tekemiseen sekä mahdollisten tietokantayhteyteen liittyvien virhetilanteiden selvittämiseen. Toinen tärkeä apuohjelma on Logger, jonka avulla ohjelma kätevästi kirjaa ylös tärkeitä tietoja esim. muistin käytöstä lokiin. Ongelmatilanteissa lokista löytyy monesti nopeasti syy ongelmaan, ja se auttaa ohjelman parantamisessa sekä korjaamisessa. Lokiin merkitään myös käsiteltävä aikajakso ja käsiteltävän ajanjakson mukaan lokitiedostokin nimetään. Lokiin kirjataan aina käsiteltävän ajoneuvon tunnus, löydettyjen paikannustietorivien määrä, paikantimen tunnus sekä laskettujen tapahtumien määrä. Jos ohjelma kaatuu kesken suorituksen tai on epäily ohjelman toimivuudesta, voi lokista tarkistamalla nopeasti löytää tiedon siitä, minkä paikantimen kohdalla oli ongelma. Lokia tarkistamalla löytyy myös nopeasti tieto siitä, jos jokin paikannin ei ole luonut paikannustietoa lainkaan.

Ohjelmani laskuosa kaipaa käynnistyessään kolme parametria. Näistä kaksi ensimmäistä ovat käsiteltävä vuosi ja kuukausi kokonaislukumuodossa. Jos ohjelman käynnistää ilman näitä ensimmäisiä parametreja, olettaa ohjelma, että käynnistäjä haluaa sen käyvän läpi nykyisen kuukauden tiedot. Kolmas syötettävä parametri on merkkijono, joka määrää ylikirjoitetaanko vanha laskettu tieto vai ei. Mikäli ylikirjoitusta ei kytketä päälle, jättää ohjelma väliin käsiteltävän paikantimen, jos se löytää sille tapahtumia ajotapahtumataulusta haetulla aikavälillä. Käyttäjän syöttäessä vuoden ja kuukauden, muuttaa ohjelma nämä arvot unix-aikaleimamuotoon, joka siis kertoo, kuinka monta sekuntia on kulunut aloitushetkestä, joka oli tammikuun ensimmäinen päivä vuonna 1970 UTC-aikavyöhykkeessä. Aikaleimoja tehdään ensiksi kaksi, ensimmäinen on syötetyn vuoden ja kuukauden alkuhetki ja jälkimmäinen on kyseisen vuoden ja kuukauden viimeinen sekunti. Näitä aikaleimoja käytetään myöhemmin sekä suodattamiseen että tiedonhaussa. Esimerkki aikaleimasta: `1388527200`. Aikaleima on päivämäärälle 1.1.2014 kello 00:00:00 Suomen aikaa.

5.1.1 Autokantaa käsiteltäessä

Autokantaa käsittelevässä ohjelmassa tehdään kaksi aikaleimaa lisää, joiden avulla kuukausi pilkotaan kolmeen osaan. Ensimmäinen niistä on kuun kymmenentenä ja toinen kahdentenakymmenentenä päivänä. Näiden avulla haettava tietomäärä ei auto-

kannassa nouse liian suureksi haettaessa paikannustietoa noin kymmenen päivän lohkoina.

Autokantaa käsittelevän ohjelman ensimmäinen versio haki valmiita kirjastoja käyttäen kaikki järjestelmän paikantimet sekä autot tietokannasta. Kirjasto teki paikantimista ja autoista olioita, joten niiden käsittely on tästä syystä huomattavasti helpompaa, kuin jos käsiteltävänä olisi suoraan MySQL:n palauttamat taulukot. Autoilla sekä paikantimilla on myös aina alku- ja loppuaikaleima, joka kertoo siitä, milloin ajoneuvo on tullut käyttöön ja milloin se on otettu käytöstä. Mikäli loppuaikaleima on tyhjä, on se vielä käytössä.

Aikaisemmin ohjelma suodatti tässä vaiheessa haetut ajoneuvot sekä paikantimet. Tämä tarkoitti sitä, että ajoneuvojen ja paikantimien piti olla käytössä suodattimen määrittämän aikarajan sisällä. Mikäli ajoneuvo tai paikannin ei ollut käytössä määritetyssä aikajaksossa, poistettiin se tässä vaiheessa käsiteltävien ajoneuvojen tai paikantimien joukosta. Esimerkiksi jos auto oli ollut käytössä aikaisemmin mutta ehditty jo poistamaan käytöstä ennen haettua aikajaksoa, ei sen tietoja tarvitse tietenkään käsitellä, sillä paikannustietoa ei kyseisen auton osalta ole olemassa. Ajoneuvoja pyrittiin tässä vaiheessa suodattamaan pois eri menetelmin ajoneuvotaulukon vielä sisältäessä traktoreita ja vastaavia ajoneuvoja. Nämä menetelmät olivat epätehokkaita ja koodirivimäärä kasvoi huolestuttavan nopeasti ilman että toiminnallisuus oli niin hyvä ja tehokas kuin se voisi olla. Tiesin että ongelmaan oli parempi ratkaisu, joten tein tämän osan kokonaan uusiksi.

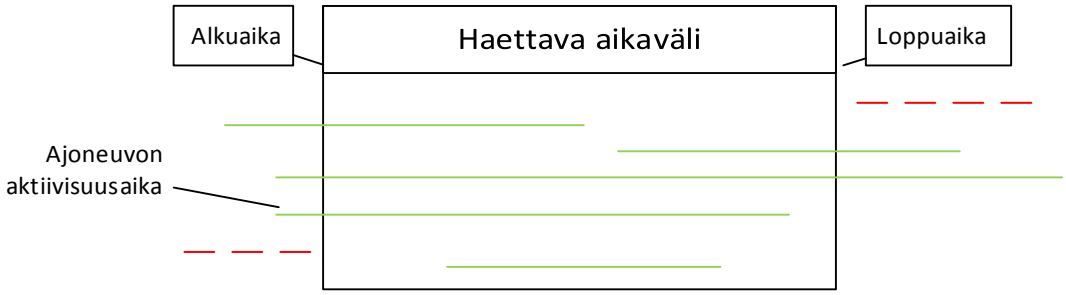
Uudessa versiossa käytetään edellisen sijaan MySQL-kantakyselyä, joka on varta vasten tätä ohjelmaa varten suunniteltu. Se hakee tietokannasta ainoastaan tuotantoajoneuvojen tiedot. Tuotantoajoneuvolla tarkoitetaan tässä yhteydessä tavallista työautoa, joka yleensä on henkilöauto tai pakettiauto. Suodattamalla jo tässä vaiheessa pois ajoneuvot, jotka ovat traktoreita, katulakaisuajoneuvoja ja vastaavia ajoneuvoja, käsiteltävien ajoneuvojen määrä laski noin 500 ajoneuvolla. Hakuun syötetään parametrit, jotka ovat aikaleimoja. Aikaleimoilla tarkistetaan, että ajoneuvo on ollut käytössä haetun aikavälin aikana, eikä se ole poistunut käytöstä. Ajoneuvoja ei nyt käsitellä olioina, koska ohjelma ei tarvitse koko olion sisältämää tietoa, ohjelmalle riittää tieto ajoneuvon tunnuksesta tässä vaiheessa. Seuraavassa on esimerkki, mitä tietoa autot-taulukko voisi sisältää, missä 123 on ajoneuvotunnus. Samaa tunnusnumeroa käytetään koko luvun aikana.

```
autot[123] = 123 (ajoneuvontunnusnumero)
```

Traktorit, katulakaisuajoneuvot sekä muut vastaavat ajoneuvot suodatetaan pois, koska ajoneuvojen nopeusvaihtelut ovat niin pienet, että ne eivät kerro meille lainkaan ajotavasta. Lisäksi ongelmana on viiden sekunnin päivitystiheys, joka on aivan liian harvoin, kun nopeudet pysyvät säännöllisesti erittäin matalina. Lisäksi käsittelemällä nämä ajoneuvot kasvaa ohjelman suoritusaika huomattavasti, ilman että ohjelman tuottama hyöty kasvaa. Palvelimia kuormitettaisiin turhaan tekemällä kyselyjä näiden ajoneuvojen paikannustiedosta.

Parametreista luoduista aikaleimoista tehdään suodatin. Suodatin toimii matemaattisen leikkausmenetelmän mukaisesti. Tämä tarkoittaa sitä, että esim. paikantimen alku- ja loppuaikaleiman käsittävä aika, on löydyttävä suodattimen määrittämän ajan sisältä. Loppuaikaleima voi myös olla määrittämätön, joka tarkoittaa, että se on yhä voimassa.

Aikasuodatus (kuva 5) näyttää miten leikkausmenetelmä toimii. Yhtenäisellä viivalla on merkitty ajoneuvot, jotka valitaan käsittelyyn mukaan, aktiivisuusajan ollessa haettavan aikavälin sisällä. Katkoviivalla on merkitty ajoneuvot, jotka eivät olleet aktiivisina haetulla aikavälillä, eikä niitä siten käsitellä.



Kuva 5. Aikasuodatus leikkausmenetelmällä.

Seuraavaksi ohjelma hakee suodattimen määrittelemän ajanjakson mukaan, kirjastoa käyttäen, autoja paikantimiin yhdistävät tiedot. Tämä tieto tulee oliomuotoisena käsiteltäväksi, ja voimme jo haussa suodattaa pois ne ajoneuvojen ja paikantimien väliset yhteydet, jotka eivät olleet voimassa määritetyssä aikajaksossa. Tämä ajoneuvojen-Paikantimet-niminen taulukko kertoo siis, mikä paikannin oli kiinni missäkin autossa haetussa aikajaksossa. Tässä vaiheessa ohjelma tallentaa lokiin tiedon muistinkäytös-

tä. Seuraavassa on esimerkki siitä, mitä tietoa ajoneuvojenPaikantimet-taulukko voisi sisältää. Sama paikantimen tunnusnumero esiintyy jatkossakin esimerkkinä.

```
ajoneuvojenPaikantimet[123] =
                                123 (ajoneuvontunnusnumero)
                                99999 (paikantimen tunnusnumero)
                                1383527200 (alkuaikaleima)
                                null (loppuaikaleima)
```

Nyt ohjelma tietää, mitkä ajoneuvot olivat käytössä haetulla aikavälillä sekä mikä paikannin oli kiinni missäkin ajoneuvossa kyseisellä aikavälillä. Ohjelma käy silmukassa läpi kaikki ajoneuvot haetusta ”autot”-taulukosta. Ennen silmukkaa ohjelma alustaa uuden taulukon. Tähän taulukkoon tallennetaan tieto suodatetuista ajoneuvoista paikantimineen. Mikäli ajoneuvo löytyy ajoneuvojenPaikantimet-taulukosta, luodaan uusi tieto autotPaikantimet-taulukkoon, johon tallennetaan sekä ajoneuvon että paikantimen tunnusnumero. Tämä taulukon läpikäynti tehdään, koska ajoneuvojenPaikantimet sisältävät kaikki ajoneuvot haetulta aikaväliltä, kuin autotaulukko sisältää ainoastaan aiemmin määritetyn kaltaiset autot. Näin ollen ohjelmalla on taulukko, johon on tallennettu yksitellen jokaiseen alkioon tietopari, joka sisältää kaiken tietokäsittelyä varten olennaisen tiedon. Seuraava on esimerkki siitä, mitä tämä taulukko voisi sisältää.

```
autotPaikantimet[0] =
                        123 (ajoneuvontunnusnumero)
                        99999 (paikantimen tunnusnumero)
```

Tämän jälkeen ohjelma käy läpi yksitellen alkiot autotPaikantimet-taulukosta. Paikannustiedot on sidottu paikantimeen paikantimen tunnusnumeron perusteella. Ohjelma pitää muistissa, kuinka monetta kertaa se käsittelee samaa alkiota. Ensimmäisellä kierroksella haetaan paikannustietoa paikantimen tunnusnumeron perusteella, kuun kymmenestä ensimmäisestä päivästä. Toisella kierroksella haetaan seuraavasta kymmenestä päivästä, ja viimeisellä kierroksella sitten kuun ylijäävistä päivistä.

5.1.2 Linja-autokantaa käsiteltäessä

Linja-autokantaa käsittelevässä ohjelmassa tiedonhaku suoritetaan heti alustuksen ja aikaleimojen luomisen jälkeen. Haku suoritetaan MySQL-kantaan kyselyllä, joka on

varta vasten ohjelmaa varten luotu. Haussa suodatetaan pois linja-autot, jotka eivät ole olleet käytössä haetulla aikavälillä sekä testikäytössä olleet linja-autot. Testikäytössä olleilla linja-autoilla on omat tunnukset, joten suodatus hakutilanteessa on helppoa. Haun tuloksena on taulukko linja-autojen tunnuksia sekä paikantimien tunnuksia. Molemmat löytyvät samasta taulukosta, ja koska ajoneuvotyyppi ei linja-auto tietokannassa vaihtele, haku pystyttiin kohdistamaan suoraan tähän ajoneuvoja paikantimiin yhdistävään tauluun. Seuraavana on esimerkki siitä, mitä tämä taulukko voisi sisältää.

```
bussitPaikantimet[0] =
                        123 (linja-autontunnusnumero)
                        99999 (paikantimentunnusnumero)
```

Linja-autokannassa tietomäärä ajoneuvoa kohti on teoreettisesti vähintään viisinkertainen autokantaan verrattuna. Teoreettinen alaraja johtuu päivitystiheyden erosta. Linja-autot päivittivät paikannustietoaan sekunnin välein kun taas autot autokannassa viiden sekunnin välein. Todellisuudessa ero on paljon suurempi kuin viisinkertainen. Linja-autot lähettävät paikannustietoa ympäri vuorokauden riippumatta siitä, ovatko ne ajossa vai ei, eikä niitä kytketä pois päältä muutoin kuin huollon yhteydessä. Autokannan ajoneuvot lähettävät paikannustietoa vain, kun ajoneuvot ovat käytössä, ja ne suljetaan yleensä työpäivän päätteeksi muutenkin. Tämän takia paikannustietoa ei voi hakea kymmenen päivän erissä, niin kuin autokantaa käsittelevässä ohjelmassa.

Paikannustietoa joudutaan hakemaan yhden vuorokauden erissä. Ongelmana on, että mitä useampi kysely, sitä hitaampi ohjelma on, koska tietokantakyselyt vievät käytännössä kaiken ohjelman suoritusajasta jo autokantapuolella. Linja-autopuolella ongelma korostuu suuremman tietomäärän takia. Useamman kuin yhden vuorokauden haku taas veisi niin paljon muistia, että kehitysympäristössä muisti loppuu kesken ja ohjelma kaatuu. Tämän takia päädyin siihen, että yhden vuorokauden aikaväli olisi hyvä kompromissi ohjelman kannalta. Yhden vuorokauden hauissa muisti loppui silti kesken tiettyjen ajoneuvojen kohdalla testiympäristössä. Tämän takia kyselyyn lisättiin vaatimus, että ainoastaan paikannustietorivit, joiden nopeuskentän arvo on yli 1, haetaan tietokannasta.

Ohjelman täytyy tietää, kuinka monta päivää on kuukaudessa, joten se tarkistaa päivien lukumäärän ”Date”-funktiota käyttäen. Tämän jälkeen silmukassa haetaan paikannustieto puolelta vuorokaudelta kerralla kuun alusta alkaen. Silmukan lähestyessä

kuun loppua on hyödyllistä tietää kuukauden päivien määrä, jotta silmukka ei hae seuraavan kuukauden tietoa.

5.2 Laskenta

Ennen kuin ohjelma hakee paikannustietoa, tarkistaa ohjelma, löytyykö tietokannan ajotapahtumataulusta paikantimelle jo tietoa sen tunnusnumeron perusteella. Haku tehdään kantakyselyllä, ja se tarkistaa ainoastaan, löytyykö ajotapahtumia käsiteltävällä aikavälillä. Ohjelman käynnistyessä on parametrilla määritetty, miten ohjelman tulee edetä, jos tapahtumia löytyy. Vaihtoehtoja on kaksi, ensimmäinen vaihtoehto on, että vanhat tiedot poistetaan sekä ajotapahtumataulusta että ajon keskinopeustaulusta haetulta aikaväliltä. Toinen vaihtoehto on, että tapahtumien löytyessä käsiteltävä paikannin jätetään väliin ja siirrytään seuraavaan. Jälkimmäinen vaihtoehto on etenkin hyödyllinen, jos ohjelman suoritus on katkennut edellisellä suorituskerralla. Näin ei ohjelman tarvitse käydä kaikkia paikantimia uudestaan läpi.

Paikannustietohaku suoritetaan MySQL-kyselyllä tietokantaan, johon on ohjelman alussa avattu yhteys. Haku vaatii parametreiksi paikantimen tunnusnumeron, jonka perusteella haku suoritetaan, sekä haettavan aikavälin alku- ja loppuaikaleiman. Haun tuloksena palautetaan taulukko paikannustietoa, joka on järjestelty paikannustiedon luomisajanhetken mukaan. Paikannustiedosta työn kannalta olennaista on ainoastaan nopeus sekä luomisajanhetki, joten haku kohdistuu ainoastaan näihin kenttiin, eli palautettava paikannustietotaulukko sisältää ainoastaan nämä tiedot. Mikäli ohjelma palauttaisi muittenkin kenttien tiedot, hidastuisi ohjelman suoritusaika. Lisätiedolla ei ole analyysi- tai laskentaohjelman kannalta hyötyä tässä tilanteessa. Alla on esimerkki siitä, mitä tämä taulukko voisi sisältää.

```
paikannusTiedot[0] =
                    56 (nopeus, km/h)
                    1388527200 (paikannustiedon luomisaikaleima)
```

Jokaisen kierroksen jälkeen tarkistetaan, löytyikö paikannustietoa lainkaan. Mikäli paikannustietoa ei löytynyt, siirrytään suoraan seuraavaan kierrokseen. Paikannustiedon löydyttyä sitä ruvetaan käsittelemään. Ohjelma käy läpi löydetyn paikannustiedon silmukassa kunnes jokainen paitsi viimeinen paikannustietorivi on käsitelty. Jokaista riviä

verrataan seuraavaan, joten viimeistä riviä ei käsitellä erikseen, koska sitä ei voi verrata seuraavaan.

Ohjelma tallentaa muistiin käsiteltävän paikannustietorivin nopeuden. Tätä tietoa käytetään kuukauden keskinopeuden laskemiseen. Ohjelma tarkistaa käsiteltävän paikannustietorivin nopeuden ja vertaa sitä taulukossa seuraavaksi olevan nopeuteen. Mikäli nopeusero on autokantaa käsiteltäessä yhtä suuri tai suurempi kuin 8 km/h jokaista sekuntia kohti, joka erottaa paikannustietojen luomisajankohtaa, syntyy siitä tapahtuma. Linja-autokannassa vastaava arvo on 5 km/h. Tapahtuma-taulukkoon talletetaan nopeusero sekä molempien keskenään verrattavien paikannustietorivien luomisajankohdat. Luotu tapahtuma talletetaan tapahtumat-taulukkoon myöhempää tietokantaan tallennusta varten. Alla on esimerkki siitä, mitä tapahtumat-taulukko voisi sisältää.

```
tapahtumat[0] =
          tapahtuma['nopeus']    = 45 (aikaleimojen nopeusero)
          tapahtuma['alkuaika']  = 1388527200 (alkuaikaleima)
          tapahtuma['loppuaika'] = 1388527205 (loppuaikaleima)
```

Selvästi virheellisiä tapahtumia ei tässä vaiheessa vielä poisteta. Ohjelman tarkoitus raja-arvoineen on saada mahdollisimman laajasti tietoa käsiteltäväksi. Raja-arvo toimii minimiarvona, eli kaikki nopeudet, jotka ylittävät tai täsmäävät, tallennetaan, mutta niihin ei oteta sen enempää kantaa tässä vaiheessa. Raja-arvo on tarkkaan harkittu, jotta tietokantaan tulisi mahdollisimman vähän hyödytöntä tietoa. Tavallisista ajotapahtumista ei kuulu syntyä tapahtumaa järjestelmään. Analyysivaiheessa voi muuttaa raja-arvoja tai luokitella eri raja-arvojen mukaisesti ajoneuvoja, mikäli siihen on tarvetta. Järjestelmän kannalta tärkeintä on olla suodattamatta liikaa tietoa laskentavaiheessa. Myöhemmässä vaiheessa on helpompaa suodattaa pois tietoa, kun suorittaa uudestaan koko raskas laskenta, jos raja-arvo onkin ollut liian matala, eli luonut liian herkästi tapahtumia.

Linja-autoja käsittelevässä ohjelmassa tallennetaan myös aktiivisten paikannustietorivien määrä. Aktiivinen paikannustietorivi tarkoittaa riviä, jossa nopeus on yli yksi. Raja ei ole nolla, koska paikallaankin ollessaan paikannin voi lähettää nopeuksia, jotka ovat nolla ja muutamia kymmenyksiä. Aktiivisten paikannustietorivien määrä tallennetaan siksi, että linja-autot eivät tallenna tietokantaan tietoa niiden aktiivisuusajasta, eli milloin ne ovat olleet ajossa. Tämän johdosta luotuja kiihdytys/jarrutus- tapahtumia verrattaan

aktiivisten paikannustietorivien määrään metriikkaa laskettaessa. Linja-autojen aktiivisuusaikoja olisi kovin raskasta tämän ohjelman laskea, ilman että olisi takeita siitä, että se toisi analyysiin lisäarvoa, joten ohjelmaan tehtiin vaihtoehtoinen ratkaisu.

Ohjelman käytyä läpi tämän hetkisen kierroksen paikannustiedot siirtyy se tiedon tallennusvaiheeseen. Mikäli tapahtumia luotiin tapahtumat-taulukkoon, käydään tapahtumat-taulukko läpi silmukassa. Jokainen tapahtuma tapahtumat-taulukossa käsitellään ja syötetään parametrina yhdessä autotPaikantimet- tai bussitPaikantimet-taulukon kanssa tallenna ajotapahtumat -funktiolle. Taulukot sisältävät käsiteltävän ajoneuvon ja siihen liitetyn paikantimen tunnusnumerot. Tämän jälkeen tallennetaan myös tieto muistinkäytöstä lokiin.

Tallenna ajotapahtumat-funktio tallentaa MySQL-tietokantaan omaan ajotapahtumatauluun tarpeelliset tiedot tapahtumasta. Tapahtuman tiedoista talletetaan tietokannan tauluun tieto tapahtuman luomisajankohdista, eli alkuajankohta ja loppuajankohta sekä nopeusmuutos kyseisen tapahtuman aikana. Tauluun tallennetaan myös ajoneuvon tunnus, jonka avulla yhdistämme tapahtuneen nopeusmuutostapahtuman oikeaan ajoneuvoon. Tietokantaan tallennetaan myös tieto, mistä paikantimesta tieto on peräisin, jotta mahdollinen selvitys olisi mahdollista. Tämän jälkeen ajotapahtuma ei liity enää paikantimeen vaan ajoneuvoon, joten analyysiohjelman ei tarvitse tietää paikannustiedosta tai paikantimista mitään, kun laskenta on jo suoritettu.

Jokaisella kierroksella tallennetaan myös tieto keskinopeudesta tietokantaan. Nopeus on laskettu jokaisen paikannustietorivin kohdalla ja ohjelmalla on myös tieto paikannustietorivien määrästä. Tämän avulla lasketaan keskinopeus jakamalla nopeus paikannustietorivien lukumäärän kanssa. Keskinopeus tallennetaan tietokantaan analyysiä varten keskinopeustauluun. Keskinopeutta voi ohjelma käyttää luokitteluun, mikäli huomaamme, että keskinopeuden mukaan voimme rajata ajoneuvoja eri luokkiin ja että se on merkittävä tarkkailtava muuttuja.

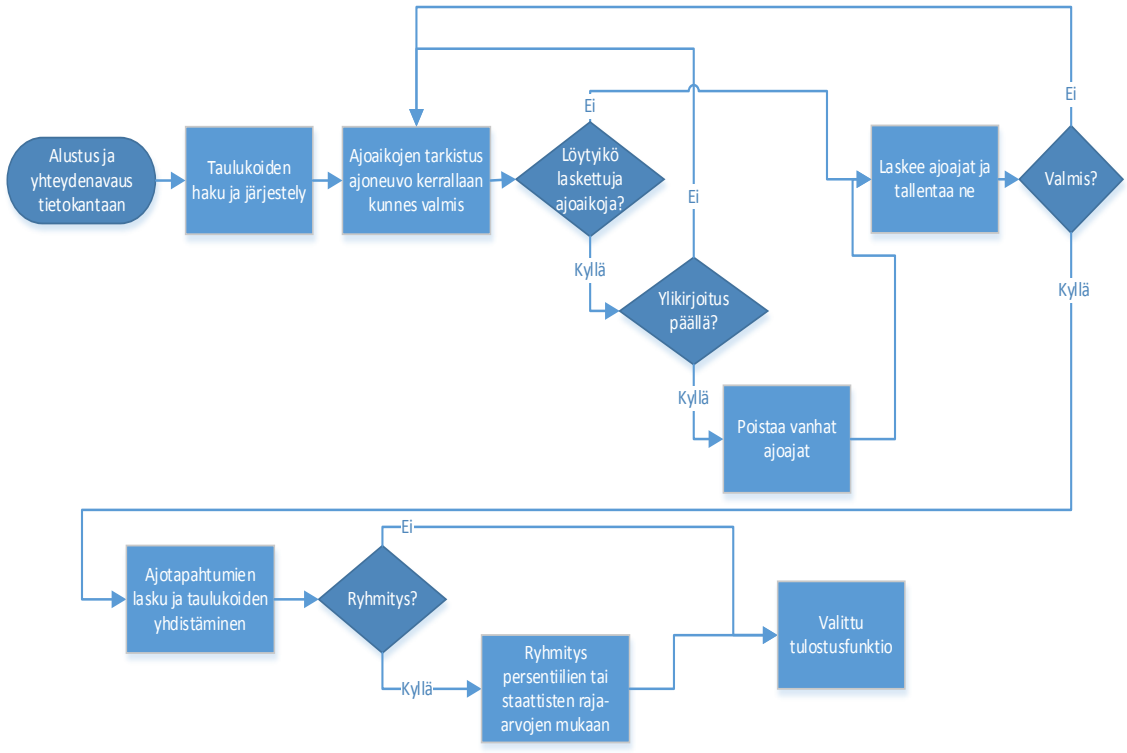
Linja-autoja käsittelevässä ohjelmassa on myös erillinen funktio, joka tallentaa tietokantaan tiedon aktiivisten paikannustietorivien määrästä. Funktio ottaa vastaan seuraavat parametrit: taulukon, joka sisältää käsiteltävän paikantimen ja ajoneuvon tunnusnumerot, aktiivisten paikannustietorivien määrän sekä aikaleimat käsiteltävän aikajakson alusta ja lopusta. Aikaleimojen tallennus tehdään siksi, että analyysiohjelma tietäisi minkä ajanjakson aktiiviset paikannustietorivit laskettiin laskettaessa metriikkaa.

6 Analyysiohjelman kuvaus

6.1 Tiedonhaku ja alustus

Päällisin puolin analyysiohjelmat ovat samanlaisia autokantaa käsiteltäessä kuin linja-autokantaa käsiteltäessä. Suurin ero on se, että linja-autoja käsiteltäessä tarkistetaan ajoneuvon aktiivisten paikannustietorivien määrää ja vastaavasti autokannassa taas ajoaikoja. Autokannassa ajoajat myös haetaan auto-olioiden avulla, mikäli niitä ei ole aiemmin laskettu.

Yksinkertaistettu prosessikaavio (kuva 6) näyttää, miten analyysiohjelma etenee autokantaa käsiteltäessä. Suurin ero prosessikaavioon tulee linja-autoja käsiteltäessä, kun ajoaikoja ei tarvitse laskea lainkaan, vaan aktiiviset paikannustietorivit haetaan suoraan kannasta.



Kuva 6. Yksinkertaistettu prosessikaavio autokantaa käyttävästä analyysiohjelmasta.

Ohjelmaan sisällytetään samat apuohjelmat, jotka olivat myös laskentaohjelmassa käytössä, eli DbHandler ja Logger, mainitakseni niistä tärkeimmät. Ohjelma avaa MySQL-

tietokantaan yhteyden ja käy sitten läpi saamiaan parametreja. Ohjelma hyväksyy kolme tai viisi parametria. Ensimmäisen niistä on oltava merkkijono, joka määrittää asetetaanko ylikirjoitusasetus päälle. Näin ollen ohjelma ei poista vanhaa tietoa, ellei erikseen niin määritetä ohjelmaa käynnistäessä.

Seuraavat kaksi parametria ovat kokonaislukuna syötettynä kuukausi ja vuosi, jonka perusteella ohjelma luo aikaleiman. Luotu aikaleima toimii haun alkuajanhetkenä. Aikaleima on syötetyn kuukauden ensimmäinen sekunti. Suorittaessa ohjelmaa kolmella parametrilla haku kohdistuu ainoastaan syötettyyn kuukauteen, eli loppumisajankohdan aikaleimaksi tulee kuukauden viimeinen sekunti.

Ohjelman voi myös käynnistää viidellä parametrilla, joista kaksi viimeistä parametria ovat myös kokonaislukuina kuukausi ja vuosi. Viimeisten parametrien mukaan luodaan haulle aikaleima loppuajankohdalle. Lopetusajankohta on syötetyn kuukauden viimeinen sekunti.

Tämän jälkeen ohjelma hakee taulukkoon kaikki ajoneuvot käyttäen valmiita kirjastoja autokantaa käsiteltäessä. Taulukkoon tallennetaan ajoneuvot olioina, eikä tehdä omaa kyselyä, niin kuin laskentaohjelmassa tehtiin. Ajoneuvot suodatetaan samalla tavalla kuin laskentaohjelmassa käyttäen suodattimena ohjelmalle syötettyjä alku- ja loppuaikaleimoja. Tämän jälkeen taulukossa on vain aikavälin sisällä aktiivisena olleet ajoneuvot.

Linja-autokannassa ajoneuvot haetaan tarkistamalla tallennettujen ajotapahtumientaulusta kaikki ne ajoneuvojen tunnukset, jotka aiheuttivat ajotapahtumia. Tämän avulla ohjelma tietää kaikki ajoneuvot haetulta aikaväliltä, jonka tiedot pitää analysoida. Linja-autokannassa ei tarvita ajoneuvoja oliomuodossa, koska ajoaikoja ei haeta olion metodia käyttäen.

Seuraavaksi ohjelmaan haetaan funktiota käyttäen kaikki ajotapahtumat haetulta aikaväliltä MySQL-tietokannasta tietokantakyselyä käyttäen. Kysely on varta vasten tätä ohjelmaa varten luotu. Valmista kirjastoa ei voitu käyttää, koska haku kohdistuu ohjelmaa varten luotuun tauluun. Parametreina funktioon syötetään alku- ja loppuaikaleimat, jotka määrittävät minkä aikavälin sisältä ajotapahtumat haetaan. Haetut tapahtumat tallentuvat väliaikaiseen taulukkoon, mitä käsitellään funktion sisällä. Tässä vai-

heessa taulukkoa ei ole vielä helppo käsitellä, koska jokainen tapahtuma on omassa alkiossaan.

Tulostaulukkoa käydään vielä läpi tarkempaa lajittelua varten funktion sisällä. Jokainen tulosrivi käsitellään yksitellen ja tallennetaan tapahtuma-taulukkoon. tapahtuma-taulukko tallennetaan kokonaisuudessaan ajoTapahtumat-taulukkoon. ajoTapahtumat-taulukko indeksoidaan ajoneuvotunnuksen mukaan. Näin ollen ohjelma voi tarkistaa löytyykö ajoneuvolle jo tapahtumia, jonka seurauksena tapahtuma-taulukko lisätään ajoTapahtumat-taulukkoon samalla indeksinumerolla, mutta uuteen alkioon. Ajoneuvon ensimmäinen tapahtuma sitä vastoin alustetaan ajoTapahtumat-taulukkoon ajoneuvotunnuksen mukaan. Alla on esimerkki siitä, mitä ajoTapahtumat-taulukko voisi sisältää.

```
ajoTapahtumat[123] =
          [0] =
          tapahtuma['nopeus']    = 45 (aikaleimojen nopeusero)
          tapahtuma['alkuaika']  = 1388527200 (alkuaikaleima)
          tapahtuma['loppuaika'] = 1388527205 (loppuaikaleima)
          [1] =
          tapahtuma['nopeus']    = 50 (aikaleimojen nopeusero)
          tapahtuma['alkuaika']  = 1388527285 (alkuaikaleima)
          tapahtuma['loppuaika'] = 1388527290 (loppuaikaleima)
```

Seuraavaksi ohjelma hakee ajoneuvojen keskimääräiset ajonopeudet haetulta aikaväliltä. Kantakyselyn tekevän funktion parametreina toimivat haetun aikavälin määrittävät aikaleimat. Kysely hakee kaikki tallennetut keskinopeusrivit tietokannasta haetulta aikaväliltä.

Funktiossa käsitellään kyselyn tulosta rivi kerrallaan samalla tavalla kuin tapahtumia hakevassa funktiossa. Tämä tarkoittaa sitä, että nopeudet-taulukkoon tallennetaan yksittäisiä nopeustaulukoita. Nopeudet-taulukko on myös indeksoitu ajoneuvotunnuksen mukaan. Toisin kuin ajotapahtumat-funktiossa, tieto ei vielä ole tarpeeksi käteväs-sä muodossa vain tällä lajittelulla. Nyt nopeudet-taulukko sisältää kaikki ajoneuvon keskinopeudet 10 päivän jaksoina yksittäisinä alkioina, niin kuin laskentaohjelmassa ne tallennettiin. Ohjelman kannalta on vain olennaista se, mikä oli keskinopeus haetun aikavälin sisällä, joten nopeudet-taulukkoa käydään vielä läpi rivi kerrallaan. Ohjelma laskee, kuinka monta keskinopeusriviä aikajakso sisältää, sekä yhteenlasketun keski-

nopeuden ja laskee sen perusteella koko aikajaksolle keskinopeuden. Tämä tieto tallennetaan uuteen ajoNopeudet-taulukkoon, joka myös indeksoidaan ajoneuvotunnuksen mukaan, mutta kukin alkio sisältää ainoastaan yhden rivin tietoa, eli aikavälin keskinopeuden. Tämä ajoNopeudet-taulukko on siten se taulukko, jonka funktio palauttaa. Seuraavassa on esimerkki, mitä ajoNopeudet-taulukko voisi sisältää.

```
ajoNopeudet[123] = 45,56 (keskinopeus, km/h)
```

Autokannassa ohjelman on tarkoitus käsitellä ajoneuvojen ajoaikaa. Tämä toimii niin, että ohjelma käy läpi löytämänsä ajoneuvotapahtumat rivi kerrallaan, johon on siis tallennettu ajoneuvojen tunnukset sekä niiden ajotapahtumat. Ajoneuvojen ajoaikaa käsiteltäessä olennaista on, että taulukko sisältää kaikki ne ajoneuvot tunnuksineen, joiden ajoajat on syytä tarkistaa. Lokiin kirjataan jatkuvasti, monesko ajoneuvo on kyseessä ja mikä on ajoneuvon tunnusnumero. Lokiin kirjataan myös tieto muistinkäytöstä. Ohjelma tarkistaa ensi töikseen, onko haettavalle ajanjaksolle määritetty käyttäjän toimesta lopetuskuukautta vai ei. Mikäli käyttäjä on näin tehnyt, luodaan aikaleima, joka vastaa haetun aloituskuukauden viimeistä sekuntia. Koska ohjelmaa on tarkoitus ajaa kokonaisina kuukausina, on tietoakin syytä käsitellä ja hakea kuukausittain. Ohjelma siirtää haun itsestään seuraavaan kuukauteen, kunnes luotu aikaleima on uudempi tai sama, kuin ohjelman alussa määritetty lopetusaikaleima.

Kuukausi kerrallaan suoritettavassa silmukassa tarkistetaan ensiksi, löytyykö kyseiseltä ajanjaksolta entuudestaan laskettuja ajotunteja. Haku tehdään funktiolla, joka suorittaa kantakyselyn ajoajat tauluun. Kyseinen taulu on varta vasten tätä ohjelmaa varten tehty analyysiohjelman nopeuttamiseksi. Funktioon syötetään parametreiksi ajoneuvotunnus sekä aikaleimat käsittelyssä olevalta aikaväliltä. Ajotuntien löytyessä ohjelma tarkistaa, onko se käynnistetty ylikirjoitusparametrilla. Ylikirjoitusparametrin ollessa tosi poistetaan vanhat ajoajat käsittelyssä olevalta aikaväliltä kyseiseltä ajoneuvolta. Poisto tapahtuu MySQL-kantakyselyllä. Poistofunktio käynnistetään samoilla parametreilla kuin hakufunktio. Ylikirjoitusparametrin ollessa epätosi siirretään käsiteltävä aikaväli seuraavaan kuukauteen ja suoritetaan uudestaan ajoaikojen tarkistus.

Ajoneuvoaikojen löytyessä ja ylikirjoitusparametrin ollessa tosi tai jos ajoneuvolle ei löytynyt aikoja, valitsee ohjelma auto-olion suodatetusta autotaulukosta. Valinta tehdään käsiteltävässä olevan ajoTapahtumat-taulukon alkion indeksinumeron, eli ajoneuvotunnuksen, avulla. Oliota käyttäen ohjelma hyödyntää olemassa olevaa kirjastoa

hakiessaan ajoneuvon ajotunnit käsittelyssä olevalta aikaväliltä. Haku palauttaa taulukon ajoaikoja alku- ja loppuaikaleiman mukaan. Näiden aikojen erotus lasketaan kyseisen aikavälin ajoajaksi. Koska kirjasto palauttaa aina kokonaisen taulukon arvoja, kun analyysiohjelman kannalta vain erotuksella on merkitystä, ja haun kestäessä pitkään sen kohdistuessa isoon tauluun, on ohjelmaa varten päätetty tehdä oma taulunsa. Tauluun tallennetaan ajoneuvotunnus, paikantimen tunnusnumero, laskettu ajoaika sekä aikavälin alku- ja loppuaikaleimat.

Ajoaikojen tarkistuksen ja mahdollisen laskemisen jälkeen tallennetut ajoajat haetaan tietokannasta ajoajat-taulukkoon. Haku tehdään funktiolla, jossa on MySQL-tietokantaan kohdistuva kysely. Haun parametreina ovat ainoastaan haetun aikavälin alku- ja loppuaikaleima. Kyselyn tulos tallentuu paikalliseen taulukkoon, jonka ohjelma käy läpi rivi kerallaan.

Ohjelma luo ajat-taulukon, johon se tallentaa ajoaika-taulukoita. Jokainen tulosrivi tallennetaan ajoaika-taulukkoon, joka tallennetaan tulosrivin ajoneuvotunnus indeksinä ajat-taulukkoon. Yhdellä ajoneuvotunnuksella voi löytyä monta riviä, joten ajoaikoja voi olla monta ajoneuvoa kohti. Tämän takia ohjelma käy vielä syntyneen ajat-taulukon läpi alkio kerrallaan. Ohjelma laskee kunkin ajoneuvon ajoajat yhteen ja tallentaa tuloksen ajoAjat-taulukkoon indeksinä taas ajoneuvo tunnusnumero. ajoAjat on se taulukko, jonka funktio lopulta palauttaa muun ohjelman käyttöön. Alla on esimerkki siitä, mitä ajoAjat-taulukko voisi sisältää.

```
ajoAjat[123] = 68,43 (yhteenlaskettu ajoaika, h)
```

Linja-autokannassa ei ole tietoa ajoajoista, joten siellä ajoajan tilalla on aktiiviset paikannustietorivit. Aktiivisten paikannustietorivien määrä haetaan tietokantakyselyllä; tämä tieto on aina luotu laskentaohjelmassa, mikäli ajoneuvoa on käsitelty. Kyseinen aktiivisten paikannustietorivien määrä jaetaan sitten tuhannella, jotta metriikaksi muodostuu syntyneet ajotapahtumat tuhatta aktiivista paikannustietoriviä kohden. Nämä arvot tallennetaan sitten aktiiviset paikannustietorivit -taulukkoon, joka indeksoidaan ajoneuvotunnusnumerolla.

Nyt ohjelmalla on kaikki tarvitsemansa, että se voi yhdistää eri tiedot yhteen yhtenäiseen taulukkoon analysointia varten. Ohjelma alkaa käydä alkio kerrallaan läpi ajoTapahtumat-taulukkoa. Kaikki käytössä olevat taulukot käyttävät indeksinään ajoneuvo-

tunnusta, joten eri taulukoiden tietojen yhdistäminen on yksinkertaista. Ensiksi ohjelma tarkistaa, löytyykö käsiteltävän ajotapahtuman ajoneuvotunnuksella ajoaikaa/aktiivisia paikannustietorivejä ollenkaan. Mikäli ajoaikaa/aktiivisia paikannustietorivejä ei löydy tai ajoaika on alle yhden tunnin, ei kyseistä ajoneuvoa käsitellä sen enempää, vaan ohjelma siirtyy seuraavaan. Aktiivisissa paikannustietoriveissä raja on tuhannessa, eli aktiivisia paikannustietorivejä pitää olla vähintään tuhat. Rajoilla ehkäistään ääriarvot, jotka ajoneuvo voi aiheuttaa matalalla ajoajalla tai vähäisellä aktiivisten paikannustietorivien määrällä. Esimerkkinä jos ajoneuvo ajaa viisi minuuttia ja saa yhden ajotapahtuman aikaiseksi, johtaisi se siihen, että ajotapahtumat tuntia kohti arvo olisi poikkeuksellisen korkea, ilman että siitä voisi päätellä mitään näytteiden lukumäärän ollessa näin matala.

Ajoajan/aktiivisten paikannustietorivien löytyessä ohjelma luo uuden paikallisen taulukon, johon se tallentaa ajoajan/aktiivisten paikannustietorivien määrän sekä keskinopeuden. Nämä arvot löytyvät aiemmin haetuista ajoNopeudet ja ajoAjat/aktiivisetPaikannusTietoRivit -taulukoista ajoneuvon tunnusnumerolla. Paikalliseen taulukkoon luodaan myös alkiot kiihdytys- ja jarrutustapahtumille, sekä alustetaan näiden arvoiksi nollat. Ohjelma käy rivi kerrallaan läpi kaikki ajoneuvoon liittyvät ajotapahtumat. Ajotapahtumista lasketaan aikaerotus tapahtuman alku- ja loppuaikaleiman mukaan sekä ajotapahtuman nopeusmuutos. Mikäli nopeus on muuttunut yli 20 (km/h)/s ei tulosta tallenneta, koska nopeusmuutos on epärealistisen suuri, joten todennäköisesti kyseessä on virhe paikannuksessa. Muissa tapauksissa taulukon kiihdytys- tai jarrutustapahtumien lukumäärää nostetaan yhdellä. Näin ohjelma toimii, kunnes kaikki ajoneuvon ajotapahtumat on käyty läpi.

Kun ohjelma on käynyt läpi kaikki ajoneuvon ajotapahtumat, laskee se vielä arvot sille, kuinka monta kiihdytys- ja jarrutustapahtumaa on ajoneuvolle luotu ajettua tuntia / tuhatta aktiivista paikannustietoriviä kohti. Nämä tiedot tallennetaan paikalliseen taulukkoon omiin alkioihin. Tämän jälkeen paikallinen taulukko tallennetaan uuteen ajoTietotaulukkoon, jonka indeksinä toimii taas ajoneuvon tunnusnumero. Tallennuksen jälkeen ohjelma siirtyy seuraavaan ajoTapahtumat-taulukon alkion käsittelyyn ja tämä toistuu kunnes kaikki on käyty läpi. Seuraavassa on esimerkki siitä, mitä ajoTieto-taulukko voisi sisältää.

```
ajoTieto[123] =
          ['nopeusmäärä']   = 50 (kiihdytystapahtumat)
```

```
['jarrutusmäärä'] = 40 (jarrutustapahtumat)
['ajoaika']       = 10,00 (ajotunnit)
['ajonopeus']     = 36,56 (keskinopeus)
['nopeus/h']      = 5 (kiihdytystapahtumat/ajoaika)
['jarrutus/h']    = 4 (jarrutustapahtumat/ajoaika)
```

6.2 Luokittelu

Ohjelmaan on toteutettu kaksi eri tavalla suoritettavaa ryhmitystä. Ensimmäinen niistä on persentiileihin perustuva toteutus, jälkimmäinen taas on staattisia raja-arvoja käyttävä toteutus. Koska suurin osa ajoneuvoista saa erittäin matalat arvot, on persentiilit valittu korkeiksi ja staattisia raja-arvoja käyttävässä toteutuksessa ovat raja-arvot yleensä vieläkin korkeampia. Tämä on tehty siitä syystä, että ne ajoneuvot, jotka aiheuttavat kaksin- tai kolmenkerroin enemmän ajotapahtumia muihin verrattuna, pääsisivät omaan luokkaansa, vaikka kyseinen luokka onkin sitten pienikokoinen verrattuna muihin luokkiin. Persentiilejä käyttävä toteutus ilmentää tätä ilmiötä hyvin, kun erot ajoneuvojen välissä korkeimmassakin persentiilissä voivat olla merkittäviä.

Persentiileihin perustuva toteutus on oma funktionsa, joka tarvitsee toimiakseen ainoastaan ajoTieto-taulukon. Funktion alussa alustetaan kaksi uutta taulukkoa, kiihdytys- ja jarrutustaulukot. Nämä taulukot täytetään käymällä alkio kerrallaan läpi ajoTieto-taulukon. Taulukkoihin tallennetaan ainoastaan tieto siitä, kuinka monta jarrutus- tai kiihdytystapahtumaa on syntynyt ajotuntia / tuhatta aktiivista paikannustietoriviä kohti, ja ne indeksoidaan ajoneuvo tunnusnumeron mukaan. Tämän jälkeen molemmat taulukot järjestetään PHP-funktiota käyttäen nousevaan järjestykseen.

Uudelleenjärjestelyn jälkeen ohjelma laskee indeksinumeron taulukoille, minkä avulla pyritään löytämään mikä on 80. persentiilin ja 95. persentiilin raja-arvoluvut. Indeksinumero lasketaan kaavalla 2.

$$\text{indeksinumero} = 0{,}8 * \text{N} \qquad (2).$$

Kaavassa 2 0,8 on haetun persentiilin prosentti desimaaleina ja N on taulukon kaikkien alkioiden lukumäärä. Tämä luku pyöristetään aina ylöspäin. Tällä menetelmällä selviää, kuinka mones luku taulukossa on kyseisen persentiilin raja-arvoluku, eli minkä arvon

alle jää tässä tapauksessa 80 % tai 95 % arvoista. Molemmille taulukoille lasketaan omat persentiilin raja-arvolukemat.

Tämän jälkeen funktio käy läpi rivi kerrallaan molemmat taulukot, kunnes silmukoissa on päästy aiemmin laskettuun indeksinumeroon. Ohjelman päästyä indeksinumeroihin osoittamiin paikkoihin taulukoissa tallentaa se niissä olevat arvot persentiilien raja-arvolukemiksi. Näitä raja-arvolukemia käytetään myöhemmin funktiossa ryhmitykseen.

Ennen kuin funktio alkaa luokitella ajoneuvoja niiden ajotapahtumien perusteella eri ryhmiin, ohjelma käy hakemassa valmiiksi alustetun taulukon funktiota käyttäen. Tämä tehdään siksi, että varmistetaan myöhemmässä tulostusvaiheessa, että taulukon ryhmät tulostuvat oikeassa järjestyksessä. Niitä on myöhemmin vaikea järjestellä, sillä ryhmien nimet eivät ole aakkosellisessa järjestyksessä ja luokitteluvaiheessa funktion olisi turha alustaa nimi joka kerta uudestaan, kun taulukkoon asetetaan ajoneuvo ajotapahtumineen. Samaa taulukkorakennetta käytetään myös staattisia raja-arvoja käyttävässä luokittelussa, joten on senkin puolesta käytännöllistä että taulukon alustaminen on viety erilliseen funktioon. Taulukkoon tallennetaan myös raja-arvolukemat myöhempää tulostusta varten.

Taulukkoon on luokittelua varten nimetty valmiiksi eri ryhmät. Ryhmiä on yhdeksän, ja ne kuvastavat kiihdytys- ja jarrutustapahtumien määrää ajoaikaan nähden. Ensimmäinen ryhmä on ”vähän kiihdytys- ja jarrutustapahtumia” -ryhmä. Seuraava ryhmä on ”vähän kiihdytystapahtumia ja kohtalaisesti jarrutustapahtumia”. Näin ryhmien nimet menevät aina viimeiseen ryhmään asti, joka on ”paljon kiihdytys- ja jarrutustapahtumia”.

Seuraavaksi funktio käy rivi kerrallaan läpi ajoneuvot ajoTieto-taulukosta. Se tarkistaa ensiksi, mihin ryhmään ajoneuvo kuuluu kiihdytysmäärien puolesta, kiihdytyksen persentiilin raja-arvolukuun verrattuna. Tämän jälkeen tarkistus tehdään jarrutusmäärien puolesta samalla tavalla, mutta jarrutusten persentiilin raja-arvolukuun verrattuna. Tarkistus tehdään siis sillä logiikalla, että jos arvo on alle 80. persentiilin raja-arvon, kuuluu se vähän tapahtumia sisältävään ryhmään. Mikäli se on 80. ja 90. persentiilin raja-arvojen välissä, kuuluu se kohtalaisesti tapahtumia sisältävään ryhmään. Yli 90. persentiilin raja-arvoluvun verran tapahtumia omaavat ajoneuvot, kuuluvat paljon tapahtumia sisältävään ryhmään. Tämä luokitusmenetelmä perustuu päätöspuu luokittelual-

goritmiin. Tämän luokittelun jälkeen palautuu taulukko, joka sisältää jokaisen ryhmän, sekä ryhmien ajoneuvot, jotka ovat ryhmänsä alkion alla.

Staattisia raja-arvoja käyttävä funktio toimii hyvin pitkälti samalla tavalla kuin persentiileihin perustuva funktio, paitsi että siinä on paljon vähemmän laskettavaa ja järjesteltävää, kun raja-arvot eivät muutu eikä niitä lasketa uudestaan joka kerta. Omia taulukoita ei tarvitse luoda, kun kerran indeksinumeroita ei lasketa lainkaan. Sama valmiiksi alustettu taulukko ryhmien nimineen haetaan tässäkin funktiossa ja luokittelufunktio toimii tismalleen samalla tavalla. Ainoa ero on, että raja-arvoluvut on etukäteen määritetty. Ohjelma palauttaa tismalleen samalla tavalla rakennetun taulukon kuin persentiilejä käyttävä funktio, paitsi että ajoneuvot ovat hyvin todennäköisesti jakautuneet ryhmiin eri tavalla.

6.3 Tulostukset ja tiedostot

Tiedostot luodaan kirjoittamalla tulostettavan merkkijonon sisältö tiedostoon. Uusi tiedosto luodaan, jos samannimistä tiedostoa ei löydy. Muutoin vanha tiedosto ylikirjoitetaan. Merkkijonoon tallennetaan tulostusfunktioihin tulevien taulukoiden sisältö, sekä kenttien nimet joiden mukaan sisältö järjestetään. Tiedostotyyppinä toimii CSV joka tarkoittaa sitä, että jokainen tietokenttä on erotettu puolipiste eli ”;”-merkillä. Rivien loppuun pitää myös lisätä rivinvaihto eli ”\n”-merkkijono. Tämän myötä tiedostot on helppo avata suoraan taulukkolaskentaohjelmissa, kuten Microsoft Excel -ohjelmassa. Liukulukujen desimaalipisteenä toimii PHP:ssä piste, mutta Suomessa se merkitään pilkulla, joten ohjelma suorittaa ennen tiedostoon kirjoittamista valmiin funktion, joka korvaa kaikki merkkijonon pisteet pilkulla. Ennen tiedostoon kirjoittamista ohjelma suorittaa myös merkistön muunnoksen, siltä varalta että ”ä”- tai ”ö”-kirjaimet esiintyisivät merkkijonossa ja aiheuttaisivat ongelmia.

Ohjelmassa on kolme erilaista tulostusfunktiota eri tarpeisiin. Yhteistä niillä on, että ne kaikki ottavat vastaan neljä yhteistä parametria mahdollisten omien parametrien lisäksi. Ensimmäinen parametri on taulukkoparametri. Tähän parametriin syötetään taulukko, jonka sisällön tulostusfunktio tulostaa. Muut kolme parametria ovat, alku- ja loppuaikaleimat sekä tieto siitä, määrittikö käyttäjä analysoitavalle aikavälille loppukuukauden vai käsittelikö ohjelma ainoastaan yhden kuukauden tietoja. Aikaleimoja käytetään tulostustiedoston luontiin niin, että tiedostonimestä käy ilmi, minkä kuukauden tietoja tiedos-

toon on tallennettu. Mikäli käyttäjä on määrittänyt analysoitavalle aikavälille loppukuukauden, lisätään tiedostonnimeen myös loppuaikaleiman määräämä kuukausi ja vuosiluku.

Ensimmäinen tulostusfunktio saa taulukkoparametrikseen lasketun ajoTieto-taulukon, jonka luonti kuvattiin luvussa 6.1. Merkkijonon alkuun lisätään kenttien nimet ”Index”, ”Tunnusnumero”, ”Kiihdytys”, ”Jarrutus” ”Kiihdytys/h”, ”Jarrutus/h” ja ”Ajoaika” / ” Aktiiviset paikannustietorivit”. Index-kenttä kertoo, kuinka mones ajoneuvo on taulukossa ja tunnusnumero taas auton tunnusnumeron. Kiihdytys- ja jarrutuskentät kertovat kyseisten tapahtumien kokonaismäärän, kun taas kiihdytys/h ja jarrutus/h kertovat nimensä mukaisesti tapahtumien määrän ajotuntiin nähden. Ajoaika-kenttään kirjoitetaan käsitellyn ajanjakson yhteenlaskettu ajoaika. ”Aktiiviset paikannustietorivit” kertoo, kuinka monta tuhatta aktiivista paikannustietoriviä ajoneuvolla on. Tämän jälkeen ohjelma käy taulukon läpi rivi kerrallaan ja lisää jokaisen ajoneuvon tiedot merkkijonoon. Tuloksena on tiedosto, jonka avulla analysoitavalta ajanjaksolta saa yleisnäkymän siitä, miten ajoneuvoille on kertynyt ajotapahtumia.

Toinen tulostusfunktio saa myös taulukkoparametrikseen lasketun ajoTieto-taulukon. Merkkijonon alkuun lisätään muutoin samat kenttien nimet ”Index” ja ”Tunnusnumero” mutta ”Kiihdytys/h” ja ”Jarrutus/h” on korvattu ”Ajotapahtumat/h” ja ”Keskinopeus (km/h)” kentillä. Ajotapahtumat/h kertoo nimensä mukaisesti kiihdytys- ja jarrutustapahtumien yhteenlasketun määrän ajotuntia kohden. Keskinopeuskenttä taas kertoo analysoitavalta ajanjaksolta lasketun keskinopeuden. Tämän jälkeen merkkijonoon lisätään ajoneuvojen tiedot rivit kerrallaan. Merkkijono tallennetaan tiedostoon kun viimeinen taulukon rivi on käsitelty. Tulostusfunktion tiedoston avulla näkee helposti miten keskinopeus vaikuttaa ajotapahtumien määrään.

Kolmas tulostusfunktio saa taulukkoparametrikseen luvussa 6.2 mainitut taulukot. Tulostus tehdään samalla tavalla riippumatta siitä, kumpi luokitusfunktio ohjelmassa on suoritettu, koska taulukon rakenne on sama. Funktio saa myös parametriksi tiedon siitä, onko luokittelu tehty persentiilimenetelmällä vai staattisilla raja-arvoilla. Tämä valinta määrittää, minkä niminen tiedosto luodaan, kun menetelmän nimi lisätään tiedostonnimeen. Funktio tallentaa merkkijonoon ensin raja-arvot, joita käytettiin luokittelussa.

Seuraavaksi merkkijonoon lisätään tieto siitä, miten ajoneuvot jakautuivat eri ryhmiin tulostamalla ryhmän numero, nimi ja ajoneuvojen lukumäärän jokaisessa ryhmässä.

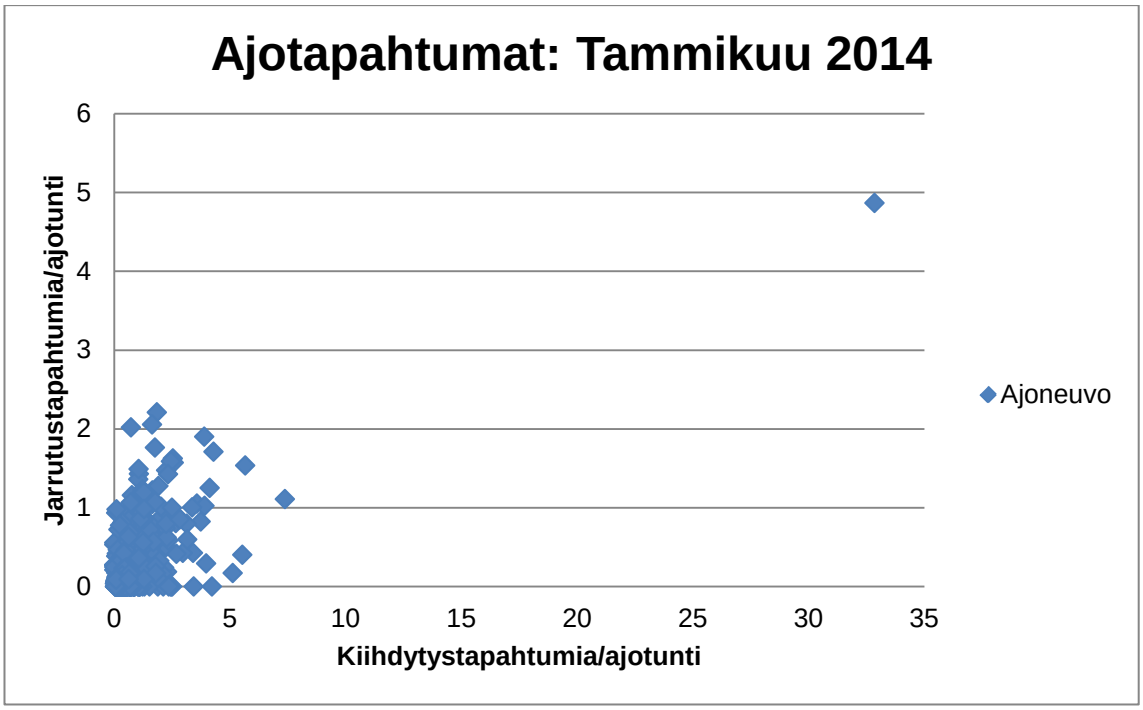
Ryhmänumero on juokseva numero, missä se ryhmä jonka ajoneuvoilla oli vähiten kumpaakin tapahtumatyyppiä, on ryhmä 1 ja niin eteenpäin. Lopuksi merkkijonoon vielä lisätään jokaisen ajoneuvon perustiedot ryhmitettyinä. Tämä tarkoittaa sitä, että merkkijonoon lisätään, mihin ryhmään kukin ajoneuvo kuuluu, ajoneuvon tunnus sekä kiihdytys/h- ja jarrutus/h-arvot. Tästä voi helposti ja nopeasti luoda kuvaajia, joista näkee, miten ajoneuvot ovat jakautuneet eri ryhmiin. Vaihtoehtoisesti tuloksesta voi myös luoda kuuvaajan, josta näkee, miten eri ryhmien ajoneuvot ovat jakautuneet.

7 Analyysimenetelmien valinta ja vertailua

7.1 Yleisesti

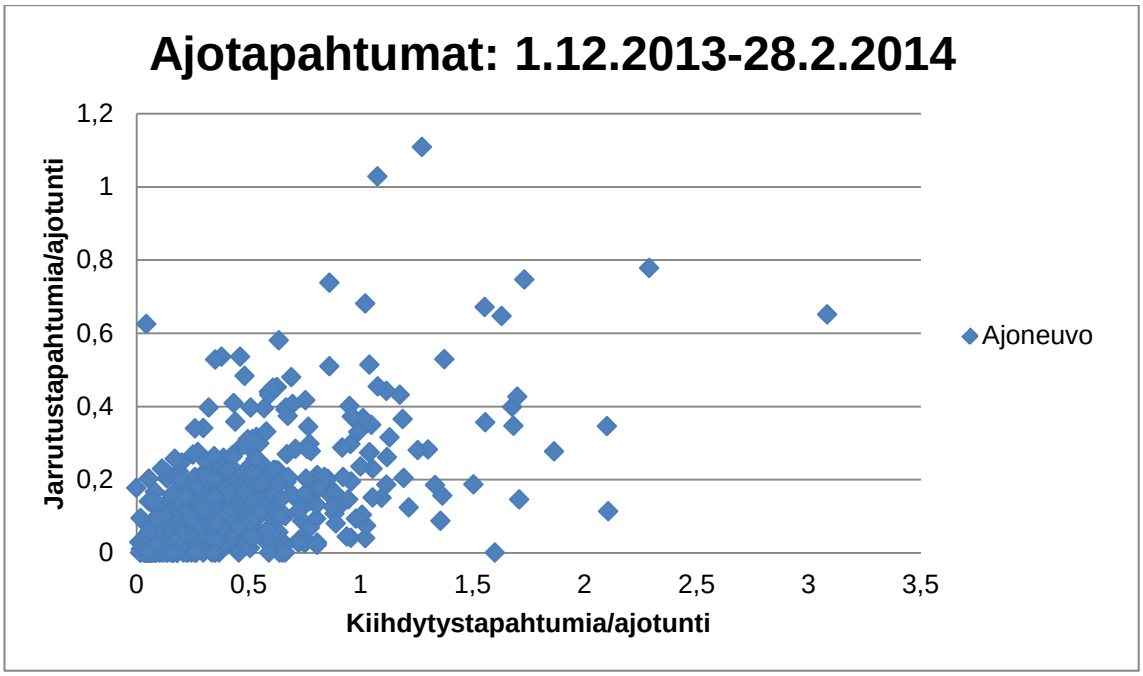
Analyysiohjelmaa varten piti tehdä päätös, miten saatuja tietoja tulisi analysoida ja ryhmitellä. Ensiksi oli tärkeää selvittää, miten paljon kutakin tarkastettavaa ajotapahtumatyyppiä ajoneuvot keskimäärin tuottivat. Tämä tarkoitti, että ohjelman piti pystyä tulostamaan lasketut tiedot helposti käsiteltävään muotoon, ensiksi vain yhdeltä kuukaudelta. Kaikki luvussa 7 esitellyt kuvaajat ovat autokannan ajoneuvoista tehtyjä ja termit sen mukaiset. Yhteensä noin 630 ajoneuvoa tuotti ajotapahtumia, joita voitiin käyttää analysointiin tarkastettavalta aikaväliltä. Autokannassa päivitettiin syksyn 2013 aikana paikantimet niin, että päivitystiheys kasvoi vanhasta kymmenen sekunnin päivitystiheydestä viiden sekunnin päivitystiheyteen. Tämän takia analysoitava ajanjakso ei ole vielä kovinkaan kattava. Linja-autot tuottavat niin paljon paikannustietoa, että tietoa ei säilytetä pitkiä aikoja, joten tietoa ei ole saatavilla pitkältä aikaväliltä.

Ohjelman tuottamasta kuvaajasta (kuva 7) paljastuu monta tärkeää asiaa, jotka pitää ottaa huomioon analyysin ja ryhmittelyn teossa. Suurin osa ajoneuvoista sijoittuu viiden kiihdytys- ja kahden jarrutustapahtuman välille ajotuntia kohden. Yhden kuukauden otoksella voi yksittäinen ajoneuvo, joka poikkeaa reilusti keskiarvosta, tehdä kuvaajasta vaikeammin hahmotettavan. Tätä tärkeämpi huomio on kuitenkin se, että näin pienellä otoksella ajoneuvo voi saada äärimmäisen korkean arvon, mutta siitä yksittäisestä tuloksesta ei pidä vielä tehdä liikaa päätelmiä kuljettajan ajotavasta. Tämän takia oli tärkeää saada ohjelma tuottamaan vastaavanlainen kuvaaja, mutta pidemmältä aikaväliltä. Pidempi aikaväli lisää tapahtumien ja ajotuntien määrää, joka johtaa siihen, että tulokset tasoittuvat ja ääripäät pienenevät.



Kuva 7. Ensimmäisen tulostusfunktion tuottaman tiedon kuvaaja ajotapahtumien jakautumisesta ajoneuvoittain yhden kuukauden ajanjaksolta autokannasta.

Aikavälin kasvaessa huomamme kuvaajasta (kuva 8), että nyt ei yhdelläkään ajoneuvolla enää ole kymmenittäin enemmän ajotapahtumia/ajotunti kuin muilla. Nyt kuvaajasta näkee selkeästi, että suurin osa ajoneuvoista on alle yhden kiihdytystapahtuman/ajotunti ja alle puoli jarrutustapahtumaa/ajotunti. Tämä antaa kuvan siitä, miten persentiilejä kannattaa määrittää niin, että siitä on asiakkaalle hyötyä. Suurimman osan ajoneuvoista ollessa lähellä nollaa, ei asiakkaan kannalta ole varmastikaan hyödyllistä ryhmitellä esim. 20 % prosentin välein. Alhaisimmat ryhmät tulisivat koostumaan melkein identtiset arvot saaneista ajoneuvoista. Asiakkaan näkökulmasta mielenkiintoisempaa on saada tietoa siitä, kuinka moni ajoneuvo aiheuttaa paljon tapahtumia ajotuntia kohti. Tämän takia ohjelma voisi ryhmitellä kaikki matalat eli ”hyvät” arvot omaan ryhmään ja keskittyä ryhmittelyssä kuvaajassa (kuva 8) nähtyihin keskiarvosta poikkeaviin ajoneuvoihin.

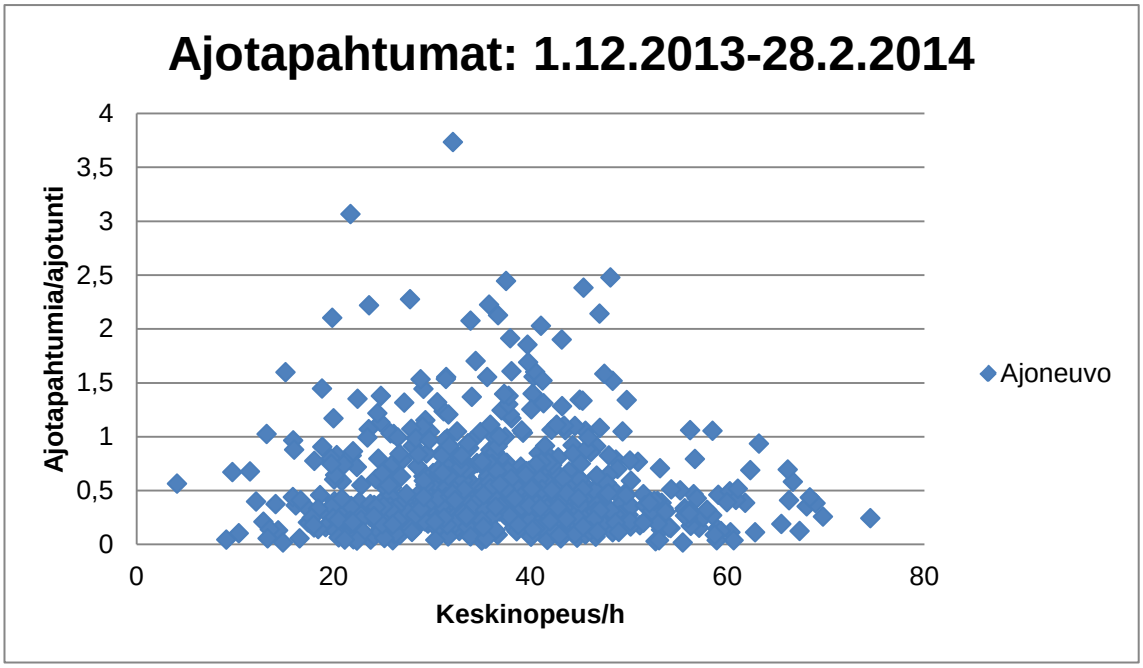


Kuva 8. Ensimmäisen tulostusfunktion tuottaman tiedon kuvaaja ajotapahtumien jakautumisesta ajoneuvoittain kolmen kuukauden jaksolta autokannasta.

Ennen ryhmittelyä on syytä tarkistaa, onko klusteroinnista hyötyä. Klusterointi olisi tapa hoitaa ryhmitys, jos huomaamme, että huomionarvoisia muuttujia on enemmän kuin kaksi. Saatavilla olevasta tiedosta on edellisissä kuvaajissa (kuva 7, kuva 8) kokonaan sivuutettu ajoneuvon keskinopeus. Syytä olisi tarkistaa, vaikuttaako keskinopeus ajotapahtumien lukumäärään. Kuvaajaa varten kiihdytys- ja jarrutustapahtumat on yhdistetty.

Tuloksena on kuvaaja (kuva 9), joka osoittaa, että keskinopeus ehkä voi vaikuttaa ajotapahtumien syntyyn. Kuvaajan (kuva 9) mukaan niiden ajoneuvojen keskuudessa, joilla on korkeimmat keskinopeudet, ei ole yhtäkään selkeästi poikkeavaa ajoneuvoa ajotapahtumien osalta. Tämä voi olla sattumaa, koska ylivoimaisesti suurimmalla osalla, tarkalleen ottaen 90 %:lla, ajoneuvoista on keskinopeus alle 50 km/h. Toinen selitys voi olla se, että korkean keskinopeuden ajoneuvot ajavat pääosin maanteillä ja moottoriteillä, eikä sen takia synny koviin kiihtyvyyksiin ja jarrutuksiin perustuvassa järjestelmässä ajotapahtumia. Kuvaaja (kuva 9) on mielenkiintoinen, mutta ei osoita varmasti, että keskinopeus olisi huomionarvoinen muuttuja ryhmittelyssä. Paljon ajotapahtumia aiheuttavilla ajoneuvoilla vaikuttaa olevan keskinopeus noin 15–50 km/h välillä tämän otoksen pohjalta. Ylipäänsä paljon ajotapahtumia aiheuttavat ajoneuvot ovat harvassa,

joten niin pieniä ryhmiä ei ole järkevää enää luokitella pienempiin ryhmiin keskinopeuden mukaan. Tämän takia klusterointia ei käytetä tässä työssä.



Kuva 9. Toisen tulostusfunktion tuottaman tiedon kuvaaja ajotapahtumien jakautumisesta keskinopeuteen nähden ajoneuvoittain kolmen kuukauden jaksolta autokannasta.

Ryhmittelyyn olen valinnut kaksi eri tapaa, joilla on eri vahvuudet ja heikkoudet. Ensimmäisenä niistä toteutin persentiileihin perustuvan ratkaisun. Tarkoituksena oli löytää sopivat rajat persentiileille, niin että ryhmityksessä olisi mahdollisimman paljon järkeä asiakkaan kannalta. Toinen ratkaisu on käyttää staattisia raja-arvoja. Toista ratkaisua varten käytin hyväkseni persentiiliratkaisun raja-arvoja ja muutin niitä mielestäni paremmiksi.

7.2 Persentiilit

Persentiilitoteutusta varten tärkeintä on määrittää sopivat persentiilit. Tämän selvittämistä varten ohjelma ajettiin toistuvasti, kunnes ryhmitys vaikutti järkevältä. Alun perin ajateltu 25. persentiili, 50. persentiili, 75. persentiili, ja 95. persentiili ei toiminut erityisen hyvin, koska jopa kolmella alimmalla persentiilillä oli melkein identtiset raja-arvot. Tämä johtuu siitä että enemmistöllä ajoneuvoista, on hyvin vähän kiihdytys- tai jarrutustapahtumia ajotuntia kohti. Kolme ryhmää, joiden ajoneuvoilla on hyvin samankaltaiset arvot, olisi turha erottaa toisistaan. Tämän sijaan toteutuksessa käytettiin ainoastaan 80. per-

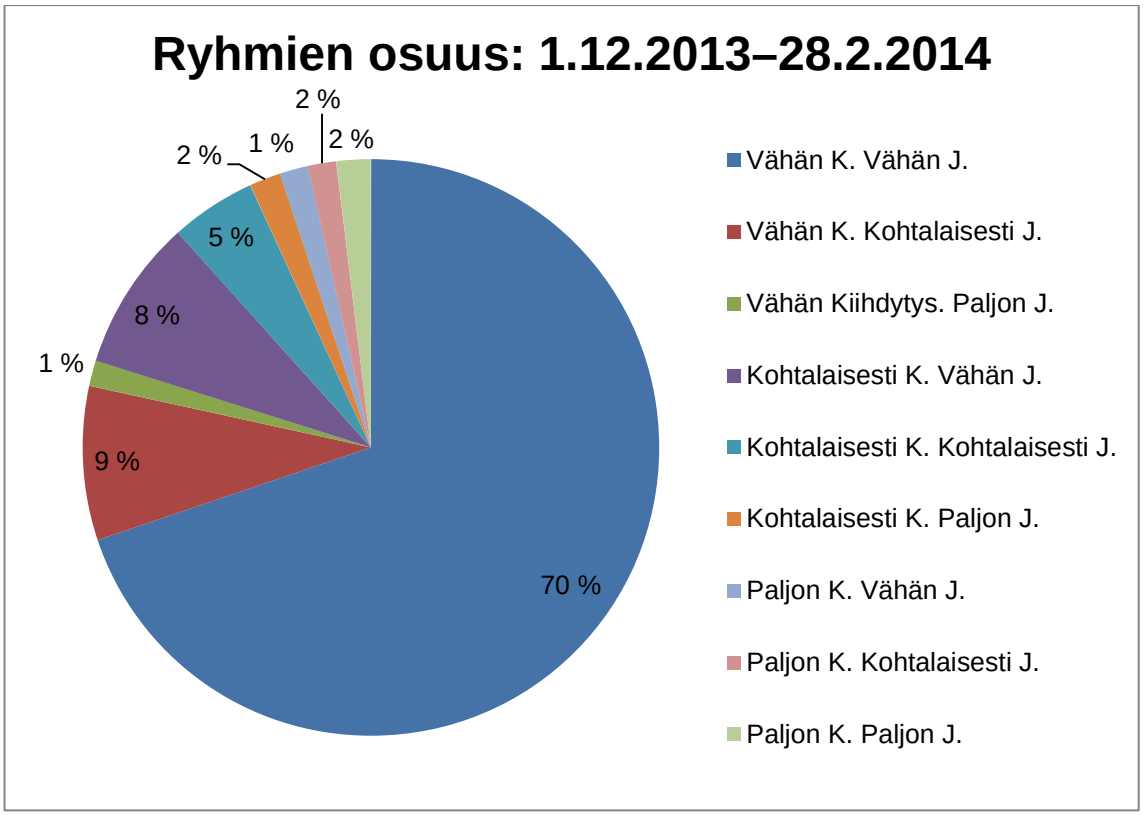
sentiiliä ja 95. persentiiliä. Ajoneuvot, jotka kuuluivat 80. persentiilin, ovat pääsääntöisesti hyvin lähellä toisiaan ajotapahtumien määrissä ajotuntia kohti.

Keskihajontataulukko (taulukko 2) osoittaa todeksi aiemmat väitteet siitä, että arvot 80. persentiilissä ovat hyvin lähellä toisiaan. Sama toistuu myös 80. persentiilin ja 95. persentiilin välissä. Ainoastaan ylimmässä persentiilissä, minne joutuvat poikkeavat ajoneuvot, on keskihajonta reilusti suurempi, aivan kuten oletettu. Tulokset vahvistavat käsitystä siitä, että persentiilit on valittu järkevästi. Ryhmittely tehdään sitten sen mukaan, miten paljon ajoneuvolla on kiihdytystapahtumia ja jarrutustapahtumia ajotuntia kohden näiden rajojen mukaan.

Taulukko 2. Keskihajonta eri persentiileissä

1.12.2013–28.2.2014	Keskihajonta	Keskihajonta
Persentiili	Kiihdytys/ajotunti	Jarrutus/ajotunti
80.	0,151772822	0,055099
80. – 95.	0,140529456	0,059079
Yli 95.	0,430502866	0,173563

Ryhmien osuuksista (kuva 10) näemme, miten ajoneuvot jakaantuvat ryhmittäin, kun ajoneuvot ryhmitellään sekä kiihdytys- että jarrutustapahtumien mukaan. Ylivoimaisesti suurin ryhmä on vähän kiihdytys- ja jarrutustapahtumia sisältävä ryhmä. Pienimmät ryhmät olivat ne, jotka koostuivat vastakkaisista ääripäistä, eli vähän jarrutustapahtumia mutta paljon kiihdytystapahtumia, sekä paljon jarrutustapahtumia, mutta vähä kiihdytystapahtumia sisältävät ryhmät.



Kuva 10. Ryhmitys persentiilitoteutuksessa, tieto tullut kolmannesta tulostusfunktiosta. K lyhenne tarkoittaa kiihdytystapahtumia ja J jarrutustapahtumia ajotuntia kohti.

Raja-arvot näille ryhmille olivat varsin matalat, niin kuin taulukosta (taulukko 3) huomataan. Etenkin jarrutustapahtumissa on syytä kyseenalaistaa, että joutuu huonoimpaan ryhmään, vaikka liian äkillisiä jarrutustapahtumia ajotuntia kohti olisi vain yli 0,398716. Persentiilien raja-arvoluvut ovat aina suhteellisia. Ne lasketaan joka kerta uudestaan saatavilla olevan tiedon perusteella. Tämän myötä voidaan sanoa, että eniten tapahtumia aiheuttavien ajoneuvojen kuljettajat ajavat suhteessa vähän tapahtumia aiheuttaviin suuremmalla nopeudenvaihtelulla. Mikäli ohjelman myötä asiakas saa kuljettajat ajamaan pienemmällä nopeudenvaihtelulla, laskisivat raja-arvoluvut myös aina siihen pisteeseen asti, että tämän toteutuksen käytännöllisyys olisi kyseenalaistettavissa.

Taulukko 3. Raja-arvot ryhmien luokittelua varten.

Raja-arvoluvut ryhmille	Kohtalaisesti	Paljon
Kiihdytystapahtumia/ajotunti	0,612114028	1,076921
Jarrutustapahtumia/ajotunti	0,20000424	0,398716

7.3 Staattiset raja-arvot

Staattiset raja-arvot määräytyivät tarkistamalla aiempia tuloksia, ja päättelemällä, mitkä raja-arvot ovat loogisia. Loogisena voi pitää sen, että huonoimpaan ryhmään ei päädy turhan moni ajoneuvo siinä tapauksessa, että keskiarvo on erittäin matala. Käyttämällä staattisia raja-arvoja voidaan eri ajanjaksoja helpommin verrata toisiinsa. Tuloksista näkee nopeasti, miten suuri osa ajoneuvoista kuuluu mihinkin ryhmään ja miten nämä määrät vaihtelevat ajanjaksoittain. Persentiilitoteutus taas aina luo ryhmät suhteessa toisiinsa, joten sen tuottamaa tietoa ei ole yhtä käytännöllistä käyttää tähän tarkoitukseen. Menetelmän toinen hyöty on, että jos asiakkaan kanssa on päätetty nämä raja-arvot, näkee tuloksista nopeasti, mitkä ajoneuvot ylittävät asiakkaan hyväksymät rajat sekä näiden ajoneuvojen tarkan määrän. Tämän tulostusfunktion tilastoja seuraamalla voi huomata, jos ajotapakoulutus tai mahdolliset muut toimenpiteet ovat toimineet.

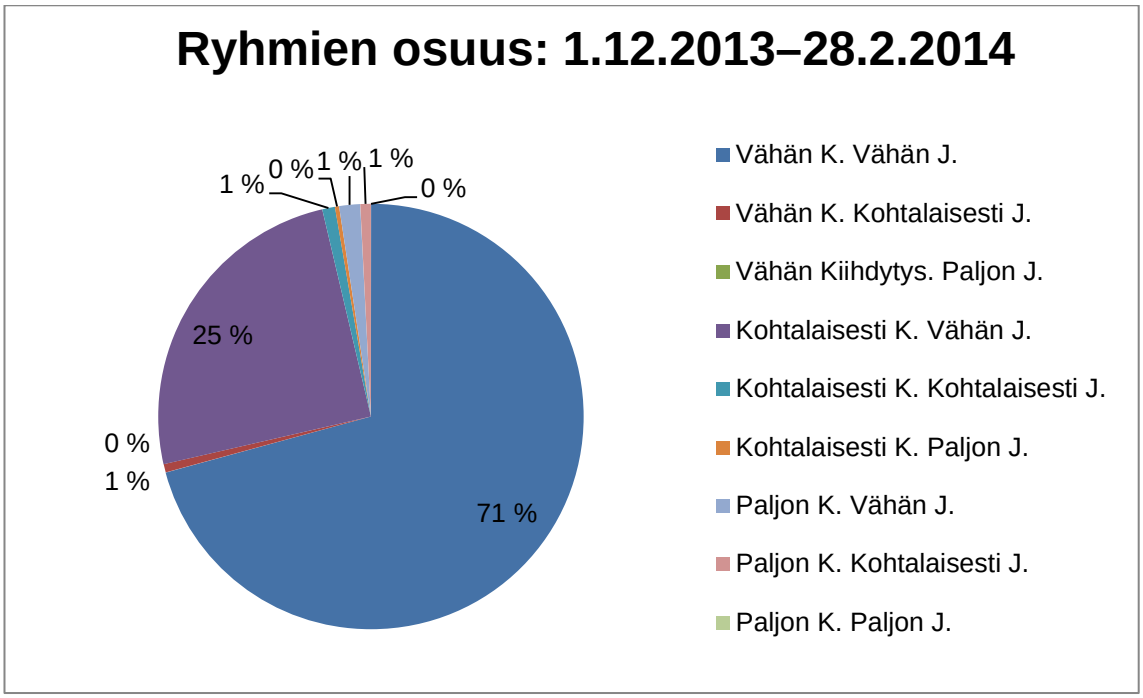
Keskihajonta (taulukko 4) oli kiihdytystapahtumissa samaa luokkaa kahdessa alhaisimman raja-arvon -ryhmässä ja reilusti suurempi ”paljon” -ryhmässä, aivan kuten tarkoituksena olikin. Menetelmän tarkoitus on ennen kaikkea löytää poikkeavat ajoneuvot, ja ne selkeästi ovat nyt ryhmitettyinä ylimmässä ryhmässä. Jarrutustapahtumissa sama ilmiö ei aivan toistu. Suurilta osin tämä johtuu siitä, että jarrutustapahtumia ajotuntia kohti on keskimäärin paljon vähemmän kuin kiihdytystapahtumia ajotunti. Tämä johtaa siihen että kun staattiset raja-arvot asetetaan asiakkaan kannalta loogisesti, ovat ne silti niin korkeita, että käytännössä kaikki ajoneuvot sijoittuvat kahteen matalimman raja-arvon -ryhmään. Tällä tarkoitetaan sitä että ”paljon” -ryhmään ajoneuvo ei joudu jarrutustapahtumien suhteen, ellei niitä oikeasti ole edes kerran ajotuntia kohti.

Taulukko 4. Keskihajontataulukko staattisissa raja-arvoissa.

1.12.2013–28.2.2014	Keskihajonta	Keskihajonta
Raja	Kiihdytystapahtumat	Jarrutustapahtumat
Alle kohtalainen	0,204314	0,100704
Kohtalaisen ja paljon välissä	0,222569	0,202572
Yli paljon	0,59319	0,223369

Ryhmien osuuksista (kuva 11) näkee, että korkeimpiin ryhmiin pääsee ainoastaan harva ajoneuvo. Korkeimmissa kiihdytysluokissa on tässä ryhmityksessä ainoastaan 15 ajoneuvoa. Tämän menetelmän avulla on helppo seurata, jos kuljettajat muuttavat ajotapansa parempaan tai huonompaan suuntaan, kun rajat pysyvät samoina. Tarkastel-

taessa yksittäisiä kuukausia, näyttäisi jakauma (kuva 11) hyvin erilaiselta: kuvaajassa olisi paljon enemmän ajoneuvoja kohtalaisesti ja paljon nimetyissä ryhmissä. Tarkastelemalla pidemmältä aikaväliltä tasaantuvat ajoneuvojen ajotapahtumien määrät ajotuntia kohti.



Kuva 11. Ryhmitys staattisia raja-arvoja käyttäen, tieto tullut kolmannesta tulostusfunktiosta. K lyhenne tarkoittaa kiihdytystapahtumia ja J jarrutustapahtumia.

Raja-arvot (taulukko 5) ovat staattisia raja-arvoja käyttävässä toteutuksessa selkeästi korkeammat kuin persentiilitoteutuksessa. Syynä tähän oli tahto saada erilleen, selkeästi paljon ajotapahtumia ajotuntia kohti aiheuttavat ajoneuvot, muista ajoneuvoista. Alle kohtalaisen rajan sijoittuu suurin osa ajoneuvoista molempia ajotapahtumatyyppiä tarkasteltaessa. Raja-arvoja valittaessa ei ollut oleellista se että ryhmät olisivat samankokoisia. Mikäli ajoneuvo aiheuttaa valitut raja-arvot ylittävän määrän ajotapahtumia ajotuntia kohti, on se perusteltua luokitella sen mukaan riippumatta siitä miten muut ajoneuvot siinä kuussa ovat ajaneet.

Taulukko 5. Raja-arvot staattisten raja-arvojen toteutuksessa

Raja-arvot ryhmille	Kohtalainen	Korkea
Kiihdytys	0,5	1,5
Jarrutus	0,5	1

8 Toimivuus ja suorituskyvyn arviointi sekä kehitysideat

8.1 Toimivuus

Työni ensimmäinen tavoite oli kehittää soveltuva laskentamenetelmä, jonka tulosten perusteella ajoneuvoja pystyisi luokittelemaan ja vertaamaan toisiinsa. Toinen tavoite oli kehittää ohjelma, joka kävisi läpi suuren määrän paikannustietoa ja suorittaisi sille laskentaa ja analyysiä. Ohjelman toimivuutta arvioitaessa ensimmäiseksi pitää arvioida laskentamenetelmän toimivuutta.

Laskentamenetelmä perustui nopeusvaihteluun paikannustietorivien välissä. Tämän avulla pystyy päättelemään, onko ajoneuvo kiihdyttänyt tai jarruttanut liian kovaa, jos se keskimäärin on ylittänyt asetetun raja-arvon nopeuden vaihtelulle. Menetelmä ei huomioi sitä, että kiihdyttäminen ja jarruttaminen eivät ole lineaarisia tapahtumia. Toinen mahdollisesti ongelmallinen kohta on, että suuret nopeudenmuutokset mitataan samoilla raja-arvoilla ajoneuvon vauhdista riippumatta. Kolmas ongelmakohta on, että autokannassa ajoneuvot päivittävät vain viiden sekunnin välein paikannustietoa, mikä voi aiheuttaa epätarkkuutta laskennassa ja rajoittaa ajotapa-analyysin tekoa.

Laskentamenetelmäni on edellä mainituista mahdollisista ongelmista huolimatta pätevä analyysin tekoon. Tarkistamalla, onko ajoneuvo keskimäärin kiihdyttänyt tai jarruttanut liian kovaa ajanjakson aikana, vähenee virheellisten ajotapahtumien määrä. Parempaa menetelmää on myös vaikeaa kehittää, kun tiedon päivitystiheys asettaa omat rajoituksensa. Työni tarkoitukseen menetelmä on varsin riittävä mahdollisista puutteista huolimatta. Tärkeintä oli todistaa menetelmän toimivuus yleisellä tasolla, ja että sen avulla on mahdollista suorittaa ajotapa-analyysiä.

Ohjelmani pystyi käymään läpi ajoneuvojen paikannustietoja ja luokittelemaan laskentamenetelmän tulosten perusteella ajoneuvot. Tuloksena nähdyt kuvaajat (kuva 7, kuva 8, kuva 9, kuva 10, kuva 11) osoittavat, että ohjelmani toimii tarkoituksenmukaisesti. Ajoneuvot on luokiteltu ja niille on laskettu ajotapahtumiin ajotuntia kohti perustuvat arvot, joiden avulla niitä voi myös vertailla keskenään. Ohjelmalle asetetut toimivuustavoitteet täyttyivät mainiosti autokantaa käsittelevällä puolella.

Linja-autoja käsittelevässä osassa ohjelma törmäsi jatkuvasti haasteisiin. Tietomäärän moninkertaistuessa autokantaan verrattuna aiheutti selkeitä ongelmia. Ohjelmaa muokattiin jatkuvasti, mutta muistimäärä loppui siitä huolimatta joka kerta kesken kehitysympäristössä. Testi- ja tuotantoympäristössä menetelmä on tarpeeksi nopea, koska ympäristöt ovat todella paljon tehokkaampia kehitysympäristöön verrattuna. Menetelmä osoittautui toimivaksi linja-autojakin käsiteltäessä mutta ainoastaan testiympäristössä.

8.2 Suorituskyky

Laskentaohjelman ensimmäinen toimiva versio käytti autokannassa laskemiseen 12,5 tuntia kehitysympäristössä, ja varasi huipussaan muistia 205,7 MB. Ohjelma tosin kaatui herkästi muistirajan ollessa 256 MB, ja joidenkin ajoneuvojen kohdalla tämä raja ylittyi. Ongelmana oli, että paikannustieto haettiin kuukausi kerrallaan, ja tämä oli tiettyjen ajoneuvojen kohdalla liikaa. Viimeistelty versio hakee paikannustiedon noin kymmenen päivän erissä, mikä tarkoittaa, että tietokantaan kohdistuvia kyselyitä on roimasti enemmän. Viimeistellyn laskentaohjelman version suorittaminen vie noin 17 tuntia kehitysympäristössä ja varasi huipussaan muistia 118,5 MB. Tuotanto- ja testiympäristössä taas aika on 15 minuuttia. Tuotanto- ja testiympäristö on merkittävästi kehitysympäristöä tehokkaampi. Viimeistelty versio suorittaa olemassa olevan tiedon tarkistuksia ja asetuksista riippuen myös poistoja. Tämän lisäksi se myös tekee ajonopeuden tallennuksia aikaisemmasta poiketen. Kaikki ajotapahtumat tallennetaan myös kymmenen päivän erissä, joten tallennuksiakin suoritetaan enemmän.

Laskentaohjelmaa kehitettäessä suorituskykyä parannettiin useampaan otteeseen optimoimalla ohjelmaa. Tärkeimpänä oli hakukyselyn optimointi sen viedessä melkein kaiken suoritusajan. Kyselyä parannettiin vähentämällä haettavan tiedon määrää; kaikkien kenttien tiedot eivät ole ohjelman kannalta olennaisia. Ohjelma käy läpi noin 90 000 paikannustietoriviä sekunnissa. Paikannustiedon hakukyselyiden teko kestää paikannustiedon määrästä riippuen paristakymmenestä sekunnista useampaan minuuttiin kehitysympäristössä, mutta alle kaksi sekuntia testiympäristössä. Ohjelman laskiessa kerralla autokannan koko kuukauden tiedot alle vuorokaudessa voidaan suorituskykyä pitää hyvänä.

Analyysiohjelman suorittaminen autokannassa lopullisessa versiossa kestää testiympäristössä noin 20 sekuntia ensimmäisellä kerralla. Tämä johtuu siitä, että ensimmäisellä

kerralla ohjelma laskee ajoaikoja ajoneuvoille, mikäli niitä ei ole valmiiksi tallennettu analyysiohjelman käyttämään tauluun tietokannassa. Ajoajan haku, jos se on aikaisemmin laskettu ja tallennettu tietokantaan, on melkein välitöntä. Tietokannasta tallennetut ajoajat haettaessa ohjelma suorittaa haut, lajittelut, ryhmitykset, luokittelut sekä neljän tiedoston tulostuksen alle sekunnissa.

Linja-autoja käsittelevää ohjelmaa parannettiin lukuisia kertoja. Se oli silti liian hidas kehitysympäristöön. Yhden ajoneuvon yhden kuukauden laskenta kesti jo 6 tuntia kehitysympäristössä, joten menetelmää ei voi siinä ympäristössä käyttää. Testi- ja tuotantoympäristössä menetelmä on tarpeeksi tehokas. Yhden ajoneuvon yhden kuukauden laskenta kestää testiympäristössä arviolta reilut 5 minuuttia. Lopullisissa ympäristöissä ohjelma suoriutuu laskennasta mallikkaasti.

8.3 Kehitysideat

Ohjelmaani voisi parantaa lukuisilla eri menetelmillä; osa niistä tosin vaatisi myös muutoksia järjestelmään. Ensimmäiseksi ohjelman tuottamat tulokset voisi tallentaa tietokantaan niin, että niitä voisi myös esittää käyttöliittymässä suoraan. Vaihtoehtoisesti siirtämällä autokantaa analysoivan analyysiohjelman suorittama ajo-ajan laskenta laskentaohjelman alaiseksi, voisi tulokset laskea aina uusiksi haussa. Tämä ei olisi käyttäjän kannalta liian hidasta, mutta toistuvassa käytössä turhan raskasta kylläkin. Analyysiohjelmanlaskenta kestää noin puoli sekuntia, mutta suorittamalla kerran analyysiohjelman jokaiselle lasketulle kuukaudelle ja tallentamalla nämä tulokset, olisi tulokset saatavilla käytännössä välittömästi. Hyvin pienillä muokkauksilla voisi ohjelma kerralla laskea määrätyltä aikaväliltä jokaisen ajoneuvon tulokset ja tallentaa ne tietokantaan.

Linja-autoja käsittelevä laskentaohjelma kaipaa vielä lisää optimoimista suoritusajan parantamiseksi. Yksi asia joka parantaisi tilannetta jo huomattavasti, olisi jos tallennusvaiheessa tieto kopioitaisiin erilliseen tauluun, jossa peräkkäiset nollavauhtirivit poistettaisiin. Tämä vähentäisi läpikäytävää tietomäärää suunnattomasti. Mikäli tämmöiseen muutokseen olisi mahdollisuuksia, voisi muitakin ylimääräisinä pidettäviä kenttiä jättää kopioimatta, kuten vastaanottoaika mainitakseni yhden niistä. Aikaleimoista ainoastaan tiedon luomisajankohta on arvokas ohjelmani kannalta.

Autokantaa käsittelevissä ohjelmissa voisi olla hyödyllistä tarkistaa, onko raja-arvo sopiva sitten, kun tietoa on pidemmältä aikaväliltä kuin kolmelta kuukaudelta. Raja-arvoa voisi myös kokeilla empiirisillä testeillä kiinnittämällä paikannin testiajoneuvoon. Tämän avulla voisi myös olla mahdollista huomata, tarvitseeko raja-arvon olla eri korkeissa nopeuksissa vai ei.

Autokantapuolella järjestelmää voisi parantaa tihentämällä viestien päivitysväliä. Tämä mahdollistaisi entistä tarkemman seurannan ja ajotapa-analyysin. Haittapuolena se myös nostaisi viestiliikenteen määrää ja tallennettavan tiedon määrää. Päivitystiheyttä nostamalla voisi ohjelmaani kehittää myös laskentaa keskikiihtyvyydelle ja muille kiinnostaville ajoneuvokohtaisille arvoille. Niin kauan kun järjestelmä ei saa kiihdytysanturitietoa tai suoraan tietoa ajoneuvojen tietokoneilta, tämä olisi paras tapa parantaa seurantaa. Tämä tosin nostaisi entisestään tietokantakyselyitten kestoa kasvattamalla tietomäärää.

Molemmissa ohjelmissa voisi tarkoituksella kytkeä pois virhearvojen suodatuksen, ja käyttää ohjelmaa näiden löytämiseen. Tämä tarkoittaisi että ohjelma voisi erikseen pitää kirjaa millä ajoneuvoilla on näitä poikkeuksellisen paljon. Tämän tiedon pohjalta voisi tutkia, onko paikantimessa jotain vialla, kun virhearvoja syntyy niin paljon.

Asiakkaan näkökulmasta ohjelmaa voisi autokantapuolella kehittää vielä niin, että tulostuksissa ajoneuvot ryhmitetään myös organisaatioittain tai lääneittäin. Tämä mahdollistaisi sisäisen vertailun ja mahdollisesti leikkimielisen kilpailun. Ohjelman tallennettuja tietoja voisi käyttöliittymän avulla käydä läpi ja löytää suurimmat parantajat. Laajennusmahdollisuuksia on runsaasti.

9 Yhteenveto

Työllä oli kaksi tarkoitusta. Ensimmäiseksi oli tarkoituksena kehittää laskentamenetelmä, jonka tulosten perusteella ajoneuvoja pystyisi luokittelemaan ja vertaamaan toisiinsa. Laskentamenetelmän piti olla hyödyllinen riippumatta päivitystiheydestä. Toiseksi tarkoitus oli kehittää ohjelma, joka kävisi läpi suuren määrän paikannustietoa ja suorittaisi sille laskentaa ja analyysiä. Tulosten perusteella ajoneuvot piti myös ryhmittää.

Työni tuloksena on laskentaohjelma, joka tekee juuri niin kuin työn alussa määritettiin. Se käy tehokkaasti läpi paikannustietorivit ja tallentaa ajotapahtumat ja muut olennaiset tiedot tietokantaan. Analyysiohjelma hakee nämä tiedot, ja käy tehokkaasti läpi laskien sitten ajoneuvoille arvot, jotka perustuvat ajotapahtumiin ajotuntia tai tuhatta aktiivista paikannustietoriviä kohti. Tulokset ryhmitellään ja tulostetaan käyttäjälle. Nämä tiedot ovat asiakkaalle varmasti mielenkiintoisia ja hyödyllisiä. Tästä näkökulmasta työ on varsin onnistunut.

Oppimisprosessina työ antoi tekijälleen paljon ja lukijan se perehdyttää ajotapa-analyysin tekoon paikannustiedon pohjalta. Ajotapa-analyysiä voi tehdä viidenkin sekunnin päivitystiheydellä, mutta sitä ei kannata pitää absoluuttisena totuutena. Mikään järjestelmä ei voi huomioida kaikkia muuttujia, mutta hyvän yleiskuvan tilanteesta ohjelma kyllä tuottaa. Tieto tarkentuu tiedon lisääntyessä, joten otannan kasvaessa autokannassa kolmesta kuukaudesta, on syytä odottaa entistä hyödyllisempiä tuloksia ohjelmalta. Työssä tuli esille menetelmän vahvuudet ja heikkoudet, sekä mihin suuntaan ohjelmaa kannattaa tulevaisuudessa viedä.

Lähteet

1 Mayer-Schönberger, Viktor & Cukier, Kenneth. 2013. Big Data A Revolution That Will Transform How We Live, Work and Think. New York: Houghton Mifflin Harcourt.

2 Constantinescu, Zoran. Marinoiu, Christian. Vlâdoiu, Monica. 2010. Driving Style Analysis Using Data Mining Techniques. <www.researchgate.net/publication/228654786_Driving_Style_Analysis_Using_Data_Mining_Techniques/file/9fcfd5060768ca268f.pdf> Luettu 26.1.2014.

3 Rigolli, Marco. Brady, MIchael. 2005. Towards a Behavioural Traffic Monitoring System. Verkkodokumentti. University of Oxford <http://pdf.aminer.org/000/056/581/towards_a_behavioural_traffic_monitoring_system.pdf> Luettu 10.2.2014.

4 Liimatainen, Heikki. Metsäpuro, Pasi. Joukkoliikenteen energiatehokkuuden seuranta, raportointi ja kehittäminen. Verkkodokumentti. Tampereen teknillinen yliopisto. <http://www.motiva.fi/files/4043/JOLEN_Joukkoliikenteen_energiatehokkuuden_seuranta_raportointi_ja_kehittaminen.pdf> Luettu 11.2.2014.

5 H.Y. Tong , W.T. Hung & C.S. Cheung, 2000. On-Road Motor Vehicle Emissions and Fuel Consumption in Urban Driving Conditions, Journal of the Air & Waste Management Association Vol. 50, Iss. 4, 2000.

6 Hoffmann, Eivind. Chamie, Mary. 1999. Standard statistical classifications: basic principles <https://unstats.un.org/unsd/class/family/bestprac.pdf> Luettu 19.3.2014.

7 Dua, Summet & Du, Xian. 2011. Data Mining and Machine Learning in Cybersecurity. Boca Raton: Taylor and Francis Group

8 Ärje, Johanna. Tilastollisia luokittelumenetelmiä koneelliseen tunnistamiseen – sovellus pohjaeläinaineistoon. Pro gradu –tutkielma. <http://urn.fi/URN:NBN:fi:jyu-201006102027> Luettu 27.3.2014

9 Jain, Anil & Dubes, Richard 1988. Algorithms for clustering data. New Jersey: Prentice Hall.

10 Abhisek, Mudgal. 2011. Modeling driving behavior at traffic control devices. Verkkodokumentti. Iowa State University <http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=1373&context=etd> Luettu 13.2.2014.

11 Liimatainen, Heikki. Rauhamäki, Harri. Liedes, Matti. 2009Kuljetusalan energia-tehokkuuden hallinta- ja kannustinjärjestelmät. Verkkodokumentti. Tampereen teknillinen yliopisto. <http://www.tut.fi/verne/wp-content/uploads/rasturaportti.pdf> Luettu 20.3.2014.

12 MacKenzie, Don. Heywood, John. 2012. Acceleration Performance Trends and the Evolving Relationship Between Power, Weight, and Acceleration in U.S. Light-Duty Vehicles. Verkkodokumentti. Massachusetts Institute of Technology. <http://web.mit.edu/sloan-auto-lab/research/beforeh2/files/MacKenzie%20&%20Heywood%20-%20TRB%20-%2012-1475.pdf> Luettu 10.3.2014.

13 An, Feng. Sauer, Amanda. 2004. Comparison of passenger vehicle fuel economy and ghg emission standards around the world. Verkkodokumentti. <http://www.c2es.org/docUploads/Fuel%20Economy%20and%20GHG%20Standards_010605_110719.pdf> Luettu 10.3.2014.

14 CGI Sisäinen verkkodokumentti. Luettu 11.3.2014