AIKASARJOJEN RYHMITTELY

Pekka Nykänen

12.6.2008
Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Tiedon valtava lisääntyminen edellyttää yhä parempia tiedon analysointikeinoja. Aikasidonnainen tieto poikkeaa perinteisestä tiedosta siten, että se muuttuu ajan kuluessa. Seuraamme ajan suhteen järjestettyä tietoa muun muassa pörssikursseista ja säätiedotuksista. Jopa ihmisten puhe voidaan käsittää ajan suhteen järjestettynä tietona. Aikaan sidottu data asettaa erityisvaatimuksia käytetyille tiedonlouhintamenetelmille, joilla pyritään löytämään olemassa olevasta aineistosta kiinnostavaa lisätietoa.

Tässä tutkielmassa käytetään ryhmittelyä tiedonlouhintamenetelmänä, jolla pyritään selvittämään aikasarjadatasta sen sisäisiä rakenteita. Aikasarjojen keskinäiseen vertailuun käytetään niin sanottua dynaamista aikasovitusta, joka sovittaa kaksi aikasarjaa toisiinsa nähden muokkaamalla niitä ajan suhteen. Tutkielmassa testataan neljän erilaisen algoritmin toimintaa. Tämän lisäksi määritellään aikasarjaryhmittely kombinatorisena optimointiongelmana ja kuvataan algoritmi, jonka avulla voidaan luoda useasta aikasarjasta niitä kuvaava keskiarvo-aikasarja.

ACM-luokat (ACM Computing Classification System, 1998 version): I.5.3
Avainsanat: aikasarja, dynaaminen aikasovitus, k-medoids, satunnaistettu paikallishaku, ryhmittely

Esipuhe

Haluan kiittää professori Pasi Fräntiä ohjauksesta ja johdatuksesta aiheen pariin. Erittäin suuret kiitokset myös FM Ville Hautamäelle kärsivällisestä ja laajamittaisesta avusta tutkielman yksityiskohtia ratkottaessa.

Kiitokset myös läheisilleni heidän tuestaan tutkielman ulkopuolella.

Lyhenteet

AHC	Agglomerative Hierarchical Clustering
AHC-KM	Agglomerative Hierarchical Clustering with k-medoids
DP	Dynamic Programming
DTW	Dynamic Time Warping
DHC	Divisive Hierarchical Clustering
KM	K-Medoids
LCSS	Longest Common Subsequence
MAE	Mean Absolute Error
MSA	Multiple Sequence Alignment
MSE	Mean Square Error
PAM	Partitioning Around Medoids (k-medoids)
RLS	Randomised Local Search
UCI	University of California, Irvine

Symbolit

Ø	tyhjä joukko
<<	paljon pienempi kuin
$\boldsymbol{C}$	ryhmien edustajien joukko $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M$
$\mathbf{c}_i$	i:nnen ryhmän edustaja
D	etäisyysfunktio
D_E	euklidinen etäisyys
D_C	kumulatiivinen etäisyys
D_{DTW}	dynaaminen aikasovitus -etäisyys
I	k-medoids -algoritmin iteraatioiden lukumäärä
J	optimoitava tavoitefunktio
K	aineiston alkioiden ulotteisuus
L_A	aikasarjojen keskipituus
L	aikasarjojen maksimipituus
M	ryhmien lukumäärä, mallin koko
N	aineiston koko
N_j	j:nnen klusterin aikasarjojen lukumäärä
P	prototyypin optimointiin käytettävien iteraatioiden lukumäärä
R	RLS-iteraatioiden lukumäärä
$\boldsymbol{S}$	aikasarja-aineisto, joka koostuu aikasarjoista $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N$
$\boldsymbol{S}_j$	aikasarja-aineisto, joka kuuluu ryhmään j
$\boldsymbol{S}_j(s_i)$	aikasarja-aineiston j:nnen ryhmän i:s aikasarja
$\mathbf{s}_i$	aikasarja-aineiston $\boldsymbol{S}$ i:s aikasarja
$\mathbf{s}_i(j)$	aikasarjan $\mathbf{s}_i$ j:s otos
$\mathbf{s}_i(j_k)$	aikasarjan $\mathbf{s}_i$ j:nnen otoksen k:s havainto
$\lvert\mathbf{s}_i\rvert$	aikasarjan $\mathbf{s}_i$ pituus
T	suoritusaika
$\boldsymbol{X}$	aineisto, joka koostuu alkioista $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$
$\boldsymbol{X}_i$	aineiston $\boldsymbol{X}$ i:s ryhmä
$\boldsymbol{X}_i(\mathbf{x}_j)$	aineiston $\boldsymbol{X}$ i:nnen ryhmän j:s alkio

Sanasto

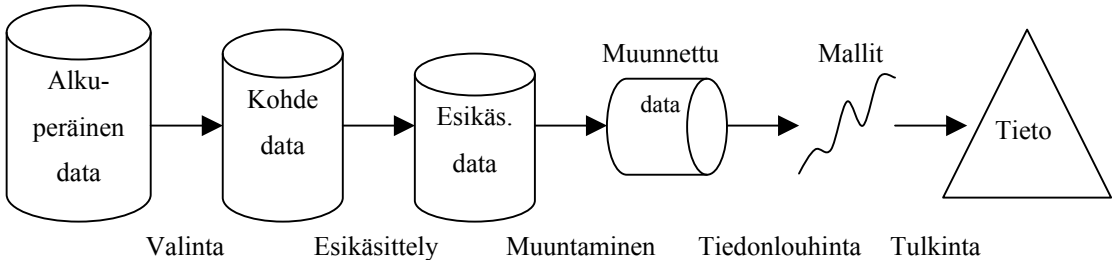
Agglomerative	Kokoava
Attribute/Feature/Variable	Ominaisuus
Centroid	Keskipiste, sentroidi
Classification error	Luokitteluvirhe
Clustering	Ryhmittely
Convex hull	Konveksi peite
Data Mining	Tiedonlouhinta
Dendrogram	Dendrogrammi, ryhmittelypuu
Distortion	Virheellisyys, ryhmittelyvirhe
Divisive	Jakava
Dynamic Time Warping	Dynaaminen aikasovitus
Feature extraction	Piirreirrotus
Feature selection	Piirrevalinta
Frame/time point	Otos
Fuzzy Clustering	Sumea ryhmittely
Hard Clustering	Kova ryhmittely
Knowledge Discovery in Databases	Tiedon havaitseminen tietokannoista
Multiple Sequence Alignment	Monen sekvenssin rinnastus
Observation	Havainto
Pairwise Nearest Neighbor	Pareittainen yhdistelymenetelmä
Partition	Ositus
Supervised Classification	Luokittelu
Stroke	Veto
Unsupervised Classification	Ryhmittely
Warping path	Sovituspolku

SISÄLLYSLUETTELO

1 JOHDANTO

Tietojärjestelmien kehityksen ja erityisesti tiedon tallennusmahdollisuuksien valtavan kasvun ansiosta keräämme ja käytämme tietoa yhä enenevissä määrin. Havainnoimaamme tietoa tallennetaan aktiivisesti tietokantoihin myöhempää analysointia varten (Xu & Wunsch, 2005). Myös kuluttajien vaatimukset esitettävälle tiedolle kasvavat havainnointi- ja tallennusmahdollisuuksien mukana. *Tiedon havaitseminen tietokannoista* (*Knowledge Discovery in Databases, KDD*) ja *tiedonlouhinta* (*Data Mining, DM*) ovat menetelmiä, joilla pyritään kuvailemaan ja havainnoimaan piilotettua, mutta merkityksellistä tietoa (Zhu & Davidson, 2007).

Tiedon havaitsemisella tietokannoista ja tiedonlouhinnalla tarkoitetaan usein samaa asiaa eikä termeille ole olemassa yksiselitteisiä määrittelyitä. Dunhamin (2002) mukaan tiedon havaitseminen voidaan määritellä prosessiksi, jossa etsitään hyödyllistä tietoa ja hahmoja datasta, kun taas tiedonlouhinta on vain yksi osa tiedon havaitsemista. Tällöin tiedonlouhinta voidaan edelleen jakaa *ennustaviin* (*predictive*) ja *kuvaileviin* (*descriptive*) menetelmiin. Ennustaviin menetelmiin kuuluvat muun muassa *luokittelu* (*classification*), *aikasarja-analyysi* (*time-series analysis*), *regressio* (*regression*) ja *ennustus* (*prediction*). Kuvaileviin menetelmiin voidaan laskea *ryhmittely* (*clustering*), *yhteenvetojen muodostus* (*summarization*) ja *sekvenssien haku* (*sequence discovery*). Kuva 1.1 havainnollistaa, kuinka KDD:n avulla pyritään havainnoimaan tietoa alkuperäisestä datasta.



Kuva 1.1: Tiedon havaitseminen tietokannoista (mukaillen lähdettä Fayyad & al., 1996)

Tämä pro gradu -tutkielma keskittyy aikasarjadatan tiedonlouhintaan ja erityisesti aikasarjojen ryhmittelyyn tiedonlouhintamenetelmänä. Selvyyden vuoksi määrittelen kuitenkin tiedon havaitsemisen tietokannoista ja tiedonlouhinnan Mörcheniä (2006) mukaillen seuraavasti:

DM: Tiedonlouhinta on varsinainen prosessi, jossa algoritmien avulla etsitään piilotettua tietoa tai rakenteita käsiteltävästä aineistosta.

KDD: Tiedon havaitseminen tietokannoista on tiedonlouhintaa, jonka tavoitteena on löytää lisätietoa aineistosta.

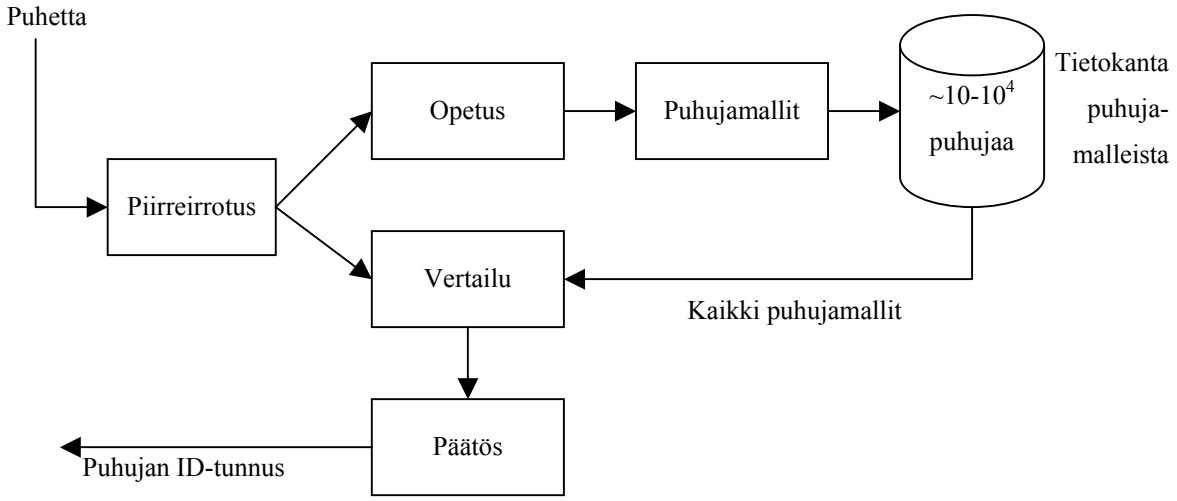
Mörchenin (2006) mukaan tiedonlouhinta koostuu useista alitehtävistä, kuten esikäsittelystä (preprocessing), ryhmittelystä ja luokittelusta. Näiden kolmen menetelmän lisäksi on olemassa myös monia muita tiedonlouhintatehtäviä. *Esikäsittelyn* tarkoitus on puhdistaa ja normalisoida käsiteltävä tieto. Mikäli esikäsittely jätetään suorittamatta, saattavat muiden menetelmien suorittamisesta saadut tulokset olla merkityksettömiä esimerkiksi datassa olevien virheellisten tai vaillinaisten tietoalkioiden takia (Mörchen, 2006). Ryhmittelyssä ja luokittelussa aineisto pyritään jakamaan pienempiin osiin ja tarkoituksena on selvittää aineiston rakennetta paremmin (Xu & Wunsch, 2005).

Esineiden ja asioiden ryhmittely luokkiin on ihmiselle luontainen toiminto. Luokittelemme asioita ryhmiin usein jopa sitä tiedostamattamme. Esimerkiksi kadulla kävellessämme luokittelemme ohikulkevat polkupyörät, henkilö- ja kuorma-autot kulkuvälineisiin, joita kannattaa väistää. Everettin (1993) mukaan luokittelu on jopa ollut osana mahdollistamassa kielten kehittymistä, sillä tarvitsemme sanoja, jotka kuvaavat erilaisia asioita – asioiden nimeäminen on luokittelua. Hartigan (1975) puolestaan käyttää uusien asteroidien löytämistä esimerkkinä ryhmittelystä: asteroideista tehdään havaintoja, jotka ovat ryhmiteltäviä alkioita. Jos havainnot muistuttavat tarpeeksi toisiaan, niin niiden voidaan olettaa olevan eri havaintoja samasta asteroidista. Tästä johtuen ryhmäksi voidaan laskea joukko eri havaintoja samasta asteroidista.

Ryhmittelyanalyysiä on käytetty useilla tieteenaloilla erilaisissa sovelluksissa. Arkeologiassa ryhmittelyalgoritmien avulla on luotu taksonomioita kaivauksista löytyneille esineille ja luille (Everitt, 1993). Biologiassa on ryhmittelyn avulla muodostettu eläin- ja kasvillisuustaksonomioita sekä ryhmitelty geenejä niiden samankaltaisuuden mukaan (Han & Kamber, 2001). Liikemaailmassa on usein tarpeellista löytää kohderyhmiä, joiden etsimiseen ja analysointiin on sovellettu ryhmittelyä. Tietojenkäsittelytieteen puolella ryhmittelyä käytetään esimerkiksi hahmon- ja puhujantunnistukseen sekä tekstidokumenttien ryhmittelyyn (Späth, 1980; Montalvo, 2007). Laaja levinneisyys tieteenalojen välillä kuvastaa

hyvin ryhmittelyn tärkeää merkitystä tiedon analysointikeinona. (Xu & Wunsch, 2005; Han & Kamber, 2001; Everitt, 1993)

Automaattinen puheentunnistus on tietojenkäsittelytieteen puolella merkittävä osa-alue. Aihe on haastava, sillä se vaatii laajaa asiantuntemusta muun muassa signaalinprosessoinnin, hahmontunnistuksen, kielitieteen, fysiologian ja tietojenkäsittelytieteen aloilta (Rabiner, 1993). Puheentunnistusta hyödyntäviä järjestelmiä on visioitu myös monissa sci-fi –elokuvissa, joista esimerkkinä Vuori (2002) ja Rabiner (1993) mainitsevat Avaruusseikkailu 2001 -elokuvan HAL-tietokoneen. Puheentunnistus voidaan jakaa puheen merkityksen tunnistukseen tai puhujantunnistukseen. Puheen merkityksen tunnistuksessa pyritään luomaan järjestelmä, jossa tietokone ymmärtää puhetta ja mahdollisesti osaa kommunikoida ihmisten kanssa puhumalla. Puhujantunnistuksessa on tarkoitus käyttää henkilön puhetta yksilöllisenä biometrisenä tunnisteena, ja tehtävä voidaan edelleen jakaa puhujantunnistukseen (*speaker identification, SI*) tai puhujanvarmistukseen (*speaker verification, SV*; Kinnunen & al., 2006). Puhujantunnistuksessa pyritään selvittämään kuka puhuja X on annetusta puhujien joukosta. Puhujanvarmistuksessa tarkoitus on vain varmistaa, että puhuja on se henkilö, joka hän väittää olevansa. Kuva 1.2 esittää tyypillistä suljetun puhujajoukon puhujantunnistusjärjestelmää. Kattavan yhteenvedon puheentunnistuksesta suomeksi voi lukea lähteestä Vuori (2001). Rabiner (1993) puolestaan käsittelee puheentunnistuksessa perinteisesti käytettyjä menetelmiä.



Kuva 1.2: Puhujantunnistukseen käytettävän järjestelmän perusrakenne (mukaillen lähdettä Kinnunen & al., 2006).

Tämän tutkielman pääpaino on aikasarjojen ryhmittelyssä ja erilaisten aikasarjaryhmien edustajien vertailussa. Aikasarjojen keskinäiseen vertailuun käytetään dynaamista aikasovitusta. Luvussa 2 käydään läpi, mitä aikasarjat ovat ja kuinka ne poikkeavat klassisesta, staattisesta datasta. Luvussa 3 perehdytään ryhmittelyyn: esitellään perusteet, määritetään ryhmittely kombinatorisena optimointiongelmana sekä luodaan katsaus ryhmittelyalgoritmien luokitteluun. Luvussa 4 keskitytään aikasarjojen ryhmittelyyn, siihen liittyviin ongelmiin ja toteutettuihin ryhmittelyalgoritmeihin, joilla testausaineistoja ryhmitellään. Luvussa 5 esitellään testausmenetelmät ja testaukseen käytetyt aineistot, jotka koostuvat puheesta, käsinkirjoitetuista merkeistä, viittomakielen merkeistä ja synteettisestä datasta. Saadut testaustulokset ja niistä tehtyjä havaintoja käydään läpi luvussa 6. Luvussa 7 luodaan yhteenveto tämän tutkielman sisällöstä.

2 AIKASARJAT

Aikasarja on aikasidonnaista tietoa jostakin tapahtumasarjasta, joka muuttuu ajan kuluessa (Zhu, 2004). Tyypillinen aikasarja koostuu joukosta havaintoja, jotka on tehty jostakin muuttujasta tasaisin välein (intervallein) ajan kuluessa (Harvey, 1993). Viime vuosikymmenen aikana mielenkiinto aikasarjojen tiedonlouhintaan on kasvanut valtavasti (Keogh & Kasetty, 2002). Ajan huomioon ottavat tietokannat voidaan jakaa kahteen luokkaan: *aikatietokantoihin* (*temporal databases*) ja *aikasarjatietokantoihin* (*time-series databases*). Aikatietokantoihin talletetaan aikasidonnaista tietoa ja tietoa järjestellään aikaleimojen avulla. Pankkien tilitietokannat ovat perinteinen esimerkki aikatietokannoista. Aikasarjatietokannat puolestaan koostuvat ajan suhteen järjestetyistä sarjoista havaintoarvoja. (Han & Kamber, 2001)

2.1 Johdatus aikasarjoihin

Aikasarja-analyysi on tiedonlouhinnan ennustaviin menetelmiin kuuluva tutkimushaara, joka pyrkii analysoimaan aikasarjojen piirteitä ja ennustamaan aikasarjoissa tapahtuvia muutoksia. Aikasarjojen tutkimisessa on perinteisesti käytetty kahta näkökantaa: analysointia ja mallintamista (Dunham, 2002). Aikasarjojen analysoinnin tarkoituksena on tehdä yhteenveto aikasarjan ominaisuuksista ja luonnehtia sen olennaiset piirteet. Analysointia voidaan suorittaa aika- tai taajuusavaruudessa. Nämä kaksi analysoinnin pääsuuntaa täydentävät toisiaan. Ajan suhteen analysoitaessa pyritään löytämään yhteyksiä eri havaintojen välillä, kun taas taajuusanalyysissä keskitytään tunnistamaan toistuvia, eli syklisiä, rakenteita aikasarjoista. Muodostettaessa malleja aikasarjoista halutaan usein ennustaa aikasarjassa tapahtuvia muutoksia sen perusteella, mitä aikasarjasta tähän asti tiedetään. Yleisesti käytetty menetelmä pyrkiä ennustamaan muutoksia on luodun mallin ekstrapoloiminen. (Harvey, 1993)

Aikasarjadata on yleistä yritysten tietokannoissa (Dunham, 2002). Tilien tapahtumista pidetään kirjaa aikaleimojen avulla: tapahtumien suoritusajat ja -päivämäärät yksilöivät kunkin tapahtuman. Tämä data ei ole tasaisin väliajoin tehtyjä havaintoja, vaan muutoksen tapahtuessa aikasarjaa päivitetään ja se pitenee. Myöhemmin esiteltävät, tässä pro gradu -

tutkielmassa käytetyt aineistot ja niiden aikasarjat puolestaan ovat kaikki tasaisin väliajoin mitattuja havaintoja. Esimerkkiaineistot sisältävät puhetta, viittomakieltä, käsinkirjoitettuja merkkejä sekä synteettistä aikasarjadataa. Tässä tutkielmassa ei keskitytä varsinaiseen aikasarja-analyysiin, vaan aikasarjoista etsitään rakenteita ryhmittelyanalyysin avulla.

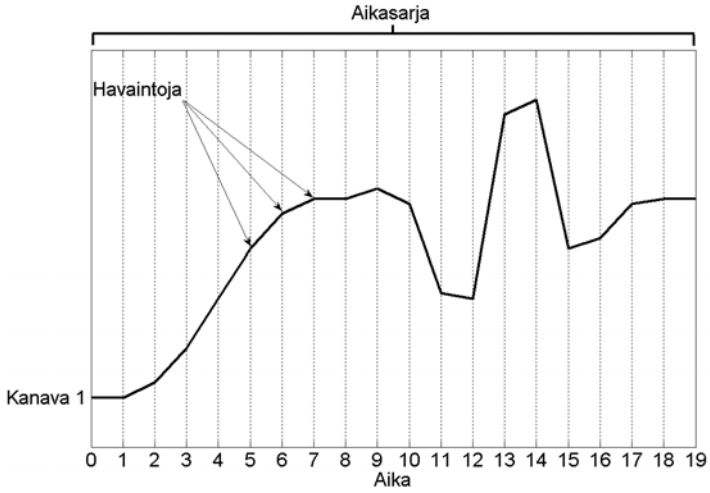
2.2 Määrittely

Määrittelen tutkielmaa varten aikasarjat ja aikasarjoihin liittyvät peruskäsitteet seuraavasti (mukaillen Mörcheniä, 2006). *Otos* (*frame*, *time point*) on yksittäinen ja uniikki havaintopiste ajassa. Kuhunkin otokseen liittyy vähintään yksi *havainto* (*observation*) tai *ominaisuus* (*feature*/*attribute/variable*) aikasarjaan liittyvästä ilmiöstä. Nämä ominaisuudet voidaan kuvata symbolisina tai numeraalisina, minkä seurauksena aikasarjat voidaan jakaa *symbolisiin* ja *numeraalisiin* aikasarjoihin. Numeraaliset aikasarjat voidaan jakaa edelleen hienompiin luokkiin, esimerkiksi sen mukaan, onko niille luonnollista määritellä laskutoimituksia. *Aikasarja* (*time series/sequence*) on ajan suhteen järjestetty joukko otoksia, joista kuhunkin liittyy vähintään yksi havaintoarvo tai -objekti. Otoksien lukumäärä ilmaisee aikasarjan pituuden. *Kanava* (*channel*) on yhden ominaisuuden järjestetty joukko havaintoarvoja ajan suhteen (Kadous, 2002). Täten voidaan sanoa, että mitattujen kanavien tai otoksessa tehtyjen havaintojen lukumäärä ilmaisee aikasarjan *ulotteisuuden* (*dimensionality*).

Taulukossa 2.1 on esimerkki yksiulotteisesta (yksikanavaisesta) aikasarjasta. Aikasarja on ilmaistu numeraalisina arvoina ja sen pituus on 20. Havaintoaika kasvaa tasaisin yhden sekunnin intervallein ensimmäisellä rivillä ja kanavasta yksi mitatut, numeraaliset havaintoarvot ovat toisella rivillä. Kuvassa 2.1 on esitetty graafisesti sama aikasarja. Pystysuorassa menevät katkoviivat kuvaavat kukin yhtä otosta ja kuvaaja esittää aikasarjaa, joka muodostuu kanavasta tehdyistä havainnoista.

Aika	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Kanava 1	1,0	1,0	1,2	1,5	2,0	2,5	2,9	3,0	3,0	3,1	3,0	2,1	2,0	3,9	4,0	2,5	2,6	3,0	3,0	3,0

Taulukko 2.1: Esimerkki yksiulotteisesta aikasarjasta.



Kuva 2.1: Esimerkki yksiulotteisen aikasarjan kuvaajasta.

2.3 Sovellusalueita

Ajan suhteen järjestettyä dataa on saatavilla valtavasti ja viime vuosikymmenten aikana tiedon määrä on kasvanut räjähdysmäisesti (Han & Kamber, 2001). Aineistoa voidaan kerätä esimerkiksi DNA-rihmoista, puheentunnistuksesta, lääketieteestä, pörssikursseista ja tekstinlouhinnasta (Xu & Wunsch, 2005). Aiemmin tuntemattomista, luokkanimikkeitä sisältämättömistä aikasarja-aineistoista halutaan usein etsiä samankaltaisten aikasarjojen ryhmiä. Suurin osa ryhmittelyä käsittelevistä tutkimuksista kuitenkin keskittyy staattiseen dataan (Liao, 2005). Tiedonlouhintatekniikoita voidaan soveltaa löytämään olennaisia piirteitä aikasarjoista. Yleinen esimerkki aikasarjoihin liittyvästä tiedonlouhinnasta on pyrkimys ennustaa pörssikursseissa tapahtuvia muutoksia. Tiedonlouhinnan tuloksia voidaan varsinaisen louhinnan jälkeen arvioida ja mahdollisesti käyttää päätöksenteon sekä strategisen suunnittelun tukena (Han & Kamber, 2001). Keogh ja Kasetty (2002) listaavat yleisiä tutkimuskohteita aikasarjojen tiedonlouhinnassa:

- *Indeksointi* (kysely, Query by Content): etsi lähin vastaavuus kyselyaikasarjalle aikasarjatietokannasta annetun etäisyysmitan mukaan.
- *Ryhmittely*: etsi annetun etäisyysmitan avulla luonnollisia ryhmiä tietokannassa olevista aikasarjoista.
- *Luokittelu*: luokittele tuntematon aikasarja johonkin kahdesta tai useammasta annetusta luokasta.
- *Segmentointi*: tavoitteena on luoda aikasarjalle karkeampi malli, joka silti muistuttaa alkuperäistä aikasarjaa.

3 RYHMITTELY

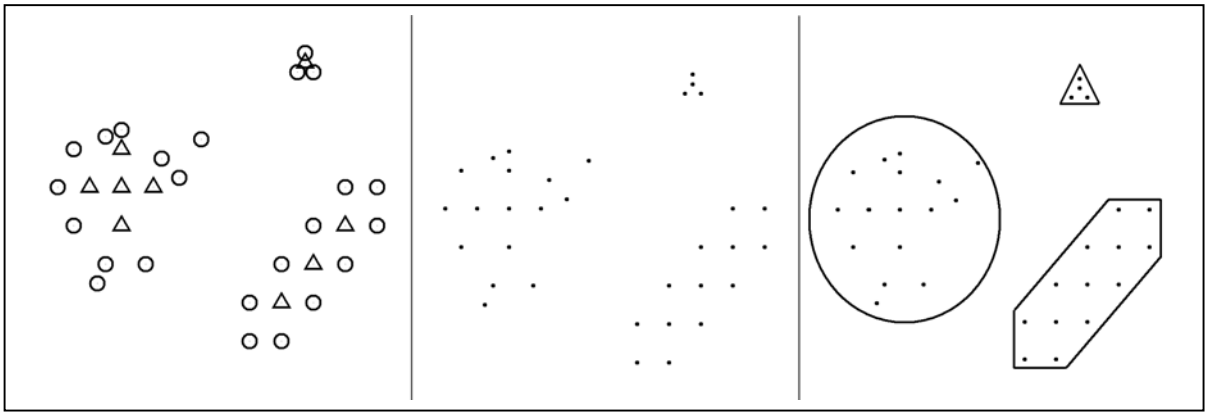
Tutkiessaan aiemmin tuntematonta ilmiötä tai opetellessaan uusia asioita ihminen pyrkii suhteuttamaan käsiteltävää aihetta aiemmin opittuun tietoon. Tämä suhteuttaminen tapahtuu tarkastelemalla tuntemattomien ja tunnettujen ilmiöiden tai asioiden samankaltaisuutta. Ilmiöiden samankaltaisuuden tai eroavaisuuden määrittely perustuu joihinkin opittuihin sääntöihin tai standardeihin. Asioiden *läheisyys* (*proximity*) kuvaa yleisesti sitä, kuinka samankaltaisia käsiteltävät asiat ovat. (Xu & Wunsch, 2005)

Tiedonlouhintamenetelmiä käytetään kuvaamaan käsiteltävää aineistoa paremmin. Aineistojen laajuuden ja moniulotteisuuden takia ihmisen on usein mahdotonta hahmottaa siitä selkeitä rakenteita. Tämän vuoksi tarvitaan tiedonlouhintamenetelmiä, joilla pyritään antamaan aineistolle rakenne, löytämään ryhmiä ja mahdollisesti muodostamaan ryhmistä luokkia. Kaikki löydetyt hahmot aineistossa eivät välttämättä ole käyttäjää kiinnostavia tai hyödyllisiä. Han ja Kamber (2001) toteavat että hahmo on kiinnostava, jos se on helposti ymmärrettävä, todellinen, hyödyllinen ja uudenlainen. Kiinnostavan hahmon voidaan todeta esittävän merkityksellistä lisätietoa aineistosta.

Ryhmittely muistuttaa paljon luokittelua, joka on toinen yleinen tiedonlouhinnassa käytetty menetelmä. Ero luokittelun ja ryhmittelyn välillä on siinä, että luokittelussa dataobjekteja luokitellaan valmiiksi määriteltyihin luokkiin. Ryhmiteltäessä puolestaan valmiita luokkatunnisteita ei ole olemassa, vaan dataobjektit on ryhmiteltävä niille luontaisiin ryhmiin objektien ominaisuuksien mukaan. Ryhmittelyä voidaan käyttää luomaan luokittelun tarvitsemat luokkanimikkeet (Han & Kamber, 2001). *Luokittelun* sanotaan olevan *ohjattua oppimista* (*supervised learning/classification*), kun taas ryhmittely on *ohjaamatonta oppimista* (*unsupervised learning/classification*; Jain & al., 1999).

Kuva 3.1 havainnollistaa luokittelun ja ryhmittelyn välisiä eroja. Vasemmalla olevassa kuvassa on luokittelu tehtynä kahteen luokkaan: kolmiolla merkattuihin keskusalkioihin ja ympyrällä merkattuihin reuna-alkioihin. Keskellä olevassa kuvassa on sama aineisto ilman luokkanimikkeitä ’keskusalkio’ ja ’reuna-alkio’. Oikealla on kuvattu yksi mahdollinen ositus aineistolle. Huomattavaa on, että luokittelu ja ryhmittely antavat erilaisen rakenteen aineistolle tässä tapauksessa. Esimerkissä luokitellaan alkiot reuna- ja keskusalkioihin niiden

sijainnin mukaan omassa ryhmässään. Ryhmittelyn avulla luodussa osituksessa on puolestaan muodostettu ryhmiä alkioista niiden euklidisen etäisyyden mukaan, jolloin tuloksena saadaan kolmen ryhmän ositus.



Kuva 3.1: Luokittelu, ryhmittelytehtävä ja ryhmittely (mukaillen Kärkkäistä, 2006).

Xu ja Wunsch (2005) määrittelevät luokittelun ja ryhmittelyn seuraavasti:

> **Luokittelu**: Ohjatussa luokittelussa pyritään kuvaamaan aineiston ***X*** datavektorit äärelliseen joukkoon eroteltuja luokkia.
>
> **Ryhmittely**: Ohjaamattomassa luokittelussa, joka tunnetaan myös tutkivana data-analyysinä tai ryhmittelynä, ei ole saatavilla luokittelun kaltaisia nimikkeitä eri ryhmille. Ryhmittelyssä jaetaan aineiston ***X*** datavektorit toisistaan erillisiin ryhmiin, joita on äärellinen määrä. Ryhmittelyn tarkoituksena on selvittää, onko aineistolla olemassa selkeitä ja luonnollisia rakenteita.

Edellisestä määrittelystä herää välittömästi kysymys, mitä tarkoitetaan ’luonnollisella rakenteella’. Usein aineistoille voidaan määritellä erilaisia rakenteita, jotka käyttötarkoituksesta riippuen saattavat kaikki olla luonnollisia. Fielding (2006) käyttää tästä esimerkkinä satunnaisesti järjestetyn korttipakan ryhmittelyä, jolloin luonnollisia ryhmityksiä ovat ositukset:

- maiden mukaan (neljä ryhmää),
- arvojen mukaan (kolmetoista ryhmää),
- parillisuuden, parittomuuden ja kuvakorttien mukaan (kolme ryhmää),
- kuva- ja numerokorttien mukaan (kaksi ryhmää),
- mustien ja punaisten korttien mukaan (kaksi ryhmää).

Osa korteista muodostuneista rakenteista vaikuttaa paremmilta kuin toiset; mahdollisesti johtuen siitä, että miellämme parhaiksi ryhmittelyiksi korttipeleissä hyödylliset ositukset. Todellisuudessa jokainen kuvattu tapa muodostaa luonnollisia osituksia korttipakalle, mutta ositukset poikkeavat käytetyn etäisyysmitan osalta. Etäisyysmitta vaikuttaa siis merkittävästi löydettyyn rakenteeseen ja samalla estää muita rakenteita löytymästä. Fielding (2006) jatkaa varoittamalla olettamasta liikaa aineiston rakenteesta, sillä ryhmittelyalgoritmit eivät osaa arvioida osituksen mielekkyyttä – ne vain keskittyvät optimoitavan tavoitefunktion minimoimiseen etäisyysmitan avulla.

Ryhmittely on erittäin käytetty menetelmä monilla tieteenaloilla. Laaja levinneisyys eri tieteenalojen parissa kuitenkin myös hankaloittaa yleisten käsitteiden ja menetelmien syntymistä, sillä eri aloilla saattavat vallita erilaiset sanastot, käsitteet ja suhtautumistavat ryhmittelyyn (Jain et al., 1999). Hyvän yleiskatsauksen ryhmittelyn historiaan, yleisiin menetelmiin ja sovellusalueisiin antavat esimerkiksi Han & Kamber (2001), Jain & al. (1999) ja Xu & Wunsch (2005).

Tyypillinen ryhmittelytehtävä käsittää seuraavat osaongelmat (Jain & Dubes, 1988):

1) Aineiston alkioiden kuvaaminen ryhmiteltävinä tietorakenteina
2) Alkioiden läheisyysmitan määrittäminen
3) Ryhmittely
4) Ryhmittelyn tuloksen esittäminen
5) Ryhmittelyn onnistuneisuuden arvioiminen

3.1 Peruskäsitteet

Ryhmittelyn tai *klusteroinnin* (*clustering*) tavoitteena on selvittää aiemmin tuntemattomasta aineistosta rakenne, jonka avulla aineisto voidaan jakaa *ryhmiin*. Ryhmiä voidaan kutsua myös klustereiksi, joukoiksi tai kategorioiksi (Xu & Wunsch, 2005). Ryhmiä muodostettaessa pyritään niiden sisäinen samankaltaisuus maksimoimaan ja ryhmien välinen samankaltaisuus minimoimaan. (Liao, 2005; Jain & al., 1999; Han & Kamber, 2001). Ryhmittelystä ei ole olemassa yleisesti hyväksyttyä määritelmää. Selkeän määritelmän puuttuessa myös ryhmittelymenetelmien luokittelu on vaikeaa. Olennaista käytetystä ryhmittelymenetelmästä

riippumatta on kuitenkin määritellä etäisyysmitta ja optimoitava tavoitefunktio. Tämän jälkeen voidaan johtaa algoritmi, joka pyrkii löytämään optimaalisen osituksen aineistolle (Xu & Wunsch, 2005; Liao, 2005).

Cormen & al. (2001) määrittelevät *algoritmin* järjestetyksi sarjaksi laskennallisia askelia, minkä avulla suoritetaan hyvin määritelty laskennallinen ongelma. Toisin sanoen algoritmi kuvaa tarkasti laskennallisen proseduurin, jolla ongelman syötteistä saadaan muodostettua ratkaisu. Ryhmittelyn tapauksessa syötteenä ryhmittelyalgoritmille annetaan ryhmiteltävä aineisto sekä mahdollisesti parametreja, kuten haluttu ryhmien lukumäärä. Ryhmittelyalgoritmi muodostaa annettujen syötteiden perusteella aineistolle osituksen. Saatujen ositusten onnistuneisuutta täytyy pystyä arvioimaan käytetystä ryhmittelyalgoritmista riippumatta ja tätä varten pitää ryhmittelyongelmalle määrittää myös optimoitava tavoitefunktio. Mitä parempi ryhmittelyalgoritmi on, sitä paremmin se optimoi tavoitefunktiota.

Ryhmittelyalgoritmin, optimoitavan tavoitefunktion ja käytetyn etäisyysmitan lisäksi tulee päättää sallitaanko alkion kuuluvan vain yhteen vai useampaan ryhmään. Mikäli alkio voi kuulua vain yhteen ryhmään puhutaan *kovasta ryhmittelystä* (*hard clustering*). Jos alkion sallitaan kuuluvan useampaan ryhmään, käytetään termiä *sumea ryhmittely* (*fuzzy clusting*). (Xu & Wunsch, 2005.) Alkion ryhmään kuulumisen aste määritellään sumeassa ryhmittelyssä usein suhdeluvuilla u_{ij} väliltä [0..1] siten, että kuhunkin tietoalkioon liittyvien suhdelukujen summa on 1 (Kärkkäinen, 2006). Tässä tutkielmassa keskitytään jatkossa pelkästään kovaan ryhmittelyyn.

Olkoon $\boldsymbol{X}$ ryhmiteltävä aineisto ja $\boldsymbol{X}_j$, j = 1, …, M sen jako M:ään ryhmään. Tällöin kovalta ryhmittelyltä vaaditaan seuraavat kolme ominaisuutta. (Xu & Wunsch, 2005).

1) $\boldsymbol{X}_j \neq \emptyset$; j = 1, …, M.
2) $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \ldots \cup \boldsymbol{X}_M = \boldsymbol{X}$.
3) $\boldsymbol{X}_i \cap \boldsymbol{X}_j = \emptyset$; i, j = 1, …, M ja $i \neq j$.

Toisin sanoen mikään ryhmä $\boldsymbol{X}_j$ ei saa olla tyhjä, ryhmien yhdisteiden tulee kattaa koko aineisto $\boldsymbol{X}$ ja missään kahden ryhmän parissa ei saa olla alkioita, jotka kuuluvat molempiin.

Ryhmiteltäessä N tietoalkioita yritetään etsitä optimaalinen M-ositus $\boldsymbol{P_O}$, joka minimoi käytetyn tavoitefunktion ja sisältää M ryhmää. Intuitiivinen ratkaisu optimaalisen osituksen $\boldsymbol{P_O}$ löytämiseksi on raakaan laskentaan pohjaava *brute force* -menetelmä, jossa muodostetaan kaikki ositukset ja valitaan se, joka minimoi tavoitefunktion arvon M:lle ryhmälle. Abramowitzin ja Stegunin (1965) mukaan *toisen lajin Stirlingin luku* (*Stirling Number of the Second Kind*) kertoo, kuinka monta erilaista M-ositusta voidaan tehdä aineistosta, jossa on yhteensä N alkiota. Tämä luku saadaan laskettua kaavalla

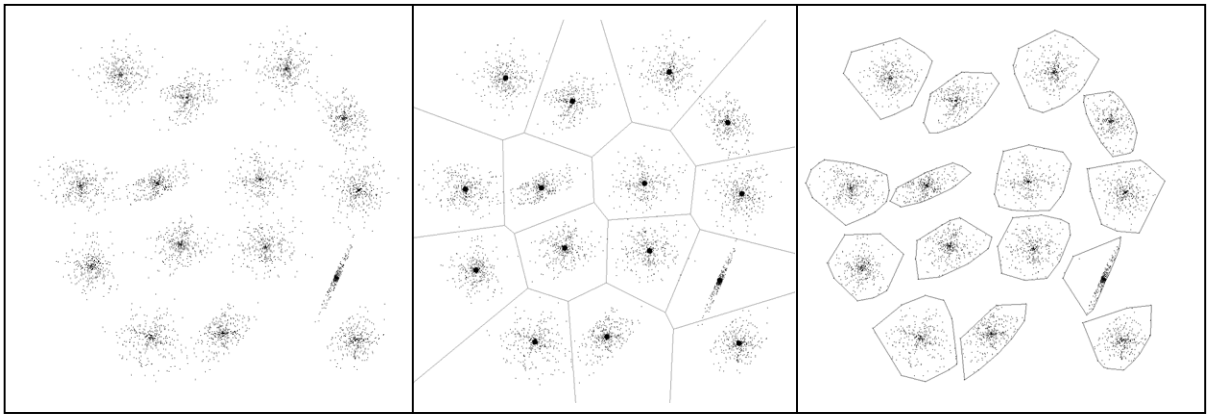
$$\left\{ {N \atop M} \right\} = \frac{1}{M!}\sum_{i=0}^{M} (-1)^{M-i} \binom{M}{i} i^N \tag{3.1}$$

, missä $\binom{M}{i} = \frac{M!}{i!(n-k)!}$ on binomikerroin.

Esimerkiksi todella pienelle 100:n tietoalkion aineistolle on mahdollisuus muodostaa 10:n ryhmän osituksia noin $2{,}755 \cdot 10^{93}$:llä eri tavalla. Vaikka ositukset ja tavoitefunktion tulokset niille saataisiin valmiina, niin parhaan tuloksen antavan osituksen etsiminen kestäisi miljardi ositusta sekunnissa tarkistavalta tietokoneelta yli $8 \cdot 10^{76}$ vuotta. Kun otetaan huomioon vielä, että aineisto oli todella suppea, voidaan todeta, ettei kaikkien ositusten muodostaminen ole mitenkään perusteltua, vaan tarvitaan toisenlainen menetelmä hyvän osituksen etsimiseen. Tästä johtuen ryhmittelyä varten on kehitetty heuristisia algoritmeja, joiden tavoitteena on löytää mahdollisimman hyvä ositus aineistolle (Xu & Wunsch, 2005). Osituksen globaalia optimaalisuutta ei kuitenkaan voida varmistaa (Fränti & Kivijärvi, 2000). Ryhmittely on kombinatorisessa muodossaan NP-täydellinen ongelma (Garey & al., 1982).

Saatuja ryhmittelyjä eli osituksia voidaan kuvata erilaisilla tavoilla. Ryhmittelyongelmasta on hankala luoda kuvaa, mikäli ryhmiteltävä aineisto koostuu useammasta kuin kolmesta ulottuvuudesta. Ihmisen hahmottamiskyky pohjautuu pitkälti visuaaliseen havainnointiin, ja tämän vuoksi ositusten kuvaaminen on tärkeää (Aurenhammer, 1991). Kuvassa 3.2 on esitelty kolme erilaista tapaa kuvata osituksia: keskipistemalli, Voronoi-diagrammit ja konveksit peitteet. Keskipistemallissa jokaisen ryhmän keskusta on merkitty isommalla pisteellä kuin varsinaiset aineiston alkiot. Voronoi-diagrammit muodostetaan vierekkäisten ryhmien keskipisteiden avulla siten, että keskipisteiden väliin piirretään viivoja, jotka ovat yhtä kaukana molemmista pisteistä. Viivojen leikkauskohtiin muodostuu pisteitä, jotka ovat yhtä kaukana vähintään kolmen ryhmän keskipisteistä. Yhdistelemällä kuvio voidaan luoda

rakenne, jossa yhden alueen sisällä ovat kaikki samaan ryhmään kuuluvat tietoalkiot. Voronoi-diagrammit on kuvannut tarkasti Aurenhammer (1991). Viimeinen kuvan 3.2 osio havainnollistaa ryhmittelyn kuvaamista konveksien peitteiden avulla. Konveksit peitteet muodostetaan ryhmille siten, että jokainen samaan ryhmään kuuluva tietoalkio kuuluu samaan alueeseen.



Kuva 3.2: Ryhmiteltävä aineisto (vasen); ositus esitettynä keskipisteillä ja Voronoi-diagrammeilla (keskellä); sama ositus esitettynä konveksien peitteiden avulla (oikea).

Han ja Kamber (2001) toteavat ryhmittelyn olevan ”*haastava tutkimusalue, jossa potentiaaliset sovellusalueet asettavat erityisvaatimuksia käytetyille ryhmittelymenetelmille*”. He luettelevat yleisiä ryhmittelymenetelmille asetettavia vaatimuksia seuraavasti:

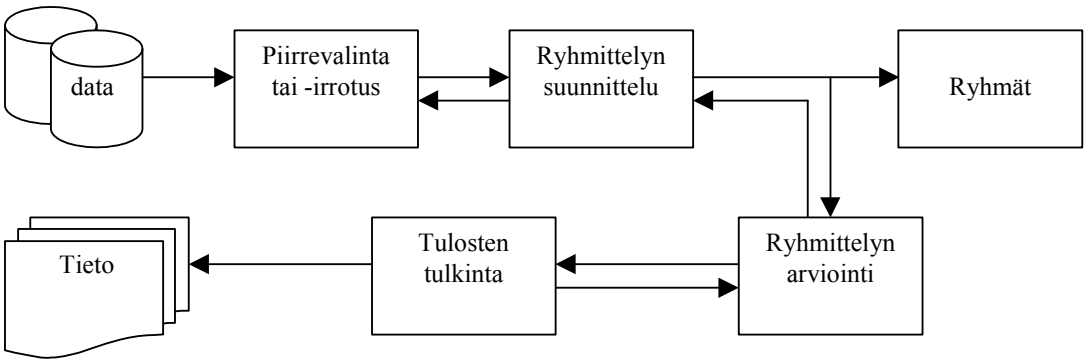
1) Skaalattavuus: ryhmittelymenetelmien tulisi toimia hyvin myös isoilla aineistoilla.
2) Kyky käsitellä eri tyyppisiä attribuutteja: sovellusalueesta riippuen ryhmittelymenetelmien saattaa pitää pystyä käsittelemään numeerisen datan lisäksi binääristä, kategorista ja järjestettyä dataa tai näiden yhdistelmiä.
3) Kyky löytää eri muotoisia ryhmiä: aineistossa oleva ryhmä voi, ja usein poikkeaa, ympyrämäisestä ryhmästä. Ryhmittelymenetelmän tulisi löytää myös satunnaisen muotoiset ryhmät.
4) Vähäiset vaatimukset ennakkotiedoille aineistosta: monet ryhmittelymenetelmät vaativat käyttäjältä ennakkotietoa aineistosta, kuten esimerkiksi ryhmien lukumäärän. Käyttäjän avustamiseksi ja ryhmittelyn laadunvalvonnan parantamiseksi tarvitaan menetelmiä, jotka osaavat tunnistaa ryhmien lukumäärän.
5) Kyky käsitellä kohinaa datassa: todelliset tietokannat sisältävät *poikkeavia havaintoja* (*outliers*), puuttuvia tietokenttiä ja virheellistä dataa.

Ryhmittelymenetelmien tulisi olla kykeneviä käsittelemään tai hylkäämään kohinaa ja virheitä datassa.

6) Riippumattomuus aineiston lajittelujärjestyksestä: järjestys, johon aineisto on lajiteltu, ei saa vaikuttaa ryhmittelymenetelmien antamiin tuloksiin.
7) Kyky käsitellä moniulotteista dataa: tietokannoissa oleva data on usein moniulotteista ja ryhmittelymenetelmien tulisi pystyä ryhmittelemään data hyvin ulottuvuuksien määrästä riippumatta. Tämän todentaminen voi olla hankalaa, sillä ihmisten silmät ovat hyviä havaitsemaan enintään kolmen ulottuvuuden ryhmittymiä.
8) Tuki ryhmittelylle asettaville rajoitteille: ryhmittelylle voidaan asettaa rajoitteita, jotka estävät alkioiden kuulumisen tiettyihin ryhmiin. Esimerkkinä Han & Kamber (2001) käyttävät pankkiautomaattien optimaalisen ryhmittelyn etsimistä kaupungissa. Rajoitteina toimivat tällöin esteet kuten kiertoliittymät, maanmuodot ja joet.
9) Tulkittavuus ja käytettävyys: saadun ryhmittelyn tulee olla käyttäjälle hyödyllinen, tulkittava ja kattava. Oleellista on tutkia vaikuttaako sovellusalueen tavoite ryhmittelymenetelmän valintaan.

On tärkeää huomata, että vaikka on suotavaa pyrkiä tavoittelemaan kaikkia yllä lueteltuja vaatimuksia suunniteltaessa ryhmittelyä, niin se on harvoin, jos koskaan, mahdollista. Mielestäni tärkeää ryhmittelymenetelmää valitessa tai suunnitellessa on tiedostaa se, mitä erityisvaatimuksia ryhmittelyprosessille asetetaan. Suoritusnopeuden ja -tarkkuuden suhteen on tehtävä usein kompromisseja.

Kuva 3.3 esittää ryhmittelyprosessia vuokaaviona. Ryhmittely aloitetaan valitsemalla aineistosta olennaiset piirteet tai ominaisuudet. Seuraavaksi suunnitellaan ryhmittelymenetelmä, joka koostuu ryhmittelyalgoritmin, etäisyysmitan ja ryhmittelykriteerin määrittämisestä. Ryhmittelyn tuloksena saadaan ositus, joka kuvaa ryhmät. Varsinaisen laskennan jälkeen ryhmittelyn mielekkyys tulee arvioida ja tulokset tulkita, mikäli ryhmittely todetaan mielekkääksi. Tulkituista tuloksista puolestaan saadaan lisätietoa aineiston rakenteesta. Paluuaskeleet mahdollistavat aiemmin käytettyjen menetelmien muokkaamisen, mikäli se koetaan tarpeelliseksi.



Kuva 3.3: Ryhmittelyprosessi kuvattuna vuokaaviona. (Xu & Wunsch, 2005)

3.2 Ryhmittelyongelman määrittely

Tässä tutkielmassa määrittelen ryhmittelyn kombinatorisena optimointiongelmana mukaillen lähteitä Fränti ja Kivijärvi (2000) sekä Fränti & al. (2006):

N Aineiston koko, tietoalkioiden lukumäärä.
M Ryhmien lukumäärä.
K Attribuuttien lukumäärä.
$\boldsymbol{X}$ Aineisto, jonka koko on N, $\boldsymbol{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.
$\boldsymbol{P}$ Ositus, joka liittää tietoalkiot $\mathbf{x}_i$ ryhmiin $\mathbf{c}_j$, $\boldsymbol{P} = \{p_1, p_2, \ldots, p_N\}$.
$\boldsymbol{C}$ Ryhmien edustajien joukko, jonka koko on M, $\boldsymbol{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M\}$.

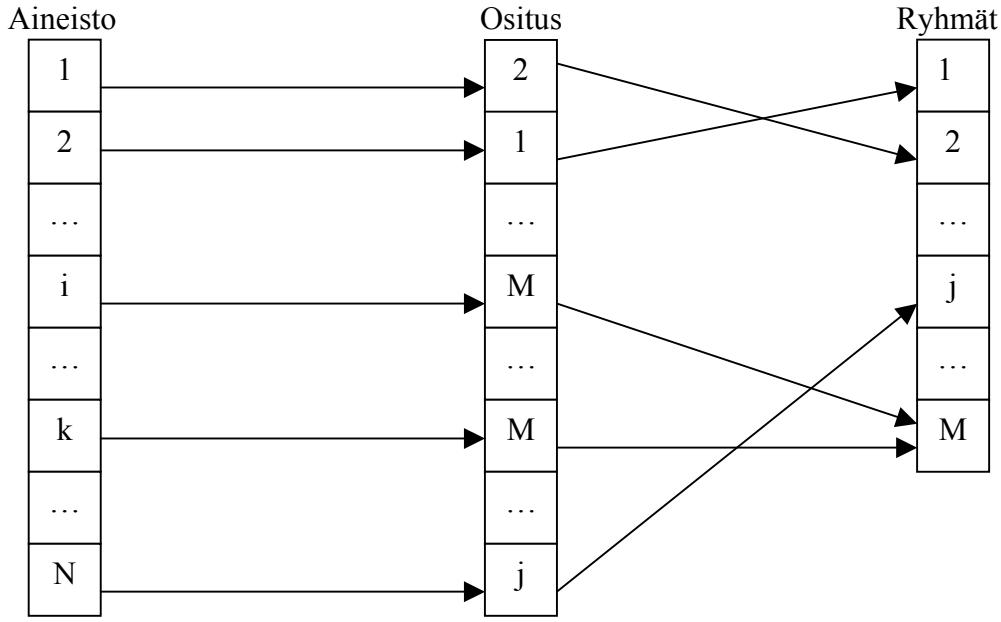
Tehtävänä on osittaa syötteenä annettu N:n kokoinen aineisto $\boldsymbol{X} = \{\mathbf{x}_i \mid 1 \leq i \leq N\}$ M:ään määrään ryhmiä siten, että samankaltaiset objektit ryhmitellään samaan ryhmään ja erilaiset piirteet omaavat objektit ryhmitellään muihin ryhmiin. Ositus $\boldsymbol{P} = \{p_i \mid 1 \leq i \leq N\}$ määrittelee ryhmittelyn osoittamalla jokaiselle tietoalkiolle $\mathbf{x}_i$ ryhmäindeksin p_i, jonka arvo kertoo, mihin ryhmään tietoalkio $\mathbf{x}_i$ on sijoitettu. Mikäli alkio $\mathbf{x}_i$ kuuluu ryhmään $\boldsymbol{X}_j$, niin $p_i = j$. Ryhmän $\boldsymbol{X}_j$ edustaja $\mathbf{c}_j$ on keskipistemallin mukaisesti keskiarvovektori siihen ryhmään kuuluvista alkioista ja se lasketaan kaavan (3.3) avulla (Fränti & Kivijärvi, 2000). Ryhmään $\boldsymbol{X}_j$ kuuluu joukko alkioita siten, että $\boldsymbol{X}_j = \{\mathbf{x}_i \mid p_i = j\}$. Kullekin alkiolle $\mathbf{x}_i$ lasketaan lähin ryhmä, eli ryhmäindeksi p_i, kaavan (3.4) avulla. (Fränti & al., 2006; Kärkkäinen, 2006)

$$p_i = j \Leftrightarrow x_i \in X_j \tag{3.2}$$

$$c_j = \frac{\sum_{\mathrm{p}(i)=j} x_i}{\sum_{\mathrm{p}(i)=j} 1} \quad \forall j \in [1..M] \tag{3.3}$$

$$p_i = argmin_{j=1}^{M} D(x_i, c_j) \quad \forall i \in [1..N] \tag{3.4}$$

Kuva 3.4 selventää aineiston, osituksen ja ryhmien keskinäistä suhdetta. Jokaista aineiston alkiota i vastaa ositustaulussa solu i, joka yksilöi ryhmän, johon tietoalkio kuuluu. Esimerkiksi ensimmäiseen tietoalkioon liittyvässä ositustaulun ensimmäisessä solussa on ryhmän 2 numero, joten alkio 1 kuuluu ryhmään 2. Kukin alkio voi liittyä täten vain yhteen ryhmään. Lisäksi jokaiseen ryhmään on liityttävä vähintään yksi alkio kovan ryhmittelyn ensimmäisen säännön mukaisesti.



Kuva 3.4: Aineiston ryhmittely toteutettuna ositustaulukon avulla.

Xu & Wunsch (2005) kuvaavat tietoalkiota seuraavasti. Yksittäinen tietoalkio $\mathbf{x}_i$ koostuu ominaisuuksista, joita on K kappaletta. Yleisin tapa esittää staattisia tietoalkioita on moniulotteinen vektori $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^K)$. Alkion ominaisuuksien lukumäärä ilmaisee alkion ulotteisuuden. Ominaisuuksien arvot voivat olla määrällisiä tai laadullisia, jatkuvia tai binäärisiä, nimellisiä tai järjestyslukuja.

Kahden alkion $\mathbf{x}_1$ ja $\mathbf{x}_2$ vertailtamista varten tarvitaan etäisyysfunktio $D(\mathbf{x}_1, \mathbf{x}_2)$, jonka valintaan alkioiden arvojen tyypit vaikuttavat. Etäisyysfunktio voidaan määritellä sovellusalueen mukaan, ja sopivan etäisyysfunktion valinta on keskeistä ryhmittelyn onnistumisen kannalta (Fränti & Kivijärvi, 2000). Etäisyysmittaa tarvitaan, jotta voidaan verrata alkioiden keskinäisiä etäisyyksiä, edustajien keskinäisiä etäisyyksiä ja alkioiden etäisyyksiä ryhmien edustajiin (Kärkkäinen, 2006). Staattiselle datalle yleisin käytetty etäisyysfunktio on euklidinen etäisyys, joka on erikoistapaus q=2 Minkowskin etäisyydestä. Euklidinen etäisyys D_E alkioiden **x** ja **y** välillä lasketaan kaavan (3.5) avulla ja Minkowskin etäisyys D_M lasketaan kaavalla (3.6), missä q on positiivinen kokonaisluku. (Hartigan, 1975; Liao, 2005).

$$D_E(x, y) = \sqrt{\sum_{i=1}^{K} (x_i, y_i)^2} \tag{3.5}$$

$$D_M(x, y, q) = \sqrt[q]{\sum_{i=1}^{K} (x_i\ ,\ y_i)^q}\ ,\ q \geq 1 \tag{3.6}$$

Yleisesti etäisyysmitan D tulee täyttää seuraavat ehdot (Xu & Wunsch, 2005):

1) Symmetrisyys: $D(\mathbf{x}_i, \mathbf{x}_j) = D(\mathbf{x}_j, \mathbf{x}_i)$,
2) Positiivisuus: $D(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ kaikilla $\mathbf{x}_i$ ja $\mathbf{x}_j$.

Jos D lisäksi täyttää seuraavat ehdot, niin sitä kutsutaan *metriikaksi*:

3) Kolmioepäyhtälö: $D(\mathbf{x}_i, \mathbf{x}_j) \leq D(\mathbf{x}_i, \mathbf{x}_k) + D(\mathbf{x}_k, \mathbf{x}_j)$ kaikille $\mathbf{x}_i$, $\mathbf{x}_j$ ja $\mathbf{x}_k$,
4) Refleksiivisyys: $D(\mathbf{x}_i, \mathbf{x}_j) = 0$ jos ja vain jos $\mathbf{x}_i = \mathbf{x}_j$.

Osituksen virheellisyyttä keskipistemallin mukaisessa ryhmittelyssä mitataan usein niin sanotun *keskivirheen* (*Mean Absolute Error*, *MAE*) avulla. Toinen tapa mitata osituksen aiheuttamaa virhettä on *keskineliövirheen* (*Mean Square Error*, *MSE*) avulla, jolloin

etäisyysfunktion arvo neliöidään. Keskineliövirhe painottaa poikkeavien tietoalkioiden osuutta osituksen virheellisyydessä. Keskivirhe voidaan laskea kaavan (3.7) avulla ja keskineliövirhe kaavan (3.8) avulla.

$$\mathrm{MAE}(X, P, C) = \frac{1}{N}\sum_{i=1}^{N} D_E(x_i, c_{\mathrm{p}(i)}) \tag{3.7}$$

$$\mathrm{MSE}(X, P, C) = \frac{1}{N}\sum_{i=1}^{N} D_E(x_i, c_{\mathrm{p}(i)})^2 \tag{3.8}$$

,

missä D_E on euklidinen etäisyys (3.5).

Algoritmissa 3.1 kuvataan keskivirheen laskeminen pseudokoodilla. Aluksi kerätään summa jokaisen tietoalkion etäisyydestä oman ryhmän keskipisteeseen, jonka jälkeen summa jaetaan alkioiden lukumäärällä. Tässä tutkielmassa käytän keskivirheen tapaista kaavaa aikasarjaryhmittelyn virheellisyyden tarkasteluun. Etäisyysmittana käytän euklidisen etäisyyden sijasta luvussa 4.2 esiteltävää etäisyysmittaa, joka soveltuu paremmin aikasarjadatalle.

```
MeanAbsoluteError(X,C,P) {

     sum := 0.0;
     FOR i := 1 TO N DO {
          j := pi;
          sum := sum + D(xi,cj);
     }

     RETURN (sum / N);
}
```

Algoritmi 3.1: Ryhmittelyvirheen laskeminen esitettynä pseudokoodilla.

3.3 Ryhmittelyalgoritmeja

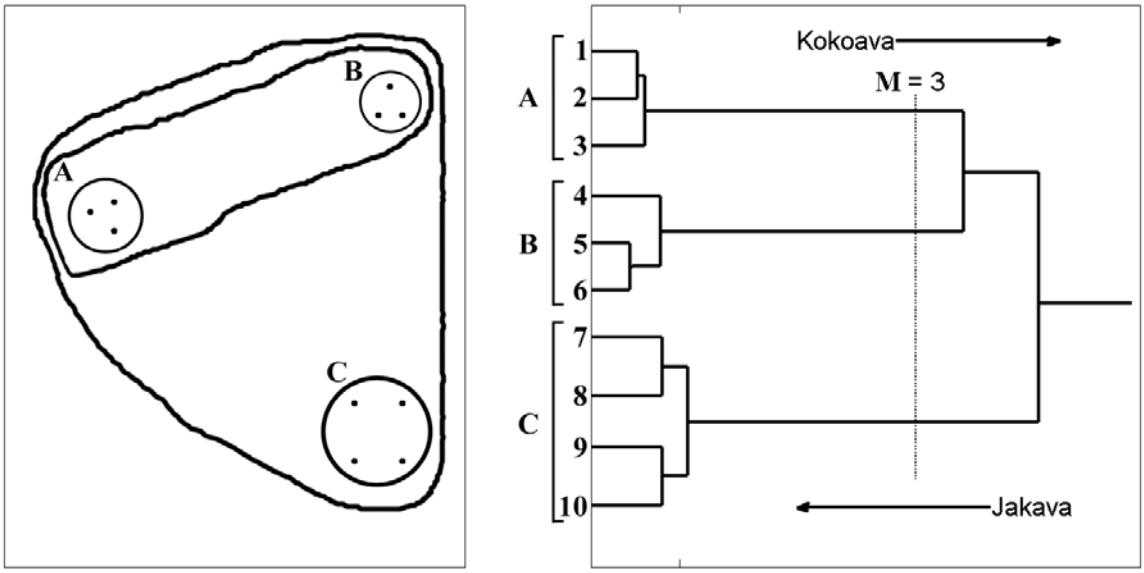
Ryhmittelyalgoritmien luokittelu on ongelmallista, sillä useat algoritmit ovat erilaisten ryhmittelymenetelmien yhdistelmiä (Han & Kamber, 2001). Kirjallisuudessa on esitetty

paljon keskenään ristiriidassa olevia luokitteluja ryhmittelymenetelmille, mikä entisestään hankaloittaa niiden luokittelua. Yleinen mielipide kuitenkin on, että ryhmittelymenetelmät voidaan jakaa kahteen ryhmään: *hierarkkisiin* ja *osittaviin* menetelmiin. (Jain & al., 1999; Jain & Dubes, 1988; Kärkkäinen, 2006; Xu & Wunsch, 2005)

3.3.1 Hierarkkiset menetelmät

Hierarkkiset ryhmittelymenetelmät muodostavat aineiston alkioille hierarkkisen rakenteen. Muodostettuja hierarkkisia ryhmittelyitä voidaan parhaiten kuvata kaksiulotteisten ryhmittelydiagrammien, *dendrogrammien*, avulla (Fielding, 2007). Dendrogrammia voidaan myös kutsua *ryhmittelypuuksi*. Ryhmittelypuu kuvaa kussakin vaiheessa tehdyt muutokset ositukseen ja sen, mihin ryhmään kukin alkio kuuluu kullakin ryhmämäärällä. Ryhmittelypuiden rakenne muistuttaa pitkälti biologien käyttämiä evoluutiopuumalleja (Fielding, 2007). Hierarkkisten menetelmien tuottamat ryhmittelypuut ovatkin olennaisia erityisesti biologisilla sovellusalueilla, joissa usein halutaan löytää hierarkkinen rakenne aineistoille. (Everitt, 1993)

Hierarkkiset menetelmät voidaan jakaa kahteen ryhmään niiden etenemissuunnan mukaan: *kokoaviin* (*agglomerative hierarchical clustering, AHC, bottom-up*) ja *jakaviin* (*divisive hierarchical clustering, DHC, top-down*) menetelmiin. Kokoavat menetelmät aloittavat tilanteesta, jossa jokainen aineiston alkio on omassa ryhmässä ja ryhmiä yhdistellään määritetyn etäisyysmitan mukaisessa järjestyksessä. Jakavat menetelmät puolestaan lähtevät alkutilanteesta, jossa kaikki aineiston alkiot ovat sijoitettuna samaan ryhmään ja olemassa olevaa ositusta pilkotaan hienompiin osiin lisäämällä ryhmien lukumäärää ja jakamalla aineisto uudestaan ryhmien kesken. Hierarkkisten ryhmittelyalgoritmien kokoamis- tai jakamisaskeleita toistetaan, kunnes on saavutettu käyttäjän antama määrä ryhmiä tai raja-arvo ryhmittelyn virheellisyydelle (Everitt, 1993). Kuvassa 3.5 on ryhmittelypuu, joka on luotu ryhmittelemällä 10 tietoalkiota kokoavan hierarkkisen algoritmin avulla. Yksi tietoalkio koostuu x- ja y-koordinaateista ja yhdessä ne muodostavat kolme selvästi toisistaan erillään olevaa ryhmää. Kuvaan on merkitty kokoavan ja jakavan menetelmän etenemissuunnat sekä ryhmittelypuun avulla valittu lukumäärä ryhmille.



Kuva 3.5: Hierarkkisen ryhmittelyn havainnollistaminen ryhmittelypuulla.

Everittin (1993) mukaan kokoavia menetelmiä sovelletaan paljon yleisemmin kuin jakavia. Kokoavat hierarkkiset menetelmät toteutetaan usein luomalla N x N –kokoinen etäisyysmatriisi, jossa pidetään yllä ryhmien välisiä etäisyyksiä. Valittaessa yhdistettäviä ryhmiä etsitään etäisyysmatriisista solu, jossa ryhmien välinen etäisyys on pienin. Tämän jälkeen soluun liittyvät ryhmät yhdistetään ($\mathbf{X}_a \leftarrow \mathbf{X}_a \cup \mathbf{X}_b$) ja etäisyysmatriisia päivitetään yhdistettyjen ryhmien osalta (Fränti & al., 2006; Jain & Dubes, 1988). Ryhmien välisenä etäisyysmittana käytetään usein *yksittäislinkitystä* (*single linkage*; *SL-AHC*), *täydellistä linkitystä* (*complete linkage*; *CL-AHC*) tai *keskiarvolinkitystä* (*average linkage*; *AL-AHC*). Yksittäislinkityksessä ryhmien välinen etäisyys määräytyy toisiaan lähimpinä olevien, eri ryhmiin kuuluvien alkioiden välisestä etäisyydestä kaavan (3.9) mukaisesti. Kaavalla (3.10) voidaan laskea täydellisessä linkityksessä käytettävä ryhmien välinen etäisyys, joka on eri ryhmiin kuuluvien, kauimpana toisistaan olevien tietoalkioiden välinen etäisyys. Keskiarvolinkityksessä puolestaan muodostetaan keskiarvoetäisyys ryhmien kaikkien alkioiden välisistä etäisyyksistä, kaavan (3.11) mukaan. (Hirano & Tsumoto, 2005). Toinen yleinen tapa muodostaa yhdistämismatriisi on niin sanottu Wardin menetelmä (Ward, 1963), joka tunnetaan vektorikvantisoinnin puolella pareittaisena yhdistelymenetelmänä (Pairwise Nearest Neighbor, PNN; Equitz, 1989; Fränti & al., 2006). Tässä metodissa lasketaan ryhmien etäisyysmittojen sijaan kahden ryhmän yhdistämisestä ositukseen aiheutuvaa *tiedon häviämistä* (*information loss*; *Jain* & Dubes, 1988).

$$D_{SL}(X_i\ ,\ X_j) = min_{\left\{ \begin{matrix} x(k)\in X(i) \\ x(l)\in X(j) \end{matrix} \right.} D(x_k, x_l) \tag{3.9}$$

$$D_{CL}(X_i\ ,\ X_j) = max_{\left\{ \begin{matrix} x(k)\in X(i) \\ x(l)\in X(j) \end{matrix} \right.} D(x_k, x_l) \tag{3.10}$$

$$D_{AL}(X_i\ ,\ X_j) = \frac{1}{N_i + N_j} \sum_{x_k \in X_i} \sum_{x_l \in X_j} D(x_k\ ,\ x_l) \tag{3.11}$$

Tätä tutkielmaa varten on toteutettu kokoava hierarkkinen ryhmittelyalgoritmi, jossa käytetään keskiarvolinkitystä ryhmien välisen etäisyyden mittana. Yksittäisten aikasarjojen etäisyyttä puolestaan mitataan niin sanotulla DTW-etäisyydellä, joka esitellään luvussa 4.2. Toteutettu hierarkkinen algoritmi käydään läpi luvussa 4.4.3.

3.3.2 Osittavat menetelmät

Osittaviin menetelmiin lasketaan tässä tutkielmassa kaikki ryhmittelymenetelmät, jotka eivät selvästi seuraa hierarkkista rakennetta ryhmitellessään aineistoa. Han & Kamber (2001) jakavat ryhmittelyalgoritmit osittaviin, hierarkkisiin, tiheyspohjaisiin, ruudukkopohjaisiin ja mallipohjaisiin menetelmiin. Xu ja Wunsch (2005) puolestaan luokittelevat klassiset osittavat menetelmät neliövirhepohjaisiin ja arviopohjaisiin menetelmiin, minkä lisäksi he luettelevat muita ryhmittelymenetelmiä, jotka pohjautuvat muun muassa graafiteoriaan, kombinatorisiin hakutekniikkoihin, sumeaan joukko-oppiin ja neuroverkkoihin. Seuraavaksi käydään läpi eri menetelmien tapoja käsitellä ryhmittelyongelmaa noudattaen Xun ja Wunschin (2005) sekä Hanin ja Kamberin (2001) jakoja.

McQueenin (1968) esittelemä *K-means* on tunnetuin *neliövirhepohjainen*, keskipistemallin mukainen ryhmittelyalgoritmi. Se tunnetaan monilla useilla nimillä eri sovellusalueilla, esimerkiksi *yleistetty Lloydin algoritmi* (*Generalized Lloyd Algorithm, GLA*), *kova c-means* (*hard c-means*) tai *Linde, Buzo, Grey* (*LBG*). K-meansin nopeus ja yksinkertaisuus ovat

selvästi vaikuttaneet sen suosioon ryhmittelymenetelmänä. Algoritmissa ryhmän edustajana käytetään ryhmään kuuluvien alkioiden laskennallista keskiarvovektoria, sentroidia. (Fränti & Kivijärvi, 2000; Kärkkäinen, 2006)

K-means perustuu kahteen optimaalisuusfunktioon, joita vuorottelemalla päästään paikallisesti optimaaliseen ositukseen. Ensimmäiseksi lasketaan kullekin tietoalkiolle lähin ryhmän edustaja. Toisessa askeleessa muodostetaan uudet ryhmien edustajat laskemalla keskiarvovektori kullekin ryhmälle siihen liittyvistä tietoalkioista. Jokainen uusi ositus on parempi tai vähintään yhtä hyvä kuin vanha. Näitä kahta askelta toistetaan haluttu määrä kierroksia tai lopetetaan, kun ositus ei enää muutu. K-means algoritmin aikavaativuus on O(KINM), missä K on aineiston ulotteisuus, I on suoritettavien k-means iteraatioiden määrä, N on aineiston koko ja M on ryhmien lukumäärä (Jain & Dubes, 1988). K-means on nopea ja yksinkertainen menetelmä, mutta se kärsii juuttumisesta lokaaliin optimiin ja se on herkkä datassa esiintyvälle kohinalle ja poikkeaville arvoille (Xu & Wunsch, 2005). Alkuositus on ratkaiseva ryhmittelyn onnistumisen kannalta ja eri alkuosituksilla saatujen osituksien laatu vaihtelee suuresti. Menetelmän etu on siinä, että sitä voidaan käyttää muiden algoritmien osana ositusten optimoinnissa. (Han & Kamber, 2001; Fränti & Kivijärvi, 2000)

K-means -algoritmin yleisyyden vuoksi sille on kehitelty paljon erilaisia muunnoksia (Jain & al., 1999). Tässä tutkielmassa toteutetaan Kaufmanin ja Rouseeuw:n (1990) esittelemä PAM -algoritmi (Partitioning Around Medoids), joka tunnetaan myös nimellä *k-medoids*. K-medoidsissa käytetään ryhmien edustajina laskennallisten keskipisteiden sijaan todellisia tietoalkioita, jotka minimoivat etäisyyden kaikkiin muihin samaan ryhmään kuuluviin alkioihin. Näitä edustajia kutsutaan tästä lähtien ryhmien medoideiksi. Toinen aikasarjojen ryhmittelyä varten toteutettu k-meansin muunnelma on niin kutsuttu *satunnaistettu paikallishaku* (*Randomised Local Search*, *RLS*; Fränti & Kivijärvi, 2000). RLS-algoritmi pyrkii välttämään lokaaliin optimiin juuttumisen vaihtamalla satunnaisesti ryhmän edustajia. RLS-algoritmia käsitellään tarkemmin luvussa 4.4.2.

Klassiset osittavat ryhmittelymenetelmät pohjautuvat kahden alkion välisiin etäisyysmittoihin. Tästä johtuen menetelmät löytävät yleensä vain pallomaisia ryhmiä. *Tiheyspohjaiset menetelmät* pohjautuvat alkioiden väliseen tiheyteen, joka yleensä määritellään lähellä sijaitsevien alkioiden lukumääränä. Tyypillinen tiheyspohjainen ryhmittelyalgoritmi on *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*; Ester & al., 1996).

Menetelmän hyvä puoli on se, että käyttäjän ei tarvitse tietää ryhmien lukumäärää ja se ei ole herkkä epämääräisen muotoisille ryhmille tai kohinalle kuten k-means. (Han & Kamber, 2001) *Ruudukkopohjaisten menetelmien* takana on pyrkimys nopeuttaa ryhmittelyä. Menetelmät kuten *STING* (*STatistical INformation Grid*; Wang & al., 1997) jakavat spatiaalisen arvoavaruuden kuutiomaisiin soluihin, jotka muodostavat ruudukon omaisen rakenteen. Ruudukkopohjaiset menetelmät käyttävät aineistosta karkeampaa esitystä, jolloin ryhmiteltäviä alkioita on vähemmän ja ryhmittely nopeutuu (Han & Kamber, 2001). Mallipohjaisten menetelmien tavoitteena on sovittaa annettu aineisto ja joku matemaattinen malli toisiinsa nähden. Hanin ja Kamberin (2001) mukaan mallipohjaiset menetelmät voidaan edelleen jakaa kahteen ryhmään: tilastollisiin ja neuroverkkopohjaisiin menetelmiin. Tässä lueteltujen menetelmien lisäksi on olemassa ryhmittelymenetelmiä, jotka on hankala luokitella mihinkään edellä mainituista ryhmistä.

4 AIKASARJOJEN RYHMITTELY

Aivan kuten staattisten tietorakenteiden kanssa, myös ajan suhteen järjestetylle datalle on kehitetty erilaisia menetelmiä ryhmittelyyn. Olennaista on ryhmittelyalgoritmin, käytetyn etäisyysmitan ja ryhmittelyn onnistuneisuutta mittaavan arviointikriteerin määrittäminen. Liao (2005) jakaa aikasarjojen ryhmittelyn kolmeen luokkaan: raa'an datan ryhmittely, piirrevektorien ryhmittely sekä datasta luotujen mallien ryhmittely. Mörchen (2006) tarkentaa raaka-data -mallin jakoa sen mukaan, miten aikasarjoja ryhmitellään. Voidaan ryhmitellä *kokonaisia aikasarjoja* (*whole series clustering*) keskenään. Toinen Mörchenin (2006) mainitsema menetelmä on yhden pitkän aikasarjan ryhmittely, tavoitteena löytää alkuperäisestä pitkästä aikasarjasta osa, joka vastaa niin kutsuttua *kyselyaikasarjaa* (*query time-series*). Tässä tutkielmassa keskitytään pelkästään raaka-data -mallin mukaiseen, kokonaisten aikasarjojen ryhmittelyyn. Tavoitteena on ryhmitellä eri pituisia, mutta samankaltaisia aikasarjoja luonnollisiin ryhmiin niiden muotojen perusteella. Kirjallisuudessa on esitetty monia heuristisia ratkaisuja aikasarjojen ryhmittelyyn, mutta näitä ei ole määritelty kombinatorisena optimointiongelmana. Tässä tutkielmassa annetaan täsmälliset määrittelyt käytetyille etäisyysfunktiolle, edustaja-aikasarjoille ja tavoitefunktiolle. Nämä määritelmät ovat yleistyksiä klassiselle ryhmittelylle käytetyistä määrittelyistä.

4.1 Aikasarjaryhmittelyn ongelmanmäärittely

Määrittelen seuraavaksi aikasarjaryhmittelyn kombinatorisena optimointiongelmana, kuten luvussa 3.2 määriteltiin ryhmittelyongelma staattiselle datalle (mukaillen Fräntiä ja Kivijärveä, 2000 sekä Fräntiä ja muita, 2006). Aikasarjoja käsiteltäessä käytän ryhmien lukumäärästä termiä mallin koko, joka tarkoittaa edustaja-aikasarjojen lukumäärää.

N Aineiston koko, aikasarjojen lukumäärä.

M Mallin koko.

K Aineiston ulotteisuus.

$\boldsymbol{S}$ Aineisto, jonka koko on N, $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\}$.

$\mathbf{s}_i$ Aineiston i:s aikasarja pituudeltaan $L_i = |\mathbf{s}_i|$, $\mathbf{s}_i = \{\mathbf{s}_i^1, \mathbf{s}_i^2, \ldots, \mathbf{s}_i^L\}$.

$\boldsymbol{P}$ Ositus, joka liittää aikasarjat $\mathbf{s}_i$ ryhmiin $\mathbf{c}_j$, $\boldsymbol{P} = \{p_1, p_2, \ldots, p_N\}$.

$\boldsymbol{C}$ Ryhmien edustajien joukko, jonka koko on M, $\boldsymbol{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M\}$.

Tehtävänä on osittaa syötteenä annettu N:n kokoinen aikasarja-aineisto $\boldsymbol{S} = \{\mathbf{s}_i \mid 1 \leq i \leq N\}$ M:ään määrään ryhmiä siten, että muodon perusteella samankaltaiset aikasarjat ryhmitellään samaan ryhmään ja erilaiset aikasarjat ryhmitellään muihin ryhmiin. Aikasarjan $\mathbf{s}_i$ pituus on $L_i = |\mathbf{s}_i|$. Ositus $\boldsymbol{P} = \{p_i \mid 1 \leq i \leq N\}$ määrittelee ryhmittelyn osoittamalla jokaiselle aikasarjalle $\mathbf{s}_i$ ryhmäindeksin p_i, jonka arvo määrää ryhmän, johon aikasarja $\mathbf{s}_i$ kuuluu. Ositus $\boldsymbol{P}$ saadaan laskemalla ryhmäindeksit p_i kaavan (4.1) avulla. Aikasarjojen etäisyysmittana käytän DTW-etäisyyttä, joka määritellään tarkemmin luvussa 4.2.

$$p_i = argmin_{j=1}^{M} D_{DTW}(s_i, c_j) \;\; \forall\, i \in [1..N] \tag{4.1}$$

Ryhmä $\boldsymbol{S}_j$ on joukko aikasarjoja, jotka kuuluvat samaan ositukseen j siten, että $\boldsymbol{S}_j = \{\mathbf{s}_i \mid p_i = j\}$. Ryhmien edustajat esitetään joukkona $\boldsymbol{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots \mathbf{c}_M\}$. Ryhmien edustajina käytetään *medoideja*. Medoidi lasketaan kaavan (4.2) avulla.

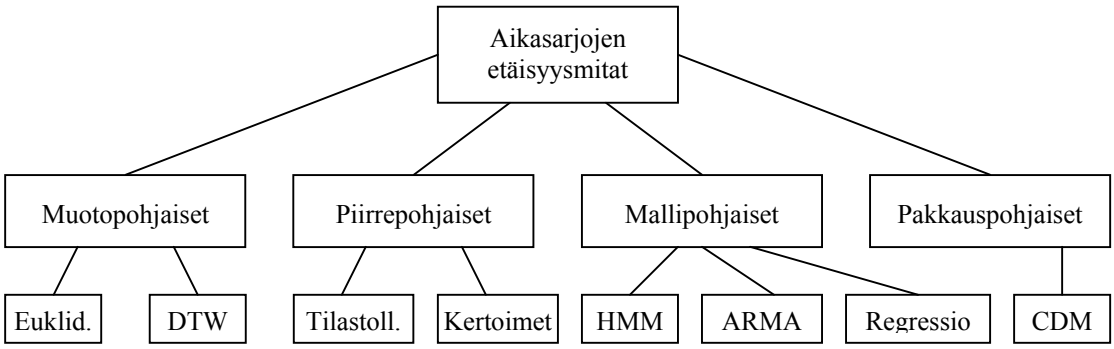
$$c_j = argmin_{s(i) \in S(j)} \sum_{s(k) \in S(j)} D_{DTW}(s_i, s_k) \;\; \forall\, j \in [1..M] \tag{4.2}$$

Aikasarjojen ryhmittelyä varten määrittelen uuden optimoitavan tavoitefunktion (4.3), joka voidaan laskea osituksen $\boldsymbol{P}$ muodostamana virheellisyytenä, kun aikasarja-aineisto $\boldsymbol{S}$ jaetaan M:ään ryhmään, joiden edustajina toimivat aikasarjat $\boldsymbol{C}$. Kyseinen funktio on suora yleistys keskivirheestä MAE. Koska aikasarjojen etäisyysmittana toimii euklidisen etäisyyden sijaan DTW-etäisyys, niin sitä käytetään myös tämän funktion osana.

$$J(S, P, C) = \frac{1}{N} \sum_{i=1}^{N} D_{DTW}(s_i, c_{p(i)}) \tag{4.3}$$

Numeerisille aikasarjoille käytettävät etäisyysmitat voidaan jakaa neljään luokkaan: muoto-, piirre-, malli- ja pakkauspohjaisiin mittoihin (Mörchen, 2006). Kuva 4.1 kuvaa aikasarjoille käytettyjen etäisyysmittojen luokittelua. Muotopohjaiset menetelmät soveltuvat aikasarjojen yleisen luonteen vertailuun. Suosituin ryhmään kuuluva etäisyysmitta aikasarjoille on euklidinen etäisyys, vaikka se ei sovellu hyvin aikasarjoille (Keogh & Kasetty, 2002). Toinen yleisesti käytetty muotopohjainen etäisyysmitta on dynaaminen aikasovitus -etäisyys, jota käytetään myös tässä tutkielmassa. Muotopohjaiset menetelmät eivät välttämättä sovellu

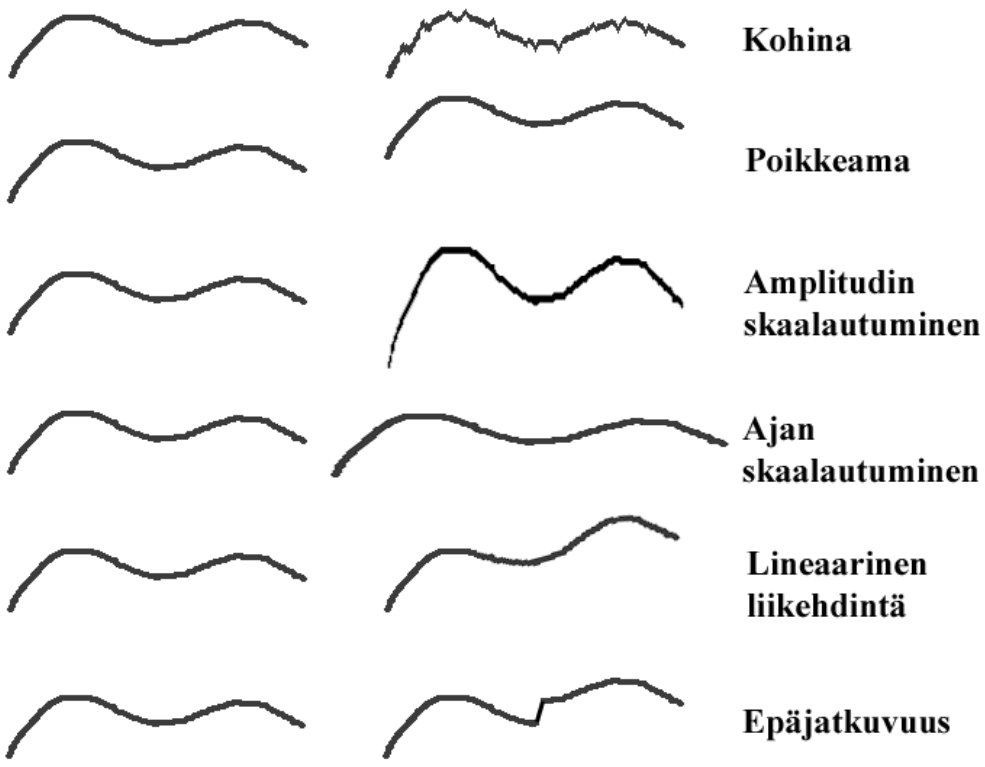
hyvin eripituisten aikasarjojen vertailuun. Tällöin voidaan harkita piirre- tai mallipohjaisten etäisyysmittojen käyttämistä. Piirrepohjaisissa menetelmissä aikasarjoista pyritään erottamaan niille ominaisimpia piirteitä. Mallipohjaiset mitat voidaan nähdä kehittyneempänä versiona piirrepohjaisista mitoista, sillä käytössä on malleja joihin aikasarjoja voidaan verrata. Pakkauspohjaiset mitat ovat uusin mittajoukko, jonka periaatteena on että samankaltaisten aikasarjojen pitäisi tuottaa yhdistettäessä ja pakatessa parempia pakkaussuhteita kuin hyvin erikaltaisten aikasarjojen. (Mörchen, 2006)



Kuva 4.1: Numeeristen aikasarjojen etäisyysmittojen luokittelu. (Mörchen, 2006)

4.2 Dynaaminen aikasovitus

Yksi keskeisimpiä haasteita aikasarjaryhmittelyssä on sopivan samankaltaisuuskriteerin asettaminen aikasarjoille (Keogh & Pazzani, 1999). Tietoalkioiden etäisyyksien vertailuun yleisesti käytetty euklidinen etäisyys aiheuttaa ongelmia aikasarjojen kohdalla, mikäli aikasarjat ovat eri pituisia tai niissä on poikkeamia, kuten kohinaa tai viiveitä. Mikäli vertailtavat aikasarjat eivät ole täydellisesti synkronoituja ja skaalattuja, niin euklidinen etäisyys tuottaa ei-toivottuja etäisyyksiä aikasarjojen välille. Tästä huolimatta euklidista etäisyyttä käytetään, usein virheellisesti, aikasarjojen etäisyyksien selvittämiseen, sillä se on helppo ja nopea laskea (Berndt & Clifford, 1994; Hirano & Tsumoto, 2004; Gaudin & al., 2006). Kuvassa 4.2 on havainnollistettu yleisiä ongelmia aikasarjojen samankaltaisuuden mittaamisen kannalta. Kohina on pienimuotoista vaihtelua havaintoarvoissa, joka saattaa johtua esimerkiksi mittauslaitteiden epätarkkuudesta. Poikkeama on tasainen muutos havaintoarvoissa kahden aikasarjan välillä. Käsiteltävät aikasarjat voivat myös poiketa toisistaan amplitudin tai ajan skaalautumisen osalta. Muita Keoghin & Pazzanin (1998) mainitsemia ongelmia ovat lineaarinen liikehdintä ja epäjatkuvuuspisteet aikasarjoissa.



Kuva 4.2: Aikasarjoissa yleisesti ilmeneviä ongelmia samankaltaisuuskriteerin osalta. (mukaillen lähdettä Keogh & Pazzani, 1998)

Dynaaminen aikasovitus (*Dynamic Time Warping*, *DTW*) on menetelmä, jolla voidaan vertailla kahta eripituista aikasarjaa keskenään niiden muotojen perusteella (Keogh & Pazzani, 1999). Dynaaminen aikasovitus voidaan kuvata kahden aikasarjan aikaulottuvuuden muokkaamismenetelmäksi, jonka tavoitteena on signaalimuotojen optimaalinen sovitus toisiinsa nähden (Hartimo & al., 1985). Dynaamisen aikasovituksen kehittäjää on hankala määrittää, sillä kirjallisuudessa esiintyy paljon samankaltaisia algoritmeja, joita esimerkiksi Kruskal (1983) käsittelee. DTW ei ole etäisyysmittana metriikka (Ruiz & al., 1985). Tätä tutkielmaa varten määritellään dynaaminen aikasovitus seuraavasti (mukaillen Ratanamahatanaa ja Keoghia, 2004):

Olkoon S ja T kaksi aikasarjaa, joiden pituudet ovat |S| = N ja |T| = M. Aikasarja S koostuu N:stä otoksesta $s_1..s_N$ ja aikasarja T koostuu M:stä otoksesta $t_1..t_M$.

$$S = \{s_1, s_2, \ldots, s_N\}$$

$$T = \{t_1, t_2, ..., t_M\}$$

Dynaamisen aikasovituksen muodostamiseksi lasketaan ensin täysi etäisyysmatriisi, jonka koko on N x M. Etäisyysmatriisin solu (i,j) vastaa otoksien s_i ja t_j välistä kumulatiivista etäisyyttä. Paras sovitus aikasarjojen välillä löytyy selvittämällä *optimaalinen sovituspolku*, joka kulkee matriisin kulmasta vastakkaiseen kulmaan. *Sovituspolku* W on jatkuva lista etäisyysmatriisin soluja ja se kuvaa siirtymän aikasarjojen S ja T otoksien välillä. *Optimaalinen sovituspolku* W_O puolestaan kuvaa parhaan sovituksen aikasarjojen S ja T otoksien välillä. Sovituspolun pituus on vähintään pidemmän aikasarjan pituus ja enintään molempien aikasarjojen pituuksien summa miinus yksi. Sovituspolun k:s solu esitetään lukuparina $(i, j)_k$.

$$W = w_1, w_2, \ldots, w_k, \ldots w_K,$$
missä $\max(M,N) \leq K \leq m+n-1$.

Bellman (1957) kehitti niin kutsutun *optimaalisuusperiaatteen* (*Principle of Optimality*), jonka nojalla jokainen ongelma on ratkaistavissa optimaalisesti dynaamisen ohjelmoinnin avulla, jos se voidaan jakaa pienempiin osaongelmiin, joille optimaalinen ratkaisu on muodostettavissa (Bertsekas, 1976). Dynaamisen aikasovituksen tapauksessa optimaalinen sovituspolku W_O voidaan laskea käyttämällä dynaamista ohjelmointia, sillä optimaalisen sovituspolun etsiminen voidaan jakaa vierekkäisten otoksien välisten optimaalisten sovituspolkujen etsimiseen. Kahden aikasarjan S ja T otoksien i ja j välinen euklidinen etäisyys D_E lasketaan kaavan (3.5) avulla. Solun (i,j) kumulatiivinen etäisyys voidaan muodostaa laskemalla yhteen solua vastaavien otoksien etäisyys $D_E(s_i,t_j)$ ja pienin kumulatiivinen etäisyys kolmesta laskettavaa solua edeltävästä solusta kyseisellä sovituspolulla:

$$D_C(i,j) = D_E(i,j) + \min(D_C(i-1,j-1), D_C(i-1,j), D_C(i,j-1))$$

Jos sovituspoluille ei aseteta muita rajoitteita, niin vertailtavina on paljon myös hyödyttömiä sovituspolkuja. Esimerkiksi kaikki polut, jotka eivät ala solusta (1,1) ovat turhia, sillä dynaamista aikasovitusta halutaan soveltaa kahden kokonaisen aikasarjan välillä. Käytännössä sovituspoluille asetetaan rajoitteita, joilla saadaan karsittua turhaa laskentaa ja siten myös nopeutettua dynaamista aikasovitusta. Tässä tutkielmassa käytetyt rajoitteet sovituspoluille

ovat *rajaehdot* (*boundary conditions*), *jatkuvuusehto* (*continuity condition*) ja *monotoonisuusehto* (*monotonicity condition*). Muita yleisesti käytettyjä lisärajoitteita ovat *kaltevuusehto* (*slope constraint condition*) ja erilaiset *sovitusikkunat* (*adjustment window condition*). Näihin keskitytään tarkemmin luvussa 4.5.

Rajaehdot: Rajaehdot määrittävät sovituspolun aloitus- ja lopetuspisteet. $w_1 = (1,1)$ ja $w_k = (N,M)$.
Jatkuvuusehto: Jatkuvuusehto rajoittaa sallitut polun askeleet vain kolmeen viereiseen soluun. Jos $w_k = (a, b)$, niin $w_{k-1} = (a',b')$ siten, että $a - a' \leq 1$ ja $b - b' \leq 1$.
Monotonisuusehto: Monotonisuusehto pakottaa sovituspolun olevan koko ajan etenevä. Jos $w_k = (a, b)$, niin $w_{k-1} = (a',b')$ siten, että $a - a' \geq 0$ ja $b - b' \geq 0$.

Koska yhden otosparin sovittamiseen tarvitsee käydä läpi kaikki aikasarjojen ulottuvuudet kaavan (3.5) mukaisesti, niin aikavaativuus otosparin sovittamiseen on O(K). Ilman lisärajoitteita sovitettavia otoksia on N x M kappaletta, joten täydellisen dynaamisen aikasovituksen laskennallinen aikavaativuus K-ulotteisille aikasarjoille S ja T on O(KNM), missä N ja M ovat sovitettavien aikasarjojen pituudet. Aiemmin mainitut lisärajoitteet pienentävät aikavaativuutta vain pienen vakion verran (Ratanamahatana & Keogh, 2005). DTW-etäisyys voidaan määritellä rekursiivisesti seuraavalla tavalla (Fu & al., 2005):

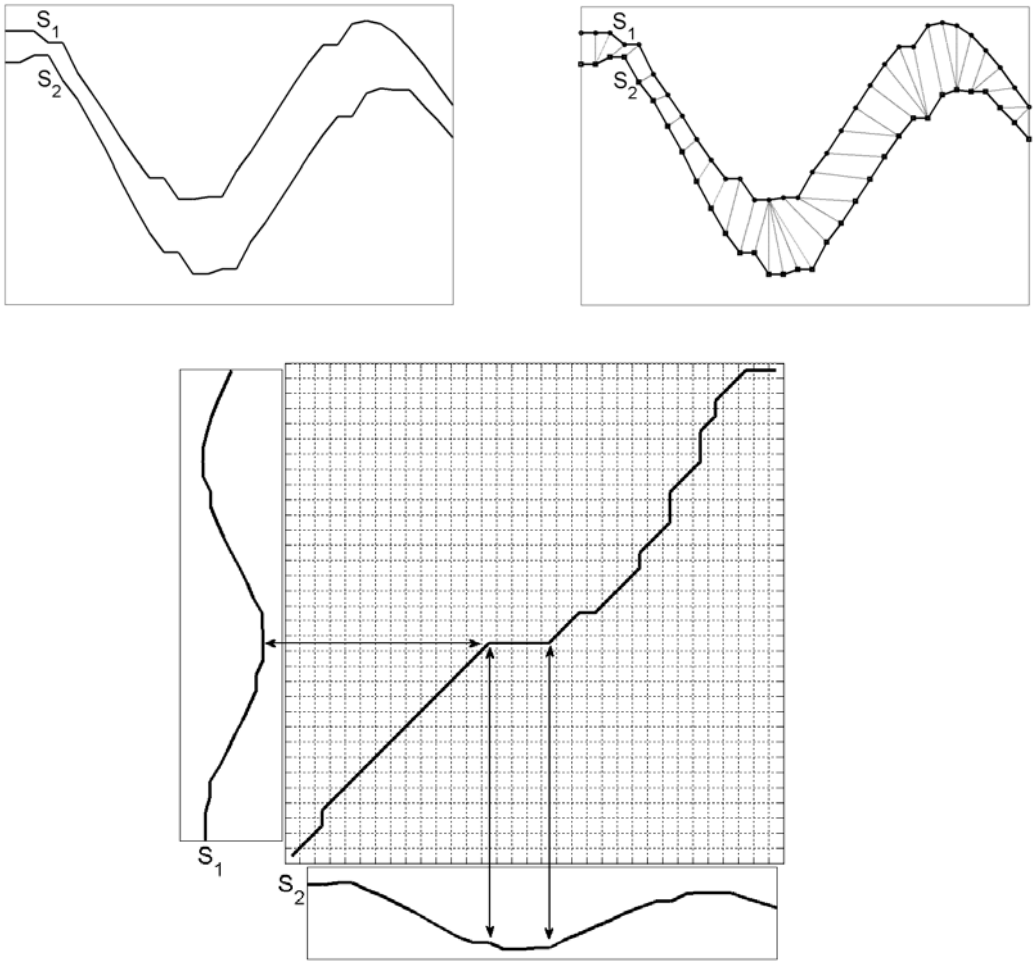
$$D_{DTW}(\varnothing, \varnothing) = 0 \tag{4.4}$$

$$D_{DTW}(S, \varnothing) = D_{DTW}(\varnothing, T) = \infty \tag{4.5}$$

$$D_{DTW}(S, T) = D_E(S_1, T_1) + min\begin{cases} D_{DTW}(S, T_{2..M}) \\ D_{DTW}(S_{2..N}, T) \\ D_{DTW}(S_{2..N}, T_{2..M}) \end{cases} \tag{4.6}$$

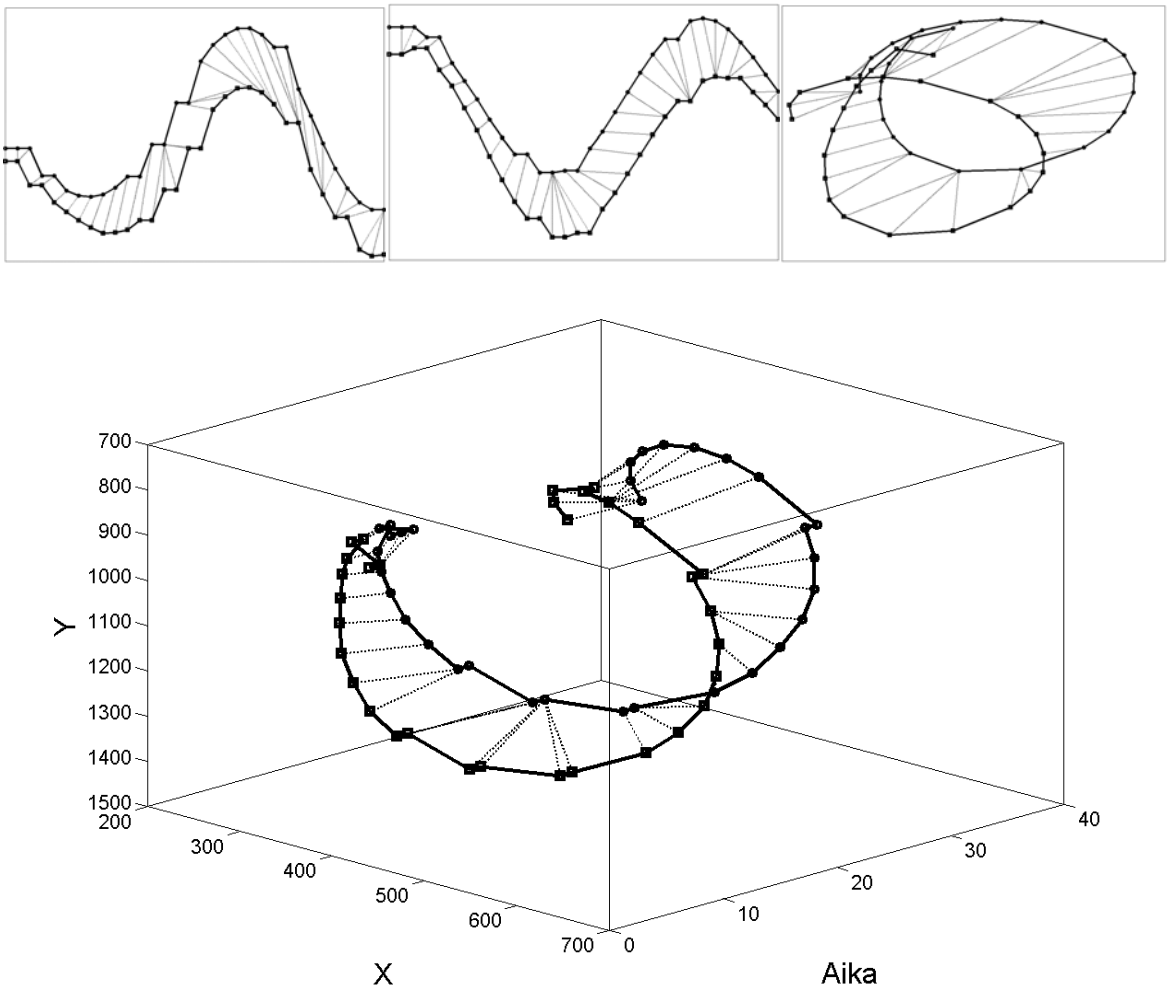
Kuvassa 4.3 on esimerkki dynaamisen aikasovituksen laskemisesta kahden aikasarjan välille. Vasemmalla ylhäällä oleva kuva esittää kahta aikasarjaa s_1 ja s_2, jotka halutaan sovittaa toisiinsa. DTW-algoritmin tuloksena saadaan sovitusmatriisiin optimaalinen sovituspolku, jota seuraamalla otokset voidaan sovittaa toisiinsa. Oikealla ylhäällä on lopputulos dynaamisesta aikasovituksesta. Katkoviivat kuvaavat otospareja, jotka asettuvat optimaaliselle

DTW-polulle. Matriisiin merkityt kaksoisnuolet osoittavat kohdan, jossa yksi aikasarjan S_1 otos sovitetaan viiteen aikasarjan S_2 otokseen.



Kuva 4.3: DTW-polun muodostaminen kahden aikasarjan välille.

Kuvassa 4.4 on esimerkki kahden käsinkirjoitetun merkin sovittamisesta toisiinsa. Merkit koostuvat x- ja y-koordinaattipisteistä. Aineisto, johon aikasarjat kuuluvat, kuvataan tarkemmin luvussa 5.1. Molemmat merkit kuvaavat numeroa nolla, ja ne on kirjoittanut sama henkilö. Vasemmalla ylhäällä olevassa kuvassa on molemmista merkeistä x-koordinaatin mukaiset aikasarjat. Keskellä ylhäällä sijaitsevat y-koordinaatit ovat samat kuin kuvan 4.3 esimerkissä. Oikealla ylhäällä on havaittavissa molemmat merkit x- ja y-koordinaattien suhteen. Tämä kuvakulma ei ota kantaa merkkien aika-ulottuvuuteen, vaan kuvaa merkkejä niin kuin ihminen ne paperilta kirjoitettuna tunnistaisi. Alhaalla sijaitsevassa isossa kuvassa on puolestaan suoritettu dynaaminen aikasovitus x- ja y-koordinaattien suhteen.



Kuva 4.4: Dynaaminen aikasovitus kahden kaksiulotteisen aikasarjan s_1 ja s_2 välillä.

Euklidisen etäisyyden laskeminen kahden otoksen välillä voidaan toteuttaa algoritmin 4.1 avulla. Kyseisen algoritmin aikavaativuus on O(K), sillä etäisyyden laskentaa varten tarvitsee käydä lävitse aikasarjojen kaikki ulottuvuudet.

```
D_E(s_i,s_j) {
     squaresum := 0.0;

     FOR d := 1 TO K {
          squaresum := squaresum + (s_i^d - s_j^d)^2;
     }
     RETURN SQRT(squaresum);
}
```

Algoritmi 4.1: Euklidisen etäisyyden laskeminen kahden otoksen välillä pseudokoodina.

DTW-algoritmi on esitetty pseudokoodina algoritmissa 4.2. Yhden etäisyysvertailun aikavaativuus on kyseistä DTW-algoritmia käyttäen $O(KL_1L_2)$, missä K on aineiston ulotteisuus, L_1 on ensimmäisen aikasarjan pituus ja L_2 on toisen aikasarjan pituus. Menetelmä on huomattavasti hitaampi kuin perinteisen euklidisen etäisyyden aikavaativuus, joka on O(KL).

```
D_DTW(s_i,s_j) {

    FOR a := 1 TO |s_i| DO {
        FOR b := 1 TO |s_j| DO {
            pathStep := MIN(D_C[a-1][b],D_C[a][b-1],D_C[a-1][b-1]);
            D_C[a][b] := D_E(s_i^a,s_j^b) + pathStep;
        }
    }

    RETURN D_C[|s_i|][|s_j|];
}
```

Algoritmi 4.2: DTW-etäisyyden laskeminen kahden aikasarjan välillä pseudokoodina.

Dynaamisesta aikasovituksesta voidaan tehdä mielenkiintoinen havainto, jos tarkastellaan aineistoa, jonka kaikkien K-ulotteisten aikasarjojen pituudet ovat 1. Tällöin aikasarjaa $\mathbf{s}_i$ voidaan ajatella K-ulotteisena tietoalkiona $\mathbf{x}_i$. Algoritmissa 4.2 määritelty DTW-etäisyys palauttaa tällöin aina tietoalkioiden välisen euklidisen etäisyyden. Tällöin aikasarjoille määritellyt ryhmittelyalgoritmit sievenevät vastaamaan staattisia vastineitaan: tavoitefunktio 4.3 vastaa tavoitefunktiota 3.7; ryhmittelyalgoritmi luvusta 4.4.1 vastaa k-medoidsin klassista versiota ja etäisyysmitta D_{DTW} vastaa euklidista etäisyyttä D_E. Voidaan todeta, että klassinen, staattinen data on erikoistapaus aikasarjadatasta, jossa aikasarjojen pituudet ovat 1. Tämän seurauksena dynaamista aikasovitusta etäisyysmittana käyttävät ryhmittelyalgoritmit soveltuvat, ainakin teoriassa, sellaisenaan myös staattisen datan ryhmittelyyn.

4.3 Käytetyt ryhmien edustajat

Usein aikasarjoja ryhmiteltäessä halutaan löytää mahdollisimman hyvät mallit aineistolle (Liao, 2005). Malli kootaan ryhmittelyn tuloksena saadun osituksen muodostamien ryhmien avulla ja ryhmät määritellään edustajien avulla. Ryhmän edustaja voi olla laskennallinen tai

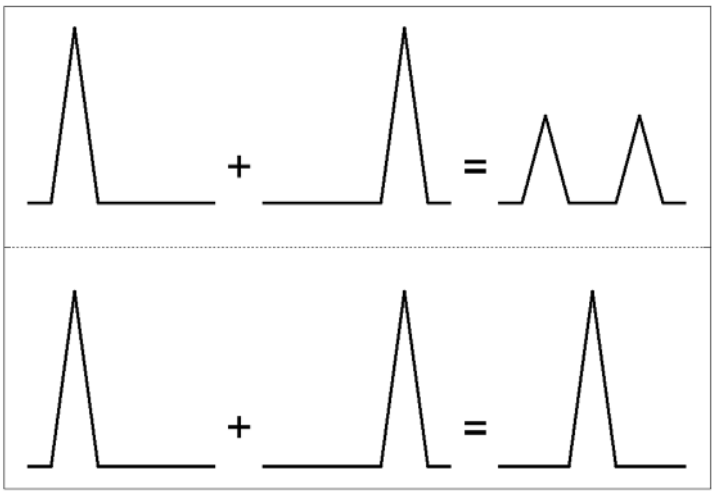
todellinen aikasarja. Seuraavaksi esitellään tässä tutkielmassa käytetyt edustajat ryhmille ja se, kuinka ne saadaan muodostettua annetulle ositukselle.

4.3.1 Medoidi

Intuitiivisesti ajatellen ryhmää parhaiten kuvaava edustaja on ryhmän alkioiden keskiarvovektori, joka on laskennallinen edustaja ryhmän keskustalle. Keskiarvoaikasarjan muodostaminen ei kuitenkaan käy samalla tavalla suoraan laskemalla otoksien euklidisista etäisyyksistä keskiarvot kuin staattisen datan kanssa, sillä aikasarjojen pituudet voivat vaihdella ja niissä saattaa olla viiveitä, jotka heikentävät lasketun edustajan laatua. Mikäli ollaan varmoja, että aikasarjat ovat synkronoituja ja yhtä pitkiä, voidaan käyttää euklidista etäisyyttä, jonka avulla myös keskiarvoaikasarja saadaan laskettua (Keogh & Kasetty, 2002). Käytännössä millään tuntemattomalla aineistolla tällaista oletusta ei voida kuitenkaan tehdä. Aikasarjadataa ryhmiteltäessä onkin parempi käyttää ryhmän edustajana todellista aikasarjaa, joka minimoi etäisyyden kaikkiin muihin ryhmän aikasarjoihin. Tätä tietoalkiota kutsutaan ryhmän medoidiksi (Kaufman & Rouseeuw, 1990). Medoidi lasketaan luvun 4.1 kaavan (4.2) avulla.

4.3.2 Keskiarvoaikasarja

Tässä luvussa määritellään ryhmän medoidin ja dynaamisen aikasovituksen avulla laskettu keskiarvoaikasarja. Kuva 4.5 osoittaa erot kahden keskiarvoaikasarjan laskemiseen käytetyn menetelmän välillä. Kuvassa halutaan muodostaa mahdollisimman hyvin kahta aikasarjaa kuvaava keskiarvoaikasarja. Molemmissa aikasarjoissa on samanlainen hyppy, hieman eri ajan hetkillä. Amplitudin mukaan suoraan vertaamalla aikasarjoja keskenään saadaan keskiarvoaikasarjaksi kaksi puolitettua hyppyä. Dynaamisen aikasovituksen avulla muodostettu keskiarvoaikasarja on sen sijaan oikean muotoinen ja oikealla paikalla, koska muutos signaaleissa on sovitettu aika-akselin mukaan. Useamman kuin kahden aikasarjan keskiarvoaikasarjan muodostamiseksi käytetään hyväksi osituksen muodostamista varten laskettuja DTW-polkuja ryhmän alkioiden ja sen edustajan välillä.



Kuva 4.5: Keskiarvon muodostaminen amplitudin ja dynaamisen aikasovituksen avulla. (Niennattrakul & Ratanamahatana, 2007)

Algoritmin 4.3 avulla voidaan luoda keskiarvoaikasarja jokaiselle ryhmälle. Samantyylisiä algoritmeja käyttävät Niennattrakul & Ratanamahatana (2007) ja Abdulla & al. (2003). Kullekin ryhmälle muodostettava edustaja saadaan asettamalla uuden edustajan pituudeksi medoidin pituus ja laskemalla muodostettavan edustajan otoksien arvoiksi kuhunkin medoidin otokseen liittyvien otoksien keskiarvo. Kuhunkin medoidin otokseen liittyvät, ryhmään kuuluvien aikasarjojen otokset saadaan seuraamalla aiemmin laskettuja DTW-polkuja. Algoritmin aikavaativuus on keskimäärin $O(ML_{CA}N_A)$, missä M on ryhmien lukumäärä, L_{CA} on medoidien keskipituus ja N_A on ryhmiin keskimäärin kuuluvien aikasarjojen lukumäärä. Laskennallisen edustajan muodostaminen on huomattavasti hitaampaa kuin medoidin, mutta sen avulla saadaan parempi edustaja ryhmälle.

```
FormAverageTimeSeries(S,C,P,paths) {
    FOR j := 1 TO M DO {
        FOR k := 1 TO |cj| DO {
            amount := 0;
            sum := 0.0;
            FOR i = 1 TO Nj DO {
                (sum,amount) := SumRelatedFrames(Si,cj(k),paths);
            }
            cj(k) := sum / amount;
        }
    }
    RETURN C;
}
```

Algoritmi 4.3: Keskiarvoaikasarjan muodostaminen käyttäen laskettuja DTW-polkuja.

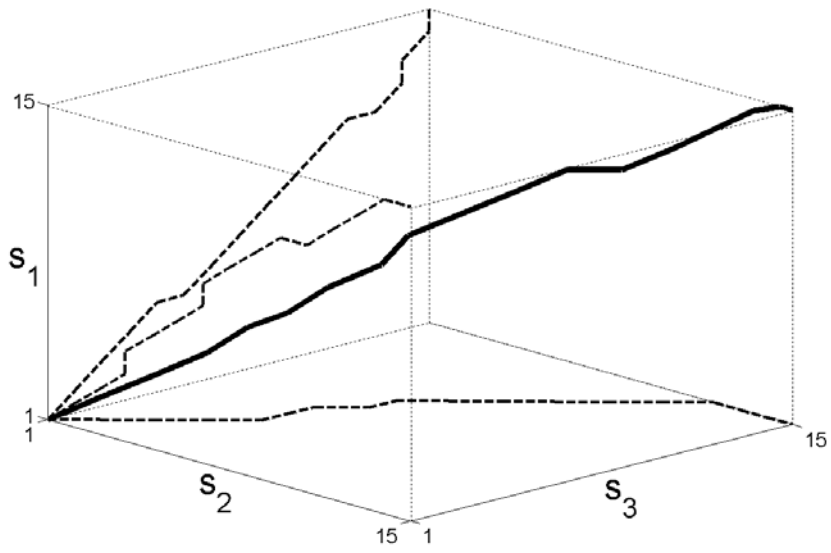
4.3.3 Ehdotettu optimointi keskiarvoaikasarjalle

Seuraavaksi esitetään uusi menetelmä approksimaatioprototyypin muodostamiselle eri pituisille aikasarjoille. Ennen varsinaisen ehdotetun menetelmän kuvaamista kuitenkin käsitellään hieman biologiaa, ja erityisesti bioinformatiikkaa, jossa esiintyy samankaltainen ongelma.

Bioinformatiikassa yleinen ongelma on kahden tai useamman rajallisesta aakkostosta koostuvan järjestetyn sarjan vertailu (Gusfield, 1997). Sarjat muodostuvat yleensä DNA-, RNA- tai proteiinirihmoista. Tätä monen sarjan sovitusta kutsutaan jatkossa nimellä *monen sekvenssin rinnastus* (*Multiple Sequence Alignment, MSA*). Ongelman ovat kuvanneet hyvin Gusfield (1997) sekä Carrillo & Lipman (1988). Carrillon & Lipmanin (1988) mukaan MSA on kriittinen tutkimuksen kohde muun muassa evoluutiotutkimuksissa ja proteiinien rakenne/tarkoitus -yhteyksien selvittämisessä. Ongelman tavoitteena on löytää niin sanottu *Steiner-konsensussarja*, joka minimoi etäisyyden rinnastettavien sekvenssien välillä (Gusfield, 1997). Konsensussarjaa ei välttämättä löydy jokaiselle monen sekvenssin rinnastukselle, sillä käytössä on rajallinen määrä merkkejä ja ’keskiarvomerkin’ muodostaminen saattaa olla mahdotonta. Optimaalisen Steiner-konsensussarjan löytäminen on NP-täydellinen ongelma äärellisellä aakkostolla käytettäessä niin sanottua Sum-of-Pairs -menetelmää (Just, 2001).

Aikasarjaryhmittelyn puolella tarkoituksena on löytää optimaalinen konsensusaikasarja, joka minimoi etäisyyden ryhmään kuuluviin todellisiin aikasarjoihin. Erona MSA:han on se, että käytettävä aakkosto on koko reaalilukujen avaruus eikä äärellinen joukko merkkejä. Intuitiivinen oletus on se, että kyseessä on myös NP-täydellinen ongelma, kuten Just (2001) on todistanut äärelliselle aakkostolle MSA:n tapauksessa. Kolmen aikasarjan ryhmän optimaalisen konsensusaikasarjan etsintää on havainnollistettu kuvassa 4.6. Kuvassa näkyvän kuution kannoille s_1-s_2, s_1-s_3 ja s_2-s_3 on piirretty kahden aikasarjan välinen dynaaminen aikasovitus katkoviivoilla. Kuution poikki diagonaalin lähellä kulkeva polku on puolestaan optimaalinen aikasarjasovitus näiden kolmen aikasarjan kesken. Useamman kuin kolmen aikasarjan optimaalisen sovituksen kuvaaminen ja toteutus on hankalaa. Tämän lisäksi, mikäli optimaalisen DTW-polun löytäminen kaikkien ryhmään kuuluvien aikasarjojen kesken on NP-täydellinen ongelma, aikavaativuus kasvaa eksponentiaalisesti aikasarjojen määrän kasvaessa, sillä jokaista aikasarjaa kohti optimaalisen ratkaisun hakuavaruus kasvaa

ulottuvuudella (N:lle aikasarjalle hakuavaruuden koko on vähintään L^N, missä L on lyhimmän ryhmään kuuluvan aikasarjan pituus). Ongelman haastavuuden takia ehdotettu konsensusaikasarjan muodostava menetelmä käyttää hyväksi luvussa 4.3.2 esitettyä algoritmia, jolla luodaan laskennallinen keskiarvoaikasarja ryhmälle.



Kuva 4.6: Optimaalisen DTW-polun etsiminen kolmelle aikasarjalle.

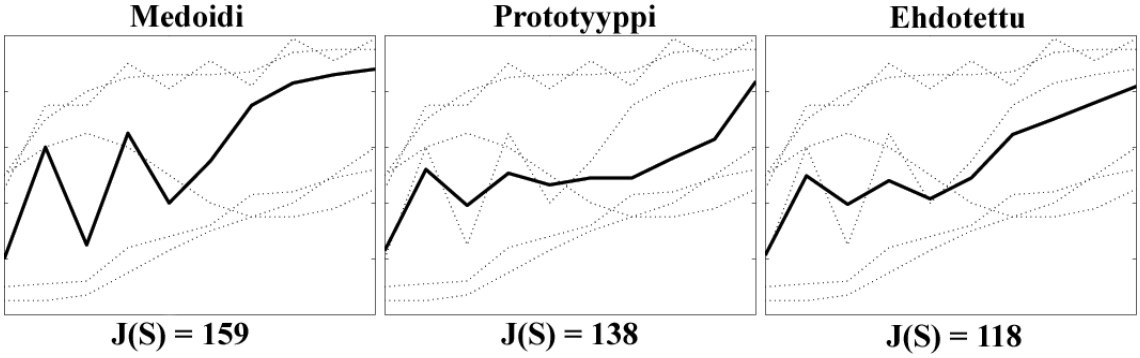
Ehdotettu menetelmä keskiarvoaikasarjan hienosäätämistä varten on kuvattu pseudokoodilla algoritmissa 4.4. Menetelmä pyrkii muokkaamaan luotua keskiarvoaikasarjaa siten, että se kuvaisi tarkemmin ryhmään kuuluvia aikasarjoja. Tämä tapahtuu sovittamalla ryhmän aikasarjat dynaamisen aikasovituksen avulla uudelleen laskennalliseen keskiarvoaikasarjaan. Sovituksen tuloksena saadaan uudet DTW-polut, joiden avulla voidaan laskea uusi keskiarvoaikasarja ryhmälle samaan tapaan kuin algoritmilla 4.3 lasketaan ensimmäinen keskiarvoaikasarja. Näitä kahta askelta toistetaan, kunnes keskiarvoaikasarjassa ei tapahdu muutosta. Ehdotetun keskiarvoaikasarjan laskemisen aikavaativuus on keskimäärin $O(MPN_A(KL_A^2 + L_{CA})) = O(MPN_AKL_A^2)$, missä M on mallin koko, P on suoritettavien optimointi-iteraatioiden lukumäärä, N_A on ryhmiin keskimäärin kuuluvien aikasarjojen lukumäärä, K on aineiston ulotteisuus, L_A on aikasarjojen keskipituus ja L_{CA} on medoidien keskipituus.

```
OptimizePrototypes(S,C,P) {

    FOR j = 1 TO M DO {
         previous := DBL_MAX;
         current := DBL_MAX – 1;
         WHILE (previous > current) {
              FOR i := 1 TO N_M DO {
                   paths[i] := DTW(s_i,c_j);
              }
              c_j = FormAverageTimeSeries(S_j,c_j,P_j,paths);
              previous := current;
              current := CalculateDistortion(S_j,c_j);
         }
    }
    RETURN C;
}
```

Algoritmi 4.4: Ehdotetun keskiarvoaikasarjan optimoinnin toteuttaminen pseudokoodilla.

Kuva 4.7 havainnollistaa käytettyjen edustajien välisiä eroja. Aineistona kuvasarjassa on kuuden aikasarjan ryhmä ja tavoitteena on muodostaa mahdollisimman hyvä edustaja tälle ryhmälle. Vasemmalla oleva medoidi on ryhmään kuuluva aikasarja, joka minimoi etäisyyden kaikkiin muihin aikasarjoihin. Medoidin avulla esitetyn ryhmän virheellisyys on J(S) = 159. Keskellä oleva prototyyppi on laskennallinen keskiarvoedustaja, joka on muodostettu laskettujen DTW-polkujen avulla. Kyseisen edustajan virheellisyys on J(S) = 138. Ryhmää parhaiten kuvaava laskennallinen aikasarja on laskettu algoritmin 4.4 avulla. Algoritmia on iteroitu yhteensä 10 kertaa keskiarvoaikasarjan hienosäätämiseksi. Kyseisen edustajan virheellisyys J(S) = 118 on käytetyistä edustajista pienin.



Kuva 4.7: Esimerkki vaihtoehtoisista edustaja-aikasarjoista kuudelle syöteaikasarjalle.

4.4 Toteutetut algoritmit

Aikasarjaryhmittelyä ja ryhmien edustajien vertailua varten toteutin kolme erilaista ryhmittelyalgoritmia aikasarjoille: k-medoids (Kaufman & Rouseeuw, 1990), RLS (Fränti & Kivijärvi, 2000) ja kokoava hierarkkinen algoritmi, joka käyttää keskiarvolinkitystä ryhmien välisen etäisyyden mittana (AL-AHC; kuvattu mm. Hirano & Tsumoto, 2005). K-medoids ja RLS ovat osittavia algoritmeja, jotka ovat muunnelmia McQueenin (1968) esittelemään k-meansiin.

4.4.1 K-medoids

Aikasarjojen ryhmittelyyn käytetty k-medoids -algoritmi on suora yleistys klassisesta k-means -algoritmista, jossa edustajina ja etäisyysmittana keskipisteiden ja euklidisen etäisyyden sijaan käytetään medoideja ja DTW-etäisyyttä. K-medoids -algoritmissa ryhmien edustajina käytetään todellisia tietoalkiota, jotka minimoivat etäisyyden kaikkiin muihin samaan ryhmään kuuluviin alkioihin (Kärkkäinen, 2006). Ryhmän edustajaa kutsutaan medoidiksi (Kaufman & Rouseeuw, 1990). Ryhmien medoidit lasketaan kaavan (4.2) mukaan. Kullekin tietoalkiolle lasketaan ositus samaan tapaan kuin k-means –algoritmissa.

$$c_j = argmin_{s(i)\in S(j)} \sum_{s(k)\in S(j)} D_{DTW}(s_i, s_k) \;\; \forall j \in [1..M] \tag{4.7}$$

$$p_i = argmin_{j=1}^{M} D_{DTW}(s_i, c_j) \;\; \forall i \in [1..N] \tag{4.8}$$

K-medoids tarvitsee syötteinä aineiston ***S***, ryhmien edustajat ***C*** sekä alkuosituksen ***P***. Tuloksena saadaan paikallisesti optimaalinen M-ositus aineistolle ***S*** sekä medoidiedustajat tälle ositukselle. Algoritmissa 4.5 on esitetty K-medoids pseudokoodina. Alkuositusta varten valitaan satunnaisesti aineistosta M aikasarjaa ryhmien alustaviksi edustajiksi. Yleinen, ja myös tässä tutkielmassa käytetty, menetelmä on valita satunnaisesti M aikasarjaa edustajiksi siten, että kukin aikasarja voi olla valittuna korkeintaan yhden ryhmän edustajaksi.

K-medoids -algoritmin ensimmäisessä askeleessa jokaiselle aikasarjalle etsitään lähin ryhmä DTW-etäisyyden avulla. Aikasarjan $\mathbf{s}_i$ lähin ryhmä löytyy laskemalla pienin DTW-etäisyys aikasarjan $\mathbf{s}_i$ ja edustajien $\mathbf{c}_1$, $\mathbf{c}_2$, …, $\mathbf{c}_M$ välillä. Tämä tapahtuu algoritmin 4.5 mukaisesti.

Vaiheen aikavaativuus on keskimäärin $O(NMKL_A^2)$, missä N on aikasarjojen lukumäärä, M on ryhmien lukumäärä, K on aineiston ulotteisuus ja L_A on aineiston aikasarjojen keskipituus.

```
assignClusters(S,C,P) {

    FOR i := 1 TO N DO {
         curr_min := DBL_MAX;
         FOR j:= 1 TO M DO {
              curr_DTW = D_DTW(s_i,c_j);
              IF (curr_min > curr_DTW) {
                   p_i := j;
                   curr_min := curr_DTW;
              }
         }
    }
    RETURN P;
}
```

Algoritmi 4.5: Aikasarjojen sovitus ryhmiin esitettynä pseudokoodina

Toinen vaihe K-medoids -algoritmissa on uusien edustajien etsintä. Jokaiselle ryhmälle etsitään paras edustaja algoritmin 4.6 avulla. Tämän vaiheen aikavaativuus on keskimäärin $O(M(N/M)^2KL_A^2) = O(N^2KL_A^2)$. Jokaista ryhmää kohti muodostetaan etäisyysmatriisi, joka koostuu ryhmään kuuluvien aikasarjojen DTW-etäisyyksistä kaikkiin muihin aikasarjoihin samassa ryhmässä. Uusi medoidi on se aikasarja, joka minimoi etäisyyden muihin aikasarjoihin.

```
assignMedoids(S,C,P) {

    FOR j := 1 TO M DO {
         sum[1..N_c(j)] := 0.0;
         currMin = DBL_MAX;
         FOR k := 1 TO N_c(j) DO {
              FOR i := 1 TO N_c(j) DO {
                   sum[k] := sum[k] + D_DTW(c_j(s_k),c_j(s_i));
              }
              IF (sum[k] < currMin) {
                   idx = k;
                   currMin := sum[k];
              }
         }
         c_j := idx;
    }
    RETURN C;
}
```

Algoritmi 4.6: Medoidin laskeminen annetun osituksen avulla.

K-medoids -algoritmin ylin taso on kuvattu pseudokoodilla algoritmissa 4.7. Algoritmissa vuorotellaan optimaalisen osituksen ja optimaalisen edustajan laskevia funktioita. Täten k-medoids algoritmin keskimääräinen aikavaativuus on $O(IKL_A^2(NM + N^2/M) + N)) = O(IKL_A^2N^2)$, missä I on k-medoids -iteraatioiden lukumäärä, K on mallin koko, L_A on aineiston aikasarjojen keskipituus ja N on aineiston koko eli aikasarjojen lukumäärä. Algoritmin aikavaativuus huonoimmassa tapauksessa on $O(IKL^2N^2)$, missä L on aineiston pisimmän aikasarjan pituus.

```
K-Medoids(S, C, P) {
    current = DBL_MAX;
    WHILE (previous > current) {
        previous := current;
        P' := AssignClusters(S,C,P);
        C' := AssignMedoids(S,C,P');
        current := CalculateDistortion(S,C',P');
        P := P';
        C := C';
    }

    RETURN (C,P);
}
```

Algoritmi 4.7: K-medoids esitettynä pseudokoodilla.

4.4.2 Satunnaistettu paikallishaku -algoritmi

Satunnaistettu paikallishaku -algoritmi tunnetaan myös nimellä RLS-algoritmi. Se on muunnelma klassisesta k-means -algoritmista. RLS pyrkii välttämään juuttumisen paikalliseen optimiin muuttamalla osituksen rakennetta radikaalisti yhden ryhmän osalta ja tämän jälkeen etsimällä paikallisesti optimaalisen osituksen uudelleenryhmitetyille edustajille k-meansin avulla. Mikäli saatu ositus on parempi kuin aiemmin tunnettu, niin se tallennetaan parhaaksi löydetyksi ositukseksi; muutoin valitaan uudelleen vaihdettava edustaja aiemmin löytyneestä parhaasta osituksesta. Käytännössä algoritmi toimii yritys ja erehdys -periaatteella, kunnes se löytää paremman osituksen.

Vaihdettava edustaja valitaan algoritmin nimen mukaisesti satunnaisesti. Satunnaisesti valittavan edustajan etu on toteutuksen yksinkertaisuus ja nopeus. Vaihdettavan edustajan valinta voidaan toteuttaa myös deterministisesti, mutta tämä lisää algoritmin aikavaativuutta.

Tämän lisäksi deterministisen version huonona puolena on juuttuminen paikalliseen optimiin tilanteessa, jossa deterministisesti valittu edustajanvaihto ei enää tuota parannusta ryhmittelyyn. Yhden medoidin satunnainen vaihtaminen voidaan toteuttaa algoritmin 4.8 avulla. Algoritmissa valitaan satunnaisesti vaihdettava edustaja ja ryhmä, jonka alueelle se vaihdetaan. Tämän jälkeen valitaan kohderyhmästä tietty aikasarja, joka toimii uutena edustajana vaihdettavalle ryhmälle. Jos medoidin sijasta käytetään laskennallista aikasarjaa edustajana, niin se täytyy laskea uudelleen suoritetun vaihdon jälkeen.

```
SwapCluster(S,C,P) {
    swapped = Random(1,M);
    DO
        destination = Random(1,M);
    WHILE (swapped == destination);
    DO
        target = Random(1,N_destination);
    WHILE (target == C_destination);

    C_swapped = S_destination(target);
}
```

Algoritmi 4.8: Medoidin satunnainen vaihtaminen esitettynä pseudokoodilla.

RLS on esitetty pseudokoodilla algoritmissa 4.9. Algoritmin aikavaativuus on keskimäärin $O(RIKNL_A^2)$ ja pahimmassa tapauksessa $O(RIKNL^2)$, missä R on RLS-iteraatioiden lukumäärä, I on k-medoids -iteraatioiden lukumäärä, K on mallin koko, N on aineiston koko, L_A on aineiston aikasarjojen keskipituus ja L on aineiston pisimmän aikasarjan pituus.

```
RLS(S, C, P, R) {

    (C, P) := K-Medoids(S,C,P);
    best = CalculateDistortion(S,C,P);

    FOR r := 1 TO R DO {
        C' := SwapCluster(S,C,P);
        (C',P') := K-Medoids(S,C',P);
        current = CalculateDistortion(S,C',P');
        IF (current < best) {
            best := current;
            (C,P) := (C',P');
        }
    }

    RETURN (C,P);
}
```

Algoritmi 4.9: Satunnaistettu paikallishaku esitettynä pseudokoodilla.

4.4.3 Kokoava hierarkkinen algoritmi

Toteutettu hierarkkinen, kokoava algoritmi (AHC) käynnistyy alkuosituksesta, jossa jokainen aikasarja muodostaa oman ryhmänsä. Ryhmiä yhdistellään keskiarvolinkityksen mukaan, jolloin yhdistettäväksi pariksi valitaan ryhmät, joiden aikasarjojen DTW-etäisyyksien summat kaikkiin toisen ryhmän aikasarjoihin ovat pienimmät. Ennen kuin ryhmiä voidaan yhdistää, niin tarvitaan taulukko, jossa on jokaisen alkion etäisyys jokaiseen toiseen alkioon. Tämän luonti DTW:n avulla on aikavaativuudeltaan keskimäärin $O(½KL_AN^2)$, missä K on aineiston ulotteisuus, L_A on aikasarjojen keskipituus ja N on aineiston koko. Kahden yhdistettävän ryhmän valinta toteutetaan algoritmin 4.10 avulla ja vaiheen aikavaativuus on $O(½N^2) = O(N^2)$. Valittujen ryhmien yhdistäminen tapahtuu algoritmin 4.11 avulla. Sen aikavaativuus on $O(N_j)$, missä N_j on yhdistettävään ryhmään kuuluvien aikasarjojen lukumäärä. Ryhmien yhdistelyä jatketaan näin, kunnes on saavutettu käyttäjän antama määrä ryhmiä. Kokonaisuudessaan AHC on kuvattu pseudokoodina algoritmissa 4.12. Sen aikavaativuus on $O(½KL_AN^2 + (N-M)(N^2+N/M)) = O(N^3)$.

```
PickShortestDistance(D) {
    FOR i := 1 TO N DO {
        FOR j := 1 TO N – i – 1 DO {
            IF (currMin > D[i][j]) {
                currMin := D[i][j];
                receiver := i;
                donater := n – j – 1;
            }
        }
    }
    RETURN ci,cj;
}
```

Algoritmi 4.10: Toisiaan lähimpinä olevien ryhmien valinta esitettynä pseudokoodilla.

```
MergeClusters(c_i,c_j) {
    FOR k := 1 TO N_c(j) DO {
        p_k := i;
    }
    N_c(i) := N_c(i) + N_c(j);
    N_c(j) := 0;
    c_j := NULL;

    RETURN (C,P);
}
```

Algoritmi 4.11: Toisiaan lähimpinä olevien ryhmien yhdistäminen esitettynä pseudokoodilla.

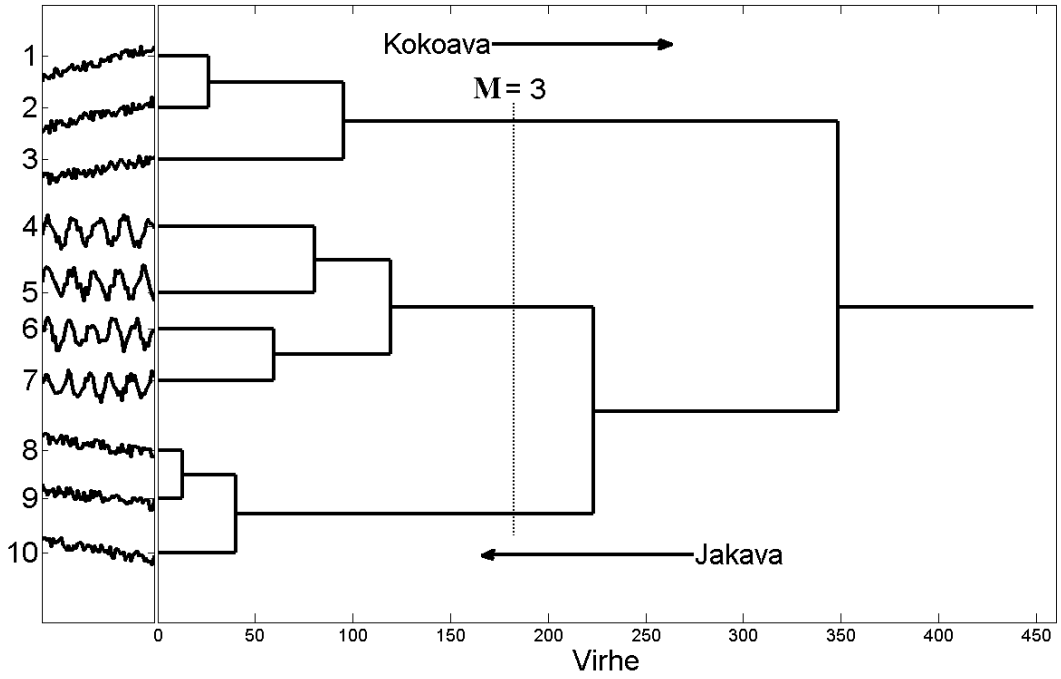
```
AHC(S,C,P,target) {

    D := CalculateClusterDistances(S,C,P);
    WHILE (C_M > target) {
        (c_i,c_j) := PickShortestDistance(D);
        (C,P) := MergeClusters(c_i,c_j);
        C_M := C_M – 1;
        D := UpdateClusterDistances(S,C,P);
    }

    RETURN (C,P);
}
```

Algoritmi 4.12: Kokoava hierarkkinen ryhmittely esitettynä pseudokoodilla.

Kokoavan hierarkkisen algoritmin lisäksi testiajoja suoritettiin algoritmien yhdistelmällä, jossa hierarkkista algoritmia käytettiin muodostamaan alkuositukset k-medoids -algoritmille. Tästä yhdistelmästä käytetään jatkossa lyhennystä AHC-KM. Hierarkkisen algoritmin avulla saatiin luotua k-medoidsille satunnaisesti valittuja alkuosituksia parempia alkuosituksia. RLS-algoritmia ei yhdistetty hierarkkisen algoritmin kanssa, sillä tällöin aineistolle luotu hierarkkinen rakenne särkyy enemmän. Kuva 4.8 esittää hierarkkisen ryhmittelyn avulla luotua ryhmittelypuuta 10 aikasarjan aineistolle. Aikasarjat ovat synteettistä dataa ja ne muodostavat kolme ryhmää. Ensimmäiseen ryhmään kuuluvissa aikasarjoissa on kasvava suuntaus, toisen ryhmän aikasarjoissa on havaittavissa syklinen rakenne ja kolmannen ryhmän aikasarjat sisältävät laskevan suuntauksen.



Kuva 4.8: Hierarkkisen ryhmittelyn havainnollistaminen aikasarjoille ryhmittelypuun avulla

4.5 Dynaamisen aikasovituksen parantaminen

Vaikka dynaaminen aikasovitus aikasarjoille on paljon vankempi etäisyyden mitta kuin euklidinen etäisyys, niin se on menetelmänä hidas (Fu & al., 2005). Kuten luvussa 4.2 todettiin, dynaamisen aikasovituksen aikavaativuus kahden K-ulotteisen aikasarjan optimaalisen sovituksen löytämiseksi on $O(KL_1L_2)$, missä L_1 ja L_2 ovat sovitettavien aikasarjojen pituudet. Sovitusta voidaan nopeuttaa lisäämällä sovituspolulle asetettavia rajoitteita, kuten kaltevuusehto tai sovitusikkuna. Rajoitteet voivat olla globaaleja tai paikallisia (Salvador & Chan, 2004).

Dynaamisen aikasovituksen lisäksi on olemassa myös muita etäisyysmittoja, jotka pyrkivät vertailemaan aikasarjojen samankaltaisuutta. Dynaaminen aikasovituksen on rajoittunut numeraalisiin aikasarjoihin ja lisäksi algoritmi vaatii, että jokainen otos on sovitettava toiseen aikasarjaan. *Pisin yhteinen alisekvenssi* (*Longest Common Subsequence*, *LCSS*) on etäisyysmitta, joka toimii myös symbolisilla aikasarjoilla eikä se DTW:n tavoin vaadi jokaisen pisteen sovittamista. Pisteiden ylitys on toivottavaa, jos tiedetään että kahden

aikasarjan pisteet eivät ole riippuvaisia toisistaan (Gaudin & al., 2006). Dynaaminen aikasovitus valittiin etäisyysmitaksi, koska se soveltuu eri pituisten aikasarjojen vertailuun. Tämän lisäksi valintaan vaikutti sen yleisyys – Gaudin & al. (2006) toteavat, että DTW on yleisimmin käytetty mitta, joka pystyy sovittamaan aikasarjoja toisiinsa nähden muokkaamalla aika-akselia.

Yleensä dynaamista aikasovitusta optimoitaessa keskitytään nopeuttamaan algoritmia kiinnittämättä huomiota sovituksen tarkkuuteen (Keogh & Pazzani, 2001). Dynaamisen aikasovitus voi kuitenkin aiheuttaa myös ei-toivottuja sovituksia, joissa algoritmi pyrkii korjaamaan havaintoasteikolla tapahtuvia muutoksia sovittamalla aika-akselia, vaikka siirtymä aikasarjojen välillä tapahtuu havaintojen arvoissa. Parannusehdotuksena Keogh ja Pazzani (2001) ehdottavat algoritmia, joka pyrkii parantamaan sovituksen laatua arvioimalla aikasarjoja niiden derivaatan mukaan.

Yleisin ja helpoiten toteutettava tapa nopeuttaa dynaamista aikasovitusta on asettaa lisärajoitteita polulle ja täten pienentää DTW-matriisia varten suoritettavien laskutoimitusten lukumäärää. Haku nopeutuu, sillä haettava alue on pienempi mutta vaarana on, että rajoitamme hakualuetta liikaa jolloin optimaalista sovitusta ei voida suorittaa aikasarjojen välille. Suosituimmat rajoitteet ovat esitelleet Sakoe & Chiba (1978) ja Itakura (1975). Rajoitteet nopeuttavat aikasovitusta ainoastaan vakiokertoimen verran ja ne ovat alttiita antamaan virheellisiä sovituspolkuja, mutta ne toimivat hyvin mikäli sovellusalue on sellainen, jossa sovitukset kulkevat lähellä etäisyysmatriisin diagonaalia. Nopein toteutettu DTW-algoritmi on FastDTW (Salvador & Chan, 2004). Algoritmi kykenee sovittamaan lineaarisessa ajassa kaksi aikasarjaa toisiinsa ilman virhesovituksia. Algoritmi käyttää hyväksi karkeuttamista, projektiota ja jalostamista. Karkeuttamisessa aikasarjat kuvataan pienemmällä määrällä havaintoja. Projektion tarkoituksena on löytää optimaalinen polku pienemmällä tarkkuudella ja käyttää tätä polkua arvauksena, kun tarkkuutta lisätään. Lopuksi jalostuksessa hienosäädetään sovituspolkua paikallisilla muutoksilla.

5 MATERIAALIT JA MENETELMÄT

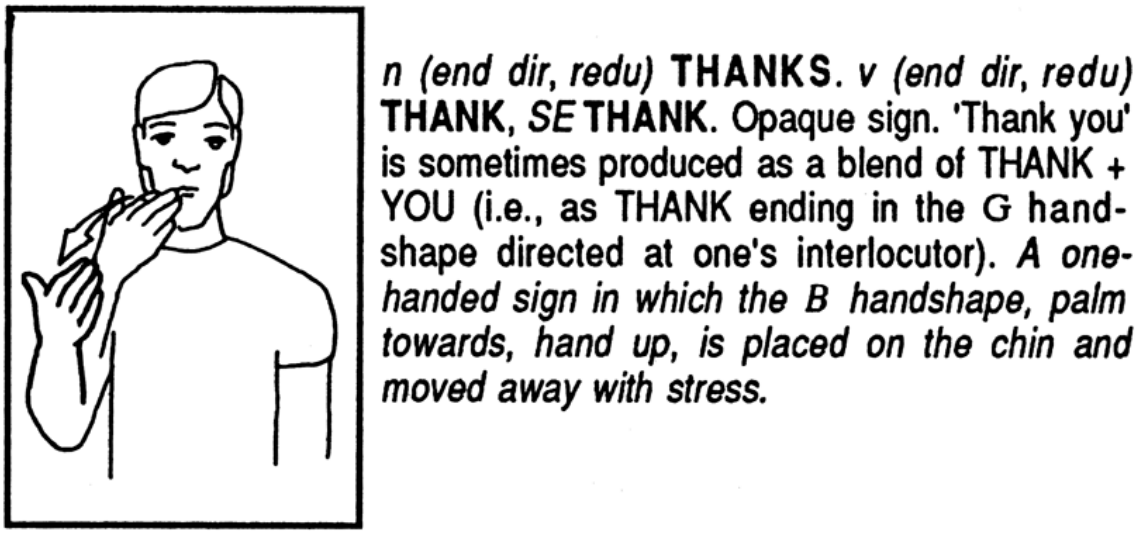
5.1 Ryhmiteltävät aineistot

Tässä tutkielmassa ryhmiteltävinä aineistoina käytetään yksi- tai moniulotteista aikasarjadataa. Kaikki aineistot koostuvat tasaisin intervallein mitatuista numeraalisista havainnoista. Taulukkoon 5.1 on koottu keskeiset tiedot eri aineistoista. Phs -aineistoa lukuunottamatta kaikki aineistot ovat ladattavissa UCI:n tietokantavarastosta (Asuncion & al., 2007).

	Auslan	Char	Japanese	Phs	Synthetic
Tyyppi	Viittomakieli	Kirjoitus	Puhe	Puhe	Synteettinen
Alkiot (N)	2565	925	640	2255	600
Ryhmät (M)	95	59	9		6
Ulotteisuus (K)	22	2	12	12	1
Minimipituus	45	15	7	5	60
Keskipituus	57	54	16	14	60
Maksimipituus	136	155	29	39	60

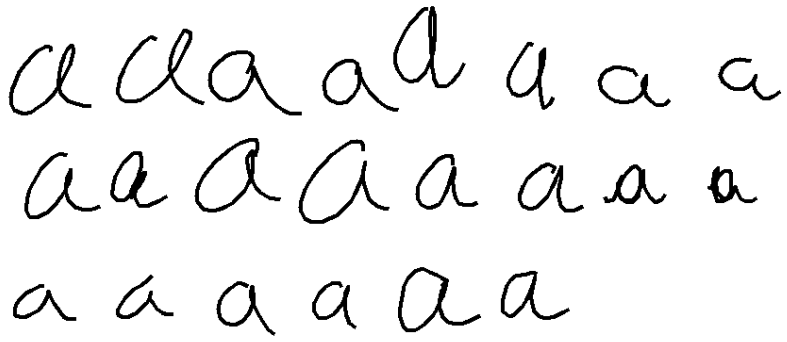
Taulukko 5.1: Ryhmiteltävät aikasarja-aineistot.

Auslan-aineisto sisältää australialaisen viittomakielen viittomismerkkejä. Tiedon keräämiseen on käytetty kahta liikkeentunnistuskäsinettä. Molemmilta käsiltä on kerätty niiden paikat x-, y- ja z-koordinaatistoissa, sekä käsien *kierteisyys*- ja *kaltevuuskulmat* (*pitch*, *roll*, *yaw*). Tämän lisäksi myös jokaisen sormen koukistuneisuus on mukana aineistossa. Yhteensä ulottuvuuksia on 2*(3+3+5) = 22. Aineisto on kerätty yhdeltä viittojalta yhdeksän viikon aikana. Erilaisia viittomakielen merkkejä on 95 kappaletta, joista jokaisesta on kerätty kolme näytettä jokaisella viikolla. Täten aikasarjoja on aineistossa yhteensä 95*3*9 = 2565 kappaletta. Aikasarjojen pituudet vaihtelevat 45 ja 136 otoksen välillä, keskipituuden ollessa 57 otosta. Kuvassa 5.1 esitetään kuinka eräs aineistoon kuuluva merkki viitotaan viittomakielellä.



Kuva 5.1: Sanan ’kiitos’ viittominen australialaisella viittomakielellä. (Kadous, 2002)

Character-aineisto koostuu käsinkirjoitetuista kirjain- ja numeromerkeistä. Kirjaimet ovat ISO/IEC 646 -standardissa määriteltyjä latinalaisia aakkosia ilman diakriittisiä merkkejä, eli kirjaimet a-z ja A-Z. Numerot ovat arabialaiset numerot väliltä 0-9. Alkuperäinen *UJI Pen Characters* -aineisto sisälsi jokaiselta henkilöltä kaksi näytettä jokaista pientä ja suurta kirjainta sekä numeroa kohti. Alkuperäisen aineiston koko oli siis 11*2*(2*26+10) = 1364 aikasarjaa. Tästä aineistosta karsin kaikki merkit, joiden piirtämiseen on käytetty useampaa kuin yhtä *vetoa* (*stroke*), sillä dynaaminen aikasovitus ei osaa huomioida useasta aikasarjasta koostuvia merkkejä. Aikasarjan muodostama kynänveto määritellään alkavaksi, kun kynä asetetaan lukupinnalle, ja loppuvaksi, kun kynä nostetaan lukupinnalta. Haluttaessa luoda malleja useista vedoista koostuville aikasarjoille tulisi etäisyysmittaa muokata tai luoda erilliset alimallit jokaiselle merkille (Vuori & Laaksonen, 2002). Karsitun aineiston koko on 925 aikasarjaa. Tämän uuden aineiston merkistö koostuu samoista merkeistä kuin alkuperäisen aineiston, pois lukien x, X ja 7. Näille kolmelle merkille ei ollut 22 näytteen joukossa yhtään yhdellä vedolla tehtyä näytettä. Aikasarjojen otokset koostuvat kahdesta ulottuvuudesta: x- ja y-koordinaateista. Aikasarjoissa on vaihteleva määrä otoksia; lyhin aikasarja on 15 otosta ja pisin on 155 otosta. Aineiston aikasarjojen keskipituus on noin 54 otosta. Kuva 5.2 sisältää kaikki aineistossa olevat näytteet merkille pienelle a-kirjaimelle. Kuvasta käy ilmi myös kuinka aineistoa ei ole normalisoitu, jonka takia kirjaimet ovat eri kokoisia ja sijaitsevat eri korkeudella muihin omalla rivillä oleviin merkkeihin nähden.



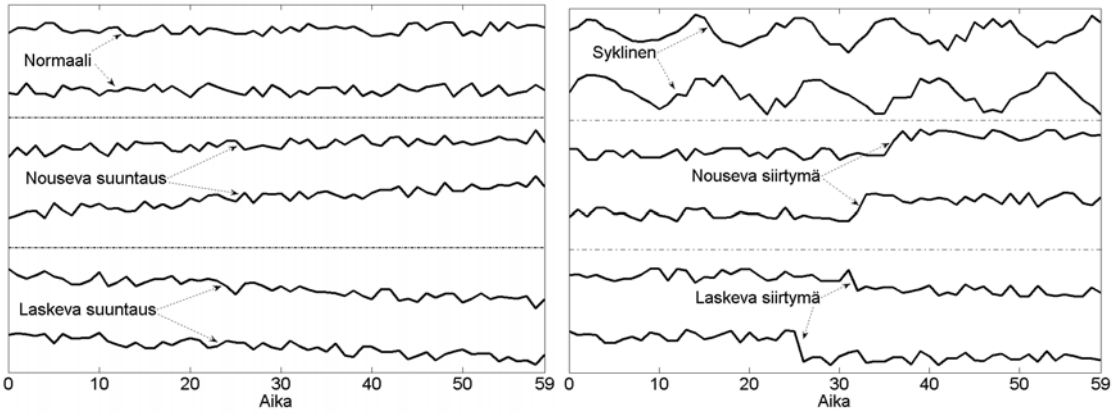
Kuva 5.2: Character-aineiston 22 näytettä a-kirjaimesta.

Japaninkielinen puheaineisto *Japanese* koostuu yhdeksän japania puhuvan mieshenkilön lausahtamista japaninkielisistä /ae/-foneemeista. Jokaiselle lausahdukselle on suoritettu 12-asteinen lineaarinen ennustusanalyysi, jonka avulla on saatu 12-ulotteisia aikasarjoja. Yksi lausahdus muodostaa aikasarjan, jonka pituus vaihtelee väliltä 7-29. Aikasarjojen keskipituus on 16 otosta. Aikasarjoja aineistossa on yhteensä 640 kappaletta ja se on alun perin suunniteltu käytettäväksi luokitteluun, mistä johtuen aineisto on jaettu opetus- ja testausaikasarjoihin. Tätä tutkielmaa varten aineistoa käsitellään yhtenä kokonaisuutena.

Phs-aineisto on hyvin samankaltainen puhetta sisältävä aineisto kuin Japanese. Kyseinen aineisto koostuu suomenkielisistä /aa/-foneemeista, mutta puhujien lukumäärää ei tunneta etukäteen. Aineisto koostuu 2255:stä aikasarjasta, jotka on saatu irrottamalla piirrevektoreita lausahduksista, aivan kuten Japanese aineistossakin. Lyhin aikasarja on pituudeltaan 5 havaintoa ja pisin puolestaan 39 havaintoa. Aineiston aikasarjojen keskipituus on 14 havaintoa. Kyseiselle aineistolle ei ole saatavissa luokkanimikkeitä, joten algoritmien toimivuutta luokitteluvirheen osalta ei testata tällä aineistolla.

Synthetic-aineistossa on 600 synteettisesti luotua, samanpituista aikasarjaa. Aineistossa on kuusi toisistaan eroavaa luokkaa: normaali, syklinen, nouseva suuntaus, laskeva suuntaus, nouseva siirtymä ja laskeva siirtymä. Aineistossa on kustakin luokasta 100 aikasarjaa. Normaalin luokan aikasarjat sisältävät satunnaista dataa, ilman havaittavissa olevaa kuviota. Syklisen luokan aikasarjoista löytyy toistuva kuvio, mikä ei kuitenkaan ole aina täsmällisesti samanlainen arvo- eikä aika-asteikolla. Nousevan suuntauksen aikasarjoissa on havaittavissa kasvava trendi, samoin kuin laskevan suuntauksen aikasarjoissa on laskeva trendi.

Aikasarjoissa, joissa on nouseva siirtymä, tapahtuu aikasarjassa satunnaisella kohdalla selvä hyppäys ylöspäin arvoasteikolla. Laskevan siirtymän aikasarjoissa hyppäys on puolestaan alaspäin. Yksikään aikasarja tai sen osa ei ole kopio toisesta aikasarjasta tai sen osasta. Kukin aikasarja koostuu 60 otoksen mittaisesta, yksi-ulotteisesta reaalilukuvektorista. Kuvassa 5.3 on jokaisesta aineiston luokasta mukana kaksi esimerkkiaikasarjaa.

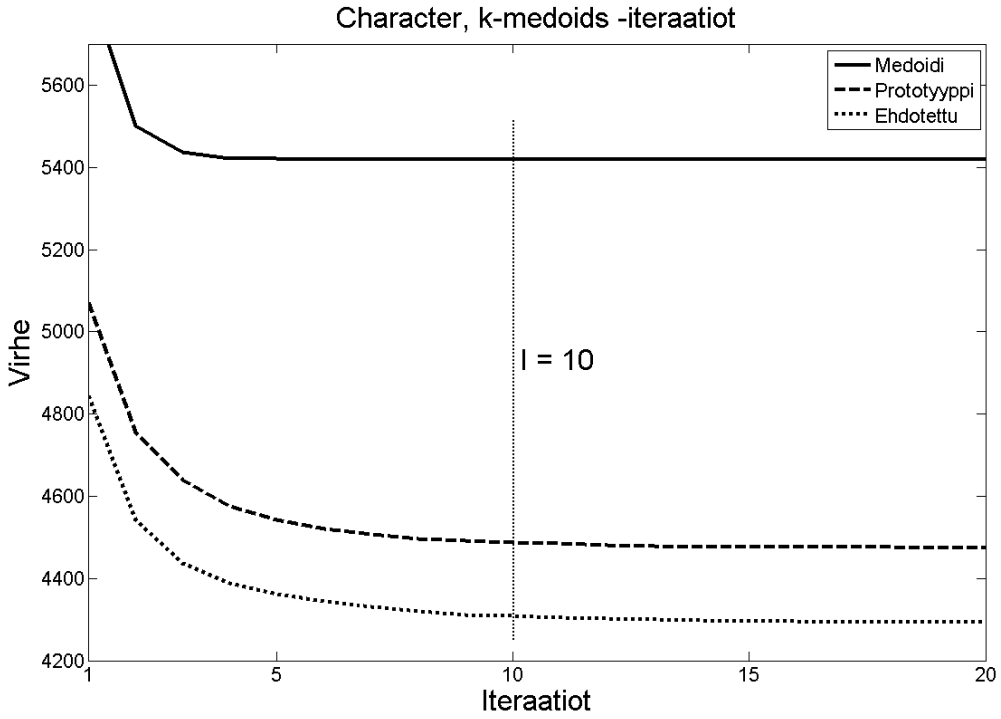


Kuva 5.3: Esimerkkijoukko synteettisesti luodusta aikasarja-aineistosta.

5.2 Testausmenetelmät

Testauksessa käytettyihin muuttujiin kuuluivat ryhmiteltävät aineistot, toteutetut algoritmit, mallien koot sekä k-medoids ja RLS-iteraatioiden lukumäärät. Menetelmiä arvioidaan optimoitavan ryhmittelykriteerin, saavutetun tehokkuuden, k-medoids –algoritmin konvergoitumisnopeuden ja luokitteluvirheen avulla.

Optimoitava ryhmittelykriteeri lasketaan kaavan (4.3) avulla. Käytettyjen edustajien konvergoitumisnopeutta puolestaan arvioidaan kuvaajista ja k-medoids -iteraatioille asetetaan yläraja kuvaajien nojalla, sillä konvergoituminen kestää liian kauan. Asetetut iteraatioiden ylärajat ovat aineistokohtaisia ja rajat ovat saatu testaamalla aineistojen ja eri edustajien konvergoitumisnopeutta. Kuvassa 5.4 on esimerkki, kuinka iteraatiomäärä on asetettu aineistolle Character. Kuvasta on silmämääräisesti arvioitu iteraatiolukumäärä I, jonka jälkeen k-medoids ei saavuta merkittävää parannusta, vaikka iteraatioita lisättäisiin. Jokaiselle aineistolle tehdyt kuvaajat on sisällytetty liitteeseen 1.



Kuva 5.4: Konvergoitumisnopeudet edustajien välillä aineistolle Character.

Luokitteluvirhe voidaan laskea kaavan (5.1) avulla. Luokitteluvirhe saa arvoja väliltä 0-100%, missä 0 on ryhmien täydellinen vastaavuus luokkanimikkeiden kanssa ja 100% puolestaan osoittaa, ettei mikään alkio ole samassa ryhmässä minkään samaan luokkaan kuuluvan alkion kanssa. Luokitteluvirheen laskeminen on esitetty pseudokoodina algoritmissa 5.1. Ositus ***P*** on laskettu ositus ja ositus ***CP*** on oikea ositus, joka saadaan luokkanimikkeiden avulla.

$$\mathrm{CE}(P, CP) = \left(\frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mathrm{Same}(\mathrm{Same}(p_i, p_j), \mathrm{Same}(cp_i, cp_j))}{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} 1} \right) \times 100\% \quad (5.1)$$

,

missä $\mathrm{Same}(x, y) = \begin{cases} 0 \Leftarrow x = y \\ 1 \Leftarrow x \neq y \end{cases}$ määrää onko pari oikea vai virheellinen.

```
ClassificationError(P,CP) {
    error := 0;
    total := 0;
    FOR i := 1 TO N DO {
        FOR j := i+1 TO N DO {
            correct := (cp_i == cp_j);
            calced := (p_i == p_j);

            IF (correct != calced) {
                error := error + 1;
            }
            total := total + 1;
        }
    }

    RETURN error / total;
}
```

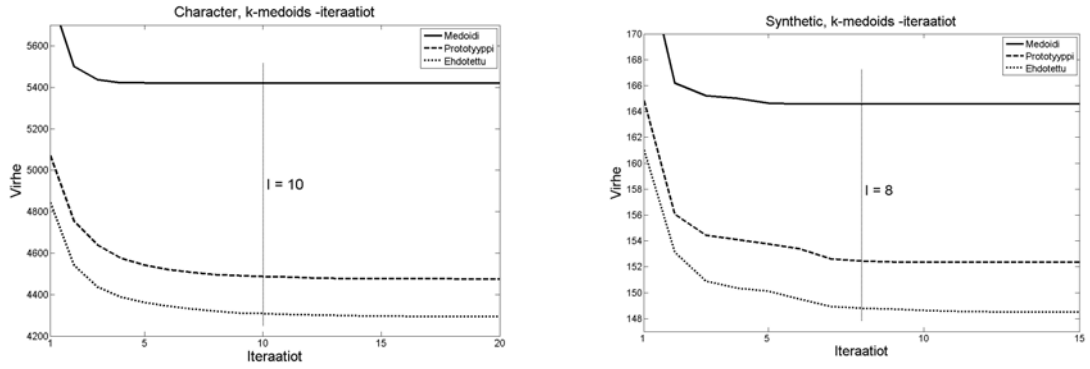
Algoritmi 5.1: Luokitteluvirheen laskeminen esitettynä pseudokoodilla.

Saavutetut testaustulokset ja niitä esittävät kuvaajat ovat 10 testiajon keskiarvoja. Tuloksista raportoidaan myös niiden keskihajonta. Toistamalla ryhmittelyjä samoilla parametreilla pyritään pienentämään satunnaistekijöiden vaikutusta testaustuloksiin. Näitä satunnaistekijöitä ovat muun muassa ryhmittelyssä käytetyn alkuosituksen hyvyys ja käytössä olevien laskentaresurssien määrä. Algoritmeille annettavat alkuositukset puolestaan ovat satunnaisesti muodostettuja, mutta samoja kaikille algoritmeille ja edustajille. Tällä pyrin takaamaan, että tulokset ovat vertailukelpoisia keskenään. Testit on pyritty suorittamaan aikoina, jolloin prosessoreilla ei ole ollut ylimääräistä kuormaa. Testiajot on suoritettu laskentaklusterilla, joka koostuu neljästä 2.8 GHz kellotaajuudella toimivasta Intel Pentium 4 -prosessorista. Laitteisto sisältää 2 gigatavua keskusmuistia ja käyttöjärjestelmänä toimii GNU/Linux (kernel: 2.4.22-openmosix3).

6 TESTAUSTULOKSET

6.1 Konvergoituminen

Konvergenssillä tarkoitetaan pistettä, jonka jälkeen k-medoids -iteraatioiden kasvattaminen ei paranna ositusta. Käytetyn edustajan vaikutus k-medoidsin konvergoitumisnopeuteen on olennaista, sillä mitä nopeammin algoritmi konvergoituu, sitä vähemmän tarvitaan iteraatioita ja sitä nopeammin algoritmi suoriutuu tehtävästään. K-medoidsin nopeus vaikuttaa myös RLS- ja AHC-KM –algoritmien nopeuteen, sillä k-medoidsia käytetään ositusten hienosäätöön kyseisissä menetelmissä. Kuva 6.1 havainnollistaa konvergoitumista aineistoilla Character ja Synthetic. Kuvista on havaittavissa, kuinka kuvaajien muodot muistuttavat paljon toisiaan, mutta laskennallisten edustajien konvergoitumisnopeus on hitaampi kuin medoidin. Pystysuora katkoviiva osoittaa pisteen, joka on valittu k-medoids -iteraatioiden lukumääräksi kyseessä olevalle aineistolle. Olennaista on myös, kuinka samalla iteraatiomäärällä ehdotettu menetelmä edustajan laskemiseksi tuottaa aineistosta riippumatta aina paremman edustajan kuin prototyyppi, joka puolestaan tuottaa aina paremman edustajan kuin medoidi. Erot medoidin ja laskennallisten keskiarvoedustajien välillä konvergoitumisnopeudessa ovat huomattavat, mutta erot prototyypin ja ehdotetun menetelmän välillä eivät ole niin merkittäviä. Saadut tulokset ovat samansuuntaisia kaikilla testiaineistoilla, ainoastaan konvergoitumisnopeudessa on havaittavissa eroja aineistojen välillä.



Kuva 6.1: K-medoidsin konvergoituminen eri edustajilla aineistoilla Character ja Synthetic.

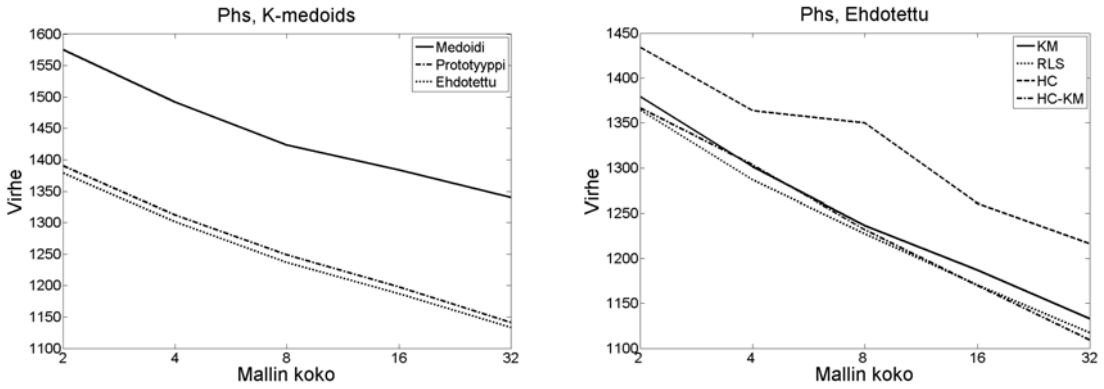
6.2 Ryhmittelyvirhe

Ryhmittelyvirheen avulla voidaan tarkastella, kuinka hyvin ryhmittelyalgoritmin avulla aineistolle saatu ositus optimoi ryhmittelykriteeriä. Muuttujana käytetään haluttua mallin kokoa M. Erikoistilanteessa, jossa mallin koko on yksi, jokainen aikasarja kuuluu samaan ryhmään. Oletuksena on, että mallin koon kasvaessa ryhmittelyvirhe pienenee, sillä ryhmään kuuluvien aikasarjojen etäisyys ryhmän edustajaan myös pienenee keskimääräisesti. Jos mallin koko on aineiston koko N, niin jokainen aikasarja muodostaa oman ryhmänsä. Tällöin ryhmittelyvirhe on nolla, mutta ryhmittely ei ole mielekäs, sillä ryhmiä on yhtä monta kuin aikasarjoja eikä ositus anna lisätietoa aineiston rakenteesta.

Todellisuudessa haluamme löytää mallin koon, joka mahdollistaa luonnollisten ryhmien muodostuksen aineistosta ja jossa M on paljon pienempi kuin N. Monet ryhmittelyalgoritmit vaativat, että niille annetaan parametrina haluttu mallin koko. Tämä on kuitenkin ongelmallista, sillä käyttäjä ei useinkaan tiedä tuntemattomasta aineisto kuinka monesta ryhmästä se muodostuu. Yleinen ratkaisu ongelmaan on ryhmitellä aineisto useasti vaihdellen mallin kokoa ja etsiä taitekohtaa, jossa mallin koon kasvattaminen ei tuo enää merkittävää parannustaositukselle (Kärkkäinen, 2006). Ryhmittelyvirheen avulla vertailen erilaisia ryhmittelymenetelmiä, mutta oletan käyttäjän tietävän oikean mallin koon. Mallien koot kiinnitetään myöhemmin suoritettavia tehokkuusmittauksia varten aineistokohtaisesti.

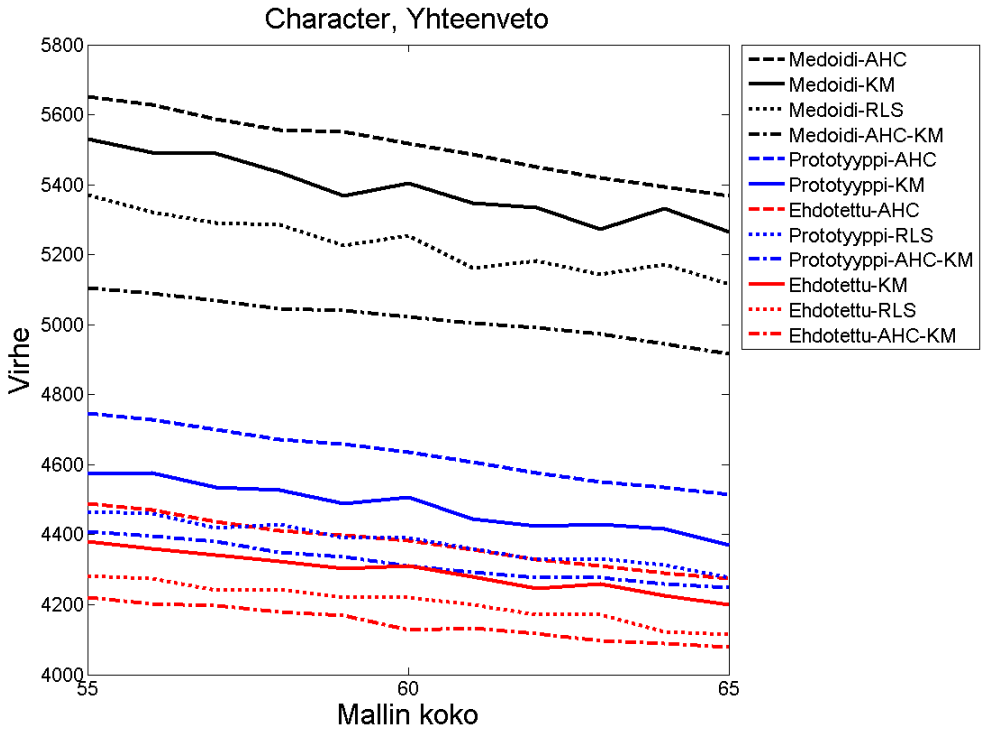
Kuva 6.2 havainnollistaa käytetyn edustajan ja ryhmittelyalgoritmin vaikutusta ryhmittelyvirheeseen aineistolla Phs. Kaikilla aineistoilla saadut tulokset edustajan vaikutuksesta ovat samansuuntaisia. Testausajojen perusteella virheellisyys pienenee aineistosta, mallin koosta ja ryhmittelyalgoritmista riippumatta aina, kun medoidin sijaan käytetään laskennallista keskiarvoaikasarjaa. Saavutetun parannuksen suuruus on riippuvainen aineistosta, mallin koosta ja käytetystä algoritmista. Pienin testaustuloksissa saavutettu parannus virheellisyyteen vaihdettaessa medoidista prototyyppiin tapahtui Synthetic-aineistolla; parannus oli noin 6 %. Sen sijaan muilla aineistoilla saavutettiin noin 12–16 % parannus. Edelleen optimoimalla edustajana toimivaa prototyyppiä ehdotetun algoritmin avulla saavutettiin aineistosta riippuen 1–4 % parannus ryhmittelyn virheellisyyteen. Algoritmista ja aineistosta riippumatta edustajien paremmuusjärjestys ryhmittelyvirheen kannalta on kiistaton: heikoiten pärjäsi medoidi, prototyypin käyttäminen

medoidin sijasta tarjoaa suuren parannuksen ja ehdotetulla algoritmilla prototyypin hienosäätö tuottaa edelleen pienen parannuksen ositukselle.



Kuva 6.2: Edustajan vaikutus k-medoidsin toimintaan aineistolla Phs (vasen); Algoritmin vaikutus osituksen virheellisyyteen aineistolla Phs käytettäessä ehdotettua edustajaa (oikea).

Ero k-medoids ja RLS-algoritmien ryhmittelyn virheellisyyden välillä on varsin pieni, noin 1–4 %. RLS-algoritmi antaa kuitenkin aina paremman tuloksen kuin k-medoids, mikäli käytössä on sama alkuositus. Tämän lisäksi RLS-algoritmi tuottaa keskihajonnaltaan pienempiä tuloksia kuin k-medoids, toisin sanoin tulosten laatu ei vaihtele yhtä paljon RLS-algoritmia käytettäessä. Ero ryhmittelyvirheessä kasvaa RLS:n hyväksi, jos RLS-iteraatioita lisätään, mutta tämä puolestaan vaikuttaa suoraan algoritmin suoritusaikaan. AHC:n ja AHC-KM:n välinen ero ryhmittelyvirheen avulla mitattuna oli 0,1–9 % aineistosta ja käytetystä edustajasta riippuen. Algoritmien yhdistelmä paransi aina pelkän hierarkkisen ryhmittelyn avulla saatua ositusta, joten sen käyttö on perusteltua. Kuva 6.3 on yhteenveto siitä, kuinka eri edustaja-algoritmi -parit sijoittuivat toisiinsa nähden ryhmittelyvirheen avulla mitattuna. Kuvan selitelaatikon järjestys on tyypillinen järjestys edustaja-algoritmi -parien sijoittumisesta toisiinsa nähden ryhmittelyvirheen avulla mitattuna. Käytetyn algoritmin vaikutus ryhmittelyvirheeseen oli pienempi kuin käytetyn edustajan. Testeissä ilmennyt algoritmien tyypillinen paremmuusjärjestys parhaimmasta huonoimpaan oli AHC-KM, RLS, k-medoids, AHC. Käytettyjen edustajien osalta paremmuusjärjestys ryhmittelyvirheellä mitattuna oli aina ehdotettu, prototyyppi, medoidi.



Kuva 6.3: Yhteenveto ryhmittelyalgoritmin ja käytetyn edustajan vaikutuksesta ryhmittelyvirheeseen aineistolla Character. Menetelmien listaus on järjestetty ryhmittelyvirheen mukaisesti heikoimmasta parhaaseen.

Taulukkoon 6.1 on koottu kunkin aineiston valitulla mallin koolla ja k-medoids -iteraatioilla saadut 10 testiajon keskiarvot ja niiden keskihajonta. Parhaat ryhmittelymenetelmät kullekin aineistolle on lihavoitu. Saatuja tuloksia ei voida vertailla aineistojen kesken, sillä havaintoarvojen suuruusluokat poikkeavat merkittävästi toisistaan aineistojen välillä. Kuten taulukosta selviää, parhaiten tavoitefunktiota optimoivat RLS- ja AHC-KM -yhdistelmä. Yhdistelmäalgoritmi tuotti paremman osituksen aineistoilla Auslan ja Character. RLS puolestaan toimi paremmin puhedataa sisältävillä aineistoilla Japanese ja Phs. Synthetic aineiston ryhmittelyssä nämä kaksi menetelmää pärjäsivät suunnilleen yhtä hyvin, kun otetaan huomioon RLS-algoritmin testiajojen hajonta ryhmittelyvirheen osalta.

Virhe (MAE)		Auslan M = 95 K = 15 x 1	Character M = 59 K = 10 x 10^2	Japanese M = 9 K = 18 x 1	Phs M = 8 K = 8 x 10^2	Synthetic M = 6 K = 8 x 10
KM	Medoidi	17,63 ± 0,15	53,68 ± 0,43	8,17 ± 0,31	14,24 ± 0,22	16,03 ± 0,54
	Prototyyppi	14,70 ± 0,11	44,88 ± 0,31	6,72 ± 0,14	12,49 ± 0,05	14,96 ± 0,43
	Ehdotettu	14,57 ± 0,10	43,02 ± 0,45	6,68 ± 0,13	12,37 ± 0,07	14,65 ± 0,38
RLS	Medoidi	17,29 ± 0,15	52,26 ± 0,43	7,69 ± 0,08	14,03 ± 0,16	15,58 ± 0,09
	Prototyyppi	14,51 ± 0,07	43,91 ± 0,31	6,57 ± 0,03	12,38 ± 0,01	14,61 ± 0,02
	Ehdotettu	14,40 ± 0,08	42,20 ± 0,28	**6,52 ± 0,01**	**12,27 ± 0,04**	**14,38 ± 0,07**
AHC	Medoidi	17,46	55,52	7,80	15,66	15,64
	Prototyyppi	14,88	46,58	6,67	13,61	14,68
	Ehdotettu	14,76	44,00	6,64	13,50	14,43
AHCKM	Medoidi	16,53	50,40	7,68	14,18	15,53
	Prototyyppi	14,25	43,38	6,58	12,48	14,66
	Ehdotettu	**14,14**	**41,68**	6,53	12,32	**14,40**
AVG	Medoidi	17,23	52,96	7,84	14,53	15,70
	Proto	14,59	44,69	6,64	12,74	14,73
	Ehdotettu	14,47	42,73	6,59	12,62	14,46

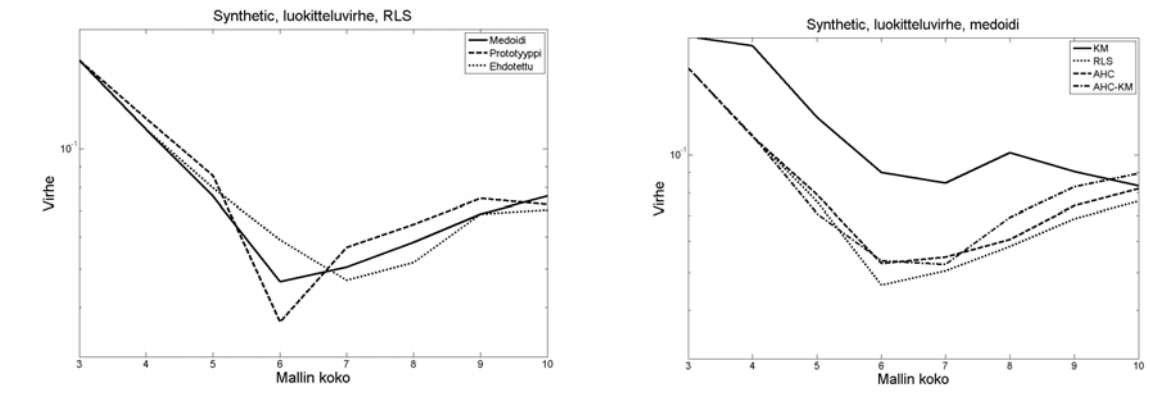
Taulukko 6.1: Käytettyjen menetelmien vaikutus ryhmittelyn virheellisyyteen.

6.3 Luokitteluvirhe

Luokitteluvirheen avulla voidaan tarkastella, kuinka hyvin ryhmittelyalgoritmi jakaa aineiston toivottuihin luokkiin. Vertailua varten tarvitaan jokaiselle aikasarjalle luokkanimike, joka määrittää mihin luokkaan tai ryhmään kyseisen aikasarjan halutaan kuuluvan. Luokitteluvirheelle ei ole olemassa yleisesti käytettyä määritelmää. Kirjallisuudessa luokitteluvirhe määritellään usein tapauskohtaisesti ja siksi eri töiden välisiä luokittelutuloksia on hankala vertailla. Tässä tutkielmassa luokitteluvirhe määritellään virheellisten ositusparien suhteella kaikkiin mahdollisiin osituspareihin ja se voidaan laskea algoritmin 5.1 avulla.

Kuva 6.4 esittää luokitteluvirheen kuvaajaa aineistolle Synthetic. Kuvasta havaitaan kuinka luokitteluvirhe on pienimmillään kun mallin koko vastaa ryhmässä esiintyvien luokkien lukumäärää. Mikäli mallin koko on pienempi tai suurempi niin todennäköisyys sille, että aikasarjat ovat eri ryhmissä lasketussa ja oikeassa osituksessa kasvaa. Tästä johtuen mallin kokoa ei ole syytä kasvattaa liikaa, sillä vaikka se pääsääntöisesti tuottaa parempia osituksia

ryhmittelyvirheen osalta, niin käytännössä muodostetuissa osituksissa käytetään liian paljon ryhmiä edustamaan kyseessä olevaa aineistoa.



Kuva 6.4: Luokitteluvirhe aineistolle Synthetic.

Taulukkoon 6.2 on koottu testeissä saadut tulokset luokitteluvirheen osalta. Hierarkkisen algoritmin deterministisyyden takia sen keskihajonta on nolla. Määritelty luokitteluvirhe ei liity ryhmittelyvirheeseen suorasti, minkä osoittavat esimerkiksi hierarkkisella algoritmilla saadut tulokset. Tämä algoritmi pärjäsi heikoiten ryhmittelyvirheen avulla tarkasteltuna, mutta yllättäen tuotti parhaat tulokset, kun osituksia tarkasteltiin luokitteluvirheen osalta. Sama ilmiö oli havaittavissa myös edustajia vaihdettaessa. Keskiarvoaikasarjan käyttäminen medoidin sijaan tuotti aineistosta riippuen noin 1–22 % parannuksen luokitteluvirheeseen. Ehdotetun keskiarvoaikasarjan hienosäädön käyttäminen yksinkertaisen keskiarvoaikasarjan sijaan puolestaan huononsi ositusta luokitteluvirheen kannalta kaikilla muilla paitsi Japanese-aineistolla. Luokitteluvirheen huononeminen oli tässä tapauksessa aineistosta ja ryhmittelyalgoritmista riippuen noin 1–12 %.

RLS-algoritmin käyttäminen k-medoidsin sijasta toi keskimäärin noin 0–50 % parannuksen luokitteluvirheeseen. Vain muutamalla edustaja-algoritmi -parilla RLS loi luokitteluvirheen osalta huonomman osituksen kuin k-medoids. Sen sijaan AHC-KM -yhdistelmäalgoritmin käyttäminen tuotti kaikilla aineistoilla huonomman luokitteluvirheen kuin pelkkä hierarkkinen ryhmittely. Yhdistelmän tuottamat luokitteluvirheet olivat testiajoissa noin 5–40 % huonompia kuin pelkän hierarkkisen algoritmin.

Luokitteluvirhe (%)		Auslan M = 95 K = 15	Character M = 59 K = 10	Japanese M = 9 K = 18	Synthetic M = 6 K = 8
KM	Medoidi	1,937 ± 0,071%	3,389 ± 0,236%	9,854 ± 2,585%	9,009 ± 2,816%
	Prototyyppi	1,818 ± 0,075%	3,175 ± 0,160%	6,593 ± 1,909%	9,236 ± 2,473%
	Ehdotettu	1,825 ± 0,082%	3,273 ± 0,159%	6,567 ± 1,866%	9,430 ± 2,341%
RLS	Medoidi	1,888 ± 0,084%	3,315 ± 0,166%	6,104 ± 1,619%	4,637 ± 1,906%
	Prototyyppi	1,782 ± 0,102%	3,197 ± 0,132%	4,627 ± 0,509%	**3,674 ± 1,408%**
	Ehdotettu	1,801 ± 0,058%	3,315 ± 0,136%	4,499 ± 0,438%	5,908 ± 2,057%
AHC	Medoidi	**1,466 %**	**2,651 %**	**3,319 %**	5,266 %
	Prototyyppi	**1,466 %**	**2,651 %**	**3,319 %**	5,266 %
	Ehdotettu	**1,466 %**	**2,651 %**	**3,319 %**	5,266 %
AHCKM	Medoidi	1,600 %	2,836 %	5,015 %	5,363 %
	Prototyyppi	1,560 %	2,752 %	4,442 %	5,870 %
	Ehdotettu	1,580 %	2,759 %	4,447 %	6,292 %
AVG	Medoidi	1,723 %	3,048 %	6,073 %	6,069 %
	Prototyyppi	1,656 %	2,944 %	4,745 %	6,011 %
	Ehdotettu	1,668 %	2,999 %	4,708 %	6,724 %

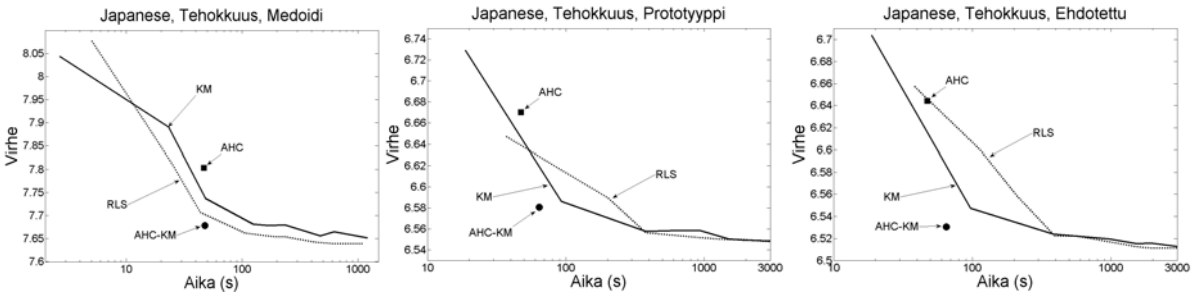
Taulukko 6.2: Käytettyjen menetelmien vaikutus luokitteluvirheeseen.

6.4 Menetelmien tehokkuus

Toteutettujen ryhmittelyalgoritmien tehokkuus määritellään saadun osituksen virheellisyydellä suhteessa käytettyyn aikaan. Käytetyn edustajan vaikutus ryhmittelyalgoritmien tehokkuusjärjestykseen ei ole merkittävä. Ainoastaan RLS-algoritmi kärsii muita algoritmeja enemmän, mikäli käytetään laskennallisia edustajia medoidien sijasta. Kuva 6.5 esittää algoritmien ja edustajien välillä saavutettuja eroja tehokkuusmittauksissa aineistolla Japanese. Kuvasta selviää, kuinka saadun osituksen laatu paranee, jos käytettävissä on enemmän aikaa. Tietyn pisteen jälkeen käytössä olevan ajan lisääminen ei paranna ositusta merkittävästi.

Testeissä käytetty k-medoids -muunnelma on algoritmi, jossa k-medoids käynnistetään peräkkäin erilaisilla satunnaisesti valituilla alkuosituksella. Algoritmi suorittaa peräkkäin käyttäjän syöttämän määrän k-medoids -ajoja. Tuloksena algoritmi antaa parhaan löytyneen osituksen aineistolle. Tätä algoritmia käytetään, koska perinteinen k-medoids antaa yksittäisiä tuloksia, joiden suoritusaikaan ja hyvyyteen alkuositus vaikuttaa. Iteratiivisen k-medoidsin avulla voidaan luoda kuvaaja, joka ilmaisee k-medoidsin tehokkuutta yksittäisiä pisteitä paremmin. K-medoids on algoritmina deterministinen, mutta saadun osituksen hyvyys riippuu

alkuosituksesta, jonka muodostus ei ole deterministinen. Hierarkkinen ryhmittelyalgoritmi on k-medoids- ja RLS-algoritmeista poiketen täysin deterministinen, joten se tuottaa aineistolle aina samanlaisen osituksen. Tästä johtuen hierarkkiselle ryhmittelyalgoritmille saadaan ainoastaan yksittäinen piste tehokkuutta kuvaaviin kuviin. Sama pätee myös hierarkkisen algoritmin ja k-medoidsin yhdistelmälle, sillä tällöin k-medoids saa aina saman alkuosituksen ja on siten deterministinen.



Kuva 6.5: Menetelmillä saavutettu tehokkuus aineistolla Japanese.

Taulukkoon 6.3 on koottu käytettyjen ryhmittelymenetelmien suoritusajat ja niiden keskihajonnat. Käytetyt k-medoids -iteraatioiden rajat ja mallien koot ovat samat kuin ryhmittely- ja luokitteluvirheitä testatessa. Aineistot voidaan jakaa suoritusaikojen perusteella kolmeen ryhmään. Ensimmäiseen ryhmään kuuluu puheaineisto Japanese, joka oli aineistosta selvästi nopein ryhmiteltävä. Tähän vaikuttavat aineiston suppeahko koko ja aikasarjojen lyhyys; aikasarjojen keskipituus on 16 otosta. Seuraavaan ryhmään kuuluu kolme aineistoa eli Character, Phs ja Synthetic. Nämä aineistot ovat verrattain lähellä toisiaan haastavuudessaan. Viimeiseen ryhmään kuuluu selvästi vaativin aineisto Auslan. Sen ryhmittelyyn kuluu ehdotetulla edustajalla ja RLS-algoritmilla pisimmillään yli 12 tuntia. Auslan on testatuista aineistoista kookkain ja se sisältää myös eniten ryhmiä sekä ulottuvuuksia. Aineiston aikasarjat ovat myös keskipituudeltaan pidempiä kuin kolmessa muussa aineistossa.

Aika (s)		Auslan M = 95 K = 15 x 10^2	Character M = 59 K = 10 x 10	Japanese M = 9 K = 18 x 1	Phs M = 8 K = 8 x 10	Synthetic M = 6 K = 8 x 10
KM	Medoidi	**2,98 ± 0,38**	**2,46 ± 0,47**	**2,78 ± 0,69**	**2,72 ± 0,39**	**2,38 ± 0,61**
	Prototyyppi	13,87 ± 0,16	7,62 ± 1,29	20,09 ± 3,40	11,01 ± 1,84	9,88 ± 1,94
	Ehdotettu	14,18 ± 0,16	8,37 ± 1,29	20,99 ± 3,76	11,54 ± 1,12	9,63 ± 1,66
RLS	Medoidi	97,66 ± 1,35	65,55 ± 11,32	75,56 ± 14,38	78,74 ± 11,95	63,09 ± 8,64
	Prototyyppi	430,86 ± 8,48	234,85 ± 33,65	635,94 ± 100,25	304,25 ± 2,93	293,91 ± 46,92
	Ehdotettu	443,28 ± 6,12	250,17 ± 42,16	658,61 ± 107,05	319,81 ± 6,85	307,75 ± 40,01
AHC	Medoidi	100,26 ± 0,02	43,28 ± 0,03	57,42 ± 0,09	70,70 ± 0,19	39,85 ± 0,01
	Prototyyppi	100,39 ± 0,06	43,18 ± 0,03	59,26 ± 0,06	72,55 ± 0,21	40,43 ± 0,01
	Ehdotettu	100,33 ± 0,03	43,43 ± 0,02	59,61 ± 0,08	72,19 ± 0,18	40,61 ± 0,01
AHCKM	Medoidi	101,68 ± 0,03	44,50 ± 0,02	58,44 ± 0,09	71,78 ± 0,12	41,06 ± 0,01
	Prototyyppi	113,33 ± 0,03	50,48 ± 0,02	79,94 ± 0,07	81,71 ± 0,28	49,27 ± 0,01
	Ehdotettu	113,81 ± 0,03	51,46 ± 0,03	80,85 ± 0,09	81,39 ± 0,04	49,91 ± 0,02

Taulukko 6.3: Käytettyjen menetelmien suoritusajat.

Tuloksista havaintaan, kuinka nopeasti medoidi voidaan laskea verrattuna keskiarvoaikasarjan muodostukseen. Tämä on merkittävä syy medoidin yleisyyteen käytettynä edustajana aikasarjoja ryhmiteltäessä. Keskiarvoaikasarjojen muodostaminen hidastaa k-medoids ja RLS -algoritmien suoritusaikoja aineistosta riippuen noin 2–8 -kertaisesti. Hidastus hierarkkisia ryhmittelymenetelmiä käyttäessä on paljon pienempi, noin 1–37 %. Tämä johtuu siitä, että kokoavaa hierarkkista ryhmittelyä käytettäessä keskiarvoedustajia ei käytetä ryhmien etäisyyksien mittaamiseen, vaan ne voidaan muodostaa vasta valmiin osituksen jälkeen. Kokoavan hierarkkisen ryhmittelyn ja k-medoidsin yhdistelmässä puolestaan merkittävin osa ajasta kuluu hierarkkisen ryhmittelyn muodostamiseen, joka pienentää k-medoidsissa tapahtuvan keskiarvoaikasarjojen muodostamiseen käytetyn ajan vaikutusta algoritmin kokonaissuoritusaikaan. Täten erityisesti hierarkkisia menetelmiä käytettäessä keskiarvoaikasarjan laskeminen on erittäin suotavaa, sillä se ei lisää algoritmien suoritusaikoja kohtuuttomasti. Myöskään ehdotettu menetelmä edustaja-aikasarjojen hienosäädölle ei aiheuta suuria lisäyksiä suoritusaikaan. Suurimmillaan erot prototyyppi-menetelmän ja ehdotetun version välillä ovat käytettäessä RLS-algoritmia, jolloin lisäys suoritusaikoihin on 3–7 %. Muilla algoritmeilla kasvu on keskimäärin noin 2–5 %.

K-medoids on odotetusti algoritmeista selvästi nopein. RLS-algoritmin suoritusaika on suoraan verrannollinen RLS-iteraatioiden määrään: 30 RLS-iteraation testiajot kasvattavat suoritusaikoja aineistosta ja edustajasta riippuen noin 28–32 -kertaisiksi. Hierarkkiset

menetelmät ovat aineistosta riippuen noin 16–33 kertaa hitaampia kuin k-medoids käytettäessä medoidia edustajana ja noin 2–6 kertaa hitaampia käytettäessä laskennallista keskiarvoaikasarjaa. Ero hierarkkisen menetelmän ja hierarkkisen menetelmän sekä k-medoidsin yhdistelmän suoritusaikojen välillä on verrattain pieni, noin 9–24 %. Tämän vuoksi onkin suotavaa käyttää hierarkkisen menetelmän sijaan yhdistelmää, jossa k-medoidsin alkuositus muodostetaan hierarkkisen menetelmän avulla.

Aikasarjojen ryhmittelyä varten käytettävän edustajan valinnassa voidaan painottaa sen laskennan nopeutta tai edustajan ryhmää kuvaavaa hyvyyttä. Mikäli tavoitteena on luoda mahdollisimman nopea ryhmittelymenetelmä, tulisi edustajana käyttää medoidia. Jos puolestaan halutaan painottaa ryhmittelyn tarkkuutta, niin voidaan harkita laskennallisen keskiarvoaikasarjan käyttämistä ryhmän edustajana. Testatuista algoritmeista parhaiten tehokkuuden osalta pärjää hierarkkisen menetelmän ja k-medoidsin yhdistelmä. RLS-algoritmi pärjää tehokkuudessa iteroitua k-medoids -algoritmia paremmin käytettäessä medoidia edustajana, mutta tilanne on huomattavasti tasaisempi käytettäessä laskennallista aikasarjaa. Tähän vaikuttavat verrattain suuriksi asetetut k-medoids -iteraatioiden rajat, jolloin RLS-algoritmi kärsii huonoista edustajien vaihdoista. Huonoiten tehokkuuden avulla mitattuna pärjää hierarkkinen ryhmittelyalgoritmi.

7 YHTEENVETO

Aikasarjojen ryhmittely on haastava ja laaja tutkimusalue, jolla on käytännön sovelluksia useilla tieteenaloilla. Puheentunnistus ja automaattinen tekstintunnistus ovat esimerkkejä sovellusalueista, joiden osana käytetään yleisesti aikasarjojen ryhmittelyä. Ryhmittelytehtävää varten tutkielmassa määriteltiin optimoitava tavoitefunktio, joka on aikasarjoille sopiva muunnelma staattiselle datalle käytetystä keskivirheestä. Tavoitefunktiota pyrittiin minimoimaan toteutettujen ryhmittelyalgoritmien avulla, joiden tehokkuutta mitattiin ryhmittelyvirheen ja suoritusajan mukaan. Algoritmien lisäksi vertailtiin erilaisten ryhmien edustajien vaikutusta ryhmittelyn onnistumiseen.

Tässä pro gradu -tutkielmassa käytettiin aikasarjojen etäisyysmittana dynaamista aikasovitus -etäisyyttä. Kyseessä oleva muotopohjainen etäisyysmitta soveltuu euklidista etäisyyttä paremmin aikasarjoille. Euklidisen etäisyyden ongelmana aikasarjojen kohdalla ovat herkkyys kohinalle ja poikkeaville arvoille sekä kyvyttömyys vertailla kahta eri mittaista aikasarjaa keskenään. Toteutettu DTW-algoritmi on aikavaativuudeltaan $O(KL^2)$, mutta kirjallisuudessa on esitetty algoritmeja, joilla DTW-etäisyys voidaan laskea tarkasti lähes lineaarisessa ajassa aikasarjojen pituuksiin nähden. Nopeamman DTW-algoritmin toteutus on välttämätöntä varsinkin isompien aineiston ryhmittelyä varten.

Käytettyinä ryhmittelyalgoritmeina toteutin kolme algoritmia. Yleisesti käytetty k-means -algoritmi ei sovellu aikasarjojen ryhmittelyyn, sillä keskiarvovektoreiden muodostaminen eri mittaisista aikasarjoista ei onnistu euklidisen etäisyyden avulla. Tästä johtuen käytetty k-meansin muunnos on k-medoids -algoritmi, jossa ryhmien edustajina käytetään todellisia aikasarjoja, jotka ovat lähinnä ryhmien keskustoja. Toinen toteutettu algoritmi on RLS-algoritmi. Kyseessä on Fräntin ja Kivijärven (2001) esittelemä muunnos k-means -algoritmiin, joka pyrkii välttämään juuttumisen paikalliseen optimiositukseen. Kolmas toteutettu ryhmittelymenetelmä on keskiarvolinkitystä käyttävä kokoava hierarkkinen ryhmittelyalgoritmi. Näiden kolmen algoritmin lisäksi käytin hierarkkisen ryhmittelyn ja k-medoidsin yhdistelmää neljäntenä ryhmittelymenetelmänä. Algoritmeja vertailtiin ryhmittelyvirheen, luokitteluvirheen ja tehokkuuden avulla.

Mikäli aikasarjoja ryhmiteltäessä halutaan optimoida nopeutta, niin medoidi on hyvä edustaja ryhmälle. Jos tavoitteena on tuottaa mahdollisimman hyvä ositus, tulisi edustajana käyttää laskennallista keskiarvoaikasarjaa kullekin ryhmälle. Ehdotettu menetelmä parantaa ryhmittelyn tuloksia, mutta on samalla hitaampi kuin medoidia käyttävä ryhmittely. Saatujen tulosten perusteella nopeimman, mutta karkeimman ryhmittelyn tuottaa k-medoids -algoritmi. Mikäli tehtävä ei ole aikakriittinen, niin RLS-algoritmi tai hierarkkisen algoritmin ja k-medoidsin yhdistelmä tuottavat parhaita osituksia ryhmittelyvirheellä mitattuna. Hierarkkinen ryhmittely tuotti heikoimmat tulokset ryhmittelyvirheen osalta, mutta pärjäsi muita algoritmeja paremmin kun tuloksia mitattiin luokitteluvirheellä. Testeissä ei yksikään algoritmi-edustaja -pari noussut selkeästi ylitse muiden. Ryhmittelyyn parhaiten soveltuvan algoritmin ja edustajan valinta riippuu sovellusalueesta ja sille liitettävistä erityisvaatimuksista.

Oleellisia lisätutkimuksen aiheita työhön liittyen ovat esimerkiksi tarvittavien k-medoids -iteraatioiden lukumäärän selvittäminen tehokkaampaa RLS-algoritmia varten, deterministisen tai puoli-deterministisen RLS-algoritmin vertailu satunnaiseen versioon nähden ja paremman keskiarvoedustajan luovan algoritmin suunnittelu. Yksi esitellyn algoritmin suurimmista ongelmista on kyvyttömyys muunnella keskiarvoaikasarjan pituutta. Keskiarvoaikasarjan pituus lukittuu samaksi kuin medoidin pituus, joka kuvastaa ryhmää parhaiten muodon, mutta ei välttämättä aikasarjojen pituuden osalta. Ratkaisu tähän voisi olla algoritmi, joka ottaa huomioon ryhmän medoidin muodon ja ryhmään kuuluvien aikasarjojen keskipituuden. Testattujen aineistojen lisäksi tulisi menetelmien toimivuutta tarkkailla todellisista sovellusalueista kerätyillä aineistolla. Aikasarjojen ryhmittelyä voisi soveltaa esimerkiksi lintujen ryhmittelyyn ääninäytteiden avulla tai tarkan viittomakielisen aineiston ryhmittelyyn.

VIITELUETTELO

Abdulla, W.H, Chow, D., Sin, G. (2003). Cross-words Reference Template for DTW-based Speech Recognition Systems. *Conference on Convergent Technologies for Asia-Pacific Region* (TENCON), **4**:1576 – 1579.

Abramowitz & Stegun (1965). *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. (http://www.math.sfu.ca/~cbm/aands/)

Asuncion, A., Newman, D.J. (2007). UCI Machine Learning Repository. Www-sivusto: http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science. (29.3.2008)

Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, **23**(3): 345–405.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, New Jersey.

Bertsekas, D. (1976). *Dynamic Programming and Stochastic Control*. Academic Press, Inc; New York.

Berndt, D.J., Clifford, J. (1994). Using dynamic time warping to find patterns in time series. *AAAI-94 Workshop on Knowledge Discovery in Databases*, Seattle, WA, USA: 359–370.

Carrillo, H., Lipman, D. (1988). The Multiple Sequence Alignment Problem in Biology. *SIAM Journal on Applied Mathematics*. **48**(5):1073–1082.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2001). *Introduction to algorithms, Second edition*. MIT Press, Cambridge.

Dunham, M.H. (2002). *Data Mining, Introductory and Advanced Topics.* Prentice Hall.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases. *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*. 226–231.

Equitz, W.H. (1989). A New Vector Quantization Clustering Algorithm. *IEEE Transaction on Acoustics, Speech and Signal Processing*. **37**(10): 1568-1575.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P (1996). From Data Mining to Knowledge Discovery in Databases. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press: 1–30.

Fielding, A.H. (2007). *Cluster and Classification Techniques for the Biosciences*. Cambridge University Press.

Fu, A., Keogh, E., Lau, L., Ratanamahatana, C. (2005) Scaling and Time Warping in Time Series Querying. *Proceedings of the 31st VLDB Conference*, Trondheim. 649–660.

Fränti, P., Kivijärvi, J. (2000). Randomised Local Search Algorithm for the Clustering Problem. *Pattern Analysis & Applications*. **3**(4): 358–369.

Fränti, P., Virmajoki, O., Hautamäki, V. (2006). Fast Agglomerative Clustering Using a *k*-Nearest Neighbor Graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **28**(11): 1875–1881.

Garey, M.R., Johnson, D.S., Witsenhausen, H.S. (1982). The complexity of the generalized Lloyd-Max problem. *IEEE Transactions on Information Theory*. **28**(2): 255-256.

Gaudin, R., Barbier, S., Nicoloyannis, N., Banens, M. (2006). Clustering of Bi-Dimensional and Heterogeneous Time Series: Application to Social Sciences Data. *Proceedings of the International Conference on Data Mining* (*DMIN*): 10–16.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press.

Han, J., Kamber, M. (2001). *Data Mining, Concept and Techniques*. Morgan Kaufmann Publishers, San Francisco.

Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons, New York.

Hartimo, I., Laine, U., Niemelä, T., Nyman, G., Oja, E, Tuovinen, P. (1985). Kuvankäsittely- ja hahmontunnistusalan sanasto, www-sivusto: http://www.it.lut.fi/kurssit/97-98/1588/references/en2fi/en2fi.html. Otaniemi, Helsinki. (29.3.2008)

Harvey, A.C. (1993). *Time Series Models, Second Edition*. Harvester-Wheatsheaf, New York.

Hirano, S., Tsumoto, S. (2005). Empirical Comparison of Clustering Methods for Long Time-Series Databases. *Lecture Notes in Computer Science*. **3430**: 268-286.

Itakura, F. (1975) Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*. **23**: 52–72.

Jain, A., Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice-Hall International. (http://www.cse.msu.edu/%7Ejain/Clustering_Jain_Dubes.pdf)

Jain, A. K., Murty, M. N., Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys (CSUR)*, **31**(3): 264–323.

Just, W. (2001). Computational Complexity of Multiple Sequence Alignment with SP-score. *Journal of Computational Biology*. **8**(6): 615–623.

Kadous, M. W. (2002). *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. Väitöskirja, School of Computer Science and Engineering, University of New South Wales.

Kaufman, L., Rouseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York : John Wiley and Sons, 1990.

Keogh, E., Pazzani, M. (1998). An Enhanced Representation of Time Series Which Allows Fast and Accurate Classication, Clustering and Relevance Feedback. *Proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*. 239-241.

Keogh, E., Pazzani, M. (1999). Scaling up Dynamic Time Warping to Massive Datasets. *3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*. Springer. **1704**: 1–11.

Keogh, E., Pazzani, M. (2001). Derivative Dynamic Time Warping. *1st SIAM International Conference on Data Mining*.

Keogh, E., Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. **7**: 349–371.

Keogh, E., Ratanamahatana, C. A. (2002). Exact Indexing of Dynamic Time Warping. *Proceedings of the 28th VLDB Conference*. 358–386.

Kinnunen, T., Karpov, E., Fränti, P. (2006). Real-Time Speaker Identification and Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, **14**(1): 277–288.

Kruskal, J. B. (1983). An Overview of Sequence Comparison: Time Warps, String Edits and Micromolecules. *Society for Industrial and Applied Mathematics Review*, **25**(2): 201–237.

Kärkkäinen, I. (2006). *Methods for Fast and Reliable Clustering*. Väitöskirja, Tietojenkäsittelytieteen laitos, Joensuun yliopisto.

Liao, T. W. (2005). Clustering of time series data – a survey. *Pattern Recognition*, **38**(11): 1857–1874.

Montalvo, S., Martínez, R., Casillas, A., Fresno, V. (2007). Multilingual news clustering: Feature translation vs. identification of cognate named entities. *Pattern Recognition Letters*, **28**(16), 2305–2311.

McQueen, J.B. (1967). Some methods of classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symp Mathemat Statist Probability*. **1**:281–296

Mörchen, F. (2006). *Time Series Knowledge Mining*. Väitöskirja, Philipps-University Marburg, Germany.

Niennattrakul, V., Ratanamahatana, C. A. (2007). On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping. *International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*. 733–738.

Rabiner, L., Juang B-H (1993). *Fundamentals of speech recognition*. Englewood Cliffs (N.J.): PTR Prentice Hall.

Ratanamahatana, C.A, Keogh, E. (2005). Three Myths about Dynamic Time Warping Data Mining. *Proceedings of the 5th SIAM International Conference on Data Mining*, 506–510.

Ruiz, V., Nolla, C., Segovia, R., (1985). Is the DTW "distance" really a metric? An algorithm reducing the number of DTW comparisons in isolated word recognition. *Speech Communication*, **4**(4): 333–344.

Sakoe, H., Chiba, S. (1978) Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing,* **26**: 43 –49.

Salvador, S., Chan, S. (2004). FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. *Workshop on Mining Temporal and Sequential Data, 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Späth, H. (1980). Cluster Analysis Algorithms for Data Reduction and Classification of Objects. Ellis Horwood Limited, England.

Vuori, V. (2001). Johdatusta puheentunnistukseen: Koska HAL ymmärtää mitä puhumme? Puheen automaattinen tunnistus ja ymmärtäminen. TKK, Informaatiotekniikan laboratorio. (http://www.tml.tkk.fi/Opinnot/Tik-111.590/2001/paperit/vuori.pdf)

Vuori, V. (2002). *Adaptive methods for on-line recognition of isolated hand-written characters*. Väitöskirja, Tietojenkäsittelytieteen laitos, Helsingin teknillinen yliopisto.

Vuori, V., Laaksonen, J. (2002). A Comparison of Techniques for Automatic Clustering of Handwritten Characters. *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)* **3**: 30168.

Wang, W., Yang, J., Muntz, R. (1997). STING: A statistical information grid approach to spatial data mining. *Proceedings of International Conference of Very Large Data Bases*. 186–195.

Ward, J.H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association* **58**(301): 236–244.

Xu, R., Wunsch, D. (2005). Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks.* **16**(3): 645–678.

Zhu, Y. (2004). *High Performance Data Mining in Time Series: Techniques and Case Studies*. Väitöskirja, Department of Computer Science, New York University.

Zhu, X., Davidson, I. (2007). *Knowledge Discovery and Data Mining: Challenges and Realities*. Information Science Reference, London.