# Barysentrisen heuristiikan ja mukautuvuusanalyysiin perustuvan algoritmin vertailu matriisien järjestämisessä

Sirkka Andersén

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Erkki Mäkinen
26.7.2013

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Sirkka Andersén: Barysentrisen heuristiikan ja mukautuvuusanalyysiin perustuvan algoritmin vertailu matriisien järjestämisessä
Pro gradu -tutkielma, 72 sivua
Heinäkuu 2013

---

**Tiivistelmä**

Tutkielmassa käsitellään matriisien järjestelyyn käytettävien algoritmien ominaisuuksia ja historiaa sekä matriisien ja graafien yhteisiä sovellus- ja ongelma-alueita. Matriisien järjestelyalgoritmeista tutustutaan tarkemmin mukautuvuusanalyysiin ja miinustekniikkaan perustuvan järjestelyalgoritmin toimintaan, minkä lisäksi tutustutaan kaksijakoisen graafin risteymien minimointiin perinteisesti käytettyyn barysentriseen heuristiikkaan ja käsitellään risteymien minimointiongelma matriisin järjestelyongelmana. Kumpaankin tekniikkaan perustuvat järjestelyt toteuttiin ohjelmallisesti ja testattiin erityyppisillä ja -kokoisilla matriiseilla. Työssä toteutettiin myös uusi barysentriseen järjestelyyn ja miinustekniikkaan perustuva järjestely, jota sovellettiin muiden tekniikoiden ohella risteymien minimointiongelmaan. Uusi järjestelytekniikka toimi erityisen lupaavasti synteettisesti generoidulla kaistamaisella aineistolla. Testiajojen perusteella myös mukautuvuusanalyysiin ja miinustekniikkaan perustuva algoritmi toimi lupaavasti risteymien minimointiin liittyvien hahmojen muodostamisessa ja risteymien minimointitehtävässä.

Tuloksien perusteella miinustekniikka ja mukautuvuusanalyysi voivat olla varteenotettavia vaihtoehtoja erityisesti sellaisilla graafeilla, joiden järjestelyyn barysentrinen järjestely tai muut perinteiset heuristiikat eivät sovi.

# Sisällys

# 1 Johdanto

Informaation visualisointi on tieteenala, joka käsittelee abstraktin tiedon koodaamista visuaalisiksi esityksiksi. Se voi tarkoittaa esimerkiksi ongelman ratkaisun muuttamista visuaalisesti ilmiselväksi [Spence, 2007] tai ihmisen kognition avustamista tietojenkäsittelytehtävissä visuaalisten representaatioiden avulla [Ware, 2004].

Spencen [2007] mukaan informaation visualisoinnin ensisijainen tehtävä on johtaa informaatiota datasta. Informaation visualisoinnin tärkeyttä hän korostaa seuraavalla esimerkillä: Vakavien petosten tutkimuksiin erikoistunut yksikkö *The Serious Fraud Office* Isossa-Britanniassa tutki erästä rakennuspetostapausta ja löysi oletetun syyllisen käytettyään kahdeksan henkilötyövuotta tapauksen selvittämiseen. Aineistona tapauksessa oli 48 vetolaatikollista paperidataa kansioissa.

Myöhemmin sama tapaus annettiin yksittäisen tutkijan selvitettäväksi ja hän sai käyttöönsä visualisointisovelluksen aineiston käsittelyyn. Sovelluksen avulla hän onnistui osoittamaan syylliseksi saman tekijän kuin toinenkin tutkimusryhmä, mutta tämän lisäksi hän onnistui sovelluksen avulla tunnistamaan päärikollisen koko vyyhdin takana.

Ihmisaivot tarjoavat todella tehokkaan rinnakkaisprosessorin tiedon käsittelylle. Asiantuntijalle lankeavassa analysointiprosessissa erinomaisia värin- ja hahmontunnistuksen ominaisuuksia voi hyödyntää monimutkaisen datan käsittelyn nopeuttamiseksi ja olennaisen tiedon havaitsemisen helpottamiseksi.

Tiedon analysoinnin sovellusalueita on lukuisia esimerkiksi bioinformatiikassa, psykologiassa ja käytettävyydessä. Monille sovellusalueille yhteinen ongelma on informaation kategorisointi, jossa voi olla kysymys esimerkiksi sairauksien luokittelemisesta oireiden perusteella.

Tiedonlouhintayhteisö onkin jo pidemmän aikaa tutkinut suurien datamäärien analysointia erilaisiin tarkoituksiin, ja esimerkiksi erilaiset klusterointimenetelmät ovat olleet jo vuosikymmeniä intensiivisen tutkimuksen kohteena. Samaan aikaan informaation visualisointi on vakiinnuttanut asemaansa tutkimusalueena. Informaation visualisoinnin parissa tutkimus on keskittynyt ihmisaivojen kognitiiviseen toimintaan, visuaalisten ärsykkeiden havaitsemiseen ja käsittelyyn. Merkittäviä tutkimustuloksia on saavutettu kummallakin alalla. Olisikin luontevaa nähdä nämä kaksi tutkimusalaa yhteistyössä keskenään, jolloin tiedonlouhintayhteisössä kehitettäisiin algoritmeja datan analysointiin ja visualisointisovelluksia toteutettaisiin näiden pohjalta ja vastavuoroisesti tiedonlouhinta-algoritmien

suuntaviivoja haettaisiin informaation visualisoinnin alan uusimpien tutkimusten pohjalta. Informaation visualisointi ei ole kuitenkaan kuin vasta viime aikoina saanut huomiota tiedonlouhintayhteisössä. [Liiv, 2008]

Informaation visualisointi tähtää visuaalisten esitysten muodostamiseen laajamuotoisesta datasta ihmiselle ymmärrettävään muotoon, kun taas tiedonlouhinta on enemmän laskenta- ja tietokonekeskeistä [Bertini & Lalanne, 2010]. Visuaalinen analytiikka tutkii visualisoinnin käyttöä tiedon analysointiin ja se on jonkinlainen yhdistelmä tiedonlouhintaa, informaation visualisointia ja ihmistekijöitä [Keim, 2002].

Tässä tutkielmassa tarkastellaan matriisin järjestelyalgoritmeja ja graafien piirtoalgoritmeja. Matriisien järjestäminen on menetelmä, jolla tutkittava aineisto voidaan järjestää lineaariselle jatkumolle, esimerkiksi kronologiseen järjestykseen. Graafien piirtoalgoritmien eräänä keskeisenä tavoitteena on löytää ns. hyvä esitysmuoto, ja hyvä voi tarkoittaa matemaattisemmin ilmaistuna esimerkiksi mahdollisimman vähäistä määrää graafin särmien välisiä risteymiä, joka tekee esitysmuodosta selkeämmän.

Sekä matriisien järjestämisen että kaksijakoisen graafin piirto-ongelman yhtenä motiivina on toiminut tiedon visualisoinnin näkökulma. Siinä missä risteymien minimointiongelma yrittää saada kaksijakoisen graafin visuaalisen esitysmuodon selkeämmäksi, niin matriisien järjestämisellä yritetään mahdollistaa datan esittäminen siten, että tulos heijastaa paremmin niin yksilöllisellä että globaalilla tasolla datan keskinäisiä suhteita. [Liiv, 2008]

Tutkielmassa esitellään graafien ja matriisien järjestelemiseen liittyviä sovellusalueita ja ongelmien yhtenevyyksiä ja tutkitaan esimerkkitapauksena kokeellisesti harvoille binäärimatriiseille sovellettavan järjestelyheuristiikan soveltuvuutta kaksijakoisen graafin risteymien minimointiongelman ratkaisemiseen.

Barysentrinen heuristiikka on valittu kaksijakoisen graafin minimointialgoritmeista vertailualgoritmiksi. Se on usein käytetty heuristinen menetelmä risteymien minimointiongelman ratkaisemiseen, jossa graafin solmuja järjestetään barysentrisen arvon perusteella.

Tutkielman lähdetiedot matriisien järjestelymenetelmistä pohjautuvat pitkälti Innar Liivin [2007; 2008; 2010a; 2010b] kattavaan tutkimustyöhön matriisien järjestelymetodeista. Hän on erityisesti panostanut yhtenäistämään järjestelymenetelmiin liittyvää hajanaista terminologiaa ja tuomaan aihetta tiedonlouhintayhteisön tietoisuuteen. Toteutettu järjestelyheuristiikka perustuu Liivin [2008] väitöstyössä hahmoteltuun inventaarionhallinnan sovellusalueen algoritmiin harvoille matriiseille.

Tutkielmassa toteutetaan heuristiikat ohjelmallisesti, tehdään kokeellinen vertailututkimus ja arvioidaan heuristiikoiden soveltuvuutta ja keskinäistä paremmuutta erityyppisille matriiseille. Tarkastelunäkökulmaksi on valittu informaation visualisointi, koska se on yhä keskeisempiä tutkimusalueita. Motiivina yhdistää tutkittavia ongelma-alueita.

Luvussa 2, *Tutkimuksen taustaa*, tehdään katsaus tutkimusalueen historiaan ja esitellään sovelluksia matriisin järjestelyyn. Luvussa 3, *Matriisin järjestäminen algoritmisena ongelmana*, johdatellaan lukijaa jo tarkemmin aiheeseen esittelemällä matriisin käsittelyyn liittyvää algoritmista ongelmaa sekä matriisin järjestelyyn liittyviä tavoitteita ja algoritmeja. Luvussa 4, *Barysentrinen heuristiikka*, tutustutaan barysentrisen heuristiikan ominaisuuksiin ja algoritmiin sekä määritellään ongelma matriisin järjestelyalgoritmina. Luvussa 5, *Liivin järjestelyalgoritmi*, tutustutaan vastaavasti Liivin [2008] järjestelyheuristiikan ominaisuuksiin ja algoritmiin. Luvussa 6, *Ohjelmallinen testaus*, käydään läpi testausohjelmistoa ja aineistoja, ja luvun lopussa tulkitaan testauksen tulokset. Luvussa 7, *Loppusanat*, on lyhyt yhteenveto keskeisistä tuloksista ja tutkimuksen tarjoamista suuntaviivoista.

# 2 Tutkimuksen taustaa

## 2.1 Matriisit

Taulukko on riveistä ja sarakkeista koostuva kaksiulotteinen tiedon esitysmuoto, jonka matemaattinen vastine on matriisi. Matriisi on suorakulmainen taulukko, jossa kuhunkin (sarake, rivi)-pariin liittyy tietty arvo kunnasta $F$, usein reaali- tai kompleksilukujen joukoista. Yleisesti $m \times n$-matriisi on muotoa

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix},$$

kun $m$ on matriisin rivien määrä ja $n$ on sarakkeiden määrä. Matriisin alkio $a_{i,j}$ on alkio rivillä $i, 1 \leq i \leq m$, sarakkeessa $j, 1 \leq j \leq n$.

Listattuna on muutamia perusominaisuuksia, joita tutkittavilla matriiseilla usein on:

- Matriisi on harva, jos siinä on ei-merkitseviä arvoja enemmän kuin merkitseviä arvoja.
- Matriisi on symmetrinen, jos $a_{i,j} = a_{j,i}$ kaikille $1 \leq j \leq n$ ja $1 \leq i \leq m$.
- Matriisi on epäsymmetrinen, jos $a_{i,j} \neq a_{j,i}$ joillekin $1 \leq j \leq n$ ja $1 \leq i \leq m$.

Taulukko on yleinen tapa visualisoida dataa. Sitä käytetään laajalti tilastollisen datan esittämiseen jo yksinkertaisesti siksi, että se on hyvin yleisesti käytetty tietorakenne metatiedon ja itse tiedon esittämiseen teknisissä sovelluksissa. Esimerkiksi relaatiotietokannat käyttävät riveistä ja sarakkeista koostuvia tietokantatauluja eli relaatioita.

Tietokoneiden, tietoverkkojen ja tabulaarista esitystapaa käyttävien sovellusten yleistymisen vuoksi taulukoitua elektronista dataa on runsaasti saatavilla, ja näin ollen matriisit ovat hyvin keskeisessä roolissa teknisissä sovelluksissa.

## 2.2 Datamatriisin rakenne

Matriisin rakenne ei itsestään ota kantaa siihen, miten matriisin sisältöä tulkitaan. Riippuen siitä, ovatko viitatut joukot samat vai toisistaan poikkeavat, datamatriiseilla on erilaisia nimityksiä.

Eräs hyvin perinteinen ja yleinen tapa matriisien esittämiseen sovelluksissa on muodostaa yksittäisistä kohteista ns. ominaisuusvektoreita, ja usean rivin muodostama kokonaisuus kuvaa näin ollen useaa yksilöä. Tällöin rivit viittaavat kuvattaviin kohteisiin ja sarakkeet ominaisuuksiin, joten joukot ovat selvästi toisistaan poikkeavat.

Tämä ei kuitenkaan luonnollisesti ole aina käytännöllisin tapa esittää dataa, vaan usein käytetään matriiseja, joiden rivit ja sarakkeet viittaavat kohteisiin ja matriisin arvot viittaavat kahden eri kohteen väliseen suhteeseen. Esimerkiksi websivustoja ja kuvia on helpompi esittää muilla keinoin kuin metristen avaruuksien tarjoamien keinojen avulla. [Seo & Obermayer, 2004]

Matriiseja, joissa rivit ja sarakkeet viittaavat samaan dataan, sanotaan yksimoodiseksi dataksi, kun taas kaksimoodinen data viittaa matriiseihin, joissa rivit ja sarakkeet viittaavat eri arvoihin. Jos rivit ja sarakkeet viittaavat samaan dataan, niin ne ovat neliömatriiseja, kun taas eri joukkoihin viittaava data voi olla dimensionaalisuudeltaan millaista tahansa.

Esimerkkinä yksimoodisesta matriisista on etäisyysperusteisessa klusteroinnissa apuvälineenä käytetty $n \times n$ -etäisyysmatriisi. Alkioiden välinen etäisyys lasketaan etäisyysmatriisiin, jonka perusteella klusterointi suoritetaan. Kaksimoodisia matriiseja ovat tyypillisesti ominaisuusvektoreista koostuvat datajoukot, joissa kiinnostavia ominaisuuksia on usein kourallinen, mutta tutkimusdataa sadoista moniin tuhansiin yksilöihin.

### 2.3 Datamatriisin arvojoukot tilastollisessa datassa

Tilastollisessa datassa esiintyy usein numeerista dataa ja matriisin arvojoukot ovat metrisistä avaruuksista, esimerkiksi euklidisista avaruuksista. Euklidisten arvojen käsittelyssä voidaan hyödyntää metristen avaruuksien ominaisuuksia, mistä on hyötyä esimerkiksi etäisyysperusteisessa klusteroinnissa.

Binäärimatriiseissa matriisin arvot ovat binäärisiä, esimerkiksi *1* ja *0*. Matriisien arvojen tulkita riippuu merkityksestä, joka sille on annettu dataa kerättäessä. Matriisin arvo 0 voi merkitä matriisista riippuen esimerkiksi puuttuvaa arvoa, loogisesti negatiivista arvoa epätosi tai sen arvo voi olla jopa tutkimusdatassa oleva mittausvirhe, jolloin arvo 0 onkin todellisuudessa 1.

Matriisit, joilla esitetään kahden eri datajoukon välisten suhteiden olemassaoloa, ovat usein binäärimatriiseja. Tällaisissa binäärimatriisessa 0-arvot voidaan usein jättää huomiotta, sillä arvolla ei ole itsessään muuta funktiota kuin kertoa, ettei suhde ole voimassa.

Binäärimatriiseja tyypillisesti käytetään mm. ostoskoridatassa, jota luodaan kauppaketjun keräämästä asiakkaiden ostoskäyttäytymisdatasta. Tällaisessa matriisissa alkion arvo kuvaa ostostapahtumien ja tuotteiden välistä suhdetta. Arkeologiassa vastaavasti binäärimatriisia voidaan käyttää kuvaamaan tietyntyyppisten artifaktien ja löydöksien välisiä suhteita.

Tällaisia binäärimatriiseja kutsutaan mm. insidenssimatriiseiksi tai yhteisesiintymädataksi.

Tilastollisen arkeologian parissa tunnetaan termi määrämatriisi (engl. *abundance matrix*) [Kendall, 1971]. Insidenssimatriisi on erikoistapaus määrämatriisista, jossa alkion arvo kertoo riviobjektin määrän tai frekvenssin tiettyä sarakeobjektia kohtaan.

## 2.4 Tilastollisten datamatriisien käsittelytapoja

### 2.4.1 Matriisin alkioiden ryhmittely

Klusterointi merkitsee datajoukon jakamista *klustereihin*. Klusterissa data-alkiot mielletään keskenään samanlaisiksi. Esimerkiksi etäisyysperusteisessa klusteroinnissa määritetään samanlaisuuden mittariksi alkioiden etäisyys toisistaan ja samassa klusterissa olevat alkiot ovat siten lähellä toisiaan ja eri klustereissa olevat etäällä toisistaan. Klusterointi on helppo rinnastaa vastaaviin intuitiivisiin ihmismielen prosesseihin, joilla luokitellaan asioita.

Klusterointialgoritmit usein jaetaan sen perusteella, minkälaisia niiden tuottamien klustereiden rakenteet ovat, kuten hierarkkinen klusterointi, jossa uusi klusteri lisätään sen vanhempi-klusterin alle. Osittavassa klusteroinnissa klusterit eivät ole suorassa suhteessa toisiinsa. [Abdu & Salane, 2009].

Tunnetuimpia ja käytetyimpiä klusterointimenetelmiä on $k$:n keskiarvon klusterointimenetelmä, joka tarkoittaa aineiston jakamista $k$:hon kappaleeseen klustereita. Esimerkiksi matriisin rivi voi olla aineiston yksilö, joka valitaan satunnaiseksi klusterin ensimmäiseksi alkioksi. Yksinkertaisimmillaan klusterointi etenee niin, että datasta valitaan satunnaisesti $k$ klusteroinpistettä $n_1, n_2, \ldots, n_k$, jotka valitaan keskittymien $C_1, C_2, \ldots, C_k$ keskipisteiksi ja loput pisteet jaetaan klustereihin laskemalla pisteen etäisyys klustereihin ja sijoittamalla se lähimpään klusteriin. [Berry, 2006]

Klusterointia käytetään lukuisiin tiedonlouhintaongelmiin, kuten datan luokitteluun ohjaamattoman oppimisen keinoin. Klusterointimenetelmät ovat tehokkaita, mutta niiden tehokkuus perustuu usein siihen, että ne eivät säilytä kaikkea

informaatiota datasta, esimerkiksi klusterissa olevat kohteet eivät ole missään järjestyksessä klusterin sisällä. [Liiv, 2008]

### 2.4.2 Matriisin järjestäminen

Matriisien järjestäminen on yksinkertaisuudessaan rivien ja sarakkeiden paikkojen vaihtamista keskenään. Liivin [2008] käyttämän määritelmän mukaan se on kokeellinen tiedonanalysointimenetelmä, jolla data järjestetään sekvenssiin yksiulotteiselle jatkumolle niin, että jokaisen yksittäisen kohteen sijainti ilmentää mahdollisimman hyvin sen suhdetta koko aineistoon ja tuo esiin rakenteellista informaatiota datasta.

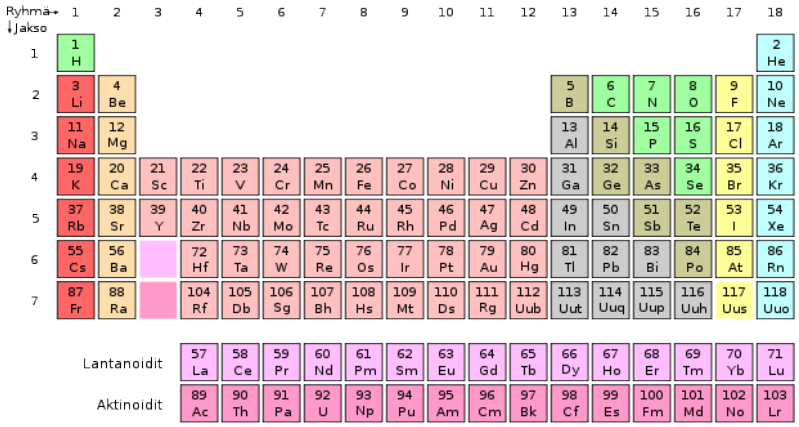
Matriisin järjestäminen viittaa joukkoon erilaisia menetelmiä, joiden tavoitteena on löytää tavoitteseen sopiva järjestys. Tavoitetta voidaan ongelma-alueesta riippuen kuvailla alustavasti tai yleisesti epätarkoilla ilmaisuilla, kuten visualisointiongelmissa järjestyksen on *vahvistettava visuaalista kykyä havainnoida hahmoja* tai rakenteita tai kuten edellisessä määritelmässä on ilmaistu, *kohteen sijainti ilmentää mahdollisimman hyvin sen suhdetta koko aineistoon*. Joissain tapauksissa, kuten datan tiivistyksessä, järjestelyn tavoite voi olla huomattavasti selkeämmin mitattavissa tai määritettävissä.

Onkin kiinnostavaa tietää, miten sopiva järjestystavoite voidaan matemaattisesti mallintaa. Sopivan järjestyksen löytämiseen liittyvästä algoritmisesta problematiikasta keskustellaan lisää luvussa 3.

Käsitteenä matriisin järjestäminen ei ole välttämättä intuitiivisesti selvä. Eroa ryhmittelyn käsitteeseen voi ainakin yrittää selventää käyttämällä esimerkkinä Mendeleevin taulukkoa alkuaineiden jaksolliselle järjestelmälle (ks. kuva 2.1).

Jaksollisen järjestelmän taulukossa alkuaineille on annettu järjestystysluvut sen perusteella, miten ne järjestyvät suhteessa muihin alkuaineisiin. Sarakkeet muodostavat ryhmiä, joiden alkuaineilla on samantyyppinen elektronikuoren rakenne, jolloin niiden kemiallinen rakenne on samantapainen. Rivit taas ovat jaksoja, jotka kertovat alkuaineen elektronikuorien määrän. Jaksollinen järjestelmä auttaa hahmottamaan alkuaineiden keskinäistä samankaltaisuutta ja on keskeisiä kemian työkaluja.

Klusterointi ja matriisin järjestely ovat optimointiongelmina myös toisistaan poikkeavia. Vaikka klusterointi ei yritäkään ratkaista samaa optimointiongelmaa kuin matriisin järjestäminen, voi se kuitenkin järjestää matriisin alkioita esimerkiksi visualisoinnin kannalta merkittävällä tavalla tai tarjota heuristisen approksimaation järjestykselle. [Hahsler *et al.*, 2008]




**Kuva 2.1** Mendeleevin jaksollisen järjestelmän taulukko. [WikimediaCommons, 2008]

### 2.4.3 Matriisin esittäminen graafina

Graafi on tietorakenne, joka koostuu solmuista ja solmujen muodostamista pareista, joita kutsutaan särmiksi. Graafi tunnetaan myös nimellä *verkko*. Graafin visuaaliseksi esitysmuodoksi on vakiintunut ns. särmä-linkki -kaavio, jossa graafin solmuja yhdistävät näiden väliset suunnatut tai suuntaamattomat särmät. Särmillä voi olla paino, kuten kauppamatkustajan ongelmassa särmiin merkityt reittien pituudet.

Matemaattisesti graafi on joukko solmuja ja näistä muodostettuja pareja, särmiä. Graafista käytetään merkintää $G = (V, E)$, jossa $V$ on joukko solmuja ja $E$ sisältää näiden väliset särmät. Graafi on *suunnattu*, kun $E$ koostuu järjestetyistä pareista. Muussa tapauksessa se on *suuntaamaton*.

Graafin $G = (V, E)$ solmut numeroidaan $1, 2, \ldots, |V|$ ja muodostetaan $|V| \times |V|$-matriisi $M = (a_{ij})$, jolle

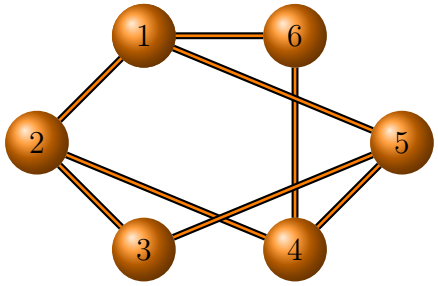
$$a_{ij} = \begin{cases} 1 & , \text{jos } (i,j) \in E \\ 0 & , \text{muutoin.} \end{cases} \tag{2.1}$$

Graafilla on vakiintuneina esitysmuotoina vierekkyyslista ja vierekkyysmatriisi. Tiheille graafeille suositaan vierekkyysmatriisiesitystä, jolloin $|E|$ on lähellä $V^2$:ta, mikä merkitsee, että solmujen välillä on paljon yhteyksiä. Vierekkyysmatriisiesityksen tilakompleksisuus on luokkaa $\Theta(n^2)$ riippumatta särmien määrästä, joten vierekkyyslista on parempi vaihtoehto esimerkiksi hyvin harvoissa graafeissa, joissa särmäjoukon koko $|E|$ on huomattavasti pienempi kuin $V^2$.

Matriisiesityksen etuna on sen yksinkertaisuus vierekkyyslistaesitykseen verrattuna, ja painottamattomilla graafeilla riittää tallentaa vain bitti kutakin joukon $V \times V$ solmuparia kohti [Cormen *et al.*, 1990].

Matriisisovelluksissa, joiden matriiseissa on tyypillisesti paljon ei-merkitseviä alkiota ja ei-merkitsevät tai nolla-alkiot tulkitaan puuttuviksi arvoiksi, voi olla hyödyllistä matriisin sijasta käyttää graafia tiedon tallentamiseen.

Kuvan 2.2 suuntaamattomasta esimerkkigraafista saadaan kuvan 2.3 vierekkyysmatriisi. Huomaa, että suuntaamattoman graafin vierekkyysmatriisi on symmetrinen.




**Kuva 2.2** Esimerkki suuntaamattomasta graafista, jossa on kuusi solmua.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | 1 | | | 1 | 1 |
| 2 | 1 | | 1 | 1 | | |
| 3 | | 1 | | | 1 | |
| 4 | | 1 | | | 1 | 1 |
| 5 | 1 | | 1 | 1 | | |
| 6 | 1 | | | 1 | | |

**Kuva 2.3** Vierekkyysmatriisi kuvan 2.2 graafista.

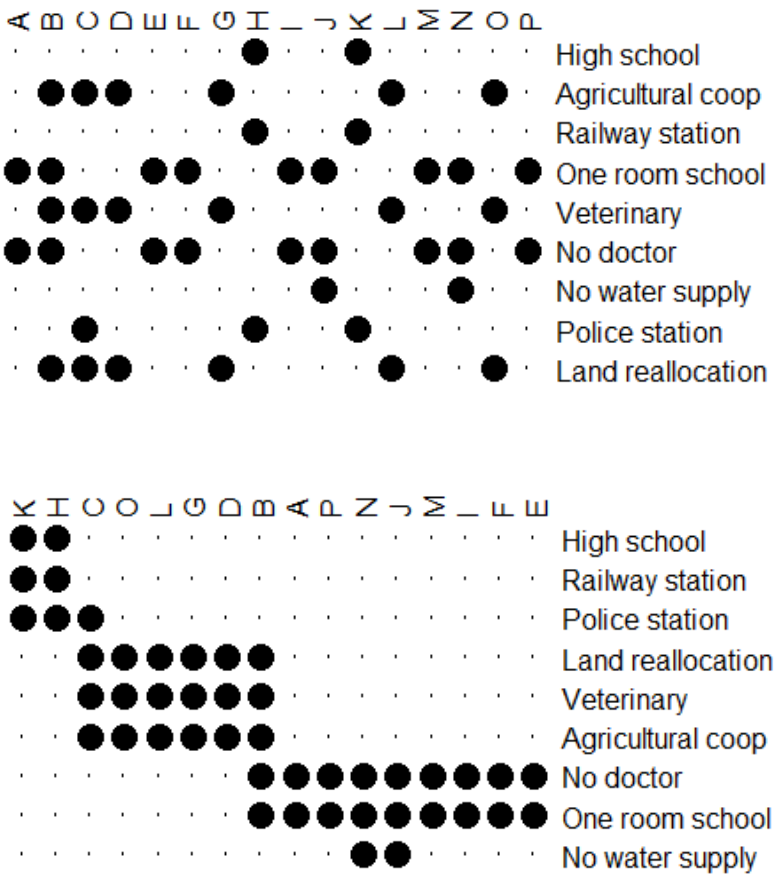
Graafin rakenteella on lukuisia sovelluksia informaation visualisoinnissa, sillä graafeja käytetään runsaasti työvälineenä havainnollistamaan rakenteellisia suhteita informaatiosta [Sugiyama, 2002], esimerkiksi prosessi- ja organisaatiokaavioissa, semanttisissa verkoissa ja ER-kaavioissa.

## 2.5 Matriisien järjestelyn historiaa

Yksi trendimuutos matriisidatan käsittelyn historiassa on käyttäjäkeskeisen datan käsittelyn merkityksen korostuminen aiemmin varsin laskentakeskeiseen käsitte-

lyyn nähden. Trendiin liittyy mm. informaation visualisoinnin nouseminen merkittäväksi tutkimusalueeksi. Kirjallisuudessa tunnetaan paljon esimerkkejä matriisien järjestelystä visuaalisen esitystavan muuttamisesta havainnollisemmaksi tai paremmaksi. Klassinen esimerkki on kaupunkidatan järjestely kuvassa 2.4.

Kuvan yläosassa on alkutilanne ja alaosassa on siitä rivejä ja sarakkeita järjestelemällä saatu uusi esitysmuoto. Järjestettyyn matriisiin on selvästi muodostunut kategorioita kylät, pikkukaupungit ja kaupungit, ja tietty ryhmä ominaisuuksia luonnehtii hyvin näitä ryhmiä. Matriisin rivien ja sarakkeiden järjestäminen on siis auttanut datan käyttäjää hahmottamaan, minkä tyyppiset palvelut ovat tyypillisiä erityyppisille taajamille.




**Kuva 2.4** Bertinin klassinen kaupunkidataesimerkki. Ylläolevassa taulukossa on alkuperäinen datamatriisi ja alemmassa kuvassa BEA-algoritmilla järjestetty matriisi. Esimerkit on generoitu R-ohjelmiston Seriation-pakkauksen palveluilla [Hahsler *et al.*, 2008]

Matriisidatan järjestelyongelman parissa ovat hieman yllättäen toimineet pioneereina muut kuin matemaatikot ja tilastotieteilijät. Arkeologian ja antropologian parissa on ollut erityistä kiinnostusta luoda informaatiota puutteellisen ja osittaisen tiedon perusteella, ja historiallisten tapahtumien, kulttuurien ja traditioiden sijoittaminen aikajärjestykseen on ollut tässä merkittävässä roolissa [Liiv, 2010a].

Egyptologi Petrie kehitti menetelmän, jolla hän ajoitti hautoja niistä löytyneen keramiikan perusteella. Hän kutsui menetelmää sekvenssiajoitukseksi. Hän ristiintaulukoi hautojen sijainnit ja niistä löydetyn keramiikan ja järjesti taulukon uudelleen vaihtamalla rivien ja sarakkeiden paikkoja kunnes kaikki suuret arvot olivat lähellä diagonaalia. Tämä oli ensimmäinen systemaattinen menetelmä matriisien järjestelyyn. [Liiv, 2008; Hahsler *et al.*, 2008]

Ilman automaattisen tietojenkäsittelyn tuomia apuvälineitä Petrie suoritti permutaatiot käsin, ja tuloksien saaminen vei päiväkausia aikaa. Lisäksi tulkinnat tehtiin hyvin subjektiivisesta ja intuitiivisesta näkökulmasta. Menetelmä sai kuitenkin runsaasti huomiota, ja se on edelleen käytössä arkeologien parissa.

Sosiologien parissa heräsi 1930- ja 1940-luvuilla kiinnostus datan järjestelytekniikoihin *sosiometrian* syntymisen myötä. Sosiometria painotti tilastollisten ja matemaattisten menetelmien käyttöä ja sosiogrammista ja sosiomatriisista tuli työkaluja suhteiden hahmottamiseen. [Liiv, 2010a]

Myöhemmin, 1950- ja 1960-luvuilla, alkoivat matriisien järjestelyn tavoitteetkin määräytyä täsmällisemmäksi. Robinsonin matriisissa permutaation tavoitteena oli matriisi, jonka alkiot olivat lähellä diagonaalia. [Liiv, 2010a].

Matriisien järjestelytekniikoilla on edelleen hyvin vakiintunut asema arkeologiassa ja antropologiassa, jossa matriisien järjestelyllä on pitkät perinteet. Lisäksi erilaisia matriiseihin liittyviä visualisointimetodeja käytetään psykologiassa ja sosiologiassa.

### 2.6 Matriisin rakenne järjestelyn lopputuloksena

Matriiseja järjestellessä datasta saattaa ilmetä säännönmukaisia hahmoja. Ilmenevät hahmot auttavat ymmärtämään dataan liittyviä sisäisiä rakenteita ja suhteita. Matriisien järjestelymetodeihin liitettyjä hahmoja ovat mm. yksiulotteinen (engl. *unidimensional*), lohkomainen (engl. *block*) ja lohkodiagonaalinen (engl. *block diagonal*) rakenne. Liiv [2008] esittelee myös *Pareto*-muodon, joka on hyödyllinen inventaarionhallintasovelluksen tuote-arvo -pareja järjesteltäessä.

Kaistamainen hahmo on tunnetuimpia ja tutkituimpia hahmoja matriiseissa.

Kaistamaisuus ilmenee mm. tunnetun matriisin kaistanleveysongelman yhteydessä ja liittyy läheisesti Robinsonin matriisin muotoon. Binäärimatriisin kaistamaisessa hahmossa datan 1-alkiot muodostavat porrasmaisen rakenteen, jossa alkiot ovat keskittyneet diagonaalin ympärille.

Kaistamaista rakennetta on havaittu tietyillä sovellusalueilla käytetyissä datamatriiseissa esimerkiksi ekologisessa ja geneettisessä datassa [Garriga *et al.*, 2008]. Kaistamainen rakenne liittyy erityisen läheisesti geenikartan muodostamiseen.

Informaation visualisoinnin kannalta hahmot ovat erityisen kiinnostavia, sillä permutoimalla matriisia saadaan datasta esiin mahdollista piiloinformaatiota. Esimerkiksi Bertinin järjesteltävään matriisiin liittyvät olennaisesti sekä paikalliset että globaalit datan väliset suhteet, jotka tulevat helpommin havaittaviksi, kun datamatriisi järjestetään. Hahmoilla ei ole merkitystä vain visualisoinnin kannalta, vaan ne liittyvät läheisesti mm. datan tiivistettävyyteen ja matriisidatan tehokkaampaan käsittelyyn.

Yleisesti tietyn hahmon tavoittelu järjestelyssä ei ole itsestäänselvää, vaan tarkoitus on saada järjestettävään dataan liittyvä luontainen rakenne tai hahmo järjestelemällä esiin. Tietyntyyppisten hahmojen tavoittelu on perusteltua tapauksissa, joissa datasta esimerkiksi tiedetään etukäteen, että sen kuuluisi noudattaa tiettyä rakennetta. Tietyillä sovellusalueilla ja datalla tietyt rakenteet ovat erittäin tyypillisiä, kuten aiemmin mainitussa geenikartoituksessa.

Seuraavaksi käydään läpi muutamia esimerkkejä matriisisovelluksista eri aloilta. Aiemmin käsiteltiin jo lyhyesti sitä, kuinka elektronisen datan määrä on kasvanut, joten käsiteltävät sovellusesimerkit sisältävät pääosin esimerkkejä, joihin kuuluu tyypillisesti suuria datamääriä.

## 2.7 Sovellusalueita

Tässä kohdassa esitellään joitakin matriisien ja matriisien järjestelyn käyttötarkoituksia. Suurin osa sovelluksista käydään yleistasolla läpi, eikä kaikkia käsitteitä välttämättä käydä tutkimusalueen laajuuden vuoksi tarkkaan läpi, joten määrittelemättömien käsitteiden selventämiseksi lukijaa suositellaan tutustumaan esimerkiksi viitattuihin lähdeteoksiin.

### 2.7.1 Bertinin mukautuva matriisi

Bertinin mukautuva matriisi on kaksiulotteinen interaktiivinen visualisointimenetelmä taulukkomuotoisen datan tarkasteluun. Matriisin järjestäminen liittyy keskeisesti Bertinin järjesteltävän matriisin käsittelyyn. Bertinin matriisilla on

sovelluksia erityisesti sellaisissa ammatillisissa tehtävissä, joissa ihmisen asiantuntemuksella ohjataan informaation käsittelyä, esimerkiksi arkkitehtuurissa ja ohjelmistosuunnittelussa [Mäkinen & Siirtola, 2005].

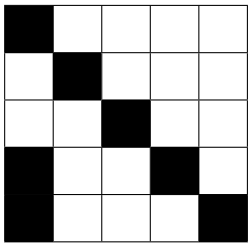
Bertinin matriisi on käytännössä kaksiulotteinen matriisi, jossa numeraaliset arvot on korvattu symboleilla. Binäärimatriisi voidaan esimerkiksi esittää muodossa, jossa arvoa 0 vastaavat tyhjät solut matriisissa ja tummennetut solut arvoa 1, kuten tehtiin jo kuvassa 2.4. Matriiseja, joiden arvot sisältävät desimaalilukuja joltain arvoalueelta, voidaan käsitellä sopivasti valitun kynnysarvon välillä myös binääriarvona. Sovelluksen käyttäjälle voidaan esimerkiksi tarjota mahdollisuus säätää kynnysarvoa sen mukaisesti, mitä arvoalueen arvoa hän pitää riittävän merkityksellisenä omiin tarpeisiinsa tai ongelmaansa nähden. [Mäkinen & Siirtola, 2000]

Matriisin arvojen jakautumista arvoalueelle voidaan mukautuvassa matriisissa havainnollistaa esimerkiksi muodostamalla eri kokoisia symboleja matriisissa. Graafisen objektin koon lisäksi voidaan myös värejä käyttää samaan tarkoitukseen.

Bertinin matriisin eräs keskeinen idea on, että sitä ei muokattaisi pysyvästi tiettyyn muotoon vaan se olisi mukautuva, kunnes kaikki matriisin rakenteeseen sisältyvä suhdetieto on havaittavissa visuaalisesti [Liiv, 2008].

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Binäärimatriisi



(b) Bertinin matriisi

**Kuva 2.5** Bertinin esitysmuoto

Kuva 2.5 esittää rinnakkain numeerisen esitysmuodon ja Bertinin matriisin.

### 2.7.2 Kaistanleveyden minimointi

Matriisin kaistanleveyden minimointi on pitkään tutkittu kombinatorinen optimointiongelma, jonka tutkimus on lähtöisin 1950-luvulta terästeollisuuden rakenneanalyysin tarpeista. Tyypillisenä lähtökohtana ongelmalle on erilaiset matriiseille tehtävät operaatiot, kuten lineaaristen yhtälöryhmien ratkaisemiseen tarvittavien harvojen kerroinmatriisien rakenne.

*Kaistanleveys* tarkoittaa ei-nolla-alkioiden maksimietäisyyttä millä tahansa matriisin rivillä. Optimointiongelmana kysytään, mikä on pienin alkioiden maksimietäisyys $b$ diagonaalilla. Esimerkiksi Gaussin eliminointi voidaan luokkaa $O(n^3)$ olevan operaation sijasta tehdä ajassa $O(bn^2)$, kun kaistanleveys on minimoitu. [Lim *et al.*, 2006]

Kaistanleveyden minimointi on NP-täydellinen ongelma, ja sille on kehitetty lukuisia heuristiikkoja. Tunnettuja kaistanleveyden minimointialgoritmeja ovat mm. leveyssuuntaiseen etsintään perustuva Cuthill-McKee-algoritmi [Cuthill & McKee, 1969] ja sen käänteisalgoritmi RCM [George & Liu, 1981], joka osoittautui käytännössä hieman alkuperäistä paremmaksi, ja myös leveyssuuntaiseen etsintään perustuva GPS-algoritmi [Gibbs *et al.*, 1976]. Hieman tuoreempia algoritmeja ovat mm. geneettiset algoritmit [Koohestani & Poli, 2010; Lim *et al.*, 2003] ja spektraalisiin menetelmiin perustuvat metodit [Del Corso & Romani, 2001]. Myös kaksijakoisen graafin minimointiongelmaan yleensä sovellettujen algoritmien soveltuvuutta kaistanleveysongelman ratkaisemiseen on testattu kokeellisesti [Abdullah & Hussain, 2006].

Kaistanleveyden minimointiongelman ratkaisut kehitettiin alunperin matriiseille, mutta myöhemmin niitä sovellettiin graafeihin. Kuten edellä jo todettiin, koska graafin voi esittää matriisina ja usein myös toisinpäin, niin graafeille sovellettuja algoritmeja voidaan soveltaa matriisien ongelmiin ja vastaavasti toisinpäin. [Cuthill & McKee, 1969; Lim *et al.*, 2006].

### 2.7.3 Inventaarion hallinta

Inventaarion hallinta on kaupallisesti merkittävä laaja sovellusalue, jossa yksilöityjen varastointiyksiköiden (engl. *stock-keeping-unit*) valtavat määrät tekevät lähtökohtaisesti mahdottomaksi niiden käsittelyn yksittäisellä tasolla. Tämän vuoksi tuotteita luokitellaan usein ryhmiin, joihin sovelletaan samoja sääntöjä.

Yhden kriteerin ABC-analyysi on perinteinen keino varastointiyksiköiden luokitteluun. Se perustuu Pareton periaatteeseen, jonka mukaan varastointiyksiköillä on niiden kysynnän ja arvon perusteella erilainen painoarvo. Yhden kriteerin ABC-analyysi nimensä mukaisesti perustuu yhden kriteerin käyttämiseen varastointiyksiköiden luokittelussa ja sillä on vaikeaa havaita kompleksisia datan keskinäisiä relaatioita, mistä huolimatta se on edelleen yleisin käytetty menetelmä inventaarion luokitteluun. Ratkaisuna kompleksisten suhteiden käsittelyyn Liiv [2008] esittää kahta menetelmää, joista ensimmäinen perustuu kattavien joukkojen louhintaan ja toinen matriisien järjestelyyn.

Kattavien joukkojen louhinta on ollut parin viime vuosikymmenen ajan tutkituimpia tiedonlouhinnan kaupallisen alan sovelluksia sen jälkeen, kun Agrawal ja muut [1993] esittelivät ostoskoridatan analysointiin assosiaatiosääntöjen louhinnan kattavien joukkojen laskennan ja Apriori-periaatteen avulla. Ostoskoridatan analyysissa tutkitaan, mitkä tuotteet esiintyvät usein yhdessä ostotapahtumissa. Syötteenä käytetään transaktiodataa, joka sisältää omina riveinään ostotapahtumista koostuvat transaktiot ja sarakkeina näihin kuuluvat tuotteet.

Transaktiodatan perusteella lasketaan algoritmisin keinoin assosiaatiosäännöt tuotteiden välille. Assosiaatiosääntö $X \Rightarrow Y$ pätee luottamusasteella $c$ transaktiodatassa $D$, kun $c$ % $D$:n transaktioista, jotka sisältää tuotteen $X$, sisältää tuotteen $Y$. Assosiaatiosäännöistä valitaan ne, joiden luottamusaste ylittää algoritmille parametrina määritetyn kynnysarvon.

Assosiaatiosääntöjen laskemiseen käytetään usein Apriori-algoritmia tai jotain sen johdannaista, joka muodostaa kattavat joukot, joiden perusteella assosiaatiosäännöt muodostetaan. Apriori-periaatteen mukaan kattavan joukon kaikkien osajoukkojen on oltava kattavia. Algoritmi käyttää annettua kynnysarvoparametria muodostamaan ensin pienimmät mahdolliset kattavat joukot, eli yhden tuotteen kattavat joukot. Seuraavalla iteraatiolla muodostetaan kandidaattijoukot, joista karsitaan sellaiset joukot, joilla on ei-kattavia osajoukkoja. Algoritmi etenee, kunnes se ei enää löydä uusia kattavia joukkoja.

Kattaviksi joukoiksi muodostuvat lopulta joukot, joiden frekvenssit ovat algoritmille asetettua kynnysarvoa suurempia. Assosiaatiosäännöistä valitaan ne, joiden luottamusaste on suurempi kuin luottamusasteen kynnysarvo, ja nämä muodostetaan kattavien joukkojen perusteella. [Agrawal & Srikant, 1994]

Kun assosiaatiosäännöt ja ABC-analyysin tulokset on laskettu, tutkitaan esiintyykö varastointiyksikölle sääntöjä, joissa $X \Rightarrow Y$, kun $X$ ja $Y$ ovat eri ABC-luokituksen mukaisia tuotteita, minkä jälkeen sovellus voi suositella uudelleenluokittelua konfliktituotteille. [Liiv, 2008]

Konfliktien laskeminen voi olla työlästä, kun riippuvuuksia on paljon ja sopivan kynnysarvoparametrin määrittäminen voi olla hankalaa. Ratkaisuna tällaisiin tilanteisiin voidaan käyttää parametritonta matriisin järjestelyyn perustuvaa menetelmää, joka muodostaa assosiaatioketjun (eng. *association chain*) transaktiodatan perusteella [Liiv, 2008]. Sovellettuun heuristiseen algoritmiin palataan tarkemmin luvussa 5.

Kattavien joukkojen käsitteeseen, sovelluksiin ja tutkimushistoriaan voi tutustua tarkemmin esimerkiksi Hanin ja muiden [2007] katsauksessa.

### 2.7.4 Klusterointilämpökartta

Bioinformatiikka on dataintensiivinen tutkimusala, jossa biologista dataa käsitellään tietojenkäsittelytieteen keinoin. Erityisesti geneettiseen dataan liittyvä tietojenkäsittely on ollut vuosikausia erittäin suuren kiinnostuksen kohteena. [Eisen *et al.*, 1998].

Klusterointilämpökartta on visuaalinen esitys datamatriisille, jonka riveihin ja sarakkeisiin liittyy klusterointipuu. Matriisinäytöllä visualisoidaan dataa eri värisävyillä. Viimeisen vuosikymmenen aikana klusterointilämpökartta on ollut erityisen suosittu bioinformatiikassa sen jälkeen, kun Eisen ja muut [1998] esittelivät sovelluksen, jossa sitä käytettiin visualisoimaan dataa geenien ilmenemisestä. Lämpökartan solun väri ilmaisi, kuinka paljon näytteessä tiettyä RNA:ta tai proteiinia oli.

Klusterointilämpökartan avulla on helppo tarkastella jopa suhteellisen suuria matriiseja: se visualisoi eri värisävyjen avulla matriisinäytöllä dataa suorakulmaisiksi väritetyiksi visuaalisesti havaittaviksi laatoiksi tai alueiksi. Matriisin toimintaan kuuluu rivien ja sarakkeiden permutointi, joten se on luonteeltaan interaktiivinen.

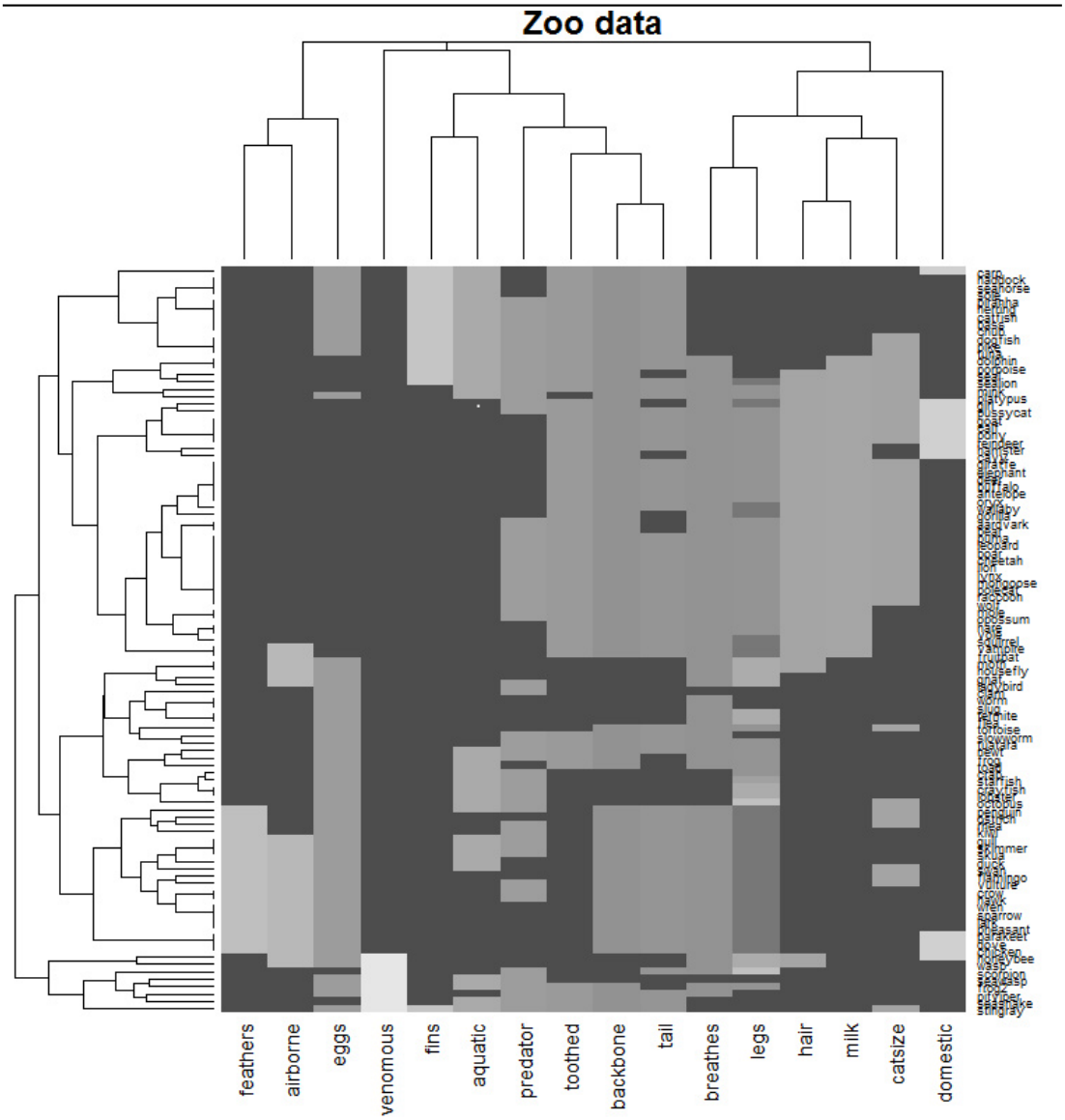
Klusterointilämpökartoissa käytetään usein hierarkkista klusterointia muodostamaan dendrogrammi, jossa lehtisolmujen järjestys tarjoaa heuristisen approksimaation matriisin objektien järjestykselle. Lehtisolmujen järjestykseen on kuitenkin tarjottu myös erillisiä optimointialgoritmeja ja se on itsessään järjestelyongelma.

Siitä huolimatta, että klusterointilämpökartta on ollut vasta viime vuosina erityisen suosittu, sillä on pitkä historia tilastotieteessä. Samantyyppisiä visualisointirakenteita on ollut esillä jo vuosikymmeniä ja klusterointilämpökartta onkin synteesi erilaisista visualisointimenetelmistä, joita tilastotieteen puolella on tutkittu jo yli vuosisadan ajan; varhaisimmat esimuodot ovat 1800-luvulta. Wilkinsonin ja Friendlyn artikkeli [2009] sisältää tarkemman katsauksen klusterointilämpökartan historiaan.

Kuva 2.6 sisältää R-ohjelmiston seriation-pakkauksella generoidun esimerkin klusterointilämpökartasta. Ohjelmistolla on mahdollista myös generoida klusterointilämpökartta järjestelyalgoritmeja hyväksikäyttäen. [Hahsler *et al.*, 2008]

### 2.7.5 Datan tiivistäminen

Suurikokoiset binäärimatriisit ovat laajasti käytettyjä erilaisissa sovelluksissa ja luonnollisesti niiden tila- ja muistivaatimukset ovat olennaisessa roolissa matriisia




**Kuva 2.6** R-pakkauksen seriation-kirjaston palveluillla generoitu klusterointi-lämpökarttaesimerkki. Datana käytetään eläintarhadataa, joka sisältää eläimiin liittyviä luonnehdintoja. [Hahsler *et al.*, 2008]

käyttävissä sovelluksissa. Useissa sovelluksissa tällaisen suuren matriisin tallentaminen levylle tai lataaminen muistiin vie runsaasti resursseja ja vaatii jotain approksimoivaa menettelyä sovellukseen. Esimerkiksi matriisien ja vektorien kertomisessa pullonkaula on usein matriisin muistivaatimukset [Willcock & Lumsdaine, 2006].

Monissa sovelluksissa iso osa sovelluksien matriiseista on harvoja matriiseja,

joten yksi tapa supistaa harvan matriisin vaatimaa tilaa on olla ottamatta huomioon nolla-alkioita matriisin tallennuksessa ja tallentaa matriisi harvana graafina [Johnson *et al.*, 2004].

Johnson ja muut [2004] hyödynsivät matriisin järjestämistä niin, että matriisien rivien peräkkäisten nollasta eroavien alkioiden esiintymien määrä minimoitiin vaihtamalla sarakkeiden paikkaa keskenään. Toteutus tehtiin esittämällä binäärimatriisin sarakkeet pisteinä korkeadimensioisessa Hamming-avaruudessa ($2^N$) ja yhdistämällä järjestelyongelma kauppamatkustajan ongelmaan. Järjestely tehtiin soveltamalla kauppamatkustajan ongelmalle kehitettyjä heuristisia menetelmiä kuten lähimmän naapurin heuristiikkaa, 2-OPT tai 3-OPT. Johnson ja muut [2004] saivat uudelleenjärjestystekniikoilla vähennettyä sekä matriisin riveihin kohdistuvaa hakua että sen käyttämää tilaa.

# 3 Matriisin järjestäminen algoritmisena ongelmana

Edellisen luvun esimerkkien tavoitteita voitaisiin ilmaista epämuodollisesti esimerkiksi käyttäen seuraavia ilmaisuja: matriisin käsittely on tehokasta, matriisin rakenteesta voidaan havaita ryhmiä tai matriisin käsittely olisi mahdollisimman esteetöntä. Matemaattisena ongelmana on muodostaa näitä tavoitteita kuvaava tavoitefunktio tai ainakin kompleksisuuden kasvaessa antaa sille jonkinlainen approksimaatio.

Rivien ja sarakkeiden järjestämistä voidaan kuvata täsmällisesti esimerkiksi Junttilan [2011] merkintätapaa käyttäen:

Vektorin $(a_1, a_2, ...a_k)$ permutaatiofunktio on kuvaus positioindeksiltä uudelle positioindeksille, eli bijektio $\Pi : \{1, 2, ...k\} \rightarrow \{1, 2, ...k\}$, tai käyttäen kokonaislukujoukosta intervallimerkintää $[k]$ vastaavalle kuvatulle joukolle $\Pi : [k] \rightarrow [k]$, jossa alkio $a_i$, $i \in 1, 2, ...k$, kuvataan alkiolle $\Pi(i)$.

Rivien järjestäminen on bijektio $\Pi : [m] \rightarrow [m]$ ja sarakkeiden järjestäminen bijektio $\Phi : [n] \rightarrow [n]$. Jos $A$ on $m \times n$ -kokoinen matriisi, niin alkio $a_{i,j}$ kuvautuu järjestetyn matriisin $A'$ alkiolle $a'_{\Pi(i),\Phi(j)}$.

Kombinatorisena ongelmana matriisin järjestelyssä on löytää sellaiset rivi- ja sarakepermutaatiot $\Pi$ ja $\Phi$, että hävikkifunktio $L$ minimoituu [Liiv, 2008]:

$$\Psi = argmin_{\Pi,\Phi} L(\Pi\Phi).$$

Etäisyysperusteisessa järjestelemisessä ongelma voidaan muotoilla käyttäen etäisyysmatriisia. Yksimoodinen ja kaksiulotteinen $n \times n$-matriisi voidaan kuvata muodollisesti seuraavasti, kun käytetään etäisyysmatriisia:

Olkoon $O = \{O_1, O_2, \ldots O_n\}$ $n$-alkioinen datajoukko ja $D = (d_{ij})$ $n \times n$-matriisi, jossa $d_{ij}$ kuvaa alkioiden $O_i$ ja $O_j$ eroa, kun $1 \leq i, j \leq n$, järjestettäessä joukon $O$ alkioita. Permutaatiofunktio on näin ollen

$$\Psi* = argmin_{\Psi} L(\Psi(D)) \text{ tai } \Psi* = argmax_{\Psi} M(\Psi(D)),$$

jossa $L$ on hävikkifunktio ja $M$ on hyötyfunktio [Hahsler *et al.*, 2008].

Monet järjestelymetodit ja kriteerit on kehitetty ensisijaisesti yksimoodiselle datalle, mutta yksimoodiselle datalle luoduilla ratkaisuilla voi kuitenkin ratkaista myös kaksimoodiseen dataan liittyviä ongelmia niin, että rivi- ja sarakejoukoista muodostetaan omat etäisyysmatriisit, joita käytetään rivi- ja sarakepermutaatioiden ratkaisemiseen erikseen. [Hahsler *et al.*, 2008]

Tutkielmassa keskitytään kuitenkin tarkastelemaan suoraan kaksimoodiselle datalle soveltuvia algoritmeja, joita pystyy soveltamaan ilman erillisiä välimuotomatriiseja. Tämä luku käsittelee kuitenkin yleiskuvauksen muodostamiseksi yksittäisten esimerkkien avulla hieman myös yksimoodisen datan järjestelyä. Kaksimoodisen datan järjestelyyn käytettäviä kriteereitä voi yleensä myös soveltaa yksimoodiseen dataan.

Useat matriisin järjestämiseen liittyvät osa-ongelmat kuuluvat NP-täydellisten tai NP-kovien ongelmien luokkaan, eikä eksaktien algoritmien käyttäminen ole näin ollen mahdollista useimmissa käytännön ongelmissa. [Mäkinen & Siirtola, 2000]

Olkoon $A$ $m \times n$- matriisi. Matriisilla $A$ on $m! \times n!$ erilaista järjestystä riveille ja sarakkeille. Esimerkiksi $8 \times 8$-matriisilla on $8! \times 8! = 1625702400$ erilaista järjestystä. Jo melko pienillä matriiseilla erilaisten järjestysten määrä on ongelmallinen, erityisesti raa'an voiman menetelmillä.

Osa järjestyksistä on isomorfisia hahmojen suhteen eli hahmot matriisissa eivät muutu, vaikka järjestykset olisivat keskenään eriävät [Mäkinen & Siirtola, 2000], joten erilaisia optimaalisia järjestyksiä voi olla useita. Tämä voi kuitenkin tavoitteesta riippuen helpottaa vain marginaalisesti tavoitteen optimointia.

### 3.1 Järjestelyn tavoite

Algoritmisena ongelmana matriisin järjestäminen tarkoittaa, että muodostetaan tavoitefunktio, jolla annettu haitta minimoituu tai hyöty maksimoituu. Erilaiset tavoitefunktiot poikkeavat siten, optimoivatko ne alkion sijaintia koko matriisin, rivin tai sarakkeen tasolla tai yksittäisen alkion naapuruston tasolla. Esimerkiksi Mooren kriteerissä optimointi määräytyy jo yksittäisen alkion naapuruston tasolla, kuten samansukuisessa von Neumannin kriteerissäkin. [Hahsler *et al.*, 2008]

Kokonaistason arviointikriteerit ovat tärkeitä, eivät vain yksittäisen järjestelytuloksen kannalta, vaan myös, kun vertaillaan eri järjestelyalgoritmien tuloksia keskenään. Useimmiten kokonaisuutta tarkasteleva tavoitefunktio muodostetaan esimerkiksi laskemalla yhteen yksittäisten alkioiden tasolla laskettuja optimointiarvoja. Tällöin kokonaistuloksen optimaalisuus on suoraan verrannollinen paikallistason optimaaliseen tulokseen. Tulos on myös riippuvainen rivi- ja saraketason optimoinnista, jos arviointiin käytetään esimerkiksi vierekkäisten rivien ja sarakkeiden etäisyyksien summaa toisistaan, mikä on suosittu hävikkifunktio järjestelymetodeissa. [Wilkinson & Friendly, 2009]

Esiteltävistä kriteeristä seuraavassa alakohdassa määritettävä vasta-Robinso-

nin muoto on yksimoodisen datan kriteeri, kun taas muut kriteerit soveltuvat lisäksi suoraan $m \times n$- matriiseille.

### 3.1.1 Robinsonin matriisin tavoitemuoto

Yksimoodisen datan järjestelyssä suosittu tapa matriisin järjestämiseen on muodostaa etäisyysmatriisi, ja etäisyysmatriisi järjestellään optimoimalla tai approksimoimalla etäisyystavoitetta, joka lasketaan etäisyysmatriisista. Tavoitefunktio voidaan muodostaa etäisyysmatriisin arvoja käyttäen.

Jo aiemmin mainittu Robinsonin matriisin muoto oli merkittävimpiä matemaattisia edistysaskeleita matriisien järjestelyalgoritmien tutkimuksessa 1950-luvulla. Siinä suurimmat alkiot ovat lähellä diagonaalia. Kuten nimi antaa jo ymmärtää, vasta-Robinsonin muoto on tälle päinvastainen, eli alkiot etääntyvät diagonaalista kasvaessaan. Vasta-Robinsonin muodossa sellaiset alkiot, joiden etäisyydet toisistaan ovat pieniä, ovat lähellä diagonaalia.

Seuraavaksi esitellään ns. kaltevuuskriteeri (engl. *gradient condition*), joka määrittää vasta-Robinsonin matriisin muodon ja pätee yksimoodiselle $n \times n$-etäisyysmatriisille: matriisi on vasta-Robinsonin muodossa, jos sen riveille pätee, että $d_{i,k} \leq d_{i,j}$, kun $1 \leq i < k < j \leq n$, ja sarakkeille pätee, että $d_{k,j} \leq d_{i,j}$, kun $1 \leq i < k < j \leq n$. Matriisin tavoitemuoto on muotoiltu rivien ja sarakkeiden tasolla.

Matriisin poikkeavuutta vasta-Robinsonin muodosta kokonaisrakenteen tasolla voi tutkia käyttäen esimerkiksi seuraavaa kriteeriä, jossa lasketaan yhteen matriisin alkiokolmikkojen etäisyysarvoja riveittäin ja sarakkeittain funktiolla $f$:

$$M(D) = \sum_{i \leq k \leq j} f(d_{i,k}, d_{i,j}) + \sum_{i \leq k \leq j} f(d_{k,j}, d_{i,j}).$$

.

Funktio $f$ määrittää, kuinka etäisyysmatriisin positioiden $i$, $j$ ja $k$ alkioiden arvot täyttävät tai rikkovat vasta-Robinsonin muotoa. Käytettävä funktio voisi olla esimerkiksi $f(x, y) = etumerkki(x - y)$. [Hahsler *et al.*, 2008].

Muihin etäisyysmatriisia käyttäviin tavoitefunktioihin voi tutustua esimerkiksi Hahslerin ja muiden [2008] yhteenvedon avulla.

### 3.1.2 Mooren ja von Neumannin kriteerit

Naapurustokriteerit optimoidaan alkion tasolla vertaamalla alkiota sen määrättyihin naapurustoalkioihin. Tunnettuja naapurustokriteerejä ovat Mooren ja von

Neumannin kriteerit, jotka ovat luultavasti tunnetuimpia käsitteitä soluautomaateista, joissa naapurustokriteerit määrittävät solun evoluution kaksiulotteisessa avaruudessa [Weisstein, 2012a]. Kriteerit tutkivat alkion ja sen naapuruston välisiä neliöityjä etäisyyksiä. Mooren kriteeri ottaa huomioon maksimissaan kahdeksan alkion naapuria kun taas Neumannin kriteeri ottaa huomioon maksimissaan neljä viereistä alkiota [Niermann, 2005].

Mooren kriteeri $m \times n$-matriisille $A$ määritellään seuraavasti:

$$s(i,j) = \sum_{k=max(1,i-1)}^{min(m,i+1)} \sum_{l=max(1,j-1)}^{min(n,j+1)} (a_{i,j} - a_{k,l})^2.$$

Vastaavasti von Neumannin kriteeri $m \times n$-matriisille $A$ määritellään seuraavasti:

$$s(i,j) = \sum_{k=max(1,i-1)}^{min(m,i+1)} (a_{i,j} - a_{k,j})^2 + \sum_{l=max(1,j-1)}^{min(n,j+1)} (a_{i,j} - a_{i,l})^2.$$

Matriisitason kriteeri saadaaan esimerkiksi laskemalla yhteen jokaisen alkion osalta tavoitefunktion arvo. Mitä lähempänä alkiot ovat naapureitaan, sitä suurempi kokonaishävikki saadaan:

$$L(A) = \sum_{i=1}^{n} \sum_{j=1}^{m} s(i,j).$$

Niermannin [2005] määritykset vastaavat von Neumannin ja Mooren naapurustoja säteellä $r = 1$ [Weisstein, 2012b]. Intuitiivisesti nämä kriteerit toimivat datoilla, joissa alkiot esiintyvät tiiviissä rykelmissä muiden alkioiden kanssa paikallistasolla.

### 3.1.3 Tehokkuuskriteeri

Tehokkuuskriteeriä (engl. *measure of effectiveness*) $m \times n$-datamatriisille $A$ käytettiin alun perin BEA-algoritmissa, ja sen tavoitteena on laskea jokaisen alkion kohdalla etäisyys neljälle vierekkäiselle naapurille ja arvo on optimaalinen, jos kaikki vierekkäiset arvot ovat lähellä toisiaan kuten von Neumannin kriteerissä.

Kokonaiskriteerissä lasketaan matriisin alkiolle $a_{i,j}$ sidosenergian vahvuus kertomalla alkion oma arvo sen samalla rivillä ja sarakkeella olevien vierekkäisten alkioiden summalla:

$$ME(A) = 1/2 \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} a_{i,j} \left[ a_{i,j+1} + a_{i,j-1} + a_{i+1,j} + a_{i-1,j} \right].$$

Kriteeri on määritelty niin, että matriisin permutaatiot, joissa esiintyy tiiviitä rykelmiä isoja arvoja, saavat suuremman tehokkuusarvon verrattuna matriisin permutaatiohin, joissa arvot esiintyvät hajanaisemmin. [McCormick *et al.*, 1972]

Tehokkuuskriteeri on sukua von Neumannin kriteerille siinä mielessä, että siinä optimoidaan tavoitefunktiota alkion paikallistasolla ottaen huomioon alkion neljä vierekkäistä alkiota. Kokonaiskriteeri on optimaalinen suhteessa alkion tason paikalliseen optimoituvuuteen. [Hahsler *et al.*, 2008]

### 3.1.4 Mukautuvuusanalyysi

Matriisin mukautuvuusanalyysissa (engl. *conformity analysis*) käydään läpi riveittäin alkioiden arvot ja vertaillaan, kuinka hyvin arvot täsmäävät vastaavassa sarakepositiossa, jolloin tavoite optimoituu. Näin riveille saadaan mukautuvuuspaino (engl. *conformity weight*). Vastaavasti tavoite voidaan laskea sarakkeittain niin, että vertaillaan alkioiden täsmäävyyttä rivipositioilla sarakkeittain.

Liiv [2008] hyödynsi tavoitetta niin, että se soveltuu harvoille binäärimatriiseille ja analyysi jättää huomiotta 0-alkiot. Tavoite sopii binäärimatriiseille, joilla 0-alkion arvo ei itsessään ole merkityksellinen, vaan viittaa puuttuvaan arvoon.

Binäärimatriisilla $A$, jolla on $m$ riviä ja $n$ saraketta, rivin $r$ mukautuvuus saadaan kaavalla

$$Conf(r) = 2 \times Conf_1(r) + ((m \times n) - e_1) - (e_r \times m), \tag{3.1}$$

kun $e_r$ on positiivisten elementtien määrä rivillä $r$ ja $e_1$ on positiivisten elementtien määrä koko matriisissa. $Conf_1(r)$ on riville 1-alkioiden eli olemassaolevien alkioiden perusteella laskettava mukautuvuus. Se saadaan kaavalla

$$Conf_1(r) = \sum_{j=1}^{n} \sum_{i=1}^{m} (\delta_{a_{r,j},a_{i,j}} \times \delta_{a_{r,j},1}), \tag{3.2}$$

kun binääriarvojen samanlaisuusfunktio on määritelty Kroneckerin delta-funktiolla:

$$\delta_{p,q} = \begin{cases} 1, & \text{jos } p = q \\ 0, & \text{jos } p \neq q \end{cases}.$$

Mukautuvuusanalyysiin palataan Liivin järjestelyalgoritmin käsittelyn yhteydessä luvussa 5.

### 3.1.5 Kolmogorov-kompleksisuus ja tiivistyksen käyttö järjestelyn arviointikriteerinä

Kolmogorov-kompleksisuus perustuu MDL-periaatteeseen, eli lyhimmän kuvauspituuden periaatteeseen, jonka mukaan datan suhteen valitaan sellainen hypoteesi, joka tarjoaa parhaimman tiivistyksen datalle. Liiv [2008] käytti Kolmogorov-kompleksisuutta yleistason kriteerinä matriisin säännöllisyyden arvioinnille. Idea perustuu siihen, että vastaavaa kriteeriä on hyödynnetty muissa tiedonlouhintaongelmissa ja datan säännöllisyyteen keskittyvän tutkimusalan läpimurtoja kannattaa hyödyntää myös matriisin järjestelyyn liittyvässä tutkimuksessa. Lisäksi motivaationa oli kehittää kokonaistason säännöllisyydelle parempi mittari.

Edellä esitetyille kokonaistason kriteereille oli tyypillistä, että niissä lasketaan yhteen paikallistason eli alkiotason optimaalisia tuloksia, jotka ovat luonnollisesti riippuvaisia määritetyn paikallistason kriteerin tuloksista. Siten muuten säännönmukainen data, joka ei esimerkiksi esiinny voimakkaasti rykelmittäin, ei välttämättä pärjää kokonaistason vertailussa. Esimerkiksi Mooren ja von Neumannin kriteerit eivät välttämättä ole vahvimmillaan datalle, joissa alkioiden lähin naapurusto on harva, vaikka kokonaistasolla matriisissa alkiot asettuisivat melko lähelle toisiaan. Paikallistason kriteerejä voisi tietysti muokata ja kehittää paremmin tunnistamaan etäisempää naapurustoa. Mooren kriteerissä on mahdollista määrätä säde, jolla naapurustoa tarkastellaan [Weisstein, 2012b].

Seuraavaksi esitellään Kolmogorov-kompleksisuuden käyttö matriisin järjestelyongelmassa formaalisti Liivin [2008] esitystä hyödyntäen.

Kaksimoodisen datan järjestelylle voidaan määritellä järjestelytavoite seuraavasti:

**Määritelmä 3.1.1** *Matriisin $A$ eri permutaatioista valitaan sellaiset permutaatiomatriisit $\Pi$ ja $\Phi$, että matriisien tulolla $\Pi A \Phi$ on lyhin mahdollinen koodaus:*

$$\Psi = argmin_{\Pi,\Phi} K(\Pi A \Phi).$$

Kolmogorov-kompleksisuus ei ole laskettavissa oleva funktio, joten tavoitetta approksimoidaan jollain tiivistysalgoritmilla, jonka jälkeen saadaan määritelmä

$$\Psi = argmin_{\Pi,\Phi} l(compress(\Pi A \Phi)),$$

kun $l$ on mielivaltaisen tiivistysalgoritmin tarjoama pituus. Kun ongelma hajotetaan rivien ja sarakkeiden suhteen erillisiksi ongelmiksi, muodostetaan hävikkifunktio, määritetään standardikoodaus $E$ ja valitaan tiivistysalgoritmiksi *gunzip*-algoritmi, saadaan muotoilu:

$$GZ(A) = 1 - \frac{min(l(compress_{GZIP}E(\Pi A\Phi)), l(compress_{GZIP}E((\Pi A\Phi)^T)))}{l(E(\Pi A\Phi))}.$$

Vertailemalla tiivistyksen tuloksia muihin algoritmeihin tiivistyksen approksimaatio antaa vertailukelpoisia tuloksia. [Liiv, 2008]

### 3.1.6 Matriisin hahmot ja hahmoja vahvistavat algoritmit

Edellisessä luvussa käsiteltiin jo lyhyesti matriisiin liittyviä hahmoja ja tuotiin esille mm. kaistamainen hahmo matriisissa, joka esiintyy esimerkiksi kaistanleveysongelmassa ja Robinsonin muodossa. Matriisin rakenteellisilla hahmoilla ei ole yhteyksiä vain informaation visualisointiin, vaan ne liittyvät läheisesti myös spesifeihin ongelmiin, mm. datan tiivistettävyyteen.

Joitakin muita tunnettuja muotoja ovat esimerkiksi jo aiemmin mainittu *yksiulotteinen*, joka on sukua kaistamaiselle hahmolle, *lohkodiagonaalinen* ja *lohkomainen* hahmo. Lohkodiagonaalista muotoa voi hyödyntää esimerkiksi QR- ja LU-matriisihajotelmien algoritmeissa, joissa tietyt aliongelmat ilmenevät lohkodiagonaalisessa hahmossa, ja nämä ongelmat voidaan ratkaista itsenäisesti. [Aykanat *et al.*, 2004]

Kaistamainen rakenne koskettaa erityisesti permutoitua matriisia, jolla on *C1P*-ominaisuus. C1P-ominaisuus tarkoittaa jatkuvien ykkösten ongelmaa (engl. *consecutive ones problem*), jossa binääridatan 1-arvoiset alkiot esiintyvät jatkuvina ilmentyminä permutoidun matriisin riveillä.

C1P-hahmo määritellään seuraavasti [Garriga *et al.*, 2008]:

**Määritelmä 3.1.1** *Binäärimatriisi $A$ on täysin kaistamainen, jos on olemassa sellaiset permutaatiot $\Pi$ ja $\Phi$ niin, että jokaisella rivillä $i$ permutoidussa matriisissa $\Pi A\Phi$ 1-alkiot sijaitsevat peräkkäisissä sarakepositioissa $\{a_i, a_{i+1}, ..., b_i\}$, kun $a_i \leq a_{i+1}$ ja $b_i \leq b_{i+1}$.*

*SCIP*-hahmo viittaa samansukuiseen hahmoon, jossa 1-alkioiden jatkuvuus pätee sekä riveille että sarakkeille.

C1P-ominaisuus edellyttää, että permutoimaton matriisi on esi-C1P-muotoista. Esi-C1P-muotoisesta matriisista on mahdollista järjestellä C1P-muotoinen matriisi polynomisessa ajassa [Atkins *et al.*, 1999], mutta reaalimaailman data on harvoin edes esi-C1P-muotoista. Tällaisista matriiseista voi olla kuitenkin mahdollista löytää alimatriiseja, joilla on C1P-ominaisuus, mikä on hyödyllistä esimerkiksi kun geenejä kartoitetaan kromosomeiksi.

Viime vuosina on tutkittu paljon myös algoritmeja, joiden tarkoituksena on maksimoida tiettyjä hahmoja tai selvittää, millä etäisyydellä matriisi on tietystä hahmosta. Rakenteen maksimointi on olennaista, sillä reaalidatassa esiintyy usein virheitä ja puutteita. Jos ongelmalla on säännöllinen hahmo matriisissa, etäisyydellä on mahdollista tutkia datassa esiintyvien virheiden määrä. [Junttila, 2011]

Binäärimatriisin etäisyys tietystä hahmosta voidaan määritellä käyttämällä etäisyysmittareina *käännä(1,0)*- ja *käännä(0,1)*-operaatioita, joiden arvolla määritellään, kuinka yksittäisten alkioiden arvojen muutoksilla matriisi tai alimatriisi saavuttaa tietyn rakenteen ja näin ollen saadaan alkuperäisen matriisin etäisyys laskemalla käytettyjen operaatioiden määrä. [Junttila, 2011; Garriga *et al.*, 2008]

Kiinnostavinta tutkielman näkökulmasta on, että graafien hahmoja pystyy esittämään matriisien hahmoina ja toisinpäin. Kaksijakoisen graafin täydellisen kaksijakoisen aligraafin, biklikin, voi yhdistää suorakulmaiseen hahmoon matriisissa. [Junttila, 2011]

## 3.2 Algoritmeja järjestelyongelmaan

Tässä kohdassa annetaan lyhyt katsaus käytössä oleviin järjestelyalgoritmeihin, keskittyen heuristisiin algoritmeihin. Joitakin metodeja on jo jossain määrin sivuttu tai esitelty etukäteen, kuten hierarkkista klusterointia. Hahsler ja muut [2008] esittelevät etäisyysmatriisia käyttäviä järjestelyalgoritmeja, mm. TSP-algoritmin, joka ratkaisee järjestelyongelman etsimällä lyhimmän Hamiltonin polun, joka vastaa ongelmana järjestelyongelmaa, hierarkkisen klusteroinnin ja pienille datoille käytettäviä eksakteja algoritmeja, joista tunnetuimpia ovat dynaamiseen ohjelmointiin perustuva algoritmi ja branch-and-bound -menetelmään perustuva algoritmi. [Brusco & Stahl, 2005].

### 3.2.1 BEA

Sidosenergia-algoritmi eli BEA (*bond energy algorithm*) on jo aiemmin tehokkuuskriteerin yhteydessä mainittu algoritmi, joka käyttää ME-kriteeriä järjeste-

lykriteerinä.

Sidosenergia-algoritmi valitsee ensin satunnaisen matriisin sarakkeen ja sijoittaa sen mielivaltaiseen kohtaan ja asettaa $i = 1$. Sen jälkeen jäljellä olevia sarakkeita yritetään sijoittaa jo sijoitettujen sarakkeiden viereen $i + 1$ vaihtoehtoiseen paikkaan. Algoritmi laskee sijoituksien tuottaman ME-arvon jokaiselle sarakkeelle ja sijoittaa eniten arvoa korottaneen sarakkeen parhaalle vaihtoehtoiselle paikalle. Lopulta laskennassa tarvittavaa arvoa $i$ kasvatetaan yhdellä ja proseduuria toistetaan, kunnes kaikki sarakkeet on sijoitettu. Sen jälkeen sama proseduuri toistetaan riveille. [McCormick *et al.*, 1972]

BEA oli aikanaan läpimurto matriisin järjestelyn alalla. Se oli tiettävästi ensimmäisiä järjestelyalgoritmeja, jotka sopivat sekä yksi- että kaksimoodiselle datalle. Ongelma oli mahdollista eriyttää rivien ja sarakkeiden suhteen eri aliongelmiksi ja lisäksi sen on todettu toimivan lähes optimaalisesti riippumatta matriisidatan rakenteesta [Liiv, 2008].

### 3.2.2 Spektraaliset algoritmit

Spektraaliset järjestelymetodit pohjautuvat spektraaliseen graafiteoriaan ja ne käyttävät Laplacen matriisin ominaisuusvektoria ja PQ-puuta matriisin järjestelyyn. Jos matriisilla on C1P-ominaisuus, niin se on mahdollista löytää polynomisessa ajassa spektraalista järjestelyä käyttäen. [Atkins *et al.*, 1999]

Vaikka Atkins ja muut [1999] alun perin osoittivatkin, että spektraalinen järjestely löytää C1P-permutaation virheettomistä esi-C1P -instansseista, niin spektraalista järjestelyä on menestyksekkäästi sovellettu myös C1P-ominaisuuden maksimointiin ja kaistamaisten rakenteiden etsimiseen datasta, joka ei ole esi-C1P-muotoista. [Garriga *et al.*, 2008]

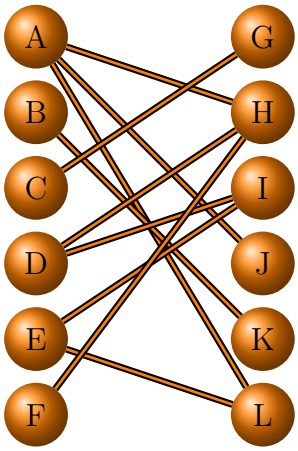
C1P-ominaisuuden maksimoinnin on todettu olevan ekvivalentti Fröbius-normin pienentämisongelman kanssa. [Vuokko, 2010].

## 3.3 Järjestelyalgoritmien ohjelmallisia sovelluksia

R [2013] on erityisesti tilastollisen dataan käytetty ohjelmisto, jonka ohjelmistoon Hahsler ja muut [2008] ovat toteuttaneet tilastollisen datan tutkimiseen *seriation*-kirjaston, jossa on käytettävissä mm. BEA-algoritmi matriisidatan järjestelyyn. Useimmat kirjaston tarjoamat algoritmit on tarkoitettu yksimoodiselle matriisidatalle, mutta *Stress*- ja *ME*-algoritmeja voi käyttää suoraan myös $m \times n$-matriiseille. Tutkielman esimerkkikuvat on toteutettu kirjaston tarjoamilla esimerkkidatoilla ja esimerkeillä.

Liiv ja muut [2012] ovat julkaisseet avoimen lähdekoodin *Visual Matrix Explorer* -ohjelmiston binäärimatriisien analysoimiseen ja tutkimiseen. Ohjelmisto on saatavilla sähköisesti SourceForgen sivustolta [VME, 2011], ja se on tarkoitettu työkaluksi tutkijoille kaikenlaisen tabulaarisessa muodossa esitettävän datan tutkimiseen.

# 4 Barysentrinen heuristiikka




**Kuva 4.1** Esimerkki kaksijakoisesta graafista, jossa on solmut $V = \{A, B, C, D, E, F, G, H, I, J, K, L\}$, jotka jakautuvat erillisiin joukkoihin $V_1 = \{A, B, C, D, E, F\}$ ja $V_2 = \{G, H, I, J, K, L\}$.

Kaksijakoisen graafin solmut ja niiden väliset särmät muodostavat rakenteen, joka esiintyy monissa käytännön sovelluksissa, kuten sosiaalisten verkkojen analyysissa ja monissa muissa ongelmissa, joissa mallinnetaan kahden eri joukon välisiä suhteita.

Kaksijakoisessa graafissa $G = (V_1 \bigcup V_2, E)$ on $V_1 \bigcap V_2 = \emptyset$. Graafi jakautuu siis kahteen erilliseen joukkoon, eikä ole olemassa sellaisia solmuja $v_i \in V_1$ ja $v_2 \in V_1$, joiden välillä olisi yhteys, ja vastaava pätee joukkoon $V_2$. Risteymien minimointiongelma kysyy, miten graafi piirretään niin, että solmut sijaitsevat kahdella samansuuntaisella suoralla ja särmien risteämiä on mahdollisimman vähän. Tämä tarkoittaa joukon $V_1$ järjestämistä toiselle ja joukon $V_2$ solmujen sijoittamista toiselle suoralle.

## 4.1 Kaksijakoisen graafin esittäminen matriisimuodossa

Kuten alakohdassa 2.4.3 esitettiin, graafin voi esittää vierekkyysmatriisiesityksenä. Kaksijakoiselle graafille $G = (V, E)$, jossa $V = (V_1 \bigcup V_2)$, voidaan muodostaa vierekkyysmatriisi esimerkiksi numeroimalla solmut seuraavasti $v_{11}$, $v_{12}$, ..., $v_{1m}$, $v_{21}$, $v_{22}$, ...,$v_{2n}$, kun $|V_1| = m$ ja $|V_2| = n$, ja liittämällä nämä taulukon riveihin ja sarakkeisiin, sekä särmät $e \in E$ taulukon arvoihin kuvan 4.2 mukaisesti. Koska $(V_1 \bigcap V_2) = \emptyset$, huomataan, että suurin osa matriisin soluista jää ilman arvoa.

| | $v_{11}$ | $v_{12}$ | $\dots$ | $v_{1m}$ | $v_{21}$ | $v_{22}$ | $\dots$ | $v_{2n}$ |
|---|---|---|---|---|---|---|---|---|
| $v_{11}$ | – | – | – | – | $e(v_{21}, v_{11})$ | $e(v_{22}, v_{11})$ | $\dots$ | $e(v_{2n}, v_{11}$ |
| $v_{12}$ | – | – | – | – | $e(v_{21}, v_{12})$ | $e(v_{22}, v_{12})$ | $\dots$ | $e(v_{2n}, v_{12}$ |
| $\vdots$ | – | – | – | – | $\dots$ | $\dots$ | $\dots$ | $\vdots$ |
| $v_{1m}$ | – | – | – | – | $e(v_{21}, v_{1m})$ | $e(v_{22}, v_{1m})$ | $\dots$ | $e(v_{2n}, v_{1m})$ |
| $v_{21}$ | $e(v_{21}, v_{11})$ | $e(v_{21}, v_{12})$ | $\dots$ | $e(v_{21}, v_{1m})$ | – | – | – | – |
| $v_{22}$ | $e(v_{22}, v_{11})$ | $e(v_{22}, v_{12})$ | $\dots$ | $e(v_{22}, v_{1m})$ | – | – | – | – |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | – | – |
| $v_{2n}$ | $e(v_{2n}, v_{11})$ | $e(v_{2n}, v_{12})$ | $\dots$ | $e(v_{2n}, v_{1m})$ | – | – | – | – |

**Kuva 4.2** Graafin vierekkyysmatriisi kaksijakoiselle graafille

Kuten alakohdassa 2.4.2 mainittiin, suuntaamattoman graafin vierekkyysmatriisi on symmetrinen. Kaksijakoiselle suuntaamattomalle graafille $A$ matriisiesitys on siis muotoa

$$A = \begin{bmatrix} 0 & B \\ B^\intercal & 0 \end{bmatrix}.$$

Suuntaamaton kaksijakoinen graafi voidaan siis vierekkyysmatriisin sijasta esittää tiivimmässä muodossa käyttämällä pelkästään vierekkyysmatriisin alimatriisia $B$. Kaksijakoisen graafin matriisiesityksessä rivit esittävät toista solmujoukkoa ja sarakkeet toista.

Kaksijakoisessa graafissa saman solmujoukon solmuilla ei ole suhteita itseensä, joten tässä esitysmuodossa ei kadoteta mitään informaatiota. Kuvassa 4.3 on esimerkkinä kuvan 4.1 graafi taulukkomuodossa.

| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ |
|---|---|---|---|---|---|---|
| $A$ | 0 | 1 | 0 | 1 | 0 | 1 |
| $B$ | 0 | 0 | 0 | 0 | 1 | 0 |
| $C$ | 1 | 0 | 0 | 0 | 0 | 0 |
| $D$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $E$ | 0 | 0 | 1 | 0 | 0 | 1 |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 |

**Kuva 4.3** Kuvan 4.1 graafin binäärimatriisi

## 4.2 Kaksijakoisen graafin piirtäminen mahdollisimman vähillä risteymillä

Kaksijakoisen graafin piirtäminen niin, että solmuryhmien väliset särmät risteytyvät mahdollisimman vähän, on tunnettu ja paljon tutkittu optimointiongelma.

Barysentrinen heuristiikka kuuluu Sugiyaman algoritmiksi kutsuttuun hierarkkisen graafin piirtotekniikkaan [Sugiyama *et al.*, 1981; Sugiyama, 2002]. Hierarkkinen graafi on monitasoinen graafi, jossa hierarkkisen tason edustaman joukon välillä on suhteita hierarkkisella tasolla ylempään ja/tai alempaan joukkoon. Hierarkkiset graafit esiintyvät monissa hierarkkisia rakenteita kuvaavissa kaavioissa ja monimutkaisten hierarkkisten rakenteiden kuvaaminen manuaalisesti voi olla työlästä ja aikaavievää, minkä takia kyseisen tehtävän automatisointi on mielekästä. Kuten monien kompleksisten rakenteiden hahmottamisessa, niin tässäkin säännölliset alirakenteet auttavat hahmottamaan kokonaisrakennetta paremmin.

Sugiyaman algoritmi on kuvaus hierarkkisen graafin piirtämiseksi mahdollisimman helposti tulkittavissa olevaan muotoon. Algoritmi kuvaa graafin piirtämisessä tarvittavat erilliset vaiheet, joita ovat

1. syklien poisto,

2. horisontaalinen sijoitus,

3. risteymien minimointi ja

4. horisontaalisten koordinaattien sijoitus.

Ensimmäisessä vaiheessa mahdolliset suunnatut sykliset graafit muutetaan asykliksiksi, jotta hierarkia olisi aito.

Toisessa vaiheessa solmut sijoitetaan horisontaalisiin tasoihin, jolloin solmut muodostavat hierarkkisen rakenteen. Lisäksi tässä vaiheessa useamman tason läpi kulkevien särmien kohdalle jokaiselle väliin jäävälle tasolle lisätään apusolmut, ja muutetaan jokainen tällainen särmä useammaksi särmäksi, jotka yhdistävät toisiinsa apusolmut sekä alkuperäisen korvatun särmän alku- ja loppusolmut.

Kolmannessa vaiheessa käydään läpi jokainen taso ja järjestellään aina samassa horisontaalisissa kerroksessa olevat solmut niin, että risteymien määrä minimoituu kahden tason välillä. Neljännessä vaiheessa solmuille annetaan vertikaaliset koordinaatit.

Yleisessä tapauksessa risteymien minimointi koostuu monitasoisen hierarkian risteymien minimoinnista. Tämä ositetetaan usein kaksitasoisen hierarkian risteymien minimoinniksi.

Jo kaksitasoisen hierarkian toispuoleinen risteymien minimointi kuuluu NP-kovien ongelmien luokkaan [Eades & Wormald, 1994], joten heuristisiin ratkaisuihin turvautuminen on perusteltua.

### 4.3 Risteymien minimointi matriisien järjestelyongelmana

Ongelmana matriisin järjestäminen sarakkeita ja rivejä permutoimalla on verrattavissa kaksijakoisen graafin risteymien minimointiongelman kanssa, sillä kaksijakoinen graafi voidaan esittää muodossa, jossa ensimmäinen ryhmä solmuja $V_1$ vastaa matriisin rivejä ja vastaavasti toinen solmujoukko $V_2$ sarakkeita. Joko matriisien rivien tai sarakkeiden järjestelyä voidaan ajatella esimerkiksi toispuoleisena risteymien minimointiongelmana.

Mäkinen ja Siirtola [2005] ovat tutkineet barysentrisen heuristiikan soveltuvuutta Bertinin mukautuvan matriisin rivien ja sarakkeiden järjestelyyn. Idea perustuu matriisien ja graafien väliseen rakenteelliseen yhteyteen, minkä vuoksi sekä graafiteorian ongelmia on voitu soveltaa matriisiongelmiin ja päinvastoin [Junttila, 2011].

Mäkinen ja Siirtola [2005] tekivät tärkeän huomion, että barysentrisen heuristiikan järjestelyllä oli taipumusta johtaa rakenteeseen, jossa alkiot ovat keskittyneet lähelle diagonaalia muistuttaen esimerkiksi kohdassa 2.6 mainittua yksiulotteista rakennetta. Hahmojen yhtenevyys antaa vihjeen, että ongelmilla on jotain yhteistä ja matriisien järjestelyongelmaan kehitettyjä ratkaisuja voisi mahdollisesti sovittaa myös risteymien minimointiongelmaan ja päinvastoin.

Myös kaistanleveyden minimointiongelman rakenteellinen tavoite muistuttaa yksiulotteista rakennetta. Abdullah ja Hussain [2006] ovatkin jo soveltaneet risteymien minimointiongelman ratkaisualgoritmeja kaistanleveyden minimointiongelmaan saman havainnon perusteella. Muun muassa barysentrinen heuristiikka, mediaaniheuristiikka ja useita metaheuristiikkoja oli tutkimuksen kohteena.

Tutkielman tavoitteena on järjestellä kaksimoodista binääristä matriisidataa risteymien minimointialgoritmilla ja formalisoida risteymien minimoinnin järjestelyongelmana. Tarkasteltava data on syklitöntä ja suuntaamatonta, joten Sugiyaman graafien piirtoalgoritmin ensimmäinen vaihe ei ole kiinnostava. Horisontaalisessa sijoituksessa solmut sijoitetaan horisontaalisiin kerroksiin, tämä vastaisi eriyttämistä ja sijoittamista positioihin. Kaksimoodisessa matriisissa tämä erot-

telu on sisäänrakennettu.

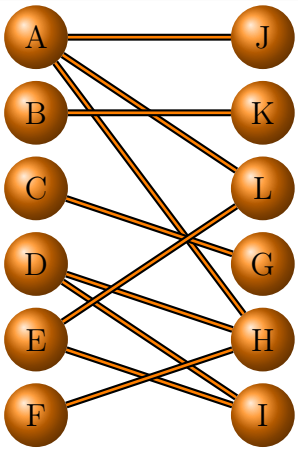
Koska tarkoituksena on käsitellä risteymien minimointiongelmaa järjestelyongelmana, muodostamme sen samaan tapaan kuin muut optimointialgoritmit. Tarkoitus on löytää sellaiset rivipermutaatio $\Pi$ ja sarakepermutaatio $\Phi$, että risteymien summa minimoituu, eli edellä määrittämämme hävikkifunktio minimoituu:

$$\Psi = argmin_{\Pi,\Phi}(L(G)). \tag{4.1}$$

Risteymien minimointi vastaa sarakkeiden ja rivien permutointia optimointialgoritmin suhteen, joka määritellään analogisesti risteymien minimoinnin kanssa.

Kaksijakoisen graafin matriisiesityksessä barysentrinen heuristiikka toimii niin, että ongelma jaetaan erillisiksi ongelmiksi sarakkeiden ja rivien järjestämisen suhteen. Jokaisella mahdollisella järjestelykierroksella vaihdetaan siis vain joko rivien tai sarakkeiden järjestystä ja toinen solmuryhmä pysyy paikallaan. Jokainen solmu $v \in V$ käydään läpi, huomioidaan solmuun yhteydessä olevien solmujen järjestysnumerot ja lasketaan niistä keskiarvo. Näin saadaan solmuille barysentrinen arvo ja solmut järjestetään nousevaan järjestykseen barysentrisen arvon mukaan. Järjestelyä voidaan toistaa vuorotellen eri solmujoukoilla kunnes risteymien määrä ei enää parane.

Kuvassa 4.4 on esimerkki barysentrisestä järjestelystä kuvan 4.1 kaksijakoiselle graafille. Risteymien määrä on vähentynyt alkuperäisestä yhdeksään.




**Kuva 4.4** Kuvan 4.1 graafin oikeanpuoleinen taso järjestettynä barysentrisen heuristiikan mukaan.

### 4.4 Risteymien määrä

Risteymien lukumäärän laskemiseksi tarvitaan tehokas algoritmi. Tämä on tunnettu aliongelma, jota kutsutaan kahden tason risteymien laskentaongelmaksi (engl. *bilayer cross counting problem*). Tämä vaihe voi osoittautua pullonkaulaksi Sugiyaman algoritmia sovellettaessa. [Barth *et al.*, 2002]

Oletetaan, että $e_1 = \{v_1, v_2\} \in E$ ja $e_2 = \{u_1, u_2\} \in E$ ja $v_1 \in V_1$ ja $v_2 \in V_2$ ja $pos_{V_i}$ on funktio, joka määrittelee solmun position tasossa $i$. Esimerkiksi, $pos_{v_i}$ tasolle 1 voisi olla bijektio $pos_{V_1} : V_1 \rightarrow [|V_1|]$ ja tasolle 2 bijektio $pos_{V_2} : V_2 \rightarrow [|V_2|]$.

Särmät $e_1 \in E$ ja $e_2 \in E$ risteävät, kun $pos_{V_1}(v_1) > pos_{V_1}(u_1)$ ja $pos_{V_2}(v_2) < pos_{V_2}(u_2)$ tai $pos_{V_1}(v_1) < pos_{V_1}(u_1)$ ja $pos_{V_2}(v_2) > pos_{V_2}(u_2)$.

Risteymien määrä olisi helppo laskea tutkimalla kaikki särmäparit ja tutkia, risteävätkö särmät. On kuitenkin olemassa tehokkaampiakin lähestymistapoja. Barth ja muut [2002] ovat osoittaneet, että ongelman voi palautta sekvenssin inversioiden laskemiseen ja esittävät algoritmin, joka on aikavaatimukseltaan luokkaa $O(|E|log|V_{min(|V_1|,|V_2|)}|)$ ja joka käyttää akkumulaattoripuuksi (engl. *accumulator tree*) kutsuttua rakennetta.

Akkumulaattoripuu $T$ on tasapainotettu binääripuu, jossa on $2^c$ lehtisolmua, joista $q$ ensimmäistä liittyy särmäjoukon loppusolmujen positioihin. Akkumulaattoripuu toteutetaan $(2^{c+1} - 1)$-kokoisessa taulukossa, jossa $2^{c-1} < q = |V_2| \leq 2^c$. Taulukossa juuri on positiossa 0 ja positiossa $i$ olevan solmun vanhempi on positiossa $\lfloor \frac{i-1}{2} \rfloor$.

Algoritmi 4.1 noudattaa Barthin ja muiden [2002] kuvausta. Algoritmissa $\pi_{E_{V_2}}$ on permutaatio $\langle v_1^2, v_2^2, .., v_q^2 \rangle$, joka sisältää joukkoon $V_2$ kuuluvien särmien päätesolmujen järjestyksen järjestyksessä $\pi_E$. $\pi_E$ on sellainen särmien järjestys, että särmät ensisijaisesti järjestetään joukon $V_1$ alkusolmun positioiden suhteen ja toissijaisesti joukon $V_2$ loppusolmun positioiden mukaan.

Algoritmissa järjestetään ensin särmäjoukko ja saadaan järjestys $\pi_E$, minkä jälkeen muodostetaan akkumulaattoripuu lisäämällä särmät yksi kerrallaan ja lisätyn särmän loppusolmua vastaavasta lehtisolmusta alkaen kuljetaan kohti juurta. Jokainen käynti puun solmussa kasvattaa solmuun tallennettua vierailujen määrää. Jokainen käynti vasemmassa lapsisolmussa kasvattaa risteymien määrää oikeanpuoleiseen sisarsolmuun tallennetulla määrällä.

Algoritmia käyttäen saamme yhtälössä 4.1 käytettävän hävikkifunktion, joka laskee kaikki risteymät yhteen:

$$L(G) = laskeRisteymat(G).$$

**Algoritmi 4.1:** laskeRisteymat

**Syöte**: kaksijakoinen graafi $G = (V_1, V_2, E)$
**Tuloste**: risteymien määrä $c$

```
radixSort(π_e) ;
treesize ⟵ 2^(q+1) − 1;
firstindex ⟵ 2^q − 1;
T ⟵ array[treesize];
for i = 0; i < treesize; t++ do
    T[i] ⟵ 0;
end
c = 0;
for k = 0; k < |E|; k++ do
    index ⟵ π_{E_{V_2}}[k] + firstindex;
    T[index]++;
    while index > 0 do
        if index%2 then
            c ⟵ c + T[index + 1];
        end
        index ⟵ (index − 1)/2;
        T[index]++;
    end
end
return c;
```

### 4.5 Heuristisia algoritmeja risteymien minimointiongelmaan

Risteymien minimointiongelmalle on toteutettu lukuisia heuristisia algoritmeja. Luultavasti tunnetuimmat algoritmit ovat jo aiemmin mainittu barysentrinen heuristiikka ja mediaaniheuristiikka. [Eades & Wormald, 1994]

Mediaaniheuristiikka on samantapainen kuin barysentrinen heuristiikka, mutta solmut järjestetään mediaanin eikä keskiarvon mukaan. Barysentrinen heuristiikka ja mediaaniheuristiikka ovat säilyttäneet keskeisen roolinsa risteymien minimointiongelman ratkaisussa siitäkin huolimatta, että useita uusia heuristiikkoja on kehitetty vuosien varrella, kuten paikalliseen etsintään perustuva sopeutuva

lisäys (engl. *adaptive insertion*) [Stallmann *et al.*, 2001] ja graafin vierekkyysmatriisiesitystä hyödyntävä algoritmi [Zhang *et al.*, 2010].

Vaikka mediaaniheuristiikka ja barysentrinen heuristiikka ovat samantyylisiä algoritmeja, niillä on teoreettisia eroavaisuuksia mediaaniheuristiikan eduksi: Eades ja Wormald [1994] ovat osoittaneet, että barysentrinen heuristiikka voi tehdä jopa luokkaa $O(\sqrt{n})$ olevan suhteellisen virheen, kun graafin solmut kuvautuvat täsmälleen yhteen $x$-koordinaattiin piirroksessa. Li ja Stallmann [2002] ovat osoittaneet, että raja on tätä luokkaa myös graafeille, joissa ei ole eristettyjä solmuja. Mediaaniheuristiikan tapauksessa tulos on korkeintaan kolme kertaa optimaalista suurempi. Mediaanit eli mediaaniheuristiikassa järjestettävät arvot ovat kokonaislukuja väliltä $[0, ..., n-1]$, jolloin lasketut arvot voidaan järjestää lineaarisessa ajassa [Stallmann *et al.*, 2001].

Vaikka muun muassa näiden seikkojen perusteella mediaaniheuristiikkaa pitäisi suosia barysentrisen heuristiikan sijasta, barysentrisen heuristiikan on osoitettu toimivan käytännössä paremmin erityisesti harvoilla graafeilla [Li & Stallmann, 2002; Jünger & Mutzel, 1996]. Myös Eades ja Wormald [1994] huomauttavat, että useissa toisistaan erillisissä tutkimuksissa barysentrinen heuristiikka on toiminut käytännössä paremmin.

### 4.6 Barysentrinen algoritmi

Algoritmi 4.2 on yksinkertainen toteutus barysentrisen heuristiikasta suuntaamattomalle kaksijakoiselle graafille. Algoritmi toimii ajassa $O(|E| + |V_1|log|V_1|)$, kun järjestelyalgoritmina on tehokas vertailuperusteinen järjestelyalgoritmi. Jotkut toteutukset ovat kompleksisuudeltaan parempiakin, kuten viistopuurakenteeseen (engl. *splay tree*) perustuva barysentrinen järjestely, jonka tasoitettu kompleksisuus on luokkaa $(|V|log|V|)$ [Eiglsperger *et al.*, 2005].

Algoritmissa suoritetaan risteymien minimointi järjestelemällä solmujoukon $V_1$ alkiot. Barysentriset arvot joukon solmuille saadaan ensin käymällä särmät läpi ja lisäämällä särmistä saatu positiotieto. Funktio $Sort()$ järjestää joukon laskettujen arvojen perusteella.

**Algoritmi 4.2:** barysentrinenJarjestely

**Syöte**: Kaksijakoinen suuntaamaton graafi $G = (V_1, V_2, E)$

**Tuloste**: Permutaatio $p$

**foreach** $u \in V_1$ **do**
    $positionsums[i] \longleftarrow 0;$
    $neighbours[i] \longleftarrow 0;$
    $barycenters[i] \longleftarrow 0;$
**end**

/* $u \in V_1, v \in V_2$ */
**while** $(u, v) \in E$ **do**
    $positionsums[u] \longleftarrow positionsums[u] + pos_{v_2}(v);$
    $neighbours[u] + +;$
**end**
**for** $i \longleftarrow 0; i < |V_1|; i + +$ **do**
    **if** $neighbours[i] \neq 0$ **then**
        $barycenters[i] \longleftarrow positionsums[i]/neighbours[i];$
    **end**
**end**
`sort(`$bc$`);`
**for** $i \longleftarrow 0; i < |V_1|; i + +$ **do**
    $p[i] \longleftarrow barycenters.removeFirst();$
**end**
**return** $p$;

# 5 Liivin järjestelyalgoritmi

Liivin [2008] järjestelyalgoritmi on harvoille binäärimatriiseille soveltuva algoritmi, joka perustuu Vyhandun [1980; 1986] ja Mullatin [1976; 1977; 1978] monotonisia järjestelmiä koskevaan tutkimustyöhön. Monotoniset järjestelmät viittaavat järjestelmiin, joissa alielementtiin kohdistuvat negatiiviset tai positiiviset toimenpiteet monotonisesti nostavat tai laskevat alielementin määriteltyä vaikutusta suhteessa koko järjestelmään. Monotonisia järjestelmiä voidaan määrittää esimerkiksi graafeille tai matriiseille. Esimerkiksi elementti voi olla graafin $G = (V, E)$ solmu $v \in V$, ja elementin poisto tarkoittaa kaikkien solmusta $v$ lähtevien särmien poistamista graafista $G$ ja vastaavasti positiivinen toimenpide solmun $v$ ja sen särmien lisäämistä graafiin $G$.

Liiv [2008] esittelee kaksi variaatiota algoritmista. Toinen on esitetty relaatiotietokantoja hyödyntävänä SQL-kyselymuodossa ja toinen on proseduraalinen algoritmi. Näistä käsittelemme vain jälkimmäistä. Liivin algoritmi perustuu binäärisen arvojakauman hyödyntämiseen, miinustekniikkaan ja mukautuvuusanalyysiin.

## 5.1 Mukautuvuuspainon laskeminen ja järjestäminen mukautuvuuspainon mukaan

Liivin algoritmissa käytetään mukautuvuusanalyysin mukaista vertailuarvoa. Mukautuvuusanalyysiä käsiteltiin jo alakohdassa 3.1.4. Mukautuvuusanalyysissä rivin mukautuvuus määräytyy sen mukaan, kuinka hyvin rivin alkiot täsmäävät matriisin muiden rivien samoissa sarakepositioissa olevien elementtien kanssa. Toisin sanoen binäärimatriisin sellaiset rivit, joiden riviobjektit liittyvät yhteen tai useampaan samaan sarakeattribuuttiin, ovat keskenään samanlaisia. Mukautuvin rivi on tyypillisesti sellainen riviobjekti, jolla on samoja sarakeattribuutteja kuin useilla muilla riveillä.

Mukautuvuuspaino lasketaan yhtälöiden 3.1 ja 3.2 mukaisesti. Algoritmi 5.1 laskee mukautuvuuden matriisin $M$ riveille. Matriisin sarakkeiden arvoille lasketaan esiintymien määrä koko matriisissa ja samaten rivien arvoille erilliseen taulukkoon. Rivin alkiot käydään läpi ja ei-tyhjien elementtien sarakkeiden arvojen frekvenssit lasketaan mukautuvuuspainoon. Lopuksi mukautuvuus lasketaan yhtälöllä 3.1.

Algoritmissa muuttuja $notExcludedElements$ sisältää esiintymien lukumäärän kaikilla riveillä, jotka ovat mukana laskennassa. Muuttuja $excluded$ sisäl-

tää mahdollisesti laskennasta poissuljettujen rivien viittaukset. Pelkän mukautuvuuspainon laskennassa tämä taulukko on tyhjä. Muuttuja *colFrequencies* sisältää sarakefrekvenssit ja *rowFrequencies* vastaavasti rivifrekvenssit. Muuttujat *M.rows* ja *M.columns* sisältävät rivien ja sarakkeiden lukumäärän matriisissa. Taulukkoon *conformities* tallennetaan lasketut mukautuvuuspainot.

**Algoritmi 5.1:** laskeMukautuvuus

**Syöte**: $m \times n$ binäärimatriisi $M$, poissuljettavat rivi-indeksit taulukossa *excluded*, rivifrekvenssit *rowFrequencies*, sarakefrekvenssit *colFrequencies*, *notExcludedElements* matriisin elementtien määrä

**Tuloste**: Mukautuvuuspainot taulukossa *conformities*

```
for i ⟵ 0; i < M.rows; i++ do
    if i ∉ excluded then
        for j ⟵ 0; j < M.columns; j++ do
            if M[i][j] ≠ 0 then
                conformities[i] ⟵ conformities[i] + colFrequencies[j];
            end
        end
        conformities[i] ⟵ 2 × conformities[i] + ((M.rows −
        excluded.size()) × M.columns − notExcludedElements) −
        (M.rows − excluded.size()) × rowFrequencies[i];
    end
end
```

Vastaavasti sarakkeille voidaan laskea mukautuvuuspainot samalla tavalla vertailemalla sarakkeen alkioita samoissa rivipositioissa.

### 5.2 Miinustekniikka

Miinustekniikka matriisien järjestelyssä on menetelmä, jossa algoritmi etenee vaiheittain niin, että aluksi jokaiselle riville tai sarakkeelle lasketaan käytetyn samanlaisuusmittarin mukainen vertailuarvo muihin riveihin tai sarakkeisiin nähden. Kuten aiemmin todettiin, tämä mittari Liivin algoritmissa on mukautuvuusanalyysi eli jokaiselle riville lasketaan mukautuvuuspaino. Tämän jälkeen tutkitaan

lasketut mukautuvuuspainot ja rivi, jolle laskettiin pienin mukautuvuuspaino, on epätyypillisin rivi, joka eliminoidaan matriisista.

Muokatulle matriisille lasketaan mukautuvuuspainot uudelleen seuraavalla iteraatiolla, ja toistetaan proseduuria kunnes kaikki matriisin rivit on eliminoitu. Eliminointijärjestys määrittää järjestelyproseduurin tuloksen riveille tai sarakkeille.

Miinustekniikan lisäksi muita sukulaistekniikoita ovat mm. plustekniikka ja sekatekniikka. Võhandu [2006] kuvaa miinustekniikan ohella myös nämä tekniikat.

Liiv [2007] toteaa, että alkuperäiset toteutukset olivat merkittävän tehottomia harvoille matriisesille, joten kummatkin algoritmit toteutettiin harvojen matriisien järjestelyyn niin, että algoritmi osasi jättää huomiotta 0-alkiot ja rakenteena käytettiin linkitettyä listaa. Algoritmi oli kuitenkin muotoiltu taulukkomuotoiselle rakenteelle.

Algoritmi 5.2 kuvaa miinustekniikan toiminnan. Algoritmi noudattaa muuten Liivin [2008] kuvausta, mutta syötejoukot poikkeavat siten, että ne eivät ole transaktiodataa, joka koostuu varastointiyksiköistä (engl. *stock-keeping-unit*), vaan binäärimatriiseja, joissa positiiviset alkiot saavat arvon 1 ja negatiiviset arvon 0.

Pseudokoodissa oletetaan, että *excluded* on alustettu taulukko ja $M$ on validi yksi-nolla-matriisi. Lopuksi palautetaan permutaatio, joka sisältää lasketun järjestyksen. Matriisin attribuutti $M.sparsity$ sisältää matriisin alkioiden määrän. $notExcludedCount$ sisältää laskennassa mukana olevien alkioiden määrän, joka vähenee joka iteraatiolla. Funktio *M.countElementsInRow()* laskee rivillä olevien ei-nolla-alkioiden määrän ja vastaavasti *M.countElementsInColumn()* sarakkeella olevien ei-nolla-alkioiden määrän.

| | $H$ | $I$ | $K$ | $G$ | $J$ | $L$ |
|---|---|---|---|---|---|---|
| $A$ | 1 | 0 | 0 | 0 | 1 | 1 |
| $E$ | 0 | 1 | 0 | 0 | 0 | 1 |
| $C$ | 0 | 0 | 0 | 1 | 0 | 0 |
| $B$ | 0 | 0 | 1 | 0 | 0 | 0 |
| $D$ | 1 | 1 | 0 | 0 | 0 | 0 |
| $F$ | 1 | 0 | 0 | 0 | 0 | 0 |

**Kuva 5.1** Kuvan 4.1 graafin binäärimatriisi järjestettynä miinustekniikalla ja mukautuvuusanalyysillä rivien ja sarakkeiden järjestelyn jälkeen. Järjestelyn laskennan vaiheet on esitetty kuvissa 5.2 ja 5.3

**Algoritmi 5.2:** miinusAlgoritmi

**Syöte**: $m \times n$ matriisi $M$
**Tuloste**: rivipermutaatio $p$
**begin**
  $k \longleftarrow 0$;
  $notExcludedElements \longleftarrow M.sparsity$ ;
  **for** $i \longleftarrow 0; i < M.rows; i++$ **do**
    $rowFrequencies[i] \longleftarrow M.countElementsInRow(i)$ ;
  **end**
  **for** $\longleftarrow 0; i < M.columns; i++$ **do**
    $colFrequencies[i] \longleftarrow M.countElementsInColumn(i)$;
  **end**
  $c \longleftarrow laskeMukautuvuus(M$,$excluded$, $rowFrequencies$, $colFrequencies$, $notExcludedElements$);
  **while** $excluded.size() < M.rows$ **do**
    $c \longleftarrow laskeMukautuvuus(M$, $excluded$, $rowFrequencies$, $colFrequencies$, $notExcludedElements$);
    $excluded[k] \longleftarrow c.min.rowIndex()$;
    $p[k] \longleftarrow c.min$;
    **for** $m_{i,j} \in C.min$ **do**
      $colFrequencies[j]--$ ;
      $rowFrequencies[i]--$ ;
      $notExcludedElements--$ ;
    **end**
    $k++$;
  **end**
  **return** $p$;
**end**

## 5.3 Järjestelyesimerkki

Käytetään kuvan 4.1 esimerkkidataa ja järjestellään se. Rivien järjestelyn vaiheet on kuvattu taulukossa 5.2 ja sarakkeiden järjestelyn vaiheet taulukossa 5.3. Riveille saatiin järjestys $(A, E, C, B, D, F)$ ja sarakkeille $(H, I, K, G, J, L)$. Kuvasta 5.1 voidaan havaita, että alkiot ovat järjestäytyneet sekundääridiagonaalin

ympärille.

### 5.4 Kompleksisuus

Mukautuvuusanalyysialgoritmin kompleksisuus on luokkaa $O(mn)$ ja miinustekniikan luokkaa $O(nm^2)$. Mukautuvuusanalyysin laskennassa pahimmassa tapauksessa käydään kaikki $m$ matriisin riviä läpi ja matriisin rivillä on kaikki attribuutit, mistä tulee $n$ sarakeposition verran laskentoja, eli $m$ läpikäyntiä ulommalle silmukalle ja $n$ sisemmälle silmukalle.

Miinustekniikassa lasketaan joka kierroksella yksi rivi vähemmän, kunnes kaikki rivit on eliminoitu $m, m-1, ..., 2, 1$, ja jokaisella läpikäynnillä käydään pienimmän elementin $O(n)$ sarakepositiota läpi eli läpikäyntejä on

$$\sum_{i=1}^{m} n \times i = n \times \sum_{i=1}^{m} i = n \times \frac{m(m-1)}{2} = O(nm^2). \tag{5.1}$$

| **1. iteraatio** | | | | | | | |
|---|---|---|---|---|---|---|---|
| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | *Mukautuvuuspainot* |
| $A$ | 0 | 1 | 0 | 1 | 0 | 1 | $2 \times 6 + (6 \times 6 - 10) - (3 * 6) = 20$ |
| $B$ | 0 | 0 | 0 | 0 | 1 | 0 | $2 \times 1 + (6 \times 6 - 10) - (1 * 6) = 22$ |
| $C$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 6 - 10) - (1 * 6) = 22$ |
| $D$ | 0 | 1 | 1 | 0 | 0 | 0 | $2 \times 5 + (6 \times 6 - 10) - (2 * 6) = 24$ |
| $E$ | 0 | 0 | 1 | 0 | 0 | 1 | $2 \times 4 + (6 \times 6 - 10) - (2 * 6) = 22$ |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 3 + (6 \times 6 - 10) - (1 * 6) = 26$ |
| **2. iteraatio** | | | | | | | |
| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | *Mukautuvuuspainot* |
| $B$ | 0 | 0 | 0 | 0 | 1 | 0 | $2 \times 1 + (6 \times 5 - 7) - (1 * 5) = 20$ |
| $C$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 5 - 7) - (1 * 5) = 20$ |
| $D$ | 0 | 1 | 1 | 0 | 0 | 0 | $2 \times 4 + (6 \times 5 - 7) - (2 * 5) = 21$ |
| $E$ | 0 | 0 | 1 | 0 | 0 | 1 | $2 \times 3 + (6 \times 5 - 7) - (2 * 5) = 19$ |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 5 - 7) - (1 * 5) = 22$ |
| **3. iteraatio** | | | | | | | |
| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | *Mukautuvuuspainot* |
| $B$ | 0 | 0 | 0 | 0 | 1 | 0 | $2 \times 1 + (6 \times 4 - 5) - (1 * 4) = 17$ |
| $C$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 4 - 5) - (1 * 4) = 17$ |
| $D$ | 0 | 1 | 1 | 0 | 0 | 0 | $2 \times 3 + (6 \times 4 - 5) - (2 * 4) = 17$ |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 4 - 5) - (1 * 4) = 19$ |
| **4. iteraatio** | | | | | | | |
| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | *Mukautuvuuspainot* |
| $B$ | 0 | 0 | 0 | 0 | 1 | 0 | $2 \times 1 + (6 \times 3 - 4) - (1 * 3) = 13$ |
| $D$ | 0 | 1 | 1 | 0 | 0 | 0 | $2 \times 3 + (6 \times 3 - 4) - (2 * 3) = 14$ |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 3 - 4) - (1 * 3) = 15$ |
| **5. iteraatio** | | | | | | | |
| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | *Mukautuvuuspainot* |
| $D$ | 0 | 1 | 1 | 0 | 0 | 0 | $2 \times 3 + (6 \times 2 - 3) - (2 * 2) = 11$ |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 2 - 3) - (1 * 2) = 11$ |
| **6. iteraatio** | | | | | | | |
| | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | *Mukautuvuuspainot* |
| $F$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 1 - 1) - (1 * 2) = 5$ |

**Kuva 5.2** Rivien järjestelyn tulos miinustekniikalla ja mukautuvuusanalyysillä kuvan 4.3 taulukolle

| **1. iteraatio** | | | | | | | |
|---|---|---|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $Mukautuvuuspainot$ |
| $G$ | 0 | 0 | 1 | 0 | 0 | 0 | $2 \times 1 + (6 \times 6 - 10) - (1 * 6) = 22$ |
| $H$ | 1 | 0 | 0 | 1 | 0 | 1 | $2 \times 6 + (6 \times 6 - 10) - (3 * 6) = 20$ |
| $I$ | 0 | 0 | 0 | 1 | 1 | 0 | $2 \times 4 + (6 \times 6 - 10) - (2 * 6) = 22$ |
| $J$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 3 + (6 \times 6 - 10) - (1 * 6) = 26$ |
| $K$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 6 - 10) - (1 * 6) = 22$ |
| $L$ | 1 | 0 | 0 | 0 | 1 | 0 | $2 \times 5 + (6 \times 6 - 10) - (2 * 6) = 24$ |
| **2. iteraatio** | | | | | | | |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $Mukautuvuuspainot$ |
| $G$ | 0 | 0 | 1 | 0 | 0 | 0 | $2 \times 1 + (6 \times 5 - 7) - (1 * 5) = 20$ |
| $I$ | 0 | 0 | 0 | 1 | 1 | 0 | $2 \times 3 + (6 \times 5 - 7) - (2 * 5) = 19$ |
| $J$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 5 - 7) - (1 * 5) = 22$ |
| $K$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 5 - 7) - (1 * 5) = 20$ |
| $L$ | 1 | 0 | 0 | 0 | 1 | 0 | $2 \times 4 + (6 \times 5 - 7) - (2 * 5) = 21$ |
| **3. iteraatio** | | | | | | | |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $Mukautuvuuspainot$ |
| $G$ | 0 | 0 | 1 | 0 | 0 | 0 | $2 \times 1 + (6 \times 4 - 5) - (1 * 4) = 17$ |
| $J$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 4 - 5) - (1 * 4) = 19$ |
| $K$ | 0 | 1 | 0 | 0 | 0 | 0 | $2 \times 1 + (6 \times 4 - 5) - (1 * 4) = 17$ |
| $L$ | 1 | 0 | 0 | 0 | 1 | 0 | $2 \times 3 + (6 \times 4 - 5) - (2 * 4) = 17$ |
| **4. iteraatio** | | | | | | | |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $Mukautuvuuspainot$ |
| $G$ | 0 | 0 | 1 | 0 | 0 | 0 | $2 \times 1 + (6 \times 3 - 4) - (1 * 3) = 13$ |
| $J$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 3 - 4) - (1 * 3) = 15$ |
| $L$ | 1 | 0 | 0 | 0 | 1 | 0 | $2 \times 3 + (6 \times 3 - 4) - (2 * 3) = 14$ |
| **5. iteraatio** | | | | | | | |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $Mukautuvuuspainot$ |
| $J$ | 1 | 0 | 0 | 0 | 0 | 0 | $2 \times 2 + (6 \times 2 - 3) - (1 * 2) = 11$ |
| $L$ | 1 | 0 | 0 | 0 | 1 | 0 | $2 \times 3 + (6 \times 2 - 3) - (2 * 2) = 11$ |
| **6. iteraatio** | | | | | | | |
| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $Mukautuvuuspainot$ |
| $L$ | 1 | 0 | 0 | 0 | 1 | 0 | $2 \times 2 + (6 \times 1 - 2) - (2 * 1) = 6$ |

**Kuva 5.3** Sarakkeiden järjestely miinustekniilla ja mukautuvuusanalyysillä kuvan 4.3 taulukolle

# 6 Ohjelmallinen testaus

## 6.1 Tekninen ympäristö

Testausohjelmisto ja algoritmit on toteutettu Java-ohjelmointikielellä. Testit ajettiin Java 6 -ohjelmistolla Ubuntu Linux -ympäristössä, jossa on 6 Gt muistia ja 64-bittinen 2.53GHz kaksiytiminen prosessori.

## 6.2 Testausaineisto

Testausaineistona käytettiin sekä valmista aineistoa että synteettistä dataa. Kuvissa 6.1 ja 6.2 on listaukset käytettyjen aineistojen ominaisuuksista: matriisien tiheydet, dimensiot ja nimet. Kuvan 6.1 aineistot ovat peräisin MatrixMarketin [Boisvert *et al.*, 1997; MatrixMarket, 2007] kokoelmista, jotka sisältävät datamatriiseja eri tutkimusalueilta. MatrixMarketin Harwell-Boeing-kokoelman aineistoja on käytetty myös kaistanleveysongelman testauksessa [Lim *et al.*, 2006].

| Matriisi | Dimensiot | Symmetrinen | Tiheys | Kokoelma |
|---|---|---|---|---|
| ASH219 | $219 \times 85$ | Ei | 0,015 | $Harwell - Boeing$ |
| ASH292 | $292 \times 292$ | Kyllä | 0,024 | $Harwell - Boeing$ |
| ASH331 | $331 \times 104$ | Ei | 0,019 | $Harwell - Boeing$ |
| ASH608 | $608 \times 108$ | Ei | 0,011 | $Harwell - Boeing$ |
| ASH958 | $958 \times 292$ | Ei | 0,007 | $Harwell - Boeing$ |
| BCSPWR06 | $1454 \times 1454$ | Kyllä | 0,002 | $Harwell - Boeing$ |
| BCSPWR08 | $1624 \times 1624$ | Kyllä | 0,001 | $Harwell - Boeing$ |
| DWT 198 | $198 \times 1984$ | Kyllä | 0,02 | $Harwell - Boeing$ |
| DWT 209 | $209 \times 209$ | Kyllä | 0,022 | $Harwell - Boeing$ |
| DWT 592 | $592 \times 592$ | Kyllä | 0,008 | $Harwell - Boeing$ |
| E05R0500 | $236 \times 236$ | Ei | 0,105 | $Sparskit$ |
| GRE 343 | $343 \times 343$ | Ei | 0,012 | $Harwell - Boeing$ |
| JGL009 | $9 \times 9$ | Ei | 0,628 | $Harwell - Boeing$ |
| JGL011 | $11 \times 11$ | Ei | 0,617 | $Harwell - Boeing$ |
| OLM100 | $100 \times 100$ | Ei | 0,04 | $NEP$ |
| OLM1000 | $1000 \times 1000$ | Ei | 0,04 | $NEP$ |
| RGG010 | $10 \times 10$ | EI | 0,76 | $Harwell - Boeing$ |
| TUB100 | $100 \times 100$ | Ei | 0,04 | $NEP$ |

**Kuva 6.1** MatrixMarketin aineistot

Kuvan 6.2 aineistot ovat leveys- ja syvyyssuuntaiseen etsintään perustuvilla heuristisilla algoritmeilla generoituja kaksijakoisia graafeja. Generointiin käytetty aineisto on alunperin Jordi Petitin väitöstutkimuksessa [2001a] käytetystä testigraafisarjasta, joka sisältää kattavan joukon graafeja eri ongelma-alueilta. Alkuperäinen testiaineisto on saatavilla sähköisesti [Petit, 2001b]. Aineistot ovat graafidatatiedostoja, joissa on solmujen ja särmien lukumäärät, solmujen naapurisolmujen määrä ja särmien päätesolmut.

| Matriisi | Dimensiot | Symmetrinen | Tiheys |
|---|---|---|---|
| Mesh33x33 (BFS) | $545 \times 544$ | Ei | 0,007 |
| C4y (DFS) | $667 \times 699$ | Ei | 0,006 |
| Hc10 (DFS) | $510 \times 514$ | Ei | 0,014 |
| RandomA3 (DFS) | $500 \times 500$ | Ei | 0,006 |
| RandomA4 (DFS) | $499 \times 501$ | Ei | 0,021 |
| Whitaker3 (BFS) | $4901 \times 4899$ | Ei | 0,001 |

**Kuva 6.2** Kaksijakoinen graafiaineisto

Synteettinen aineisto on listattu kuvassa 6.3 ja se on generoitu luomalla satunnaisia alkioita diagonaalin ympärille 30 % leveydelle niin, että matriisin tiheys on 15 %. Aineistot ovat keskenään hyvin samanlaisia, koska lähtöaineistoissa on hyvin vähän alkioita. Synteettisellä aineistolla on tarkoitus saada testattua hieman suuremmalla tiheydellä olevia aineistoja ja tutkia kaistamaisen rakenteen löytymistä satunnaisesta järjestelystä.

### 6.3 Toteutusratkaisu

Toteutuksessa noudatettiin edellisissä luvuissa esitettyjä mukautuvuusanalyysin, miinustekniikkaa ja mukautuvuusanalyysiä yhdistävän Liivin algoritmin ja barysentrisen algoritmin sekä risteymien määrän laskennan algoritmeja.

Alkuperäisten järjestelytekniikoiden ohella toteutettiin miinustekniikan ja barysentrisen järjestelyn yhdistävä tekniikka, joka mukautuvuuspainojen sijaan käytti barysentrisen järjestelyn tuloksia hyväkseen. Barysentrinen arvo toimi lasketun painon perustana, mutta rivin barysentriseen arvoon yhteenlaskettavat sarakepositiot kerrottiin vielä riviin liittyvien sarakkeiden riviviittausten määrällä. Tämän takana oli ajatus, että kun miinustekniikan iteraatioissa poistetaan rivi ja riviin liittyvät alkiot, niin seuraavilla iteraatioilla muut samoihin sarakepositioihin liittyvät rivit saisivat mahdollisesti pienemmän arvon ja poistettaisiin herkemmin.

| Matriisi | Dimensiot | Symmetrinen | Tiheys |
|---|---|---|---|
| Mesh33x33 (15%) | $545 \times 544$ | Ei | 0,15 |
| C4y (15%) | $667 \times 699$ | Ei | 0,15 |
| Hc10 (15%) | $510 \times 514$ | Ei | 0,15 |
| ASH219 (15%) | $219 \times 85$ | Ei | 0,15 |
| ASH292 (15%) | $292 \times 292$ | Ei | 0,15 |
| ASH331 (15%) | $331 \times 104$ | Ei | 0,15 |
| ASH608 (15 %) | $608 \times 108$ | Ei | 0,15 |
| ASH958 (15 %) | $958 \times 292$ | Ei | 0,15 |
| DWT 198 (15 %) | $198 \times 1984$ | Ei | 0,15 |
| DWT 209 (15 %) | $209 \times 209$ | Ei | 0,15 |
| DWT 592 (15%) | $592 \times 592$ | Ei | 0,15 |
| GRE 343 (15%) | $343 \times 343$ | Ei | 0,15 |
| OLM100 (15%) | $100 \times 100$ | Ei | 0,15 |
| OLM1000 (15%) | $1000 \times 1000$ | Ei | 0,15 |
| TUB100 (15%) | $100 \times 100$ | Ei | 0,15 |

**Kuva 6.3** Synteettisesti vahvistettu kaistamainen aineisto

Tässä järjestelytoteutuksessa barysentrista arvoa laskettaessa kerrottiin jaettavaan osaan summattavat osat kyseisen päätesolmun naapurien määrällä, joten laskettu arvo oli sitä suurempi, mitä enemmän naapureita rivin solmuilla oli.

Graafin solmulle $v$, jonka naapurusto on $N_v$, näin painotettu arvo on

$$bc_n(v) = \frac{\sum_{u \in N_v} pos(u) \times |N_u|}{|N_v|}.$$

Arvo sijoitetaan kaavaan 3.1, ja saadaan solmua $v$ vastaavan rivin $r_v$ mukautuva barysentrinen paino:

$$bc_{conf}(r_v) = 2 \times bc_n(r_v) + (n \times m - e_1) - (e_1(r_v) \times m),$$

kun $e_1$ on nollasta poikkeavien alkioiden määrä matriisissa ja $e_1(r_v)$ nollasta poikkeavien alkioiden määrä rivillä $r_v$.

Matriisit toteutettiin linkitettyyn listaan perustuvaan rakenteeseen, jossa matriisin perättäiset rivit linkitettiin toisiinsa ja matriisin alkiot linkitettiin peräkkäin jokaiselle riville. Liiv [2008] käytti vastaavaa rakennetta kuvaamaan transaktiodataa. Tämä rakenne muistuttaa graafin vierekkyyslistarakennetta. Muitakin rakenteita, kuten CCS- ja CRS-formaatteja [Silva, 2005; Shahnaz *et al.*, 2005],

harkittiin, mutta näistä ei todettu olevan merkittävää hyötyä toteutuksen kannalta.

Hyödyllisintä sekä Liivin algoritmin että barysentrisen algoritmin ajon kannalta oli, että matriisia kyettiin käymään tehokkaasti läpi joko riveittäin tai sarakkeittain sen perusteella, laskettiinko barysentriset arvot ja mukautuvuuspainot riveille tai sarakkeille. Sekä rivien ja sarakkeiden järjestelyä ei tarvinnut toteuttaa erikseen vaan sarakkeiden järjestäminen oli mahdollista toteuttaa rivien järjestelynä. Toteutusratkaisussa rivien ja sarakkeiden järjestely toteutettiin kahtena erillisenä rivien järjestelynä niin, että matriisi transponoitiin järjestelyjen välissä.

## 6.4 Testiajot

Aineistot luettiin MatrixMarketin mtx- tai gra-formaateissa olevista tiedostoista levyltä muistiin matriisirakenteeseen, joka ohjattiin järjestelykäsittelyyn. Risteymien määrä laskettiin alkuperäisestä matriisista, joka erityisesti MatrixMarketin aineistoissa oli pieni, koska nämä olivat tallennettu hyvin strukturoidussa muodossa. Tämän jälkeen matriisi permutoitiin satunnaiseen järjestykseen, ja satunnaistetun matriisin risteymien määrä laskettiin, minkä jälkeen satunnaistettu matriisi ohjattiin järjestelykäsittelyyn neljälle eri järjestelyalgoritmivariaatiolle.

Järjestelyalgoritmiluokat ensin järjestelivät syötematriisin rivit, minkä jälkeen matriisi transponoitiin ja sarakkeiden järjestely ajettiin suorittamalla rivien järjestely transpoosille. Järjestelyn jälkeen risteymien määrä laskettiin uudelleen ja otettiin talteen. Ajoajat otettiin talteen sekä rivien että sarakkeiden järjestelyille.

Ennen tulosten käsittelyä on ensinnäkin syytä huomata, että useissa tuloksissa risteymien määrä on laskemisen sijaan kasvanut, mikä oli odotettavissa syistä, joita tarkastellaan seuraavassa kohdassa. Tämän takia järjestelyjen tuloksia tarkastellaan risteymien määrän muutoksena lähtötilanteeseen verrattuna. Toisekseen, ajoaikojen vertailussa mielekkäät vertailut tehdään niiden tekniikoiden välillä, jotka oli yhdistetty miinustekniikkaan, ja toisaalta niiden tekniikoiden välillä, joissa ei tätä käytetty lainkaan. Kolmanneksi, aineistojen tarkat minimit eivät olleet tiedossa, joten vertailua optimaaliseen nähden ei tehty.

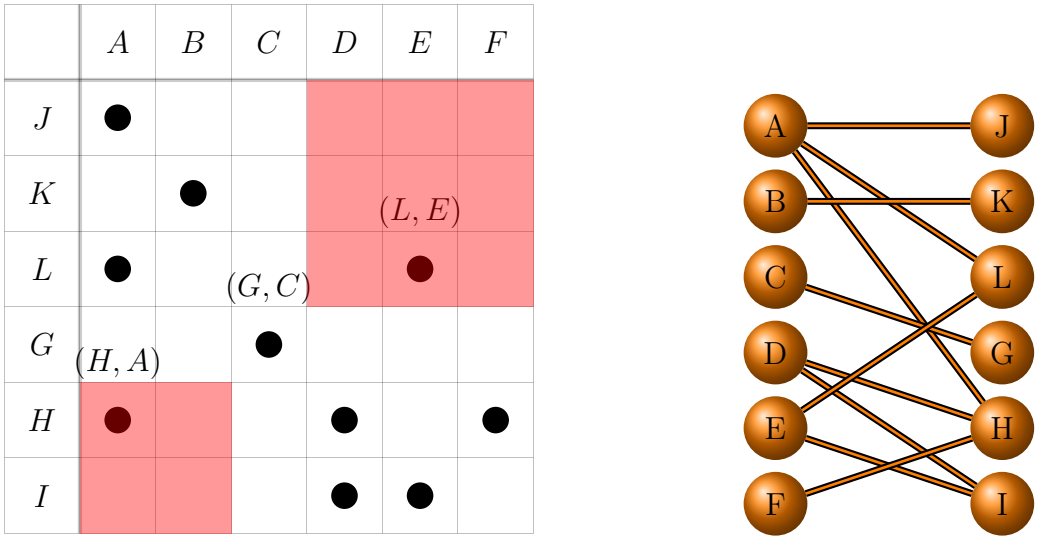
## 6.5 Risteymien määrät ja matriisin hahmo

Luvussa 4 todettiin, että graafin $G = (V, E)$ särmät $e_1 \in E$ ja $e_2 \in E$ risteävät, kun $pos_{V_1}(v_1) > pos_{V_1}(u_1)$ ja $pos_{V_2}(v_2) < pos_{V_2}(u_2)$ tai $pos_{V_1}(v_1) < pos_{V_1}(u_1)$ ja $pos_{V_2}(v_2) > pos_{V_2}(u_2)$.

Särmä $e_1 = (v_1, v_2)$ voidaan sijoittaa matriisiin niin, että rivi- ja sarakepositiot voivat kuvata särmän alku- ja loppusolmun positioita, jolloin graafin särmä kuvautuu matriisin alkioksi. Särmän kanssa risteävät muut särmät sijoittuvat matriisissa yläoikealle ja alavasemmalle suhteessa särmään. Kuvassa 6.4 on kuvan 4.4 graafi ja siihen liittyvä vierekkyysmatriisi. Särmän $(G, C)$ kanssa risteävät särmät $(L, E)$ ja $(H, A)$ sijoittuvat matriisissa alkion $(G, C)$ suhteen oikeaan yläkulmaan ja vasempaan alakulmaan. Korostamattomilla alueilla sijaitsevien alkioiden särmät eivät risteä särmän $(G, C)$ kanssa.

Mitä vähemmän risteymiä matriisissa on, sitä paremmin alkiot sijoittuvat keskenään näihin asetelmiin, kun ideaa soveltaa koko särmäjoukkoon. Kaistamaisissa hahmoissa esimerkiksi risteymiä on vähän, kuten voi huomata myöhemmin strukturoitujen testiaineistojen risteymien määrästä.

Risteymien määrän kasvu tuloksissa selittyy siis sillä, että matriisin järjestelyn tulos muuttaa matriisin hahmoa sellaiseen hahmoon, jossa risteymien määrä on suurempi. Tämän seikan voisi korjata testaamalla peilikuvaa hahmosta ja valitsemalla hahmoista sen, joka vähentää risteymiä. Tutkielman tuloksissa on kelpuutettu sekä risteymien määrän kasvu että niiden väheneminen tulokseksi.




(a) Risteymät matriisissa (b) Graafi

**Kuva 6.4** Graafin risteävät särmät matriisiesityksessä

Kuvissa 6.15, 6.16, 6.17 ja 6.18 on visualisoitu erilaisia järjestelyn tuloksia käytetyillä aineistoilla. Visualisoinnit on tuotettu eri ajoilla kuin testitulokset, joten risteymien määrät eivät täsmää taulukoitujen tulosten kanssa. Visualisoinnit on toteutettu Javan Swing-grafiikkakirjaston palveluilla.

## 6.6 MatrixMarketin ja kaksijakoisen graafidatan järjestelyjen vertailu

Kuvat 6.15, 6.16 ja 6.17 esittävät *OLM100*-, *ASH219*- ja *DWT209*-matriisin testatuilla järjestelymetodeilla järjestettynä.

Kuvassa 6.5 on sarakkeiden ja rivien järjestelyyn kuluneet ajat matriiseittain. Kuvissa 6.6, 6.7, 6.8 ja 6.9 on testiajoissa saadut risteymien määrät ja muutosprosentti satunnaiseen järjestelyyn nähden. Aineistoilla keskimääräinen järjestelytulos barysentrisellä järjestelyllä oli **4,7 %** ja mukautuvuusanalyysillä **19,2 %**. Miinustekniikan ja mukautuvuusanalyysin yhdistelmällä järjestelytulos oli keskimäärin **28,6 %** ja miinustekniikan ja barysentrisen järjestelyn yhdistävällä tekniikalla **16,0 %**.

Suurin osa järjestelyalgoritmeista jäi kauaksi alkuperäisessä muodossa luettujen aineistojen risteymien määristä. Valitut aineistot olivat lähtökohtaisesti hyvin strukturoidussa muodossa, jossa alkiot ovat keskittyneet lähelle diagonaalia ja joissain aineistoissa, kuten aineistoissa *OLM100* ja *OLM1000*, vieläpä kaistamaisessa hahmossa. Nämä rakenteet ovat edellisen kohdan perusteella sellaisia, joissa ilmenee vähemmän risteymiä. Muutama aineisto (*RandomA4* ja *RandomA3*) on lähtökohtaisesti satunnaisaineistoa, joten se, että mitkään järjestelyalgoritmit eivät onnistuneet löytämään näistä mitään säännönmukaisia hahmoja, ei ole yllätys.

Barysentrinen järjestelyalgoritmi palautti monessa tapauksessa lähes yhtä epäjärjestyksessä olevan matriisin kuin satunnaistettu aineisto, jonka algoritmi sai järjestääkseen. Parhaiten barysentrinen järjestely toimi testimatriiseilla *JGL009*, *JGL011*, *OLM100* ja *TUB100*. Lisäksi matriisilla *E05R0500* barysentrisen menetelmän tulos oli hieman parempi tulos kuin muilla menetelmillä. Näillä aineistoilla on korkeammat tiheydet kuin muilla aineistoilla, minkä vuoksi aineistot ovat mahdollisesti toimineet paremmin.

Stallmannin ja muiden [2001] kokeellisissa tutkimuksissa barysentrinen järjestely toimi mediaaniheuristiikkaan verrattuna huonosti graafeilla, joiden lähtöjärjestys on satunnainen ja jotka olivat hyvin harvoja ja rakenteisia ja joissa edellytettiin algoritmin tekevän ratkaisevia askelia solmuilla, joiden aste oli 2. Esimerkiksi matriisien *ASH219*, *ASH331* ja *ASH608* keskimääräinen nollasta poikkeavien alkioiden määrä rivillä on 2, joka vastaa solmun astetta 2. Myös Abdullah ja Hussain [2006] huomauttavat kaistanleveyden minimointiongelman tutkimuksessaan, että barysentrinen järjestely toimii paremmin, kun graafien tiheys kasvaa. Kun alkioita on vain muutamia rivillä, jolle barysentristä arvoa lasketaan, voi saatu keskiarvo olla liian epätarkka.

Pelkkä mukautuvuusanalyysi toimi erityisen hyvin *ASH*-matriiseilla, minkä lisäksi se järjesteli joitakin yksittäisiä matriiseja kohtalaisesti. *ASH*-matriiseissa on toisiinsa verrattuna hyvin samantapainen rakenne ja mukautuvuusanalyysi lasketaan sopivasti niin, että matriisi jää tyhjäksi vastakkaisista nurkista ja alkiot keskittyvät joko diagonaalin tai vastadiagonaalin ympärille, kuten kuvasta 6.16 näkyy. Miinustekniikan kanssa yhdistetyt tekniikat pärjäsivät keskenään tasaisemmin, eikä yhtä selvää eroa ollut kuin barysentrisen järjestelyn ja mukautuvuusanalyysin välillä.

Keskimääräisen järjestelytuloksen perusteella Liivin algoritmi ja mukautuvuusanalyysi tarjosi parhaimman järjestelyn näillä aineistoilla. Liivin algoritmi luonnollisesti toimi hyvin niilläkin matriiseilla, joilla mukautuvuusanalyysi toimi hyvin. Liivin algoritmi osasi muodostaa osassa tapauksia hämmästyttävän säännöllisiä rakenteita, kuten nauhamaisen rakenteen matriisin *OLM100* yläosaan, kuten kuvasta 6.15 voi huomata. Barysentristä metodia ja miinustekniikkaa yhdistelevä metodi on luonut alkioista tiiviitä rykelmiä.

Ajoaikojen perusteella barysentrinen järjestely oli ylivoimaisesti nopein testatuista heuristiikoista, mutta pärjäsi huonosti risteymien minimoinnissa (tai maksimoinnissa). Mukautuvuusanalyysi oli riippuvainen etukäteen lasketuista rivi- ja sarakefrekvensseistä ja laskentakaava hieman monimutkaisempi, minkä vuoksi se saattoi olla hieman hitaampi. Liivin algoritmi oli useimmissa tapauksissa hieman nopeampi kuin barysentrisen järjestelyn ja miinustekniikan yhdistävä tekniikka.

| Matriisi | Barysentrinen järjestely | | Mukautuvuus-analyysi | | Miinustekniikka ja mukautuvuusanalyysi | | Miinustekniikka ja barysentrinen järjestely | |
|---|---|---|---|---|---|---|---|---|
| | Rivit | Sarakkeet | Rivit | Sarakkeet | Rivit | Sarakkeet | Rivit | Sarakkeet |
| Mesh33x33 | 1 | 1 | 19 | 19 | 1904 | 1179 | 1339 | 1836 |
| C4y | 1 | 1 | 51 | 49 | 3305 | 3463 | 3635 | 4016 |
| Hc10 | 10 | 14 | 14 | 17 | 1294 | 1028 | 1020 | 976 |
| RandomA3 | 3 | 2 | 121 | 120 | 1014 | 1596 | 1146 | 1342 |
| RandomA4 | $< 1$ | $< 1$ | 35 | 33 | 1783 | 1221 | 1433 | 1885 |
| Whitaker3 | 8 | 8 | 3756 | 3268 | 2119841 | 2287746 | 1533620 | 1583587 |
| ASH219 | 1 | $< 1$ | 2 | 2 | 139 | 20 | 123 | 23 |
| ASH292 | $< 1$ | $< 1$ | 5 | 9 | 344 | 422 | 368 | 417 |
| ASH331 | $< 1$ | $< 1$ | 2 | 3 | 245 | 19 | 252 | 23 |
| ASH608 | 1 | $< 1$ | 7 | 11 | 2316 | 144 | 1896 | 157 |
| ASH958 | $< 1$ | $< 1$ | 27 | 30 | 7893 | 659 | 7857 | 644 |
| BCSPWR06 | 2 | 2 | 109 | 109 | 20782 | 26301 | 24052 | 19947 |
| BCSPWR08 | 2 | 1 | 221 | 217 | 55876 | 57662 | 54040 | 55430 |
| DWT 198 | $< 1$ | $< 1$ | 6 | 3 | 95 | 92 | 118 | 88 |
| DWT 209 | $< 1$ | $< 1$ | 7 | 6 | 122 | 104 | 126 | 123 |
| DWT 592 | $< 1$ | $< 1$ | 26 | 26 | 2404 | 2437 | 2466 | 1719 |
| E05R0500 | 4 | 2 | 12 | 11 | 164 | 109 | 991 | 150 |
| GRE 343 | $< 1$ | $< 1$ | 7 | 16 | 469 | 513 | 512 | 559 |
| JGL009 | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ |
| JGL011 | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ |
| OLM100 | $< 1$ | $< 1$ | 1 | $< 1$ | 10 | 9 | 11 | 10 |
| OLM1000 | $< 1$ | $< 1$ | 89 | 88 | 13568 | 13144 | 11781 | 11644 |
| RGG010 | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ | $< 1$ |
| TUB100 | $< 1$ | $< 1$ | 1 | 1 | 21 | 21 | 22 | 22 |

**Kuva 6.5** Järjestelyjen ajoajat graafidatalla ja MatrixMarketin aineistoilla millisekunneissa

| Matriisi | Alkuperäinen | Satunnaistettu | Barysentrinen järjestely | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 | 71988 | 1112684 | 1109035 | 99,7 |
| C4y | 1400389 | 2049846 | 2053508 | 100,2 |
| Hc10 | 2626768 | 3344768 | 3310506 | 99,0 |
| RandomA3 | 165059403 | 167560562 | 168381806 | 100,5 |
| RandomA4 | 5616836 | 6677343 | 6746540 | 101,0 |
| Whitaker3 | 1325601 | 92726662 | 92184005 | 99,4 |
| ASH219 | 8258 | 47113 | 46845 | 99,4 |
| ASH292 | 18973 | 393773 | 399147 | 101,4 |
| ASH331 | 7862 | 110666 | 112155 | 101,3 |
| ASH608 | 26314 | 362052 | 339176 | 93,7 |
| ASH958 | 40180 | 910830 | 921625 | 101,2 |
| BCSPWR06 | 201972 | 2833548 | 2832823 | 100,0 |
| BCSPWR08 | 258713 | 3707399 | 3736427 | 100,8 |
| DWT 198 | 25063 | 153811 | 154279 | 100,3 |
| DWT 209 | 57381 | 237614 | 233688 | 98,3 |
| DWT 592 | 228288 | 2015192 | 2019629 | 100,2 |
| E05R0500 | 2661449 | 8434344 | 7854912 | 93,1 |
| GRE 343 | 55454 | 520870 | 526549 | 101,8 |
| JGL009 | 458 | 495 | 636 | 128,5 |
| JGL011 | 1097 | 1148 | 894 | 79,9 |
| OLM100 | 686 | 35450 | 31558 | 89,0 |
| OLM1000 | 6986 | 4036914 | 4152272 | 102,9 |
| RGG010 | 1032 | 1144 | 1048 | 91,6 |
| TUB100 | 734 | 36172 | 30894 | 85,4 |

**Kuva 6.6** Risteymien määrät eri aineistoilla barysentrisen järjestelyn jälkeen

| Matriisi | Alkuperäinen | Satunnaistettu | Mukautuvuus-analyysi | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 | 71988 | 1112684 | 998081 | 89,7 |
| C4y | 1400389 | 2049846 | 2392896 | 116,7 |
| Hc10 | 2626768 | 3344768 | 3568078 | 106,7 |
| RandomA3 | 165059403 | 167560562 | 169044065 | 100,9 |
| RandomA4 | 5616836 | 6677343 | 6779375 | 101,5 |
| Whitaker3 | 1325601 | 92726662 | 111273603 | 120,0 |
| ASH219 | 8258 | 47113 | 69723 | 148,0 |
| ASH292 | 18973 | 393773 | 442409 | 112,4 |
| ASH331 | 7862 | 110666 | 170065 | 153,7 |
| ASH608 | 26314 | 362052 | 568401 | 157,0 |
| ASH958 | 40180 | 910830 | 1423682 | 156,3 |
| BCSPWR06 | 201972 | 2833548 | 3285632 | 116,0 |
| BCSPWR08 | 258713 | 3707399 | 4158270 | 112,2 |
| DWT 198 | 25063 | 153811 | 163548 | 106,3 |
| DWT 209 | 57381 | 237614 | 289849 | 122,0 |
| DWT 592 | 228288 | 2015192 | 2648590 | 131,4 |
| E05R0500 | 2661449 | 8434344 | 8503931 | 100,8 |
| GRE 343 | 55454 | 526911 | 366767 | 69,6 |
| JGL009 | 458 | 495 | 291 | 58,8 |
| JGL011 | 1097 | 1148 | 1148 | 100,0 |
| OLM100 | 686 | 35450 | 36377 | 102,6 |
| OLM1000 | 6986 | 4036914 | 4017229 | 99,5 |
| RGG010 | 1032 | 1144 | 1020 | 89,2 |
| TUB100 | 734 | 36172 | 35043 | 96,9 |

**Kuva 6.7** Risteymien määrät eri aineistoilla mukautuvuusanalyysillä järjestelyn jälkeen

| Matriisi | Alkuperäinen | Satunnaistettu | Miinustekniikka ja barysentrinen järjestely | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 | 71988 | 1112684 | 1029807 | 92,6 |
| C4y | 1400389 | 2049846 | 2392896 | 116,7 |
| Hc10 | 2626768 | 3344768 | 3816448 | 114,1 |
| RandomA3 | 165059403 | 167560562 | 169272059 | 101,0 |
| RandomA4 | 5616836 | 6677343 | 7110032 | 106,5 |
| Whitaker3 | 1325601 | 92726662 | 94028675 | 101,4 |
| ASH219 | 8258 | 47113 | 45898 | 97,4 |
| ASH292 | 18973 | 393773 | 475384 | 120,7 |
| ASH331 | 7862 | 110666 | 119212 | 107,7 |
| ASH608 | 26314 | 362052 | 375926 | 103,8 |
| ASH958 | 40180 | 910830 | 1110674 | 121,9 |
| BCSPWR06 | 201972 | 2833548 | 3351599 | 118,3 |
| BCSPWR08 | 258713 | 3707399 | 4478962 | 120,8 |
| DWT 198 | 25063 | 153811 | 187156 | 121,7 |
| DWT 209 | 57381 | 237614 | 325530 | 137,0 |
| DWT 592 | 228288 | 2015192 | 2681190 | 133,0 |
| E05R0500 | 2661449 | 8434344 | 8310235 | 98,5 |
| GRE 343 | 55454 | 520870 | 622925 | 118,2 |
| JGL009 | 458 | 495 | 614 | 124,0 |
| JGL011 | 1097 | 1148 | 1481 | 129,0 |
| OLM100 | 686 | 35450 | 31806 | 89,7 |
| OLM1000 | 6986 | 4036914 | 4197750 | 104,0 |
| RGG010 | 1032 | 1144 | 1452 | 126,9 |
| TUB100 | 734 | 36172 | 23553 | 65,1 |

**Kuva 6.8** Risteymien määrät eri aineistoilla miinustekniikan ja barysentrisen järjestelyn jälkeen

## 6.7 Järjestelyt kaistamaisella synteettisellä aineistolla

Kaistamaisen synteettisen aineiston järjestelyssä pärjäsi parhaiten barysentristä järjestelyä ja miinustekniikkaa yhdistelevä järjestely, joka tuotti yllättävän hyviä tuloksia. Kuvassa 6.18 visualisoidaan eri algoritmien tulokset vahvistetulla *ASH219*-aineistolla, ja siitä näkyy hyvin barysentrisen järjestelyn ja miinustekniikan yhdistävän tekniikan aikaansaama kaistamainen hahmo. Pelkällä barysentrisellä järjestelyllä ja mukautuvuusanalyysillä on satunnaisen näköinen järjestely. Liivin algoritmilla alkiot ovat keskittyneet myös matriisin vastakkaisiin nurkkauksiin, vaikka eivät ehkä niin selvästi kuin barysentrisen järjestelyn ja miinustekniikan

| Matriisi | Alkuperäinen | Satunnaistettu | Miinustekniikka ja mukautuvuusanalyysi | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 | 71988 | 1112684 | 1006491 | 90,5 |
| C4y | 1400389 | 2049846 | 2326690 | 113,5 |
| Hc10 | 2626768 | 3344768 | 3436004 | 102,7 |
| RandomA3 | 165059403 | 167560562 | 168780679 | 100,7 |
| RandomA4 | 5616836 | 6677343 | 6691994 | 100,2 |
| Whitaker3 | 1325601 | 92726662 | 111473268 | 120,2 |
| ASH219 | 8258 | 47113 | 69729 | 148,0 |
| ASH292 | 18973 | 393773 | 400663 | 101,7 |
| ASH331 | 7862 | 110666 | 161719 | 146,1 |
| ASH608 | 26314 | 362052 | 556705 | 153,8 |
| ASH958 | 40180 | 910830 | 1377502 | 151,2 |
| BCSPWR06 | 201972 | 2833548 | 3116899 | 110,0 |
| BCSPWR08 | 258713 | 3707399 | 3943192 | 106,4 |
| DWT 198 | 25063 | 153811 | 167912 | 109,2 |
| DWT 209 | 57381 | 237614 | 278774 | 117,3 |
| DWT 592 | 228288 | 2015192 | 2656528 | 131,8 |
| E05R0500 | 2661449 | 8434344 | 8335425 | 98,8 |
| GRE 343 | 55454 | 520870 | 344789 | 59,0 |
| JGL009 | 458 | 495 | 250 | 50,5 |
| JGL011 | 1097 | 1148 | 779 | 67,9 |
| OLM100 | 686 | 35450 | 56910 | 160,5 |
| OLM1000 | 6986 | 4036914 | 6201778 | 153,6 |
| RGG010 | 1032 | 1144 | 1452 | 126,9 |
| TUB100 | 734 | 36172 | 72081 | 199,3 |

**Kuva 6.9** Risteymien määrät eri aineistoilla mukautuvuusanalyysin ja miinustekniikan soveltamisen jälkeen

kan tapauksessa.

Pelkällä barysentrisellä järjestelyllä järjestetyt aineistot saivat hivenen parempia tuloksia verrattuna alkuperäisiin aineistoihin, joita käsiteltiin edellisessä kohdassa. Järjestelyjen tulokset on listattu kuvissa 6.11 barysentriselle järjestelylle, 6.12 mukautuvuusanalyysille ja kuvissa 6.14 ja 6.13 miinustekniikkaa käyttäville tekniikoille.

Synteettisillä aineistoilla keskimääräinen järjestelytulos barysentrisellä järjestelyllä on **5,8 %** ja mukautuvuusanalyysillä **2,9 %**. Mukautuvuusanalyysi pärjäsi tällä aineistoilla huomattavasti huonommin kaksijakoiseen graafidataan ja MatrixMarketin aineistoihin verrattuna. Miinustekniikan ja mukautuvuusanalyysin yhdistelmällä järjestelytulos on keskimäärin **26,2 %** ja miinustekniikan ja barysentrisen järjestelyn yhdistävällä tekniikalla **51,5 %**.

Lähtöaineistolla on selvästi vaikutusta algoritmien pärjäämiseen, sillä mukautuvuusanalyysi ei pärjännyt tällä aineistolla yhtä hyvin kuin edellisen kohdan aineistolla. Synteettisellä aineistolla rivien mukautuvuudet eivät ole todennäköisesti niin yksiselitteisiä kuin käyttämässämme lähtöaineistoissa ja satunnainen generointi ei tuota alkioita täsmälleen toisiaan vastaaviin samoihin sarake- ja rivipositioihin. Barysentrinen järjestely toimii myös silloin, kun rivin alkioiden positiot ovat lähellä toisiaan eivätkä täsmää täydellisesti keskenään kuten mukautuvuusanalyysissä.

Kuvassa 6.10 on listattu järjestelyjen ajoajat. Järjesteltävällä aineistolla ei näyttänyt olevan samalla tavalla vaikutusta ajoaikoihin kuin järjestelyjen toiminnassa, vaan ajoajoissa toistui sama kaava kuin MatrixMarketin ja graafidatan aineistoissa, eli barysentrinen järjestely oli ylivoimaisesti nopein ja miinustekniikan ja barysentrisen järjestelyn yhdistelmätekniikka oli kaikista hitain.

| Matriisi | Barysentrinen järjestely | | Mukautuvuus-analyysi | | Miinustekniikka ja mukautuvuusanalyysi | | Miinustekniikka ja barysentrinen järjestely | |
|---|---|---|---|---|---|---|---|---|
| | Rivit | Sarakkeet | Rivit | Sarakkeet | Rivit | Sarakkeet | Rivit | Sarakkeet |
| Mesh33x33 (15%) | 2 | 4 | 355 | 397 | 4301 | 4469 | 5554 | 5273 |
| C4y (15%) | 3 | 6 | 940 | 988 | 6276 | 10671 | 10924 | 13636 |
| Hc10 (15%) | 3 | 4 | 306 | 324 | 3230 | 3524 | 4351 | 4714 |
| ASH219 (15%) | 1 | 1 | 6 | 5 | 128 | 34 | 160 | 31 |
| ASH292 (15%) | 2 | $< 1$ | 42 | 48 | 491 | 436 | 686 | 708 |
| ASH331 (15%) | 1 | $< 1$ | 9 | 17 | 371 | 79 | 688 | 84 |
| ASH608 (15 %) | 2 | 2 | 36 | 97 | 2848 | 467 | 3527 | 476 |
| ASH958 (15 %) | 13 | 4 | 12 | 20 | 196 | 154 | 843 | 248 |
| DWT 198 (15 %) | 1 | 1 | 11 | 19 | 111 | 127 | 176 | 194 |
| DWT 209 (15 %) | 1 | $< 1$ | 20 | 17 | 136 | 149 | 234 | 236 |
| DWT 592 (15%) | 2 | 4 | 355 | 397 | 4301 | 4469 | 5554 | 5273 |
| GRE 343 (15%) | 1 | 1 | 78 | 75 | 886 | 822 | 1103 | 1147 |
| OLM100 (15%) | $< 1$ | 1 | 3 | 2 | 20 | 15 | 30 | 26 |
| OLM1000 (15%) | 8 | 15 | 3247 | 3706 | 24529 | 26100 | 34493 | 42840 |
| TUB100 (15%) | 13 | 4 | 12 | 20 | 196 | 154 | 843 | 248 |

**Kuva 6.10** Synteettisen kaistamaisen aineiston järjestelyjen ajoajat millisekunneissa

| Matriisi | Alkuperäinen | Satunnaistettu | Barysentrinen järjestely | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 (15%) | 94839643 | 492509625 | 449206946 | 91,2 |
| C4y (15%) | 269478723 | 1215675418 | 1139168292 | 93,7 |
| Hc10 (15%) | 100203417 | 384707050 | 366347091 | 95,2 |
| ASH219 (15%) | 380030 | 1920961 | 1812212 | 94,3 |
| ASH292 (15%) | 7755222 | 41047542 | 43922292 | 107,0 |
| ASH331 (15%) | 1165443 | 6543711 | 7049960 | 107,7 |
| ASH608 (15%) | 13875513 | 72646795 | 70228765 | 96,7 |
| ASH958 (15%) | 85331625 | 438164996 | 401654574 | 91,7 |
| DWT 198 (15%) | 1727577 | 8490771 | 9046529 | 106,5 |
| DWT 209 (15%) | 2239995 | 10540488 | 10746163 | 102,0 |
| DWT 592 (15%) | 136092585 | 690255256 | 649610238 | 94,1 |
| GRE 343 (15%) | 14760939 | 77525299 | 79032833 | 101,9 |
| OLM100 (15%) | 92684 | 572229 | 675102 | 118,0 |
| OLM1000 (15%) | 1101305416 | 5618905431 | 5633019163 | 100,3 |
| TUB100 (15%) | 92363 | 543478 | 545164 | 100,3 |

**Kuva 6.11** Risteymien määrät synteettisellä kaistamaisella aineistolla barysentrisen järjestelyn jälkeen

| Matriisi | Alkuperäinen | Satunnaistettu | Mukautuvuus-analyysi | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 (15%) | 94839643 | 492509625 | 470387089 | 95,5 |
| C4y (15%) | 269478723 | 1215675418 | 1163481674 | 95,7 |
| Hc10 (15%) | 100203417 | 384707050 | 372250323 | 96,8 |
| ASH219 (15%) | 380030 | 1920961 | 1882699 | 98,0 |
| ASH292 (15%) | 7755222 | 41047542 | 39361085 | 95,9 |
| ASH331 (15%) | 1165443 | 6543711 | 6590507 | 100,7 |
| ASH608 (15%) | 13875513 | 72646795 | 70673480 | 97,3 |
| ASH958 (15%) | 85331625 | 438164996 | 428119070 | 97,7 |
| DWT 198 (15%) | 1727577 | 8490771 | 8556603 | 100,8 |
| DWT 209 (15%) | 2239995 | 10540488 | 10628464 | 100,8 |
| DWT 592 (15%) | 136092585 | 690255256 | 658315470 | 95,4 |
| GRE 343 (15%) | 14760939 | 77525299 | 73908558 | 95,3 |
| OLM100 (15%) | 92684 | 572229 | 558338 | 97,6 |
| OLM1000 (15%) | 1101305416 | 5618905431 | 5316699697 | 94,6 |
| TUB100 (15%) | 92363 | 543478 | 545451 | 100,4 |

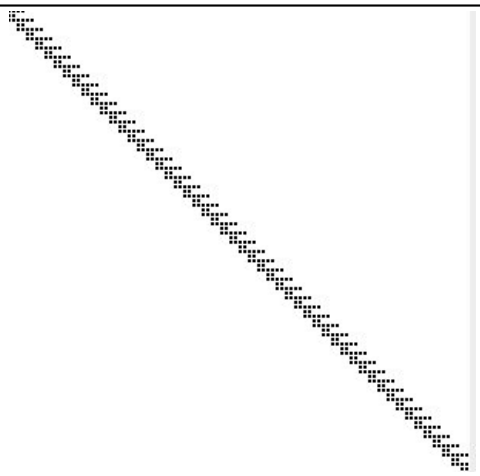
**Kuva 6.12** Risteymien määrät synteettisellä kaistamaisella aineistolla mukautuvuusanalyysin soveltamisen jälkeen

| Matriisi | Alkuperäinen | Satunnaistettu | Miinustekniikka ja mukautuvuusanalyysi | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 (15%) | 94839643 | 492509625 | 581930820 | 118,2 |
| C4y (15%) | 269478723 | 1215675418 | 632486047 | 52,0 |
| Hc10 (15%) | 100203417 | 384707050 | 369154474 | 96,0 |
| ASH219 (15%) | 380030 | 1920961 | 2351780 | 122,4 |
| ASH292 (15%) | 7755222 | 41047542 | 5104774 | 123,7 |
| ASH331 (15%) | 1165443 | 6543711 | 5104774 | 78,0 |
| ASH608 (15%) | 13875513 | 72646795 | 78405617 | 107,9 |
| ASH958 (15%) | 85331625 | 438164996 | 319798344 | 73,0 |
| DWT 198 (15%) | 1727577 | 8490771 | 11559391 | 136,1 |
| DWT 209 (15%) | 2239995 | 10540488 | 15399005 | 146,1 |
| DWT 592 (15%) | 136092585 | 690255256 | 764178729 | 110,7 |
| GRE 343 (15%) | 14760939 | 77525299 | 104631809 | 135,0 |
| OLM100 (15%) | 92684 | 572229 | 403432 | 70,5 |
| OLM1000 (15%) | 1101305416 | 5618905431 | 2493541895 | 44,4 |
| TUB100 (15%) | 92363 | 543478 | 582761 | 107,2 |

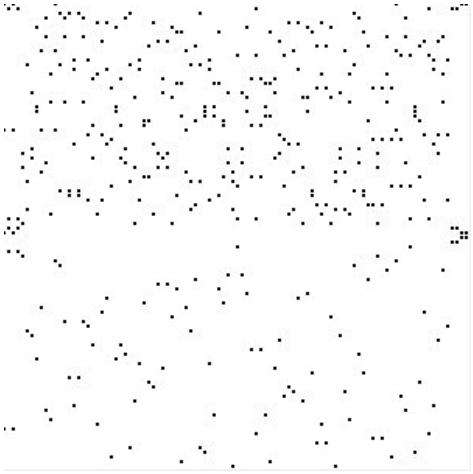
**Kuva 6.13** Risteymien määrät synteettisellä kaistamaisella aineistolla miinustekniikan ja mukautuvuusanalyysin soveltamisen jälkeen

| Matriisi | Alkuperäinen | Satunnaistettu | Barysentrinen järjestely ja miinustekniikka | Järjestelyn tulos (%) |
|---|---|---|---|---|
| Mesh33x33 (15%) | 94839643 | 492509625 | 200716333 | 40,8 |
| C4y (15%) | 269478723 | 1215675418 | 313965355 | 25,8 |
| Hc10 (15%) | 100203417 | 384707050 | 138416108 | 36,0 |
| ASH219 (15%) | 380030 | 1920961 | 694618 | 36,2 |
| ASH292 (15%) | 7755222 | 41047542 | 36399911 | 88,7 |
| ASH331 (15%) | 1165443 | 6543711 | 5654928 | 86,4 |
| ASH608 (15%) | 13875513 | 72646795 | 70228765 | 96,7 |
| ASH958 (15%) | 85331625 | 438164996 | 29957426 | 41,2 |
| DWT 198 (15%) | 1727577 | 8490771 | 14502753 | 170,8 |
| DWT 209 (15%) | 2239995 | 10540488 | 15852912 | 150,4 |
| DWT 592 (15%) | 136092585 | 690255256 | 167757322 | 24,3 |
| GRE 343 (15%) | 14760939 | 77525299 | 124458877 | 160,5 |
| OLM100 (15%) | 92684 | 572229 | 390762 | 68,3 |
| OLM1000 (15%) | 1101305416 | 5618905431 | 1364777474 | 24,3 |
| TUB100 (15%) | 92363 | 543478 | 865399 | 159,2 |

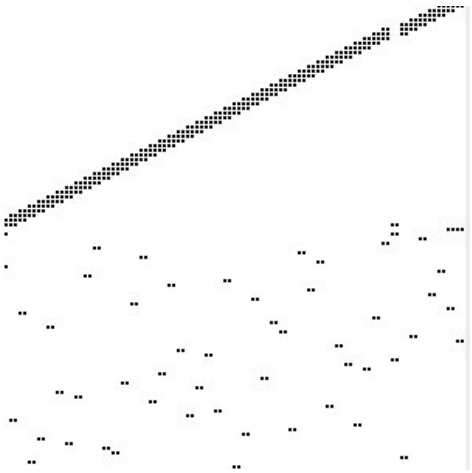
**Kuva 6.14** Risteymien määrät synteettisellä kaistamaisella aineistolla miinustekniikan ja barysentrisen järjestelyn jälkeen



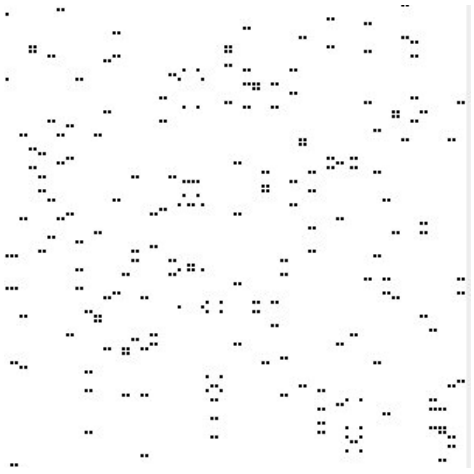
(a) Alkuperäinen



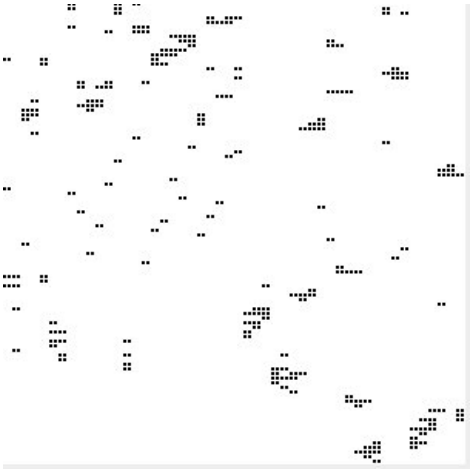
(b) Mukautuvuusanalyysi



(c) Liivin algoritmi

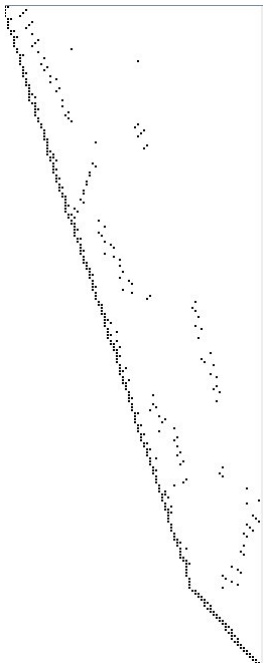


(d) Barysentrinen järjestely

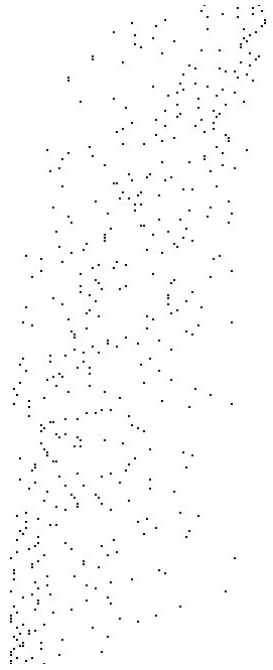


(e) Miinustekniikka ja barysentrinen järjestely

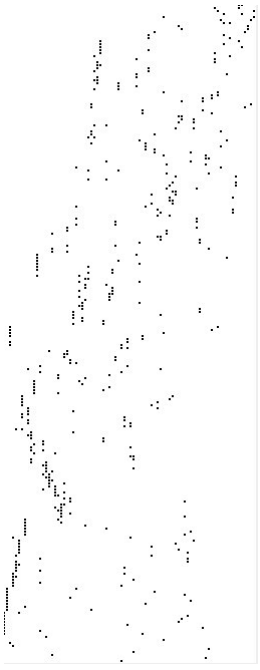
**Kuva 6.15** Aineisto OLM100 järjestettynä ja visualisoituna matriisimuodossa



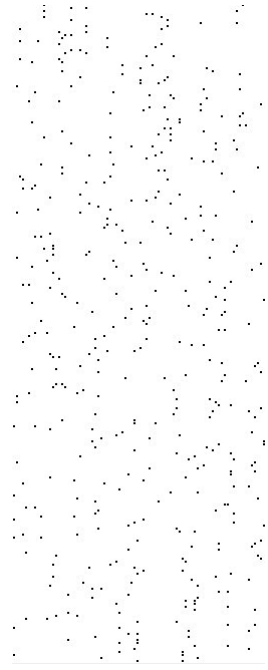
(a) Alkuperäinen



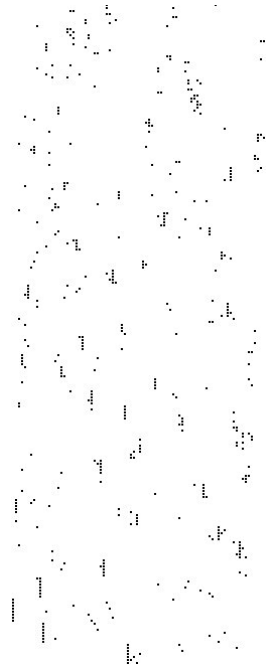
(b) Mukautuvuusanalyysi



(c) Liivin algoritmi

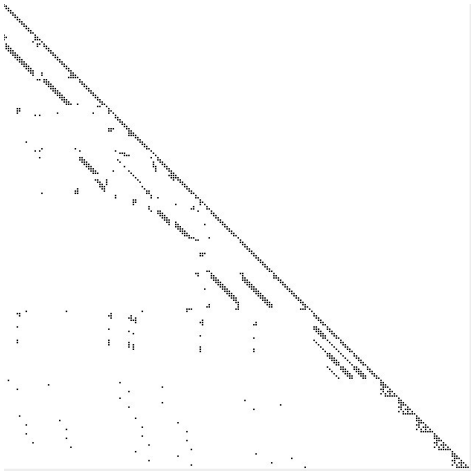


(d) Barysentrinen järjestely

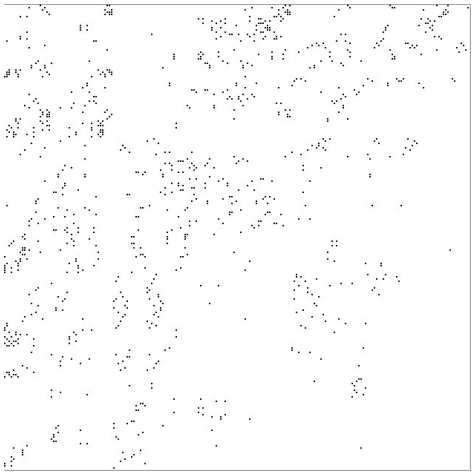


(e) Miinustekniikka ja barysentrinen järjestely

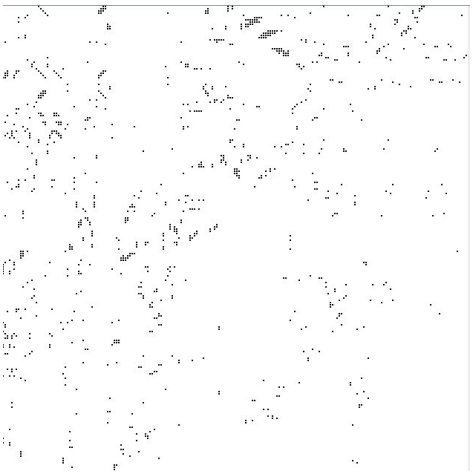
**Kuva 6.16** Aineisto ASH219 järjestettynä ja visualisoituna matriisimuodossa



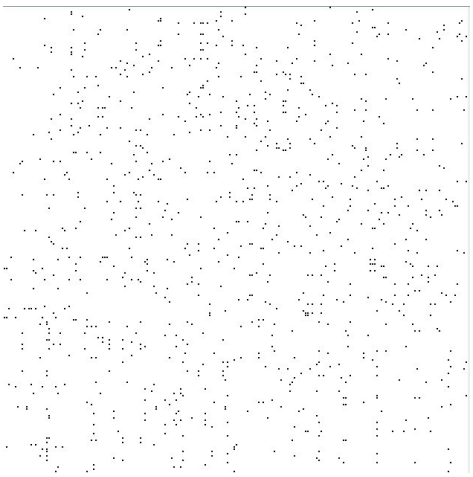
(a) Alkuperäinen



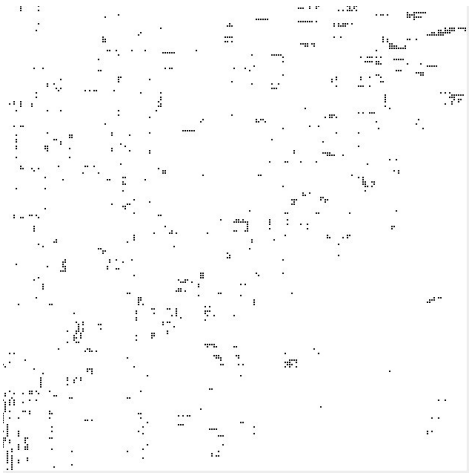
(b) Mukautuvuusanalyysi



(c) Liivin algoritmi

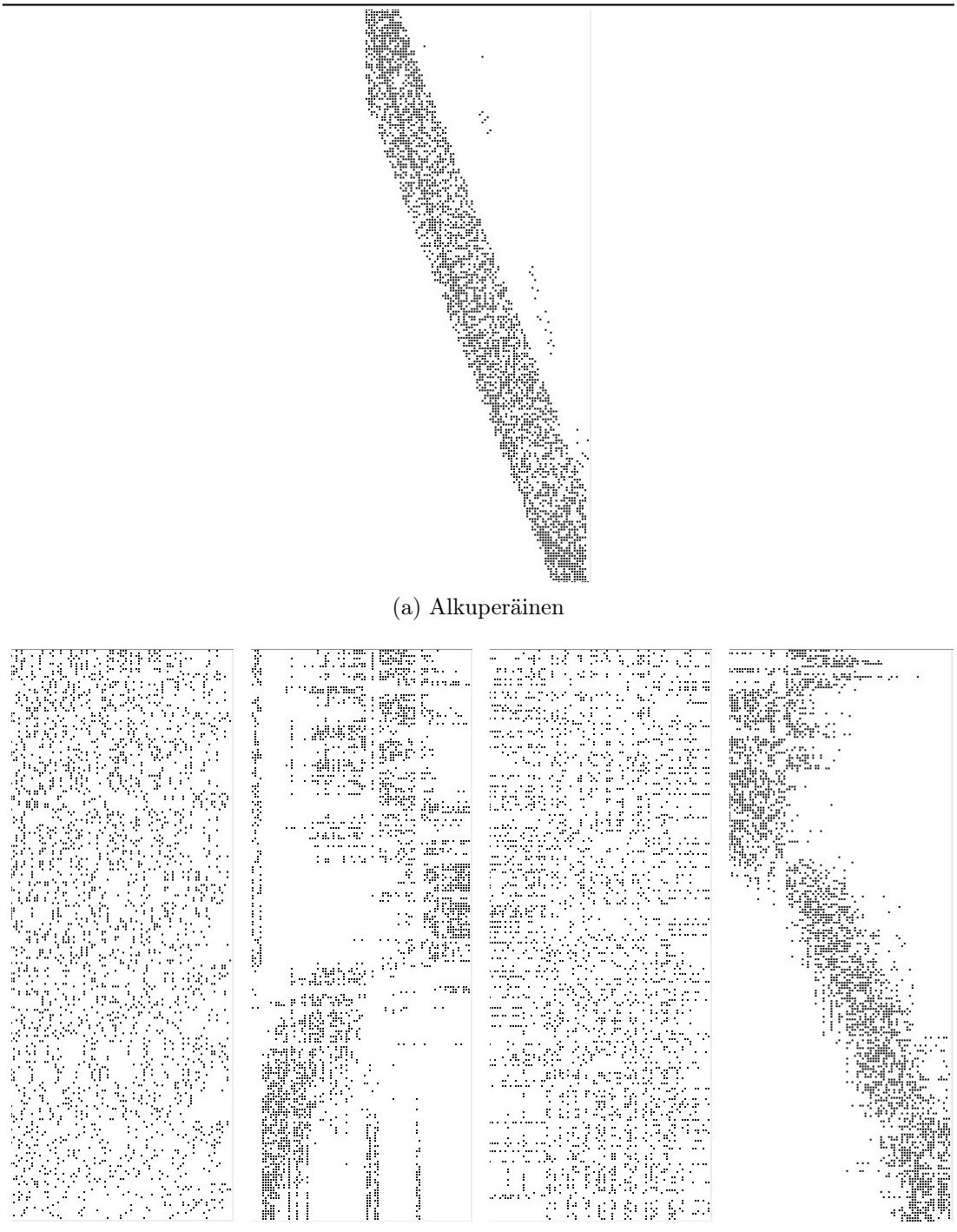


(d) Barysentrinen järjestely



(e) Miinustekniikka ja barysentrinen järjestely

**Kuva 6.17** Aineisto DWT209 järjestettynä ja visualisoituna matriisimuodossa



(a) Alkuperäinen

(b) Mukautuvuusanalyysi

(c) Liivin algoritmi

(d) Barysentrinen järjestely

(e) Miinustekniikka ja barysentrinen järjestely

**Kuva 6.18** Synteettisesti vahvistettu ASH219 järjestettynä ja visualisoituna matriisimuodossa

# 7 Loppusanat

Tulokset vahvistivat sen aiemman olettamuksen, että risteymien minimointi linkittyy kaistamaiseen rakenteeseen matriisissa, minkä Mäkinen ja Siirtola [2000] ovat tuoneet esiin. Hahmontunnistus voi myös tarjota uusia näkökulmia ongelmanratkaisuun, kuten tutkimuksessa on käytännössä todettu matriisien hahmon ja risteymien minimointiongelman välillä.

Liivin algoritmi onnistui kokeellisissa testeissämme vahvistamaan risteymien minimointiin liittyviä hahmoja useissa testimatriiseissa. Myös kokeellinen uusi tekniikka, barysentristä järjestelyä ja miinustekniikkaa yhdistelevä tekniikka, ylsi lähes yhtä hyviin suorituksiin ja toimi erityisen hyvin kaistamaisella synteettisellä aineistolla. Barysentrinen järjestely ja pelkkä mukautuvuusanalyysi olivat riippuvaisempia lähtöaineistosta, eivätkä aina onnistuneet vahvistamaan risteymien minimointiongelmaan liittyviä hahmoja, mutta onnistuivat yksittäisissä testitapauksissa.

Barysentrinen heuristiikka ei pärjännyt testeissä, mikä todennäköisesti johtui testiaineistojen luonteesta, mutta myös barysentriseen heuristiikkaan liittyvästä toteutuksesta. Barysentristä metodia voisi yrittää soveltaa iteratiivisesti esimerkiksi niin, että järjestelypuolta vaihdellaan, kunnes tulos ei enää parane, mikä järjestely on toiminut hyvin aiemmissa testeissä [Jünger & Mutzel, 1996].

Barysentrisen järjestelyn ja miinustekniikan yhdistely vaatisi vielä tarkempia testejä ja teoreettista tarkastelua, jotta sen käyttökelpoisuus ongelma-alueella selviäisi. Synteettinen aineisto, jolla järjestelymetodi pärjäsi, oli melko homogeenistä, eivätkä MatrixMarketin eri sovellusalueiden matriisit soveltuneet aivan yhtä hyvin tälle menetelmälle. Miinustekniikka lisäksi vahvisti hahmontunnistusta, mutta vaati lisää laskentatehoa, minkä takia se ei välttämättä ole sellaisenaan käytännöllinen ainakaan isoilla matriiseilla tai graafeilla, mutta tarjoaa ainakin kiinnostavan lähtökohdan jatkotutkimuksille.

# Viiteluettelo


[Abdu & Salane, 2009] Eman Abdu & Douglas Salane. A spectral-based clustering algorithm for categorical data using data summaries. In *Proceedings of the 2nd Workshop on Data Mining using Matrices and Tensors*, DMMT '09, pages 1–8, 2009.

[Abdullah & Hussain, 2006] A. Abdullah & A. Hussain. Heuristics and metaheuristics for bandwidth minimization of sparse matrices. In *Proceedings of the 2006 IEEE International Conference on Engineering of Intelligent Systems*, ICEIS 2006, pages 1–6, 2006.

[Agrawal & Srikant, 1994] Rakesh Agrawal & Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, 1994.

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imieliński, & Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, pages 207–216, 1993.

[Atkins *et al.*, 1999] Jonathan E. Atkins, Erik G. Boman, & Bruce Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1999.

[Aykanat *et al.*, 2004] Cevdet Aykanat, Ali Pinar, & Ümit V. Çatalyürek. Permuting sparse rectangular matrices into block-diagonal form. *SIAM Journal on Scientific Computing*, 25(6):1860–1879, 2004.

[Barth *et al.*, 2002] Wilhelm Barth, Michael Jünger, & Petra Mutzel. Simple and efficient bilayer cross counting. In *Graph Drawing*, volume 2528 of *Lecture Notes in Computer Science*, pages 130–141. 2002.

[Berry, 2006] Michael W. Berry. *Lecture Notes in Data Mining*. World Scientific Publishing Co., Inc., 2006.

[Bertini & Lalanne, 2010] Enrico Bertini & Denis Lalanne. Investigating and reflecting on the integration of automatic data analysis and visualization in knowledge discovery. *ACM SIGKDD Explorations Newsletter*, 11(2):9, 2010.

[Boisvert *et al.*, 1997] Ronald F. Boisvert, Roldan Pozo, Karin Remington, Richard F. Barrett, & Jack J. Dongarra. Matrix market: a web resource for test matrix collections. In *Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software: Assessment and Enhancement*, pages 125–137, 1997.

[Brusco & Stahl, 2005] Michael J. Brusco & Stephanie Stahl. Optimal least-squares unidimensional scaling: Improved branch-and-bound procedures and comparison to dynamic programming. *Psychometrika*, 70:253–270, 2005.

[Cormen *et al.*, 1990] Thomas H. Cormen, Charles E. Leiserson, & Ronald R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[Cuthill & McKee, 1969] E. Cuthill & J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172, 1969.

[Del Corso & Romani, 2001] G. M. Del Corso & Francesco Romani. Heuristic spectral techniques for the reduction of bandwidth and work-bound of sparse matrices. *Numerical Algorithms*, 28(1):117–136, 2001.

[Eades & Wormald, 1994] Peter Eades & Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, April 1994.

[Eiglsperger *et al.*, 2005] M. Eiglsperger, M. Siebenhaller, & M. Kaufmann. An efficient implementation of Sugiyama's algorithm for layered graph drawing. In *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 155–166, 2005.

[Eisen *et al.*, 1998] M. B. Eisen, P. T. Spellman, P. O. Brown, & D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, 1998.

[Garriga *et al.*, 2008] Gemma C. Garriga, Esa Junttila, & Heikki Mannila. Banded structure in binary matrices. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 292–300, 2008.

[George & Liu, 1981] A. George & J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, 1981.

[Gibbs *et al.*, 1976] Norman E. Gibbs, Jr. Poole, William G., & Paul K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13(2):236–250, 1976.

[gzip, 2003] The gzip home page, `http://www.gzip.org/`, 2003.

[Hahsler *et al.*, 2008] Michael Hahsler, Kurt Hornik, & Christian Buchta. Getting things in order: An introduction to the R package `seriation`. *Journal of Statistical Software*, 25(3):1–34, 2008.

[Han *et al.*, 2007] Jiawei Han, Hong Cheng, Dong Xin, & Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15:55–86, 2007.

[Johnson *et al.*, 2004] D. Johnson, S. Krishnan, J. Chhugani, S. Kumar, & S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *Proceedings of the 30th International Conference on Very Large Data Bases*, VLDB'04, page 23, 2004.

[Junttila, 2011] Esa Junttila. *Patterns in Permuted Binary Matrices*. PhD thesis, University of Helsinki, 2011.

[Jünger & Mutzel, 1996] Michael Jünger & Petra Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In *Graph Drawing*, volume 1027 of *Lecture Notes in Computer Science*, pages 337–348. 1996.

[Keim, 2002] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8:1–8, 2002.

[Kendall, 1971] David G. Kendall. Abundance matrices and seriation in archaeology. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 17:104–112, 1971.

[Koohestani & Poli, 2010] Behrooz Koohestani & Riccardo Poli. A genetic programming approach to the matrix bandwidth-minimization problem. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II*, PPSN'10, pages 482–491, 2010.

[Li & Stallmann, 2002] Xiao Yu Li & Matthias F. Stallmann. New bounds on the barycenter heuristic for bipartite graph drawing. *Information Processing Letters*, 82(6):293–298, 2002.

[Liiv *et al.*, 2012] Innar Liiv, Rain Opik, Jaan Ubi, & John Stasko. Visual matrix explorer for collaborative seriation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(1):85–97, 2012.

[Liiv, 2007] Innar Liiv. Visualization and data mining method for inventory classification. In *Proceedings of the 2007 IEEE International Conference on Service Operations and Logistics, and Informatics*, SOLI 2007, pages 1–6, 2007.

[Liiv, 2008] Innar Liiv. *Pattern Discovery Using Seriation and Matrix Reordering: A Unified View, Extensions and an Application to Inventory Management*. PhD thesis, Tallinn University of Technology, 2008.

[Liiv, 2010a] Innar Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, 3(2):70–91, 2010.

[Liiv, 2010b] Innar Liiv. Towards information-theoretic visualization evaluation measure: a practical example for Bertin's matrices. In *Proceedings of the 3rd BELIV'10 Workshop: BEyond Time and Errors: Novel EvaLuation Methods for Information Visualization*, BELIV '10, pages 24–28, 2010.

[Lim *et al.*, 2003] A. Lim, B. Rodrigues, & Fei Xiao. Using an evolutionary algorithm for bandwidth minimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1 of *CEC '03*, pages 678–683, 2003.

[Lim *et al.*, 2006] A. Lim, B. Rodrigues, & F. Xiao. Heuristics for matrix bandwidth reduction. *European Journal of Operational Research*, 174(1):69–91, 2006.

[MatrixMarket, 2007] Matrix market, `http://math.nist.gov/MatrixMarket/`, 2007. Visited on 20.05.2013.

[McCormick *et al.*, 1972] William T. Jr. McCormick, Paul J. Schweitzer, & Thomas W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

[Mullat, 1976] J. E. Mullat. Extremal subsystems of monotonic systems, I. *Avtomatica i Telemekhanika (Translated from Russian in Automation and Remote Control)*, 1976(5):130–139, 1976.

[Mullat, 1977] J. E. Mullat. Extremal subsystems of monotonic systems, II. *Avtomatica i Telemekhanika (Translated from Russian in Automation and Remote Control)*, 1976(8):169–178, 1977.

[Mullat, 1978] J. E. Mullat. Extremal subsystems of monotonic systems, III. *Avtomatica i Telemekhanika (Translated from Russian in Automation and Remote Control)*, 1977(1):109–119, 1978.

[Muñoz *et al.*, 2002] X. Muñoz, W. Unger, & I. Vrto. One sided crossing minimization is NP-Hard for sparse graphs. In *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 16–20, 2002.

[Mäkinen & Siirtola, 2000] Erkki Mäkinen & Harri Siirtola. Reordering the reorderable matrix as an algorithmic problem. In *Theory and Application of Diagrams*, volume 1889 of *Lecture Notes in Computer Science*, pages 453–468. 2000.

[Mäkinen & Siirtola, 2005] Erkki Mäkinen & Harri Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica (Slovenia)*, 29(3):357–364, 2005.

[Niermann, 2005] Stefan Niermann. Optimizing the ordering of tables with evolutionary computation. *The American Statistician*, 59(1):41–46, 2005.

[Petit, 2001a] Jordi Petit. *Layout problems.* PhD thesis, Universitat Politècnica de Catalunya, 2001.

[Petit, 2001b] Jordi Petit. Minla experiments, `http://www.lsi.upc.edu/~jpetit/MinLA/Experiments/`, 2001. Visited on 17.06.2013.

[R, 2013] R project home page, `http://www.r-project.org/`, 2013.

[Seo & Obermayer, 2004] Sambu Seo & Klaus Obermayer. Self-organizing maps and clustering methods for matrix data. *Neural Networks*, 17(8-9):1211–1229, 2004.

[Shahnaz *et al.*, 2005] R. Shahnaz, A. Usman, & I.R. Chughtai. Review of storage techniques for sparse matrices. In *Proceedings of the IEEE 9th International Multitopic Conference*, INMIC'05, pages 1–7, 2005.

[Silva, 2005] Malik Silva. Sparse matrix storage revisited. In *Proceedings of the 2nd Conference on Computing Frontiers*, CF '05, pages 230–235, 2005.

[Spence, 2007] Robert Spence. *Information Visualization: Design for Interaction, 2nd edition.* Pearson Education Limited, 2007.

[Stallmann *et al.*, 2001] Matthias Stallmann, Franc Brglez, & Debabrata Ghosh. Heuristics, experimental subjects, and treatment evaluation in bigraph crossing minimization. *Journal of Experimental Algorithmics*, 6, December 2001.

[Sugiyama *et al.*, 1981] Kozo Sugiyama, Shojiro Tagawa, & Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, 1981.

[Sugiyama, 2002] Kozo Sugiyama. *Graph Drawing and Applications for Software and Knowledge Engineers.* World Scientific, 2002.

[VME, 2011] Sourceforge.net: Visual matrix explorer project web page, `http://vismatexplorer.sourceforge.net/`, 2011. Visited on 16.07.2013.

[Vuokko, 2010] Niko Vuokko. Consecutive ones property and spectral ordering. In *Proceedings of the 10th SIAM International Conference on Data Mining*, SDM '10, pages 350–360, 2010.

[Vyhandu *et al.*, 2006] Leo Vyhandu, Rein Kuusik, Ants Torim, Eik Aab, & Grete Lind. Some algorithms for data table (re)ordering using monotone systems. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, AIKED'06, pages 417–422, 2006.

[Vyhandu, 1980] L. Vyhandu. Some methods to order objects and variables in data systems. *Proceedings of Tallinn Technical University*, 482:43–50, 1980.

[Vyhandu, 1986] L. Vyhandu. Fast methods for data analysis and processing. *Transactions of the Faculty of Economics, Tallinn Technical University*, 614:15–23, 1986.

[Ware, 2004] Colin Ware. *Information Visualization: Perception for Design.* Morgan Kaufmann Publishers Inc., 2004.

[Weisstein, 2012a] Eric W. Weisstein. Cellular automaton. From MathWorld—A Wolfram Web Resource, `http://mathworld.wolfram.com/CellularAutomaton.html`, 2012. Visited on 08.12.2012.

[Weisstein, 2012b] Eric W. Weisstein. Moore neighborhood. From MathWorld—A Wolfram Web Resource, `http://mathworld.wolfram.com/MooreNeighborhood.html`, 2012. Visited on 08.12.2012.

[WikimediaCommons, 2008] Wikimedia commons: Alkuaineiden jaksollinen järjestelmä (suomenkielinen versio), `http://commons.wikimedia.org/wiki/File:Periodic_table_fi.svg`, 2008. Sähköinen lähde; haettu 03.09.2011.

[Wilkinson & Friendly, 2009] Leland Wilkinson & Michael Friendly. The history of the cluster heat map. *The American Statistician*, 63(2):179–184, 2009.

[Willcock & Lumsdaine, 2006] Jeremiah Willcock & Andrew Lumsdaine. Accelerating sparse matrix computations via data compression. In *Proceedings of the 20th Annual International Conference on Supercomputing*, ICS '06, pages 307–316, 2006.

[Zhang *et al.*, 2010] Yi-kun Zhang, Hao Chen, Deng-xin hua, Ying-an Cui, & Bao-wei Zhang. An edge crossing minimization algorithm based on adjacency matrix transformation. In *Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering*, volume 1 of *ICACTE 2010*, pages 672–675, 2010.