Kandidaatintutkielma

Mallinnettu jäähdytys ja ympyräkiekkojen etsiminen kuvasta

Aaro Salosensaari

Soveltava matematiikka
Matematiikan ja tilastotieteen laitos
Helsingin Yliopisto

Helsinki, 15. maaliskuuta 2016

Sisältö

Luku 1

Johdanto

Tämä soveltavan matematiikan kandidaatintutkielma käsittelee matemaattisen optimoinnin menetelmää nimeltä mallinnettu jäähdytys ja sen soveltamista digitaalisen kuvankäsittelyn ja keinonäön ongelmaan: ympyräkiekkojen etsimiseen sumeasta ja kohinaisesta kuvasta. Kuvaongelman muotoilun yhteydessä esitellään myös digitaalisen kuvankäsittelyn perusteita.

Mallinnettu jäähdytys (engl. *simulated annealing*, suomeksi myös *jäähdytysmenetelmä* tai *simuloitu jäähdytys* [suomennokset ks. HKR93], myöhemmin lyhyesti SA tai SA-algoritmi) on tunnettu probabilistinen optimointialgoritmi globaalin, tavallisesti diskreetin tai kombinatorisen optimointitehtävän ratkaisemiseksi. (Käytettävästä terminologiasta riippuen sitä voidaan nimittää myös optimointiheurestiikaksi tai metaheurestiikaksi.) Jäähdytysalgoritmin esittelivät Kirkpatrick et al. 1983 artikkelissa [KGV83] Metropolis–Hastings-algoritmin sovelluksena erilaisiin kombinatorisiin optimointiongelmiin, kuten kuuluisaan kauppamatkustajan ongelmaan. Metropolis-algoritmi vuorostaan on klassinen Markovin ketjujen Monte Carlo -menetelmä, jonka juuret ovat jäähtyvän fysikaalisen systeemin simuloinnissa. Mallinnettu jäähdytys on yksinkertainen ja monikäyttöinen heurestiikka, jonka hyödyllinen ominaisuus on sen kyky kiivetä ylös energiafunktion paikallisten minimien ”energiamaisemaan” aiheuttamista kuopista [SSF02].

Sivuhuomiona mainittakoon että SA:lla on myös mielenkiintoisia ajankohtaisia yhtymäkohtia kvanttilaskentaan. Metodin lähisukulainen nk. kvanttijäähdytys (*quantum annealing*) perustuu siihen että monia kombinatorisia optimointiongelmia, kuten verkon kahtiajako-ongelma, voidaan esittää spin-lasien Ising-mallin matalaenergisimpien tilojen etsintänä samaan tapaan kuin SA mallintaa termodynaamisen systeemin matalaenergisten tilojen löytämistä [LA87, luku 4.5; Joh+11]. Kvanttijäähdytysalgoritmi on väitetysti tehokas optimointimenetelmä, jos käytettävissä on sopivasti konstruoitu Ising-spin-systeemiä mallintava kvanttitietokone, tai ainakin D-Wave Systems, Inc., esittää kykenevänsä huomattavaan laskentano-

peuden parannukseen tällaisella laitteella [Joh+11; Den+15], mutta nopeushypyn merkittävyydestä ei ole selvyyttä [ZBT15].

Tutkielmassa mallinnettuun jäähdytykseen tutustutaan soveltamalla sitä yksinkertaiseen kuvankäsittelyongelmaan, jossa tehtävänä on tunnistaa tunnettu määrä k tummia ympyräkiekkoja sumentuneesta ja kohinaisesta vaaleataustaisesta kuvasta. Tutkittavaa kuvaongelmaa lähestytään optimointiongelmana: Tutkittavista kuvista muodostetaan malli, jossa kuvan ympyräkiekot parametrisoidaan niiden sijantikoordinaattien ja säteiden avulla ja kiekkojen sumennosta mallinnetaan kuvamatriisin konvoluutio-operaatiolla. Minimoitavaksi *kohde- eli energiafunktioksi* valitaan sopiva parametrisoitujen kiekkojen etäisyyttä kuvadatasta mittaava funktio E. Energiafunktiota minimoimalla voidaan etsiä parametrit, jotka määräävät kuvadataan parhaiten sopivat kiekot.

Tutkielman pääpainopiste on mallinnetussa jäähdytyksessä ja sen käyttäytymisen tutkimisessa kuvatussa esimerkkiongelmassa. Esitetty ympyräongelma ja valittu kuvamalli on reaalimaailman sovelluksia ajatellen varsin rajoitettu, sillä käytännön kuvankäsittelysovelluksissa vastaavat ongelmat ovat usein monella tapaa vaikeampia: häiriölähteitä on enemmän ja kuvan kohteista ei aina voi muodostaa yhtä yksinkertaista mallia.

Ympyräobjektien tunnistaminen kuvasta on perinteinen konenäön alan ongelma, jota voitaisiin lähestyä myös monella muulla tapaa: Esimerkiksi kappaleiden erottelua voitaisiin helpottaa esimerkiksi esikäsittelemällä kuvaa soveltamalla tavallisia kohinanpoistomenetelmiä tai segmentointia. Jos sumennoksen määrä on pieni, vaihtoehtoisesti kunkin alkuperäisen kiekon keskipisteen ja säteen arviointi onnistuisi niin esikäsitellystä kuvasta jollain Houghin ympyrämuunnokseen (Circle Hough Transform) perustuvalla menetelmällä [ks esim Lea92], joille on monissa suosituissa kuvankäsittelyn kirjastoissa (esim. OpenCV, Matlab Image Processing Toolbox) valmiit toteutukset.

Mikäli kuvadata on huomattavan sumeaa, alkuperäisen sumentumattoman kuvan pinta-alan arvioiminen luotettavasti voi olla vaikeaa. Ylipäätään sumeiden kuvien tarkentaminen (*deblurring*) on myös laajalti tutkittu kuvankäsittelyn ongelma, ja tyypillinen esimerkki epätriviaalista inversio-ongelmasta [MS12]. Sumennosta usein mallinnetaan konvoluutio-operaationa, jonka *dekonvoluutioon* voidaan käyttää (riippuen kuinka hyvin konvoluution pistehajontafunktio tunnetaan) esimerkiksi regularisointi- [MS12], Wiener-suodatin-, tai tilastollisia (Bayes) menetelmiä [CVR14]. Tässä tutkielmassa sumentuneen kuvadatan ongelmaa lähestytään naiivisti olettamalla että käytössä on *a priori* -informaatiota (tai muuten hyvä arvaus) kuvaa sumentaneesta prosessista, mitä hyödynnetään optimointimallissa. (Vastakohtaisesti vaikeammassa nk. sokeassa dekonvoluutiossa tällaisia oletuksia ei tehdä, [ks. esim CVR14])

Tutkielman kokeellisen osuuden numeeriseen laskentaan käytettiin Matlab-ohjelmistoa [Matlab14] ja tulosten analysointiin sekä kuvaajien tuottamiseen myös SciPy-ohjelmistoa [J+01] ja sen Matplotlib-pakettia [Hun07].

Tutkielman rakenne on seuraava: Luvussa 2. ”Mallinnettu jäähdytys” lyhyesti todetaan joitain Markovin ketjuja koskevia esitietoja, jonka jälkeen esitetään mallinnetun jäähdytyksen algoritmi ja sen teoreettisia perusteluja. Luvussa 3. ”Kiekko-ongelman kuvamalli” esitetään kiekko-ongelman matemaattinen malli ja hieman tarvittavaa kuvankäsittelyn teoriaa, sekä malli simuloidun aineiston muodostamiseksi. Luvussa 4. ”Jäähdytysalgoritmin soveltaminen kiekko-ongelmaan” kuvaillaan algoritmin tarkemmin soveltamista optimointitehtävänä muotoiltuun kuvaongelmaan, ja luvussa 5. ”Tulokset” esitellään tuloksia sovelletun algoritmin toiminnasta erilaisilla simuloiduilla aineistoilla. Viimeisessä luvussa 6. ”Pohdinta ja johtopäätökset” tarkastellaan tuloksia ja algoritmin toimintaa sekä esitetään yhteenveto tutkielman tuloksista.

Luku 2

Mallinnettu jäähdytys

2.1 Optimointi

Yleisesti matemaattisissa optimointiongelmissa (eli optimointitehtävässä) tavoitteena on löytää annetun ongelman jossain mielessä paras ratkaisu tietystä mahdollisten ratkaisujen joukosta. Ratkaisuehdokkaan hyvyyttä kuvaa ratkaisuehdokkaiden avaruudessa Ω määritelty reaaliarvoinen funktio E (hintafunktio, kohdefunktio), ja optimointiongelman ratkaisu on avaruuden piste jossa E saavuttaa minimin. Ongelmasta riippuen halutaan löytää paikallinen (lokaali) tai globaali minimipiste. (Tehtävänä voi olla myös funktion maksimointi, mutta koska funktion E maksimointitehtävä voidaan aina esittää funktion $-E$ minimointiongelmana, voidaan optimointiongelma määritellä minimoinnin kautta.)

Määritelmä 2.1 (Globaali diskreetti optimointiongelma)**.** Olkoon optimointiongelman mahdollisten ratkaisujen s avaruus Ω. Jäähdytysmenetelmän yhteydessä ratkaisumahdollisuuksia s kutsutaan *tiloiksi*. Diskreetissä optimointiongelmassa tila-avaruus Ω on diskreetti.

Energiafunktio E (myös kohde- tai hintafunktio, engl. *objective function*, *cost function*) on kuvaus tila-avaruudelta reaaliluvuille,

$$E : \Omega \to \mathbb{R}. \tag{2.1}$$

Globaalin optimointiongelman ratkaisu on tila s_{opt}, jossa E saavuttaa globaalin minimin,

$$E(s_{\text{opt}}) = \min_{s \in \Omega} E(s). \tag{2.2}$$

Merkintä. Koska tila-avaruus Ω on diskreetti ja useimmissa sovelluksissa (erityisesti tässä tutkielmassa), kuten kombinatorisissa optimointitehtävissä, lisäksi äärellinen (tai korkeintaan numeroituva), tiloja voidaan mielekkäästi merkitä indeksoidulla

merkinnällä $s_i \in \Omega$, missä $i \in \{1, \ldots, |\Omega|\}$. Tässä tutkielmassa ratkaisuavaruus on äärellinen. Jos sekaannuksen varaa ei ole, tarvittaessa samastamme kunkin tilan indeksinsä kanssa ja merkitsemme lyhyesti s_i:n sijasta i.

Numeerisessa optimointiongelmassa funktion E arvon laskeminen kussakin pisteessä voi olla laskennallisesti kallis operaatio ja hakuavaruus Ω suuri, joten kaikkien alkioiden s_i iteroiminen on useimmiten liian hidasta. Konveksin funktion minimin tai yleisesti paikallisen minimin,

$$s_{\text{lok. opt}} = \min_s E(s), \quad \text{missä } |s - s_{\text{lok. opt}}| < \delta \text{ jollain } \delta > 0, \tag{2.3}$$

löytäminen on helppoa erilaisilla differentiaalilaskentaan pohjautuvilla menetelmillä (esimerksi kvasi-Newton-, konjugaattigradientti- tai jyrkimmän pudotuksen menetelmät). Sen sijaan ei-konveksien funktioiden optimoinnissa haasteena on välttää juuttuminen paikallisiin minimialueisiin, mihin tyypillinen vastaus on hyödyntää algoritmissa satunnaisuutta. Toisaalta haasteena on myös samalla hyödyntää ongelman rakennetta globaalin minimin löytämiseen paremmin kuin esimerkiksi yksinkertaisessa kattavassa satunnaishaussa. Erilaiset probabilistiset ja satunnaisuuteen perustuvat (meta-)heurestiikat (kuten geneettiset algoritmit tai tabu-haku, [ks. esim. GK03]) ja Monte Carlo -menetelmät (mm. *importance sampling*) ovat osoittautuneet toimiviksi tämänkaltaisissa ongelmissa. Yksi tällaisista heurestiikoista on tässä tutkielmassa käsiteltävä jäähdytysalgoritmi. [SSF02]

2.2 Esitieto: Markovin ketjut

Ennen jäähdytysalgoritmin tarkempaa kuvausta käydään läpi joitain perusteita Markovin ketjuista. [1]

Diskreetti Markovin ketju on stokastinen prosessi, jossa prosessin kukin tila riippuu vain edellisestä tilasta. Olettaen että ketjun tila nykyhetkellä tunnetaan, seuraava (tuleva) tila on ehdollisesti riippumaton menneistä tiloista. Tätä kutsutaan ketjun *Markov-ominaisuudeksi* tai *unohtuvaisuusominaisuudeksi*.

Määritelmä 2.2. Stokastinen prosessi (satunnaisprosessi) on jono ajanhetkien $k \in \mathbb{N}$ mukaan järjestettyjä satunnaismuuttujia (satunnaisvektoreita) $\mathbf{X}(k)$ joltain todennäköisyysavaruudelta tilajoukkoon Ω. Merkitään

$$\mathbf{X}(k) = \text{ketjun } k\text{. toteutunut tila.} \tag{2.4}$$

Todennäköisyys että satunnaisprosessin tila k. hetkellä on i on siis näillä merkinnöillä $\Pr\{\mathbf{X}(k) = i\}$. Tässä yhteydessä prosessi on jäähdytysalgoritmi, jonka tila-avaruus on

[1] Merkinnät [LA87] tapaan.

optimointiongelman (korkeintaan numeroituva) tila-avaruus Ω.

Markovin ketju voidaan nyt määritellä formaalisti ehdollisen riippumattomuuden avulla:

Määritelmä 2.3. Markovin ketju on satunnaisprosessi $\mathbf{X}(1), \mathbf{X}(2), \mathbf{X}(3), \ldots$, joilla pätee

$$\Pr\left\{\mathbf{X}(k+1) = i \mid \mathbf{X}(1) = j_1, \mathbf{X}(2) = j_2, \ldots \mathbf{X}(k) = j_k\right\} = \Pr\left\{\mathbf{X}(k+1) = i \mid \mathbf{X}(k) = j_k\right\} \tag{2.5}$$

kaikille tiloille $i, j_1, \ldots, j_k$.

Markov-ketjun määräävät siis ehdolliset todennäköisyydet $\Pr\{\mathbf{X}(k+1) = i \mid \mathbf{X}(k) = j\}$, jotka kertovat todennäköisyyden sille, että $k+1$. siirtymän jälkeen ollaan tilassa i olettaen, että k. askeleen tila oli j. Riippumattomuus tarkoittaa, että k:ta edeltävät tilat eivät vaikuta todennäköisyyteen $\mathbf{X}(k+1) = i$.

SA-algoritmin määrittelyä varten tarvitaan Markovin ketjuja, joilla on tiettyjä ominaisuuksia:

Määritelmä 2.4. *Homogeeni* Markovin ketju eli aikahomogeeni Markovin ketju on Markovin ketju, jossa siirtymätodennäköisyydet pysyvät samoina kaikilla k.

Termillä 'aikahomogeeni' korostetaan, että kyseessä on dynaaminen prosessi jossa siirtymätodennäköisyydet pysyvät samoina ajan suhteen.

Homogeenissa Markovin ketjussa siirtymät määrää jokaisella askeleella k sama siirtymämatriisi M

$$M_{i,j} = \Pr\{\mathbf{X}(k+1) = i \mid \mathbf{X}(k) = j\}. \tag{2.6}$$

Määritelmä 2.5. *Ergodisessa*[2] Markovin ketjussa mikä tahansa tila on saavutettavissa mistä tahansa muusta ketjun tilasta jollain siirtymien sarjalla, eli kaikilla i, j on äärellinen $n \in \mathbb{N}$, jolla

$$\Pr\{\mathbf{X}(n) = i \mid \mathbf{X}(0) = j\} > 0. \tag{2.7}$$

Sanotaan, että kaikki tilat *kommunikoivat*.

Määritelmä 2.6. *Periodisessa* Markovin ketjussa jokin tila i toistuu p siirtymän jaksoissa jollain $p \in \mathbb{N}$, eli on olemassa jakso

$$p = \operatorname{syt}\{k : \Pr\{\mathbf{X}(k) = i \mid \mathbf{X}(0) = i\} > 0\} > 1, \tag{2.8}$$

missä syt on suurin yhteinen tekijä.

[2]Käytämme teoksen [SSF02] terminologiaa; joissain muissa määritelmissä ergodisuuteen vaaditaan hieman vahvempia oletuksia, ja tässä kuvailtua ominaisuutta kutsutaan *pelkistymättömyydeksi* (engl. *irreducibility*).

Määritelmä 2.7. *Aperiodiseksi* sanotaan Markovin ketjua jossa $p = 1$ kaikilla tiloilla i, eli yhtäpitävästi ketjua, jossa kaikilla tiloilla i on sellainen $k \in \mathbb{N}$ niin, että kaikilla $k' \geqslant k$ todennäköisyys

$$\Pr\{\mathbf{X}(k') = i \mid \mathbf{X}(0) = i\} > 0. \tag{2.9}$$

Lause 2.1. *Mikäli homogeenilla ergodisella Markovin ketjulla on olemassa tila i jolla* $M_{ii} > 0$*, ketju on aperiodinen.*

Todistus. Määritelmän mukaan

$$M_{ii} = \Pr\{\mathbf{X}(k+1) = i \mid \mathbf{X}(k) = i\} > 0 \quad \text{kaikilla } k, \tag{2.10}$$

eli tila i on aperiodinen. Koska oletuksen mukaan kaikki ketjun tilat kommunikoivat, kaikki muut tilat $j \neq i$ ovat saavutettavissa tilasta i ja päinvastoin äärellisessä ajassa, eli joillain n_1, n_2 pätee

$$\Pr\{\mathbf{X}(k+n_1+n_2) = j \mid \mathbf{X}(k+n_1) = i, \mathbf{X}(k) = j\} > 0. \tag{2.11}$$

Koska $M_{ii} > 0$, tilan j todennäköisyys on positiivinen myös tapahtumille $\mathbf{X}(k+n_1+n_2+1), \mathbf{X}(k+n_1+n_2+2), \ldots$, eikä siis tilalla j voi olla periodia. □

Määritelmä 2.8. Aikahomogeenisen Markovin ketjun *stationaarinen jakauma* $\boldsymbol{\pi}$ on vektori, jolle pätee

$$0 \leqslant \boldsymbol{\pi}(i) \leqslant 1, \tag{2.12}$$

$$\sum_i \boldsymbol{\pi}(i) = 1, \quad \text{ja} \tag{2.13}$$

$$\boldsymbol{\pi}(i) = \sum_j \boldsymbol{\pi}(j) M_{i,j} = \sum_j \boldsymbol{\pi}(j) \Pr\{\mathbf{X}(k+1) = i \mid \mathbf{X}(k) = j\}. \tag{2.14}$$

Seuraavat perustavanlaatuiset ([LA87; SSF02]) Markovin ketjuja koskevat lauseet oletetaan tunnetuiksi:

Lause 2.2. *Aikahomogeenisella Markovin ketjulla on stationaarinen jakauma (tasapainojakauma) jos ja vain jos ketju on aperiodinen ja ergodinen.*

Lause 2.3. *Jos (homogeenilla) Markovin ketjulla on stationaarinen jakauma* $\boldsymbol{\pi}$*, jokainen Markovin ketju konvergoi sitä kohti kun ketjun pituus kasvaa rajatta, eli*

$$\boldsymbol{\pi}(i) = \lim_{n\to\infty} \Pr\{\mathbf{X}(n) = i \mid \mathbf{X}(0) = j\} \quad \textit{kaikilla } j. \tag{2.15}$$

2.3 Jäähdytysalgoritmi ja sen matemaattinen malli

Kuvaillaan seuraavaksi ensin jäähdytysalgoritmin toimintaidea (jaksot 2.3.1, 2.3.2) ja sitten tarkempi matemaattinen malli Markovin ketjuina (jakso 2.3.3).

2.3.1 Jäähdytysalgoritmin idea

Jäähdystysalgoritmin toiminta perustuu termodynaamiseen analogiaan kiinteiden metallikappaleiden jäähdyttämiseen tietynlaisessa metallurgian lämpökäsittelymenetelmässä (menetelmän engl. nimitys *annealing*). Menetelmässä metallikappale kuumennetaan niin korkeaan lämpötilaan, että aineen partikkelit alkavat liikkua vapaasti ja niiden keskinäinen järjestys on satunnainen. Tämän jälkeen kappaleen annetaan hitaasti jäähtyä, erityisen hitaasti jäätymispisteen lähellä. Näin jäähdytettävä aine kristalloituu matalaenergisimpään perustilaansa, ja partikkelit muodostavat säännöllisen, virheettömän rakenteen [KGV83; LA87]. Optimointiongelman kohdefunktion arvot samastetaan jäähdytettävän kappaleen muodostaman fysikaalisen systeemin energiatiloihin. Algoritmissa systeemin kuhunkin lämpötilaansa liittyvän tasapainotilan (engl. *thermal equilibrium*) lähestymistä simuloidaan Metropolis–Hastings-algoritmilla [KGV83; SSF02].

2.3.2 Algoritmin kuvaus: Metropolis-kriteeri

Oletetaan että olemme määritelleet jonkin tavoite- eli energiafunktion E. Määritelmän 2.1 mukaisesti tavoitteena on löytää tilavektori s_{opt} joka minimoi energiafunktion $E(s_{\text{opt}})$ (yhtälö 2.2). Määritellään lisäksi kontrolliparametri $t > 0$, joka vastaa fysikaalisen systeemin lämpötilaa.

SA-algoritmi simuloi jäähtyvää fysikaalista systeemiä seuraavasti: Alkutilassa lämpötilan $t = t_0$ lähtöarvo t_0 on korkea ja se vastaa fysikaalisessa prosessissa tilannetta, jossa termodynaamisen systeemin partikkelit liikkuvat vapaasti (satunnaisesti). Oletetaan että hetkellä k nykyinen kandidaatti optimoitavan parametrivektorin arvoksi (eli fysikaalisessa tulkinnassa *systeemin tila*) on s_i. Generoidaan uusi, satunnainen tila s_j nykyisen konfiguraation s_i läheltä jonkin sopivan etäisyysmitan suhteen, ja hyväksytään se uudeksi vallitsevaksi arvaukseksi *Metropolis-kriteerin* perusteella:

Jos uusi tila s_j on energiafunktion mielessä parempi kuin edellinen (eli $E(s_j) \leqslant E(s_i)$), se hyväksytään uudeksi nykyiseksi ratkaisuehdokkaaksi. Jos s_j on huonompi kuin s_i (eli $E(s_j) > E(s_i)$), hyväksymme sen joka tapauksessa lämpötilasta t riippuvalla todennäköisyydellä

$$\exp\left(-\frac{\Delta E}{t}\right), \tag{2.16}$$

missä $\Delta E = \Delta E_{ji} = E(s_j) - E(s_i)$. Mikäli ehdokasta s_j ei hyväksytä, pysytään entisessä tilassa s_i.

Kun algoritmia on iteroitu tarpeeksi kauan (ja fysikaalisessa tulkinnassa simuloitava systeemi on saavuttanut sen hetkiseen lämpötilaan liittyvän termodynaamisen tasapainotilan), lämpötilaparametria t lasketaan hieman, ja algoritmia jatketaan kunnes systeemi on ”jäähtynyt”. Tällä tarkoitetaan että algoritmi pysäytetään tarpeeksi pienellä t:n arvolla, yleensä kun voidaan jonkin valitun kriteerin avulla todeta systeemin jäähtyneen matalaenergisimpään rakenteeseensa.

Edellä kuvattu prosessi on analoginen aiemmassa alaluvussa 2.3.1 kuvaillun lämpökäsittelyprosessin fysikaalisen mallin kanssa, mitä tutkitaan tarkemmin myöhemmin (alaluku 2.4). Intuitiivisesti kontrolliparametrin eli ”lämpötilan” ollessa suuri algoritmi hyväksyy runsaasti satunnaisia siirtymiä energiafunktion suhteen ”ylämäkeen”, ja tilaehdokkaiden s_i jono on hyvin satunnainen. Lämpötilan laskiessa algoritmi hyväksyy yhä harvemmin satunnaisia poikkeammia, ja sen sijaan valitsee todennäköisemmin vain energiafunktion suhteen parempia kandidaatteja s_i. Tämä stokastisuus selittää algoritmin kyvyn paeta paikallisia minimejä.

Jäljempänä tutkielmassa alaluvussa 2.4.3 esitetään, että tiettyjen teoreettisten ehtojen vallitessa tilajono s_i konvergoi kohti globaalia minimiä s_{opt} todennäköisyydellä 1 kun $t \to 0$.

2.3.3 Algoritmin malli Markovin ketjuina

Algoritmin läpikäymiä tiloja s ajanhetkillä $k = 1, \dots, n$ voidaan mallintaa sarjana homogeeneja Markovin ketjuja kullakin lämpötilaparametrin t arvolla:

Merkitään algoritmin hetkellä k toteutunutta tilaa $\mathbf{X}(k)$, jolloin algoritmi on stokastinen prosessi $\mathbf{X}(1), \mathbf{X}(2), \dots, \mathbf{X}(n)$, jonka siirtymätodennäköisyydet hetkellä k

$$\Pr\left\{\mathbf{X}(k+1) = j \mid \mathbf{X}(k) = i\right\} \tag{2.17}$$

riippuvat Metropolis-kriteerin mukaisesti tiloista i ja j ja tuonhetkisestä lämpötilasta $t = t_k$. Jos lämpötila pysyy askeleiden $k, \dots, k+m$ ajan vakiona, $t_k, t_{k+1}, \dots, t_{k+m} = t_c$, tilat $\mathbf{X}(k), \dots, \mathbf{X}(k+m)$ muodostavat homogeenin äärellispituisen Markovin ketjun.

Koska kullakin t:n arvolla Metropolis-kriteerin määräämät siirtymätodennäköisyydet säilyvät samoina, algoritmin Markov-ketjujen siirtymätodennäköisyydet voidaan esittää lämpötilaparametrista $t > 0$ riippuvan siirtymämatriisin $M(t)$ avulla,

$$M_{ji}(t) = \begin{cases} G_{ji}(t)A_{ji}(t), & j \neq i \\ 1 - \sum_{l \neq i} G_{li}(t)A_{li}(t), & j = i, \end{cases} \tag{2.18}$$

missä *generointitodennäköisyys* $G_{ji}(t)$ ilmaisee todennäköisyyden että tilakonfiguraatio

s_j generoidaan lähtötilasta s_i, ja *hyväksymistodennäköisyys* $A_{ji}(t)$ todennäköisyyden että lähtötilasta s_i generoitu tila s_j hyväksytään.

Tässä tutkielmassa oletetaan että $G(t)$ on yksinkertaisesti (tasainen) satunnaisjakauma, jonka perusjoukko on lähtötilan s_i naapuritilat $N(i)$,

$$G_{ji}(t) = \frac{1}{m(N(i))}, \tag{2.19}$$

missä m on sopiva tn-mitta, diskreetissä tapauksessa naapureiden lukumäärä $\#N(j)$, jolloin

$$G_{ji}(t) = \frac{1}{\#N(i)}, \tag{2.20}$$

ja $A(t)$ on edellä jaksossa 2.3.2 kuvailtu Metropolis-kriteeri, joka voidaan kirjoittaa lyhyesti

$$A_{ji}(t) = \min\left\{1, \exp\left(-\frac{\Delta E}{t}\right)\right\}, \tag{2.21}$$

kun huomataan että $\exp\left(-\frac{\Delta E}{t}\right) \geqslant 1$ kun $\Delta E \leqslant 0$.

Algoritmiin liittyy myös jokin jäähdytysstrategia, johon sisältyy lämpötilaparametrin t alkuarvo t_0, funktio $T : \mathbb{N} \to \mathbb{R}_+$ joka määrää sen arvot kullakin homogeenilla ketjulla $t_k = T(k)$, kunkin homogeenin Markov-ketjujen pituus L_k, ja algoritmin pysäytyskriteeri (tyypillisesti lämpötila t_{stop} tai iteraatio i_{stop}, joka määrää ketjun tai hetken jolloin algoritmi pysähtyy).

Yllä esitettyä SA:n muotoilua Laarhoven ja Aarts [LA87] kutsuvat *homogeeniksi* algoritmiksi erotuksena epähomogeenista muotoilusta, jossa kontrolliparametrin t:n arvoa lasketaan jokaisen siirtymän jälkeen, jolloin Markov-ketjut eivät ole aikahomogeeneja.

2.4 Algoritmin teoreettinen perustelu

Perustellaan seuraavaksi hieman miksi jäähdytysalgoritmi toimii. SA on pohjimmiltaan sovellettu Metropolis–Hastings-algoritmi (tai lyhyesti Metropolis-algoritmi) joka tuottaa (pseudo-)satunnaisotoksen s yleisestä Boltzmannin jakaumasta

$$\Pr(s) \propto \exp\left(-\frac{E(s)}{t}\right). \tag{Boltzmann}$$

Tämän tutkielman piiriin ei kuulu asymptoottinen konvergenssitarkastelu kuinka Boltzmannin jakauman approksimointi vähenevillä lämpötiloilla johtaa glo-

baaliin optimiin, vaan nojaamme vain tilastolliseen mekaniikkaan perustuvaan fysikaaliseen intuitioon.

Esitetään kuitenkin tavanomainen todistus ([CG95] mukaan) miksi alaluvussa 2.3 kuvattu Metropolis–Hastings-algoritmi todella approksimoi haluttua jakaumaa (tässä Boltzmann-jakaumaa tietyssä lämpötilassa) tietyin Markov-ketjua koskevin oletuksin, ja lyhyt yhteenveto teoksessa [LA87] esitetyistä laajemmista teoreettisista konvergenssituloksista.

2.4.1 Tilastollinen mekaniikka ja Boltzmannin jakauma

Tilastollisessa fysiikassa *tilastollinen mekaniikka* tutkii tiiviin (kiinteän ja nestemäisen) aineen makroskooppista käyttäytymistä ja erityisesti termodynaamisia systeemeitä soveltamalla tilastollisia menetelmiä klassisen mekaanikan tai kvanttimekaniikan mukaisiin aineen mikroskooppisiin malleihin. Tilastollisessa mekaniikassa oletetaan että fyysisen systeemin makroskooppinen tila voidaan mallintaa todennäköisyysjakaumana sen mahdollisten mikroskooppisten tilojen (konfiguraatioiden) kokoelman yli. Aineen mikroskooppisella tilalla tarkoitetaan aineen kaikkien yksittäisen (tässä yhteydessä klassisen mekaniikan käsityksen mukaisten) hiukkasten tilaa. [SSF02]

Jäähdytysmenetelmän tarkastelun kannalta tilastollisen mekaniikan keskeinen tulos on, että mikroskooppisilla tiloilla on lämpötilassa t tasapainojakauma (termodynaaminen tasapainotila), joka voidaan johtaa [ks. esim. SSF02, luku 5] ja monissa malleissa (kun kvantti-ilmiöitä ei huomioida) on Boltzmannin jakauma (2.22). Termodynaamisessa tasapainotilassa lämpötilassa t systeemi on tilassa s jonka sisäenergia on $E = E(s)$ todennäköisyydellä

$$\Pr(s) = \frac{1}{Z(t)} \exp\left(-\frac{E(s)}{k_B t}\right), \tag{2.22}$$

missä k_B on Boltzmannin vakio, t systeemin termodynaaminen lämpötila, ja jakofunktio $Z(t)$ on lämpötilasta riippuva normittava tekijä

$$Z(t) = \sum_i \exp\left(-\frac{E(s_i)}{k_B t}\right), \tag{2.23}$$

missä summa otetaan mahdollisten energiatilojen $E(s_i)$ yli.

Tilastollisen fysiikan tunnettu tulos on, että kun jäähdytysalgoritmin simuloimassa jäähdytysprosessissa $t \to 0$ tarpeeksi hitaasti, tutkittava systeemi on saavuttaa kaikissa läpikäymissään lämpötiloissa jakauman 2.22 kuvaaman termodynaamisen tasapainotilan. Lämpötilan t laskiessa todennäköisyysjakauma 2.22 keskittyy pienienergisten tilojen s ympärille, ja kun lämpötila lähestyy nollaa, vain energioiltaan pienimpien tilojen todennäköisyys on positiivinen. Toisin sanoen kun t lähestyy nol-

laa, systeemi saavuttaa matalaenergisimmän perustilansa (engl. *low energy ground state*), jossa aineen hiukkaset kristalloituvat hilarakenteeseen.

Kuten edellä (jakso 2.3.2) on kuvailtu, mallinnetussa jäähdytyksessä optimointiongelman mahdolliset ratkaisut $s_i \in \Omega$ vertautuvat fysikaalisen systeemin tiloihin, joiden energiatilat antaa optimoitava funktio E. Vastaavasti jäähdytysalgoritmin tilojen tulkitaan kullakin lämpötilaparametrin t arvolla noudattavan Boltzmann-muotoista jakaumaa

$$\pi_t(i) = \frac{1}{Q(t)} \exp\left(-\frac{E(s_i)}{t}\right), \tag{2.24}$$

missä $Q(t)$ on fysikaalista jakofunktiota vastaava normittava tekijä

$$Q(t) = \sum_{s \in \Omega} \exp\left(-\frac{E(s)}{t}\right). \tag{2.25}$$

Mallinnettussa jäähdytyksessä fysikaalista prosessia simuloidaan Metropolis-algoritmilla, joka approksimoi yhtälön (2.24) muotoista Boltzmannin jakaumaa.

Todellisuudessa monet kombinatoriset ongelmat joihin jäähdytysalgoritmia sovelletaan eivät kuitenkaan aina käyttäydy kuin tyypilliset termodynaamiset systeemit. Tällaisissa tilanteissa algoritmin käyttäytymistä usein verrataan lasin kaltaisiin systeemeihin (*glassy systems*). Kyseisen kaltaisen aineen rakenne jää epäoptimaaliseen järjestykseen kun sen lämpötila laskee aineen nk. neste-lasi-transitiolämpötilan alapuolelle: sanotaan että systeemi käy läpi neste-lasitransition. [SSF02, luku 6.6]

2.4.2 Algoritmin tasapainojakauma ja Metropolis–Hastings-algoritmi

Lauseen 2.2 mukaan ergodisella ja aperiodisella Markovin ketjulla on olemassa jokin stationaarinen jakauma π. Määritelmän 2.5 mukaan ergodisessa Markovin ketjussa jokainen tila on saavutettavissa mistä tahansa toisesa tilasta äärellisessä ajassa. Sopivasti toteutetussa jäähdytysalgoritmissa oletus vaikuttaa mielekkäältä. SA-algoritmi voidaan myös olettaa aperiodiseksi: missä tahansa algoritmin tilassa (joka ei ole paikallinen maksimi) Metropolis-kriteerin johdosta algoritmilla on positiivinen todennäköisyys hylätä generoitu siirtymä ja säilyttää edellinen tila. Näin ollen algoritmilla on kaikissa lämpötiloissa t tiloja joilla $M_{ii}(t) > 0$, ja lauseen 2.1 mukaan algoritmin ketjut ovat aperiodisia.

Näin ollen on perusteltua olettaa että järkevästi toteutettu SA-algoritmi konvergoi kohti jotain tasapainojakaumaa. Edellisessä alaluvussa perusteltiin algoritmin kykyä löytää globaali optimi tilastolliseen fysiikkaan perustuvan intuition kautta, minkä kannalta oleellista on että mallinnettu jäähdytys approksimoi nimenomaisesti Boltzmannin jakaumaa (2.24).

Mallinnettu jäähdytys on erikoistapaus yleisestä Metropolis–Hastings-algoritmista. Metropolis–Hastings on suosittu Markovin ketjujen Monte Carlo -menetelmä (engl. *Markov Chain Monte Carlo*, lyhyesti MCMC) jonkin halutun todennäköisyysjakauman $\pi(x)$ approksimointiin, ja erityisen hyödyllinen kun todennäköisyysjakaumasta $\pi(x)$ on vaikea generoida satunnaisotantaa, mutta tunnetaan funktio f, jolle pätee

$$\frac{\pi(x')}{\pi(x)} = \frac{f(x')}{f(x)}. \tag{2.26}$$

Osoitetaan että mallinnetun jäähdytyksen tasapainojakauma on Boltzmannin jakauma johtamalla yleinen Metropolis–Hastings-algoritmi, jonka tasapainojakauma on $\pi(x)$. Toisin sanoen oletetaan, että $\pi(x)$ on määritelmän 2.8 mukainen vektori ja erityisesti halutaan, että siirtymämatriisin M määräämälle Markovin ketjulle pätee määritelmän 2.8 kolmas ehto

$$\pi(i) = \sum_j \pi(j) M_{i,j}. \tag{2.27}$$

Oletetaan, että etsittävän Metropolis-algoritmin Markovin ketju toteuttaa täydellisen tasapaino- eli kääntyvyysominaisuuden

$$\pi(j) M_{i,j} = \pi(i) M_{j,i}. \tag{2.28}$$

Nähdään, että tasapainojakauman määritelmän 2.8 kolmas ehto seuraa kääntyvyydestä:

$$\sum_j \pi(j) M_{i,j} = \sum_j \pi(i) M_{j,i} = \pi(i) \sum_j M_{j,i} = \pi(i), \tag{2.29}$$

missä viimeinen askel pätee sillä luonnollisesti $\sum_j M_{j,i} = \sum_j \Pr\{\mathbf{X}(k+1) = j \mid \mathbf{X}(k) = i\} = 1$ kaikilla i.

Johdetaan nyt yleinen Metropolis-algoritmi etsimällä Markov-ketju joka toteuttaa kääntyvyysominaisuuden (2.28), jolloin sillä on haluttu stationaarinen jakauma π.

Kirjoitetaan $M_{i,j} = \Pr\{\mathbf{X}(k+1) = i \mid \mathbf{X}(k) = j\}$ generointi- ja hyväksymistodennäköisyyksien $G_{i,j}$ ja $A_{i,j}$ avulla

$$M_{i,j} = G_{i,j} A_{i,j}, \quad i \neq j.$$

Sijoitetaan tämä yhtälöön (2.28), jolloin saadaan

$$\pi(j) G_{i,j} A_{i,j} = \pi(i) G_{j,i} A_{j,i} \tag{2.30}$$

Oletetaan että myös generointitodennäköisyydet toteuttavat kääntyvyysehdon $G_{i,j} = G_{j,i}$, mikä SA:n tapauksessa on useimmiten järkevä oletus. Tällöin yhtälöksi tulee

$$\pi(j)A_{i,j} = \pi(i)A_{j,i} \tag{2.31}$$

On vielä valittava hyväksymistodennäköisyyksille $A_{i,j}$ sellainen todennäkoisyysjakauma että tämä pätee kaikilla i, j.

Oletetaan että tilat i, j ovat sellaiset, että

$$\pi(j) < \pi(i). \tag{2.32}$$

Toisin sanoen todennäköisyys että olemme tilassa i on suurempi kuin tilassa j. Koska oletuksen mukaan $G_{i,j} = G_{j,i}$, tasapainoehdon (2.31) säilymiseksi hyväksymistodennäköisyyden $A_{i,j}$ siirtymälle $j \to i$ on oltava suurempi kuin hyväksymistodennäköisyyden $A_{j,i}$ siirtymälle $i \to j$, eli on oltava $A_{j,i} < A_{i,j}$. Toisaalta todennäköisyytenä $A_{i,j}$ ei voi olla suurempi kuin 1; *valitaan* se mahdollisimman suureksi $A_{i,j} = 1$, eli mikäli tehdään siirtymä tilasta j tilaan i joka on todennäköisempi kuin j (2.32), se hyväksytään automaattisesti.

Ehdosta $A_{i,j}$ ja yhtälöstä (2.31) saadaan kaava hyväksymistodennäköisyydelle $A_{j,i}$,

$$A_{j,i} = \frac{\pi(j)}{\pi(i)}. \tag{2.33}$$

Vastaavat yhtälöt luonnollisesti pätevät mikäli $\pi(j) > \pi(i)$.

Hyväksymimistodennäköisyydelle $A_{i,j}$ on siis näin johdettu kaava kun $i \neq j$,

$$A_{i,j} = \min\left\{1, \frac{\pi(i)}{\pi(j)}\right\}. \tag{2.34}$$

Algoritmin (2.18) johto on valmis, kun vielä asetamme tapauksessa $i = j$

$$M_{i,j} = M_{i,i} = 1 - \sum_{l \neq j} G_{l,j}A_{l,j}. \tag{2.35}$$

Kuten edellä (alaluku 2.4.1) perusteltiin, SA-algoritmissa approksimoitava jakauma π on kullakin t:n arvolla Boltzmannin jakauma (2.24)

$$\pi = \pi_t(i) = \frac{1}{Q(t)} \exp\left(-\frac{E(s_i)}{t}\right). \tag{2.36}$$

Koska

$$\frac{\pi(j)}{\pi(i)} = \frac{\pi_t(j)}{\pi_t(i)} = \exp\left(\frac{E(s_i) - E(s_j)}{t}\right), \tag{2.37}$$

yhtälö (2.34) saa tällä jakaumalla muodon

$$A_{j,i} = \min\left\{1, \frac{\pi(j)}{\pi(i)}\right\} = \min\left\{1, \exp\left(-\frac{\Delta E}{t}\right)\right\}, \quad \text{missä } \Delta E = \Delta E_{j,i} = E(s_j) - E(s_i), \tag{2.38}$$

mikä on juuri haluttu Metropolis-kriteeri (2.21).

Lauseen 2.3 mukaan Metropolis-algoritmi konvergoi kohti stationaarista jakaumaa π todennäköisyydellä 1 kun Markov-ketjujen pituudet kasvavat rajatta kullakin t.

2.4.3 Konvergenssituloksia

Yllä oletettiin että ketju toteuttaa täydellisen (myös paikallisen) tasapainoehdon (engl. *detailed balance*) (2.28). Esimerkiksi Laarhoven ja Aarts esittävät kirjallisuudessa tunnettuja todistuksia [LA87, Luvut 3.1.2, 3.1.3], joissa Markovin ketjuja koskevat oletukset voivat olla lievempiä (paikallisen tasapainon sijaan riittää olettaa esimerkiksi nk. *globaali* tasapainoehto) ja samalla osoitetaan, että algoritmi konvergoi asymptoottisesti kohti globaalia optimia s_{opt} tai (jos minimi ei ole yksikäsittäinen) globaalin minimin antavien konfiguraatioiden joukon S_{opt} (tasa-)jakaumaa kun $t \to 0$, eli

$$\lim_{t \to 0}\left(\lim_{k \to \infty} \Pr\left\{\mathbf{X}(k) \in S_{\text{opt}}\right\}\right) = 1. \tag{2.39}$$

Lisäksi voidaan osoittaa [LA87, Luku 3.2; GG84] että kun Markov-ketjujen pituus kullakin t on yksi tai yleisemmin äärellinen (epähomogeeni algoritmi), algoritmi silti konvergoi kohti optimia kunhan $t \to 0$ riittävän hitaasti, missä 'riittävän hitaasti' tarkoittaa ehtoa muotoa

$$t_n = \frac{\Gamma}{\log(n)}. \tag{2.40}$$

Vakiolle Γ tunnetaan erilaisia rajoja $\Gamma \geqslant d$, jossa d on ongelman rakenteesta riippuva parametri [LA87, s. 38]. Mielenkiintoinen SA:n ominaisuus on että myös ehtoa (2.40) nopeammat jäähdytysstrategiat toimivat usein hyvin, vaikkei niille ole yhtä varmaa teoreettista perustelua.

Luku 3

Kiekko-ongelman kuvamalli

3.1 Digitaalinen harmaasävykuva

Digitaalisessa kuvankäsittelyssä kuvan ja erityisesti valokuvan tapaisen kuvadatan tyypillinen esitysmuoto on pieninä kuvaelementteinä, *pikseleinä*, (engl. *pixel*, lyhenne termistä *'picture element'*) ruudukossa. Esitysmuotoa kutsutaan *rasterikuvaksi* tai *bittikarttakuvaksi*. [1] Harmaasävykuvassa (engl. *greyscale image*) kullakin pikselillä on numeerinen arvo, joka kertoo pikselin kattaman kuvapinnan alueen keskimääräisen valoisuuden eli intensiteetin. Tästä nimi "bittikartta": yleisissä tietokoneohjelmistojen käyttämissä tallennusmuodoissa pikselin valoisuus ilmaistaan 8-bittisellä kokonaisluvulla väliltä 0 ja $2^8 - 1 = 255$ (binäärinä 0_2 ja 1111111_2). Vuorostaan RGB-värikuvassa pikseliin liittyy väri-informaation antava RGB-arvo (kolme erillistä 8-bittistä kanavaa punaiselle, vihreälle ja siniselle).

Harmaasävykuvan luonnollinen vastaava matemaattinen malli on $m \times n$-matriisi

$$A \in [0,1]^{m \times n}, \tag{3.1}$$

missä m ja n antavat kuvan koon. Alkioiden arvot ovat reaalilukuja välillä $[0,1]$, jotka vastaavat kyseisen pikselin valoisuutta: 0 on täysin musta pikseli, 1 valkoinen. [Jäh02, s. 29–32]

Matriisin koordinaateissa (i,j) sijaitsevaa pikseliä merkitään $A(i,j)$.

Vaikka matemaattisissa malleissa kuvamatriisin alkiot käsitetään reaaliluvuiksi, tietokone laskee diskreetissä lukuavaruudessa. Tässä tutkielmassa esitellyn algoritmin käytännön Matlab-toteutuksessa laskenta tapahtuu liukulukumatriiseilla, jotka esitetään yllä kuvaillun kaltaisina pikselikuvina. Vaikka liukulukulaskenta pystyy mallintamaan reaalilukuja vain tiettyyn tarkkuuteen asti, ovat siitä aiheutuvat virheet niin pieniä ettei niitä tarvitse tämän tutkielman piirissä huomioida

[1] Mainittakoon, että on myös olemassa vektorigrafiikkaa, jossa kuva esitetään geometristen objektien (kuten polkujen) avulla. Jatkossa 'kuvalla' ja 'kuvadatalla' tarkoitetaan rasteri- / bittikarttakuvaa.

tarkemmin.

3.2 Ongelmanasettelu ja kuvamalli

Oletetaan että tutkittavana on digitaalinen harmaasävyvalokuva, joka esittää k kappaletta erillisiä mustia objekteja valkoisella taustalla, ja objektit voidaan tulkita tasavärisiksi ympyräkiekoiksi. Tehtävänä on määrittää kuvadatasta objektien $w_1, \ldots, w_k$ määrä, sijainti ja koko. Tutkittavien harmaasävykuvien ajatellaan syntyneen tavallisen (digitaalisen) kameran kaltaisessa systeemissä, joten kuvamallissamme otetaan huomioon tyypillisiä tällaisessa systeemissä kuvadataa huonontavia virhelähteitä.

Matemaattinen mallimme k ympyräkiekkoa $W = w_1, ..., w_k$ tuottavasta prosessista on

$$A_{\text{havaittu}} = P * A_W + E, \tag{3.2}$$

missä havaittu lopullinen kuvadata A_{havaittu} on ideaalinen ympyräkiekkoja esittävä kuva A_W, johon on kohdistunut kahdenlaisia virheitä: kuva on sumentunut, mitä mallinnetaan konvoluutio-operaatiolla $P * A_W$, ja siinä on additiivista kohinaa E.

Seuraavaksi määritellään ideaalisen kuvamatriisin A_W tuottaminen kiekkokokoelmasta W, konvoluutio-operaatio ja additiivinen kohina. Kuvaillaan myös kuinka tätä yleistä kuvamallia sovelletaan simuloidun aineiston tuottamiseen ja aineiston tutkimiseen käytetty 'havaintomalli', jota hyödynnetään luvussa 4 kiekkojen löytämiseen simuloidusta aineistosta.

3.3 Ympyräkiekot

Tavoitteena on rekonstruoida kuvadatan pohjalta ympyräobjektit, joista kukin voidaan yksilöidä järjestettynä kolmikkona (tai vektorina)

$$w = (x_0, y_0, r) \in m \times n \times \mathbb{R}, \tag{3.3}$$

missä x_0, y_0 on ympyrän keskipisteen koordinaatti $m \times n$ kuvamatriisissa A ja r ympyrän säde.

3.3.1 Yksi kiekko

A priori -tietomme kuvan esittämien kappaleiden muodosta ja luonteesta mahdollistaa niiden oleellisen informaatiosisällön esittämisen kolmella parametrilla. Tulkitaan että kolmikko (x_0, y_0, r) määrää yhden mustan ympyrän valkoisella pohjalla kuvamatriisissa A seuraavan funktion avulla:

Määritelmä 3.1. Ympyrä, jonka keskipiste on (x_0, y_0) ja säde $r > 0$, voidaan kuvata kuvamatriisiksi A funktiolla $f : m \times n \times \mathbb{R} \to [0,1]^{n \times n}$,

$$f(x_0, y_0, r) = A, \text{missä } \begin{cases} A_{i,j} & = 0, \text{ jos pikselin } (i,j) \text{ keskipiste on ympyrän sisällä} \\ A_{i,j} & = 1, \text{ muulloin} \end{cases}$$

missä ympyrän (ympyräkiekon sisällä) jos $\|(i,j) - (x_0, y_0)\| \leqslant r$.

Teimme yllä muutaman tietoisen yksinkertaistuksen numeerisen laskennan helpottamiseksi. Ensinnäkin oletamme että ympyräkiekkojen keskipisteet sijaitsevat aina tarkalleen jonkin pikselin kohdalla (x_0, y_0 kokonaislukuja jotka vastaavat pikselien koordinaatteja); realistisemmin olettaisimme koordinaatit x_0 ja y_0 reaaliluvuiksi kuvamatriisin alueella, ja laskisimme mitkä pisteet kuuluvat ympyräkiekkoon laskemalla niiden etäisyydet niihin. Toiseksi mallimme ympyräkiekosta on varsin karkea graafisessa mielessä: pikseli on joko täysin musta tai valkea.

3.3.2 Monta kiekkoa

Malli voidaan yleistää useammalle kiekolle. Oletetaan että meillä on k kappaleen (järjestämätön) kokoelma kiekkoja $W = \{w_1, \ldots, w_k\}$. Intuitiivisesti vastaavassa kuvamatriisissa pikseli on valkea, jos se ei kuulu mihinkään ympyräkiekkoon, ja musta jos se kuuluu johonkin kiekkoon. Kokoelma on järjestämätön, sillä tulokseksi saadaan samanlainen kuva riippumatta missä järjestyksessä sen ympyrät luetellaan.

Määritelmä 3.2. k ympyräkiekkoa $w_1, \ldots, w_k$ voidaan kuvata kuvamatriisiksi A funktiolla $f_{\text{monta}} : (m \times n \times \mathbb{R})^k \to [0,1]^{m \times n}$,

$$f_{\text{monta}}(w_1, \ldots, w_k) = A, \text{missä } \begin{cases} A_{i,j} = 0, \text{jos pikselin } (i,j) \text{ keskipiste on} \\ \qquad\qquad \text{jonkin ympyrän } w_i \text{ sisällä} \\ A_{i,j} = 1, \text{muulloin} \end{cases}$$

Suoraan nähdään että kokoelman W vektoreiden $w_1, \ldots, w_k$ järjestyksellä funktion f_{monta} parametreina ei ole väliä,

$$f_{\text{monta}}(w_1, \ldots, w_i, w_j, \ldots, w_k) = f_{\text{monta}}(w_1, \ldots, w_j, w_i, \ldots, w_k), \quad i \neq j. \tag{3.4}$$

Näin ollen merkintä $f_{\text{monta}}(W)$ on järkevä.

Näin määritellyn kiekkomallin avulla voidaan tulkita ympyräkokoelma W vastaavaksi kuvamatriisiksi $A = A_W = f_{\text{monta}}(W)$, ja jatkossa teemme näin suoraan mainitsematta funktiota f_{monta} eksplisiittisesti.

Funktio on helppo toteuttaa ohjelmallisesti ”ja” -operaatiolla $\wedge$ matriiseille:

Määritelmä 3.3. Jos $A = B \wedge C$, missä A, B, C ovat $m \times n$ -matriiseja, niin

$$A_{i,j} = \begin{cases} 1, \text{ jos } B_{i,j} = 1 \text{ ja } C_{i,j} = 1 \\ 0, \text{ jos } B_{i,j} = 0 \text{ tai } C_{i,j} = 0. \end{cases} \tag{3.5}$$

Jos A_i on ympyräkiekkoa w_i vastaava kuvamatriisi eli $A_i = f(w_i), i = 1, \ldots, k$, voimme käsitellä kuvamatriiseja A_i *maskeina* ja kirjoittaa koko kiekkokokoelmaa vastaavan kuvamatriisin A_W niiden avulla

$$A_W = A_1 \wedge \ldots \wedge A_k. \tag{3.6}$$

3.4 Konvoluutio

Digitaalisessa kuvankäsittelyssä kuvan sumentaminen (engl. *blurring*, tai pehmentäminen, engl. *smoothing*) tehdään soveltamalla kuvamatriisiin A sopivaa (lineaarista) suodatinta P (engl. *filter*). Tuloksena saadaan summennettu kuva A_{sumea}, jossa kunkin pikselin $A_{\text{sumea}}(i, j)$ arvo perustuu vastaavan alkuperäisen, terävän kuvamatriisin A pikselin $A(i, j)$:n ja sen ympäristön arvoihin. Tämä on intuitiivinen analogia fysikaaliselle ilmiölle, jossa virheellisesti tarkennetun optisen systeemin linssi- tai peilijärjestelmä kohdistaa havaittavasta kohteesta peräisin olevan valon epätarkasti kameran CCD-kennon tai silmän verkkokalvon kaltaiselle havaintopinnalle ja tuottaa sumean kuvan.

Käytännössä $A_{\text{sumea}}(i, j)$ saadaan ottamalla $A(i, j)$:n ympäristön pikseleistä painotettu keskiarvo. Painotetun keskiarvon painot ja sen, mitä oikeastaan tarkoitetaan 'ympäristöllä' kertoo suodatinmatriisi P. Formaalisti tällainen suodatus vastaa A:n ja P:n (lineaarista) *konvoluutiota* $P * A$, jonka määrittelemme seuraavaksi. Sanomme joskus lyhyesti että kuvamatriisille A tehdään (lineaarinen) konvoluutio-operaatio $P*$. [BB09]

Yleisesti funktioteoriassa konvoluutio on kahden funktion f ja g välinen operaatio, joka määrittelee uuden funktion $f * g$.

Määritelmä 3.4. Funktioiden f ja g *konvoluutio* $f * g$ on

$$f * g = (f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x - t)\,\mathrm{d}t$$

Lause 3.1. *Konvoluutio on vaihdannainen,* $f * g = g * f$*:*

Todistus. Todistus seuraa soveltamalla muuttujanvaihtokaavaa epäoleelliselle inte-

graalille:

$$\begin{aligned} f * g = (f * g)(x) &= \int_{-\infty}^{\infty} f(t)g(x-t)\,\mathrm{d}t \\ &= \lim_{a\to\infty} \int_{-a}^{a} f(t)g(x-t)\,\mathrm{d}t, \end{aligned}$$

tehdään muuttujanvaihto $t = x - s$, $\mathrm{d}t = -\mathrm{d}s$, jolloin

$$\begin{aligned} &= \lim_{a\to\infty} \int_{x+a}^{x-a} -f(x-s)g(s)\,\mathrm{d}s \\ &= \lim_{a\to\infty} \int_{x-a}^{x+a} f(x-s)g(s)\,\mathrm{d}s \\ &= \int_{-\infty}^{\infty} g(s)f(x-s)\,\mathrm{d}s, \end{aligned}$$

missä viimeinen integraali on haluttua muotoa ja määritelmän mukaan $g * f$. □

Mikäli funktioiden f, g määrittelyjoukko on kokonaislukujen joukko, integraali voidaan luonnollisella tavalla korvata summana. Toisin sanoen voidaan määritellä *diskreetti konvoluutio*:

Määritelmä 3.5. Diskreetti konvoluutio on

$$(f * g)(n) = \sum_{m=-\infty}^{\infty} f(m)g(n-m),$$

joka yleistettynä kahteen ulottuvuuteen $\mathbb{Z}^2$ on

$$(f * g)(n_1, n_2) = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} f(m_1, m_2)g(n_1 - m_1, n_2 - m_2).$$

Funktioille määritelty konvoluutio voidaan helposti laajentaa matriiseille samastamalla yleinen reaalinen matriisi $B \in \mathbb{R}^{m\times n}$ sen määräämän funktion $f_B : m \times n \to R^{m\times n}$ kanssa, missä f_B saa arvonsa matriisista B

$$f_B(i,j) = b_{i,j} = B(i,j), \tag{3.7}$$

missä $b_{i,j}$ on B:n alkio i. rivillä ja j. sarakkeessa.

Toisin sanoen merkintämme A:n konvolvoimiselle P:llä

$$A_{\text{sumea}} = P * A \tag{3.8}$$

on järkevä. Määritellään vielä konvoluutiomatriisi P tarkemmin ja tutkitaan kuinka sumennos käytännössä konvoluutiossa tapahtuu.

Määritelmä 3.6. Konvoluution $P*$ ydin (engl. *kernel*) eli konvoluutiomatriisi eli suodatinmatriisi P on äärellinen $K \times L$ -matriisi joillekin luonnolliselle luvulle $K, L \in$

$\mathbb{N}$. Konvoluutio-operaatiossa P:n ajatellaan olevan 0 $K \times L$ -matriisin ulkopuolella.

Oletetaan että P on 3×3 konvoluutiomatriisi, joka on indeksoitu niin että keskimmäinen alkio on $p_{0,0}$, ja A on $m \times n$ -kuvamatriisi. Tällöin diskreetti kaksiulotteinen konvoluutio $P*A$ saadaan määritelmän mukaan seuraavasta yhtälöstä, kun $1 \leqslant k \leqslant m$ ja $1 \leqslant l \leqslant n$:

$$(P * A)(k,l) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} P(i,j)A(k-i, l-j) \tag{3.9}$$

Ottaen huomioon että P:n arvo konvoluutiomatriisin ulkopuolella on 0 eli 3×3 matriisin tapauksessa $P(i,j) = 0$ kun $|i| > 1$ tai $|j| > 1$, voidaan kirjoittaa:

$$= \sum_{i=-1}^{1} \sum_{j=-1}^{1} P(i,j)A(k-i, l-j) \tag{3.10}$$

$$= \sum_{i,j=-1,0,1} \sum P(i,j)A(k-i, l-j) \tag{3.11}$$

Toisin sanoen $P*A$ voidaan tulkita $n \times n$ -kuvamatriisiksi, jonka kunkin (k,l)-pikselin $(P * A)(k,l)$ arvo on painotettu summa $A(k,l)$:sta ja sen naapuripikseleistä, missä painot saadaan konvoluutiomatriisista.

3.4.1 Reunat

Vielä on sovittava kuinka käsitellään A:n reunat, eli tapaukset jolloin kaavassa 3.9 summattava $A(i,j)$ ei ole määritelty koska indeksit (i,j) menevät A:n rajojen ulkopuolelle, $i < 1$ tai $i > m$ tai $j < 1$ tai $j > n$.

Yksinkertaisin vaihtoehto olisi yksinkertaisesti jättää summasta 3.9 pois ne pikselit joissa konvoluutio-operaatio ei näin tulkittuna olisi määritelty; tällöin konvolvoitu matriisi $P * A$ olisi mitoiltaan pienempi kuin alkuperäinen A, mikä ei ole toivottavaa.

Toinen usein käytetty vaihtoehto on 'laajentaa matriisia nollalla', ja tulkita matriisin A:n pikseleiden $A(i,j)$ arvot nollaksi kun i,j ovat matriisin ulkopuolella. Koska valitussa kuvamallissa kuvamatriisin A:n arvot 'neutraalia' tyhjää taustaa esittävillä alueilla ovat 1 ja kappaleen kohdalla 0 (kuvamme esittää tummia kappaleita valkealla taustalla), käytetään 'nollana' lukua 1. [2] Ongelmaksi jää tällöin tilanne, jossa ympyräkiekko ei ole kokonaan kuvan sisällä: tällaiset kiekot saisivat kuvan laidassa valkean reunan.

Kolmas yleinen ratkaisu on laajentaa kuvamatriisia konvoluutiota laskettaessa toistamalla 'nollan' sijasta A:n olemassaolevia reunapikselejä. Koska puuttuvien konvoluution laskemiseen tarvittavat alueet ekstrapoloidaan, tämäkin aiheuttaa

[2] Tai vaihtoehtoisesti tuottaa kuvadatasta käänteiskuva ja operoida sillä: matriisin alkio on 1 = kuuluu ympyräkiekkoon, 0 = ei kuulu kiekkoon.

pienen vääristymän. [3] Tutkielmassa käytetyillä pienillä konvoluutiomatriiseilla vääristymä on kuitenkin hyvin pieni. [BB09]

Joissain tapauksissa on hyödyllistä tulkita kuva *toruspintana*, jolloin kunkin reunan yli menevät pikselit tulkitaan otettavaksi vastakkaisesta reunasta. Näin määriteltyä konvoluutiosuodatinta sanotaan myös *sykliseksi* konvoluutioksi. Käytännössä ero siihen että kuvamatriisia jatkettaisiin on myös pienillä konvoluutiomatriiseilla pieni. [Jäh02, s. 104–106]

Vast'edes tutkielmassa sovitaan, että konvoluutio-operaatiossa reunat käsitellään kolmannella esitetyllä tavalla eli jatkamalla olemassaolevia reunapikseleitä.

3.5 Kohina

Tavoitteena on mallintaa prosessia, jonka tuottamat kuvat sumentuneen lisäksi kohinaisia, mihin käytämme additiivista kohinamallia.

Yleisesti kohinalla tarkoitetaan havaitussa kuvassa A_{havaittu} ei-toivottua satunnaista komponenttia. Additiivisessa kohinamallissa tätä mallinnetaan lisäämällä kohinattomaan kuvaan satunnaismatriisi E, jossa jokainen pikseli alkio (pikseli) on jokin satunnaismuuttuja. Jos sumennettu (konvolvoitu) kuva on $P * A$, lopullinen sumentunut ja kohinainen kuva on

$$P * A + E.$$

Tällaisessa mallissa kohinamatriisin E jakauma tyypillisesti riippuu mallinnettavasta häiriölähteestä. Monia yleisiä kohinatyyppejä mallinnetaan normaalijakautuneella kohinamallilla (*Gaussian noise*), kuten esimerkiksi laitteiston elektronisten komponenttien lämpökohinaa ([Bon05]) tai approksimaationa Poisson-jakautuneesta otoskohinasta [Ks. esim. Jäh02, luku 3.4.5]. Toinen yleinen kohinatyyppi on nk. suola ja pippuri -kohina (*salt and pepper noise*), jota syntyy esimerkiksi digitoinnissa tai digitaalisessa tiedonsiirrossa: kuvan pikseleistä vain harvaan on kohdistunut virhe, mutta häiriöt ovat hyvin suuria suuntaan tai toiseen, jolloin kuva näyttää siltä ikään kuin sille olisi ripoteltu suolaa ja pippuria [Bon05].

Yhteistä edellämainituille virhelähteille on, että kohina syntyy vasta kuvan digitaalisessa talteenottolaitteistossa: kuva on jo sumea havaintopinnan tulkitessa sen digitaaliseksi signaaliksi. Tässä mielessä additiivinen malli on mielekäs.

[3] Parhaassa tapauksessa konvolvoitava raakakuva olisi suurempi kuin tarpeen, jolloin siitä voitaisiin leikata reunat pois ja syntyvä pienempi kuva olisi toivottun kokoinen.

3.6 Havaintomalli ja simuloidun aineiston malli

Edellä määriteltiin yleinen kuvamalli

$$A_{\text{havaittu}} = P * A_W + E \tag{3.2}$$

sumentuneita ja kohinaisia kiekkokuvia tuottavasta prosessista.

Tutkielmassa esitellyn algoritmin toimivuutta esitettyyn ongelmaan testataan kokeilemalla sitä simuloituun aineistoon. Oikeasssa tutkimustilanteessa havaintomalli olisi muodostettu approksimoimaan jotain todellista prosessia, ja fysikaalisesta prosessista muodostettu malli on usein ainakin hieman epätarkka, sillä tutkittava prosessi harvoin tunnetaan aivan täydellisesti.

Näin ollen olisi epärealistista sekä tuottaa simuloitu aineisto että sitten analysoida sitä täsmälleen samanlaiseen malliin perustuen. Oletetaan kuitenkin että havaintomallimme on approksimoi todellisuutta varsin hyvin, jolloin simuloituun aineistoon A_{data} voidaan käyttää periaatteeltaan samaa mallia

$$A_{\text{data}} = P_{\text{data}} * A_W + E, \tag{3.12}$$

missä datan luomiseen käytetty sumentava suodatin on kuitenkin hieman erilainen kuin havaintomallissa odotetaan.

Kuvittelemme siis että tutkimme yleisen mallin 3.2 mukaista prosessia, jonka ”todelliset parametrit” (konvoluutiomatriisi) ovat mallin 3.12 mukaiset. Ympyröiden tunnistamiseen (seuraavasssa luvussa) kuitenkin käytetään nk. ”havaintomallia”, joka on hieman yksinkertaisempi kuin aineiston luomiseen käytetty ”todellinen” malli.

Käytännössä ero mallien välillä on pieni, mutta tarkoituksena on esitellä parhaita periaatteita kuinka tällaisissa tutkimusongelmissa ”realistinen” simuloitu aineisto kannattaa tuottaa. Erityisesti monissa inversio-ongelmien laskennallisissa ratkaisuissa aineiston tuottaminen ja inversioratkaisun laskeminen samalla laskentamenetelmällä voi johtaa epärealistisen hyviin tuloksiin [MS12].

3.6.1 Havaintomallin konvoluutiomatriisi

Valitaan havaintomallissa käytettäväksi konvoluutiomatriisiksi P

$$P = \frac{1}{26} \cdot \begin{pmatrix} 2 & 2 & 2 \\ 2 & 10 & 2 \\ 2 & 2 & 2 \end{pmatrix}. \tag{3.13}$$

Yllä matriisi kerrotaan skalaarilla $\frac{1}{26}$, jolla *normitetaan* matriisi siten että konvoluutio vastaisi *painotetun keskiarvon* ottamista. Skalaari 1/26 saadaan yhtälöstä

$$\frac{1}{26}(8 \cdot 2 + 10) = 1.$$

Valittu P muistuttaa hieman kuvan- ja signaalinkäsittelyssä käytettyä Gauss-sumennosta (engl. *Gaussian low pass filter, Gaussian blur*), jossa konvoluutiomatriisin arvot saadaan kaksiulotteisesta Gaussin normaalijakaumasta.

3.6.2 Havaintomallin kohina

Havaintomallissamme oletetaan yksinkertaisesti että kohina E on tasaisesti jakautunutta valkoista kohinaa. Käytettävä simuloitu data (joka määritellään tarkemmin alaluvussa 3.6) ei pyri mallintamaan mitään tiettyä erityistä kamerasysteemiä, joten voimme valita yleisen esimerkin kohinalähteestä joka huonontaa kuvan laatua varsin paljon. Oikeassa sovelluksessa kohinamallia todennäköisesti olisi tarpeen tarkentaa käytettävän mittaustilanteen ja -välineiden oikeasti tuottaman kohinan perusteella.

Kohinamatriisi E ajatellaan satunnaismatriisiksi, jonka alkiot $E(i,j) = E_{i,j}$ ovat riippumattomia tasaisesti jakautuneita satunnaismuuttujia keskiarvolla 0,

$$E(i,j) \sim U(-0.5, 0.5) \tag{3.14}$$

Tällaista kohinaa, jossa $E_{i,j}$ ovat samoin jakautuneita (riippumattomia) satunnaismuuttujia ja niiden keskiarvo on 0, kutsutaan valkoiseksi kohinaksi.[4]

Visuaalisesti summamatriisin $P * A + E$ alkiot joiden arvot ovat välin $[0,1]$ ulkopuolella tulkitaan luvuksi 0 tai 1 riippuen olisiko alkio välin $[0,1]$ ala- vai yläpuolella.

3.6.3 Simuloidun aineiston sumennos ja kohina

Sumentavaksi suodattimeksi P_{data} valitaan hieman monimutkaisempi konvoluutiomatriisi, jota havaintomallin konvoluutiomatriisi yrittää ”epätarkasti” mallintaa,

$$P_{\text{data}} = \frac{1}{36}\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 8 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{3.15}$$

[4]Valkoisen kohinan tarkka määritelmä joka perustuu E:n tehospektrin tarkasteluun ei kuulu tämän tutkielman piiriin. Tehospektri voidaan määritellä esimerkiksi E:n autokorrelaatiofunktion Fourier-muunnoksen kautta, [ks. Jäh02, s.96 – 97].

Simuloituun aineistoon lisättävä kohina on havaintomallin mukaista tasaista satunnaiskohinaa

$$E(i,j) \sim U(-0.5, 0.5) \tag{3.16}$$

jota on helppo generoida laskennellisilla ohjelmistoilla. Havaintomallin mukaisesti lopullisessa kuvassa harmaasävykuvan pikseleiden arvorajojen $[0, 1]$ ylle tai alle arvoja saavat matriisin alkiot tulkitaan pikseleiksi

$$(P * A + E)_{\text{kuva}}(i,j) = \begin{cases} 0, & \text{jos } (P * A + E)(i,j) < 0 \\ 1, & \text{jos } (P * A + E)(i,j) > 1. \end{cases} \tag{3.17}$$

Luku 4

Jäähdytysalgoritmin soveltaminen kiekko-ongelmaan

4.1 Kiekko-ongelman muotoilu optimointiongelmana

Oletetaan että kuva A_{data} esittää edellisessä luvussa esitellyn mallin mukaisesti yhtä ympyräkiekkoa $w_0 = (x_0, y_0, r_0)$, jonka parametrit ajatellaan tuntemattomiksi vakioiksi jotka halutaan selvittää. Tämä yhden kiekon kiekko-ongelma voidaan ajatella määritelmän 2.1 mukaisena globaalina diskreettinä optimointiongelmana, jossa pyritään löytämään muuttujille x, y, r arvot jotka *minimoivat* niitä vastaavan havaintomallin kuvan A_{havaittu} eron A_{data}:aan. Monen kiekon tapauksessa vastaavasti pyritään minimoimaan jokaista kuvan A_{data} esittämän kiekkokokoelman $W = (w_0, \dots, w_k)$ kiekkoa vastaavat parametrit.

Jos kuvien resoluution ajatellaan olevan tarpeeksi suuri, kiekko-ongelmaa olisi myös perusteltua mallintaa *jatkuvan* optimoinnin tehtävänä eikä kombinatorisena optimointiongelmana, johon tässä tutkielmassa kuvattu ”klassinen” SA soveltuu. Esimerkiksi Press et al. [Pre+07] ehdottavat jatkuvassa tapauksessa hieman hienostuneempaa menetelmää uuden tilan s_j valitsemiseen. Käytännössä diskreetti algoritmi kuitenkin toimii kiekko-ongelmaan, ja joka tapauksessa teoreettisesti jatkuva parametriavaruus on diskreetti laskentatarkkuuden puitteissa.

Muotoillaan k kiekon ongelma äärelliselle $k \in \mathbb{N}$ (josta yhden kiekon ongelma saadaan erikoistapauksena $k = 1$) optimointiongelmana, johon sovelletaan jäähdytysalgoritmia. Määritellään algoritmin tilat ja valitaan sopiva energiafunktio ja jäähdytysstrategia:

Määritelmä 4.1. Yhden kiekon tapauksessa sanomme että kiekon w_i määrittävät parametrit x_i, y_i, r_i muodostavat yhdessä jäähdytysalgoritmin tilan (tai ratkaisueh-

dokkaan tai konfiguraation) s_i,

$$s_i = (x_i, y_i, r_i) \in m \times n \times \mathbb{R} \tag{4.1}$$

Vastaavasti monen kiekon ongelmassa kun kiekkojen määrä k on tunnettu, systeemin tila s_i on kokoelma kiekkoja W eli ne määritteleviä parametrivektoreita,

$$s_i = \{s_{i,1}, \dots, s_{i,k}\} \tag{4.2}$$

$$= \{(x_{i,1}, y_{i,1}, r_{i,1}), \dots, (x_{i,k}, y_{i,k}, r_{i,k})\}, \tag{4.3}$$

jotka voidaan tarpeen vaatiessa esittää esimerkiksi matriisina

$$s_i = \begin{bmatrix} (x_{i,1}, y_{i,1}, r_{i,1}) & \dots & (x_{i,k}, y_{i,k}, r_{i,k}) \end{bmatrix}. \tag{4.4}$$

4.2 Energiafunktio

Määritellään seuraavaksi edellisessä luvussa 3. ”Kiekko-ongelman kuvamalli” esitellyn havaintomallin (3.2)

$$A_{\text{havaittu}} = P * A_W + E \tag{3.2}$$

pohjalta sopiva optimoitava energiafunktio.

4.2.1 Naiivi energiafunktio

Kuten aiemminkin, merkitään kuvadatamatriisia A_{data}, ja parametrien x, y, r määräämää ympyrää w esittävä matriisia $A_{x,y,r} = A_w$. Mallin (3.2) ytimellä konvolvoitu kuvamatriisi on $P * A_w$, jonka avulla voidaan määritellä eräs yksinkertainen energiafunktio:

Energiafunktio E_{naiivi} on $P * A_w$:n ’pikselikohtainen’ ero matriisiin A_{data}, eli erotuksen $A_{\text{data}} - P * A_w$ alkioittainen euklidisen normin neliö

$$E_{\text{naiivi}}(s) = E_{\text{naiivi}}(x, y, r) = \sum_i \sum_j \left(A_{\text{data}}(i, j) - P * A_w(i, j) \right)^2. \tag{4.5}$$

Luvussa 3 määriteltiin myös malli usean k kiekon kokoelmaa $W = \{w_1, \dots, w_k\}$ vastaavalle kuvalle A_W, ja E_{naiivi} yleistyy suoraan kokoelmalle W kun tila s yleistetään monelle kiekolla kuten yllä yhtälössä (4.2):

$$E_{\text{naiivi}}(s) = \sum_i \sum_j \left(A_{\text{data}}(i, j) - P * A_W(i, j) \right)^2. \tag{4.6}$$

Suoraan nähdään, että energiafunktion kannalta parempi on sellainen tila s, jonka parametrit ovat lähellä käyttämämme mallin mukaisia "todellisia" kiekkoparametreja $W = s_{\text{opt}}$, ja vastaavasti tilat s', joiden kiekot kaukana A_{data} todellisista kiekoista ovat energiafunktion E_{naiivi} huonompia.

Ilman datan kohinaa ja mallin epätarkkuutta ($P \neq P_{\text{data}}$) edellä määritelty energiafunktio saavuttaisi globaalin minimin $E_{\text{naiivi}}(s_{\text{opt}}) = 0$. Virheilähteiden vuoksi tämä ei ole luultavasti täysin mahdollista, mutta voidaan olettaa että suurella todennäköisyydellä energiafunktio on tässä suhteessa edelleen hyvä kohinasta ja epätarkasta sumennosmallista huolimatta.

4.2.2 Energiamaasto

Tarkastellaan seuraavaksi energiafunktion E_{naiivi} 'energiamaastoa' eli sen parametriavaruudessaan saamia arvoja (*energy landscape* [SSF02]):

Ensinnäkin E_{naiivi} ei erota toisistaan kahta eri tilavektoria s ja s', mikäli niitä vastaavat ympyrät eivät osu päällekkäin minkään A_{data}:n ympyräobjektin kanssa. Toisin sanoen, vaikka toinen tilavektori s olisi paljon lähempänä optimia kuin s'. (Ks. kuva n.) Käytännössä algoritmi ei pysty parempaan kuin satunnaisiin arvauksiin kunnes tilakandidaattien s_i satunnaiskulku osuu sellaiseen kandidaattiin, jota vastaava ympyrä on osittain A_{data}:n tavoiteympyrän päällä; vasta tämän jälkeen energiafunktio pystyy mittaamaan uuden tilaehdokkaan s_j paremmuutta suhteessa edelliseen s_i.

Hyvin pienillä ympyrän säteillä tehtävä muistuttaa yhtä enemmän sellaista optimointiongelmaa, jota Salamon, Sibani ja Frost [SSF02] kutsuvat 'golf-reikäongelmaksi': tasaisessa energiamaastossa on pieni alue, joka ei näy ympäröivässä energiamaastossa (ikään kuin golf-kentän reikä), jossa saavutetaan globaali minimi. (Kuva 4.1.) Tälläisen yksittäisten, algoritmin näkökulmasta satunnaisten minimipisteiden löytämiseen on vaikea löytää satunnaishakua parempaa strategiaa. SA-algoritmi pyrkii vastaamaan tähän korkean lämpötilan ratkaisujen satunnaisuudella: sopivalla jäähdytysskenaariolla ratkaisu löydetään vielä lämpötilan t ollessa korkea ja suuri osa parametriavaruudesta on satunnaiskulun tavoitettavissa, mikä on kuitenkin kiekkojen säteiden kutistuessa pistemäisiksi yhä vaikeampaa.

Toiseksi kuvan reunat muodostavat E_{naiivi}:n suhteen *lokaalin* minimin (ks. kuva 4.1c), samoin tilavektorit s joissa kiekkokandidaatit asettuvat osittain päällekkäin pyrkien kattamaan esimerkiksi vain yhden suurempia A_{data}:n kiekon mahdollisimmaan tarkkaan (kuva 4.2b).

4.2.3 Paranneltu monen kiekon energiafunktio

Kiekkojen lukumäärän ja ratkaisuavaruuden Ω ulottuvuuksien määrän kasvaessa päällekkäisten kandidaattien lokaalien minimien määrä moninkertaistuu nopeasti ja vaikeuttaa optimaalisen ratkaisun s_{opt} löytämistä. (Kuva 4.2b.) Tämän vuoksi määritellään myös paranneltu energiafunktio E_{par}, joka rankaisee päällekkäisistä kandidaattikiekoista. (Tutkituissa aineistoissa A_{data}:ssa ei ole päällekkäisiä kiekkoja.)

Määritellään ensin kuvamatriisin A intensiteettikäänteiskuva

$$A' = 1 - A = \begin{cases} A(i,j)' & = 0, \text{ jos } A(i,j) = 1 \\ A(i,j)' & = 1, \text{ jos } A(i,j) = 0 \end{cases}$$

ja reaalilukuarvoinen apumatriisi $\Delta A \in \mathbb{R}^{m \times n}$,

$$\Delta A = 1 - \left(A'_{w_1} + \ldots + A'_{w_k} \right), \tag{4.7}$$

missä 1 on A:n kokoinen ykkösmatriisi ja yhteenlasku- ja erotus tavallisia matriisioperaatiota. Matriisin A avulla voidaan nyt määritellä paranneltu energiafunktio

$$E_{\text{par}}(s) = E_{\text{par}}(w_1, \ldots, w_k) = \sum_i \sum_j \left(A_{\text{data}}(i,j) - P * \Delta A(i,j) \right)^2. \tag{4.8}$$

Nähdään, että jos kiekot $w_1, \ldots, w_k$ ovat erillisiä, pätee $1 - \left(A'_{w_1} + \ldots + A'_{w_k} \right) = A_{w_1,\ldots,w_k} = A_W$, ja E_{par} saisi kyseisillä parametrin arvoilla saman arvon kuin E_{naiivi}. Sellaisten pikseleiden (i,j) joiden kohdalla useampi kiekko osuu päällekkäin, apumatriisin pikselit $\Delta A(i,j)$ ovat negatiivisia, toisin kuin vastaavassa kuvamatriisissa A_W, jonka elementeille pätee

$$A_W(i,j) \in [0,1].$$

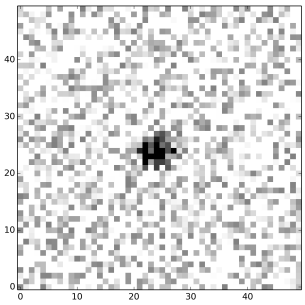
Näin ollen yhtälön (4.8) energiafunktio E_{sov} saa päällekkäisten kiekkojen kohdalla suuremman arvon kuin E_{naiivi}, sillä

$$A_{\text{data}}(i,j) - P * \Delta A(i,j) > A_{\text{data}}(i,j) - P * A_W(i,j).$$

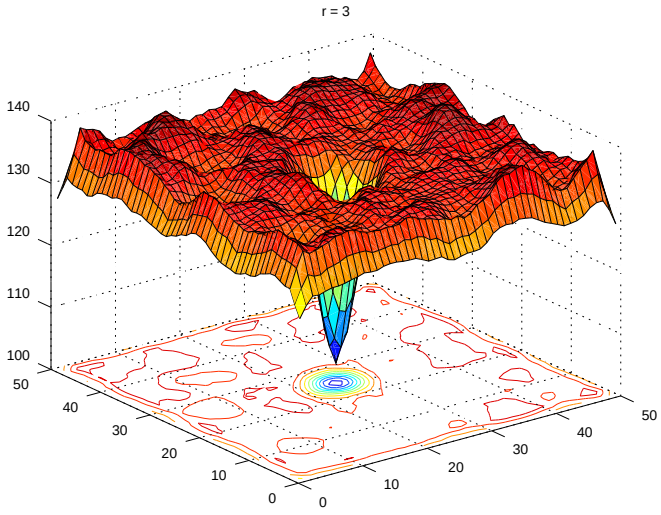
Käytännön esimerkki kuvassa 4.2.

4.3 Siirtymät

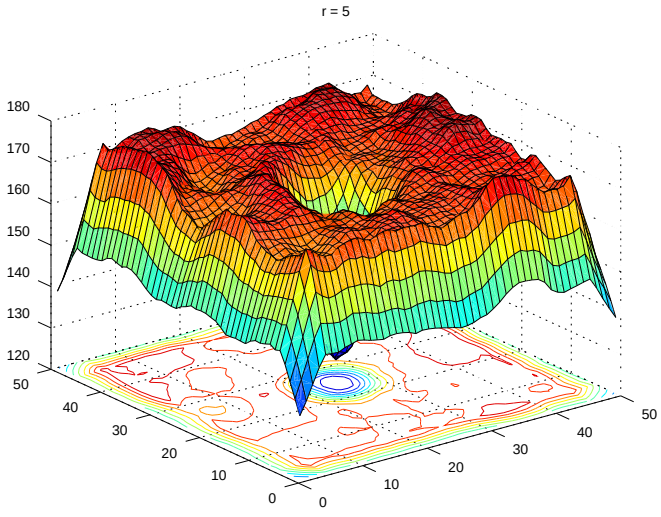
Yksinkertaisin tapa toteuttaa jäähdytysalgoritmin 'pieni satunnainen siirtymä' olisi satunnaisesti valita jokin muuttujista x, y, r ja lisätä tai vähentää siitä jokin kiinteä, pieni luku $\delta > 0$. Käytännössä näin saataisiin jatkuvan parametriavaruuden Ω diskretisointi. Tällöin kuitenkin joko menetettäisiin ratkaisutarkkuutta (jos δ on



(a) A_{data}, $r = 3$.

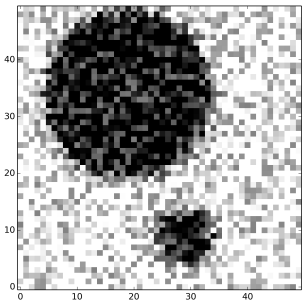


(b) Kiinteä $r = 3$.

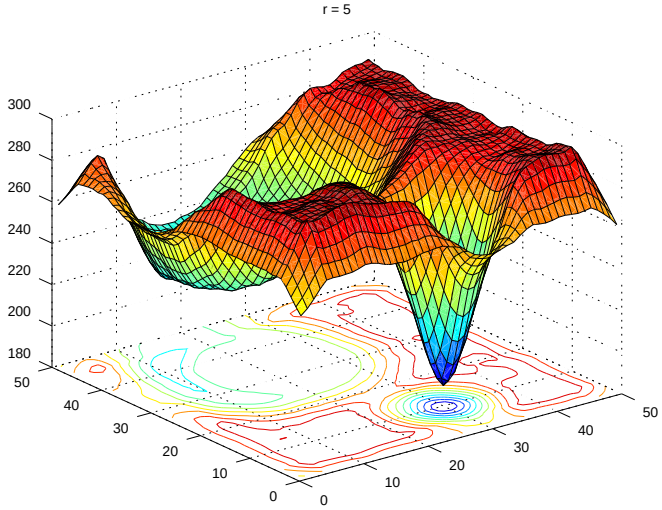


(c) Kiinteä $r = 5$.

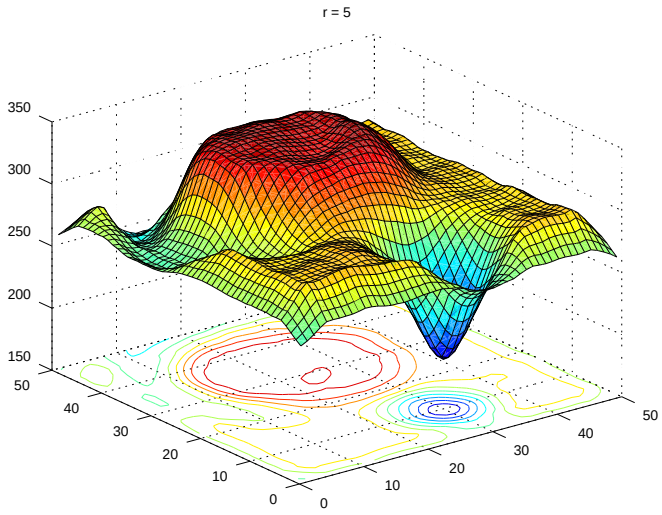
Kuva 4.1: Energiafunktion E_{naiivi} energiamaasto xy-avaruudessa. Energiamaasto on xy-ulottuvuudessa golf-reikämäinen (kuva 4.1b), mutta r-ulottuvuudessa globaalin minimin 'kuoppa' näkyy alati laajemmalla alueella kun r kasvaa. Huomaa myös reunojen lokaalit minimit.



(a) A_{data}. $(x_i, y_i, r_i) = (35, 20, 15), (10, 30, 5)$.



(b) E_{naiivi}.



(c) E_{sov}

Kuva 4.2: Energiafunktioiden E_{naiivi} ja E_{sov} arvoja toisen kiekon x_2y_2-avaruudessa kun toisen kiekon säde $r_2 = 5$ ja ensimmäisen kiekon parametrit pidetään vakioina $(x_1, y_1, r_1) = (34, 22, 14)$. E_{sov} rankaisee tiloista, joissa toinen parametrikiekko on ensimmäisen päällä.

suuri) tai pienellä vakion arvolla $\delta \approx$ pikseli siirtymät olisivat liian pieniä jotta algoritmi kävisi ratkaisuavaruutta läpi tehokkaasti.

Tämän vuoksi käytetään seuraavaa menetelmää siirtymien generointiin: Valitaan satunnainen kiekko, jonka parametrit ovat (x, y, r). Uusi piste (x', y', r') valitaan (x, y, r) -keskisen 'ellipsoidin' sisältä parametriavaruudessa Ω:

$$r' = \delta \cdot r, \qquad \delta \sim U(-1, 1), \tag{4.9}$$

$$(x', y') = (x, y) + \delta \cdot (\cos(\phi), \sin(\phi)), \qquad \phi \in U(0, 2\pi). \tag{4.10}$$

Lisäksi rajoitetaan x, y-siirtymiä siten ettei kiekon keskipiste voi mennä kuva-alueen A ulkopuolelle. Vastaavasti estetään arvot $r < 1$ pikseli ($r = 0$ muodostaisi myös lokaalin minimin) ja $r >$ kuvan koko.

4.4 Jäähdytysstrategia

SA-algoritmin jäähdytysstrategian valitsemiseen yleisesti kirjallisuus ([LA87], [SSF02], [Pre+07]) tarjoaa lukuisia hyvin erilaisia vaihtoehtoja. Tässä tutkielmassa käytetään yksinkertaista eksponentiaalista jäähdytysstrategiaa.

4.4.1 Lämpötilan laskeminen

Lämpötilaa homogeenilla ketjulla k päivitetään yhtälön

$$t_{k+1} = \alpha \cdot t_k, \qquad k = 0, 1, 2, ... \tag{4.11}$$

mukaisesti, missä α on vakio $1 - \epsilon$ jollekin pienelle $\epsilon > 0$. (Käytännössä 'pienellä' tarkoitamme että kokeilemme luvussa 5 arvoja $\alpha = 0.90 ... 0.99$.)

Markov-ketjun pituus kullakin kontrolliparametrin arvolla t_k asetetaan vakioksi $L_k = 10$.

4.4.2 Lämpötilan alkuarvo

Kontrolliparametrin alkuarvo t_0 eli lähtölämpötila pyritään valitsemaan siten, että lämpötilan ensimmäisillä arvoilla hyväksyttyjen tilamuutosten suhde kaikkiin yritettyihin tilamuutoksiin q on lähellä yhtä, eli

$$q_0 = \exp -\overline{\Delta E}/t_0 \approx 1, \tag{4.12}$$

missä $\overline{\Delta E}$ on 'keskimääräistä' satunnaista siirtymää vastaava energiafunktion muutos lämpötilassa t_0. Toisaalta ei ole toivottavaa, että t säilyy korkeana (ja $q \approx 1$) *liian*

pitkään, jolloin siitä ei enää hyötyä algoritmin toimivuuden kannalta ja simulointi korkeissa lämpötiloissa vie turhaan laskenta-aikaa.

Näin ollen kunkin algoritmin ajokerran alussa valitaan sopiva arvio saadaan yhtälön 4.12 avulla:

$$\exp\left(-\overline{\Delta E}/t_0\right) = q_0 \quad (\leqslant 1) \tag{4.13}$$

$$t_0 = -\frac{\overline{\Delta E}}{\log q_0}, \quad q_0 \leqslant 1, \tag{4.14}$$

missä keskimääräistä energiafunktion muutosta $\overline{\Delta E}$ kullakin A_{data} arvioidaan simuloimalla muutamia jaksossa 4.3 kuvailtuja siirtymiä.

4.4.3 Loppukriteeri

Käytännön varotoimenpiteenä asetetaan maksimiraja kuinka monta iteraatiota algoritmia suoritetaan ($n = 1000$ homogeenia 10 iteraation Markov-ketjua).

Varsinaiseksi lopetuskriteeriksi asetamme kuitenkin sen lämpötilan, jossa Markov-ketjut alkavat toistua samankaltaisina, eli uudet iteraatiot pienemmillä lämpötiloilla eivät enää vaikuta johtavan merkittävästi parempiin ratkaisuihin. Käytännössä kelvolliseksi osoittautui katkaisuehto, jossa algoritmin iteraatio i jää viimeiseksi $i := i_{\text{final}}$, jos

$$\frac{\sum_{j=0,\ldots,3} q_{i-j}}{4} < 0.05. \tag{4.15}$$

Luku 5

Tulokset

5.1 Testiaineisto

Algoritmin tutkimista varten luotiin simuloitu testiaineisto jaksossa 3.6 kuvaillun mallin mukaisesti. Aineistoa varten generoitiin suuri joukko erilaisia parametreiltaan mielivaltaisten $k = 1, ..., 5$ kiekon harmaasävykuvia, joiden koko oli 50×50 pikseliä, ja algoritmin toiminnan tarkastelua varten poimittiin esimerkinluontoisesti kuusi kuvaa (a, b, c, d, e, f), jotka näkyvät kuvassa 5.1. Testikuvien a, b, ja c avulla esitellään algoritmin toimintaa kiekkomäärän kasvaessa. Kuvan d avulla taas vaikuttavatko eri jäähdytysskenaariot ratkaisun tarkkuuteen löytää pieni kiekko ($x = 35, y = 35, r = 2$), ja testikuvia e, f vertailemalla tarkastellaan miten yhden kiekon säteen muutos algoritmin toimintaan.

5.2 Ratkaisun onnistumisen mittaaminen

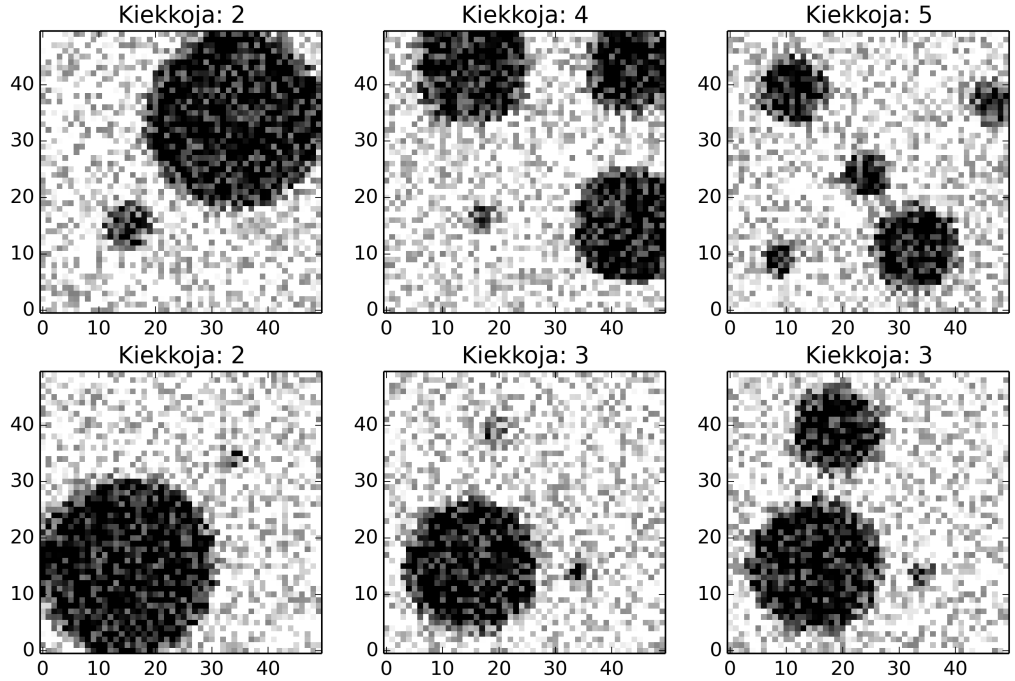
Kuvaillaan vielä mitat, joita käytetään ratkaisun onnistumisen arvioimiseen.

5.2.1 Energiafunktio

Energiafunktion arvon tarkastelu kullakin iteraatiolla kunkin kulkijan edetessä on luonnollinen menetelmä arvioida optimointialgoritmin toimintaa. Varjopuolena kuva energiafunktion tarkastelu ei välttämättä kerro paljonkaan kuinka hyvä ratkaisu on tarkasteltavan kuvakäsittelyongelman suhteen.

5.2.2 Pinta-alan symmetrinen erotus

Toinen vaihtoeheto on verrata todellisen ja ratkaisun kiekkojen symmetristä erotusta. Olkoon W_{final} kulkijan lopullinen ratkaisu ympyräparametreille ja W_{orig} alkuperäiset datan luomiseen käytetyt kiekkojen parametrit, ja A_W luvun 3 mallin mukaisesti



Kuva 5.1: Tutkielmassa lähemmin tarkasteltavat testikuvat. Vasemmalta oikealle, ylärivi: Testikuvat a, b, c. Alarivi: d, e, f.

kiekkoparametreja vastaava häiriötön kuva jossa $A_{ij} = 0$, jos pikseli on kuvan sisällä. Olkoon myös vastaava kuvaa A vastaava käänteiskuva $A' = 1 - A$ kuten kappaleessa 4.2.3. Tälloin symmetriseen erotukseen perustuva mitta on

$$m_{\text{symm}}(W_{\text{final}}, W_{\text{orig}}) = (A'_{W_{\text{final}}} \setminus A'_{W_{\text{orig}}}) \cup (A'_{W_{\text{orig}}} \setminus A'_{W_{\text{final}}}),$$

kun kuvamatriisit mielletään joukoiksi, johon joku tietty pikseli kuuluu ($A'_{ij} = 1$) tai ei ($A'_{ij} = 0$). Mitta vastaa naiivia energiafunktiota ja melko samankaltainen parannellun energiafunktion kanssa.

5.2.3 Sovellettu etäisyysmitta

Määritellään lisäksi toisenlainen ratkaisukiekkojen etäisyyttä todellisista mittaava mitta joka ei perustu pinta-aloihin, mutta kuitenkin vastaa intuitiivisista käsitystä kiekkojen etäisyydestä: Valitaan löydetyn ratkaisun ja todellisten kiekkojen väliseksi etäisyydeksi summa kunkin ratkaisukiekon keskipisteen ja säteen Euklidinen etäisyys alkuperäisten kiekkojen keskipisteeseen ja säteeseen kun ratkaisun kiekkojen todelliset vastineet valitaan kaikkien mahdollisten kiekkoparien etäisyyksien pienuusjärjestyksessä.

Toisin sanoen käytetään seuraavanlaista algoritmia. Olkoon $W_{\text{final}}, W_{\text{orig}}$ kiekko-

joukot kuten yllä, ja lasketaan etäisyys m_{sov} seuraavasti:

Data: $W_{\text{final}}, W_{\text{orig}}$
Result: m_{sov}

$n_{\text{circles}} \leftarrow |W|$;
$m_s \leftarrow 0$;
for $i \leftarrow 1$ **to** $n_{circles}$ **do**
  $d_{\min} \leftarrow \min_{a \in W_{\text{final}}, b \in W_{\text{orig}}} ||a.x - b.y|| + ||a.y - b.y|| + ||a.r - b.r||$;
  $a_{\min}, b_{\min} \leftarrow \operatorname{argmin}_{a \in W_{\text{final}}, b \in W_{\text{orig}}} ||a.x - b.y|| + ||a.y - b.y|| + ||a.r - b.r||$;
  $m_{\text{sov}} \leftarrow m_s + d_{\min}$;
  $W_{\text{final}} \leftarrow W_{\text{final}} \setminus a_{\min}$;
  $W_{\text{orig}} \leftarrow W_{\text{orig}} \setminus b_{\min}$;
end

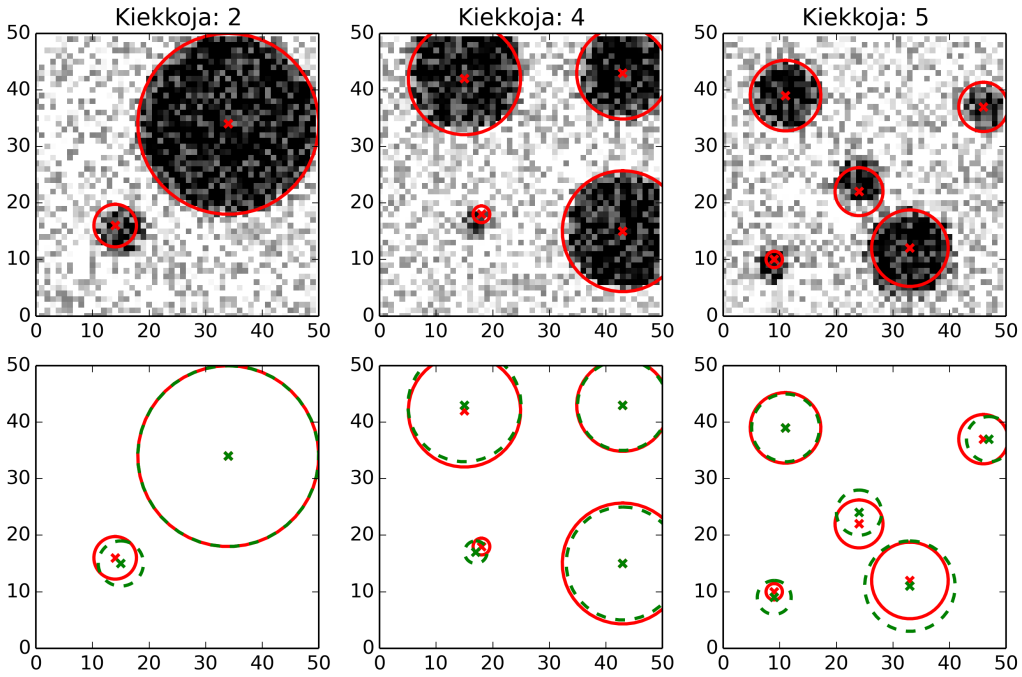
5.3 Algoritmin tulokset

Testiaineiston kuville ajettiin 100 kertaa luvussa 4 kuvailtu algoritmi vaihtelevalla määrällä jäähdytysskenaarioita $\alpha = 0.90 ... 0.99$. Jokaista algoritmin ajokertaa tietyin parametrein eli toteutunutta Markov-ketjua nimitetään jatkossa *kulkijaksi*.

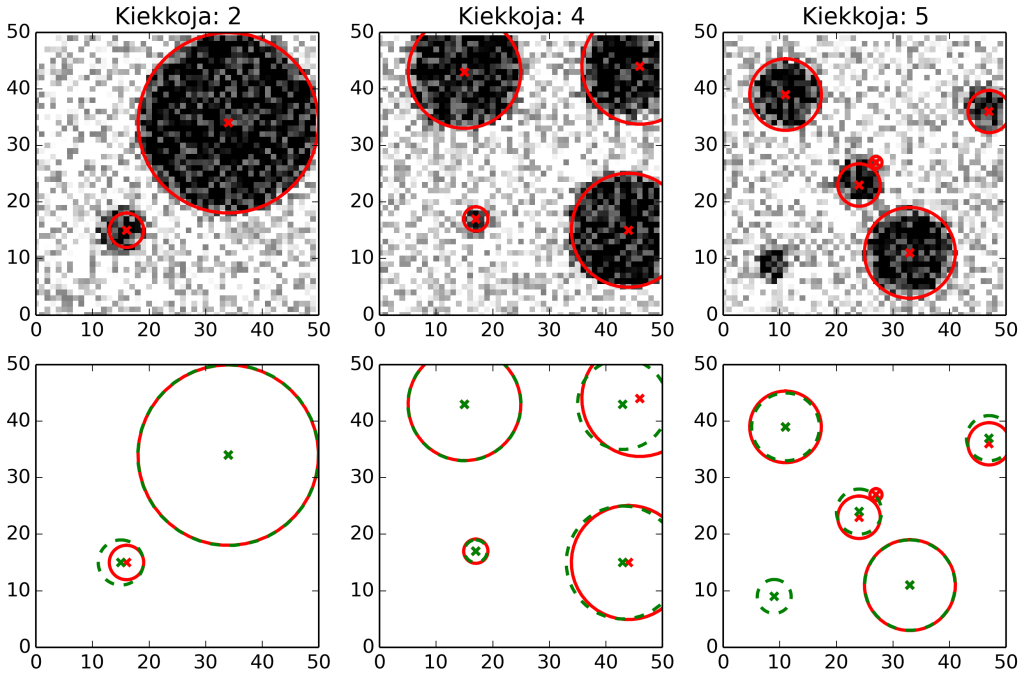
Koska kulkijat ovat itsenäisiä, kunkin 100 kulkijan kokoelman simuloinnissa voitiin hyödyntää rinnakkaislaskentaa. Numeeriseen laskentaan käytettiin Helsingin yliopiston Tietojenkäsittelytieteen laitoksen Ukko-laskentaklusteria.

Taulukko 5.1: Testiaineiston kuvat a, b, c. Kuvien 5.2 ja 5.3 energiafunktion suhteen parhaiden ratkaisujen lopulliset energiat, virheet ja kulkijoiden pituus (homogeenien Markov-ketjujen määrä). Jäähdytysskenaariot $\alpha = 0.90$ ja $\alpha = 0.99$.

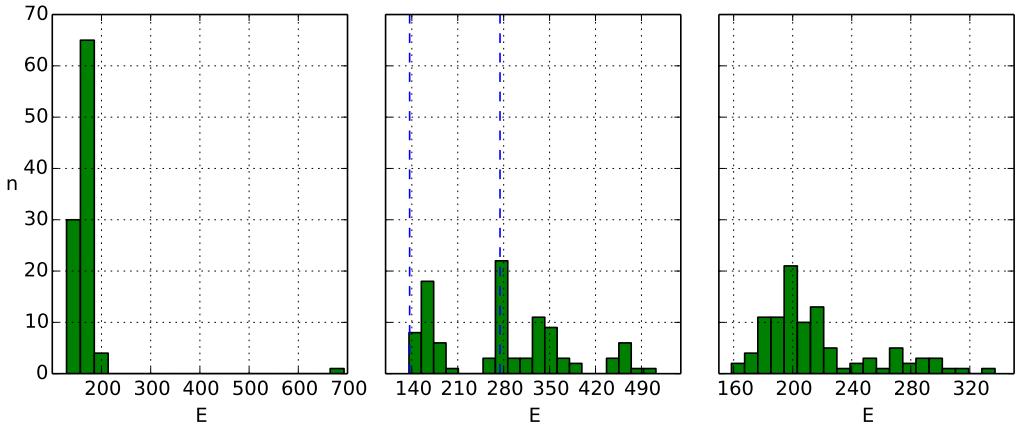
α	Aineisto	Markov-ketjuja	Energia	m_{sov}	m_{symm}
0.90	a	56	129.982734083	1.43765180533	22
	b	52	136.670018567	3.36654077973	66
	c	60	159.514844916	6.75832581266	131
0.99	a	679	129.277573595	1.41934200707	22
	b	411	135.156720071	5.0613981837	49
	c	438	149.503374454	27.974956417	73



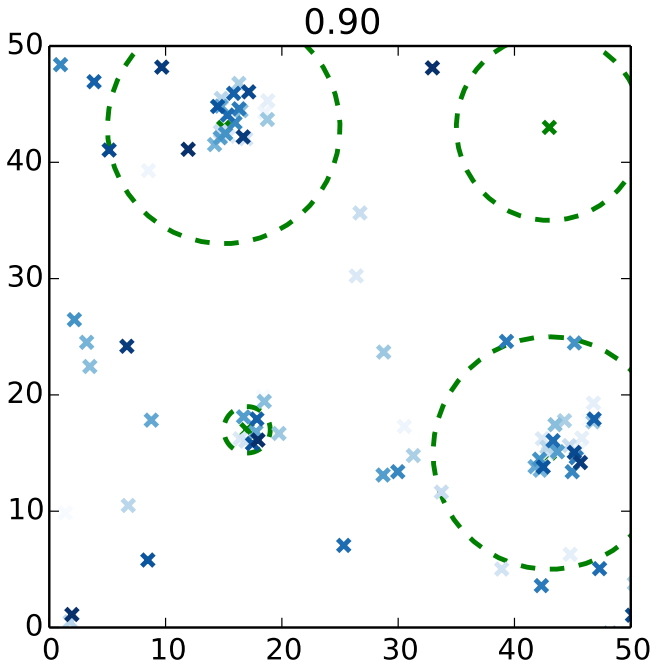
Kuva 5.2: Testiaineiston kuvat a, b, c: Parhaat ratkaisut energiafunktion suhteen (punaiset ympyrät) 100 kulkijan kokoelmasta jäähdytysskenaariolla $\alpha = 0.90$. Todelliset kiekot alarivillä vihreällä katkoviivalla.



Kuva 5.3: Samat testikuvat a, b, c kuin kuvassa 5.2: Parhaat ratkaisut energiafunktion suhteen (punaiset ympyrät) 100 kulkijan kokoelmasta jäähdytysskenaariolla $\alpha = 0.99$.



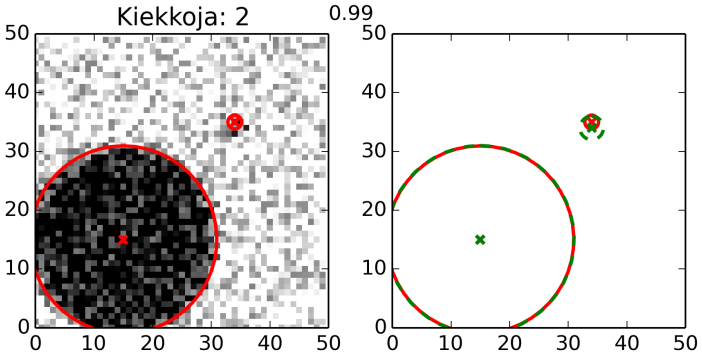
Kuva 5.4: Kuvan 5.2 skenaariota vastaavien kokoelmien kulkijoiden lopullisten energioiden jakaumat. Testikuvaa b vastaavaan histogrammiin merkitty kokoelman energialtaan paras ratkaisu sekä histogrammin moodi ($E \approx 280$).



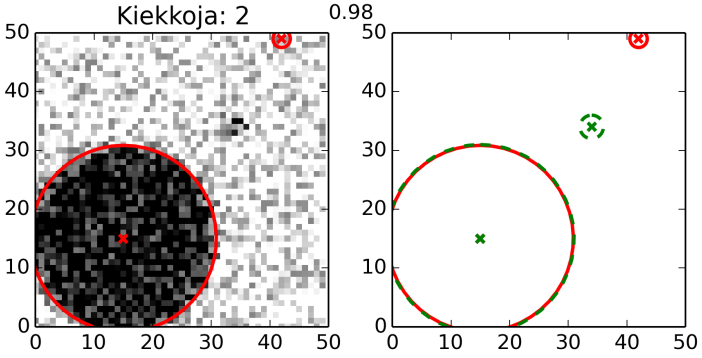
Kuva 5.5: Kuvan 5.4 testikuvan b histogrammin energiamoodin kulkijoiden ($n = 22$ kpl) löytämät keskipisteet, saman kulkijan pisteet samalla värillä. Keskipisteiden sijaintiin lisätty $U(-2, 2)$-tärinää päällekkäisten merkkien näkemiseksi. Toisin kuin kuvassa 5.4 näkyvässä parhaassa ratkaisussa, loppuenergiamoodin kulkijat eivät näytä löytävän neljättä kiekkoa ($x = 44, y = 44, r = 8$).

Taulukko 5.2: Testiaineiston kuva d. Kuvien 5.6 – 5.10 energiafunktion suhteen parhaiden ratkaisujen lopulliset energiat, virheet ja kulkijoiden pituus.

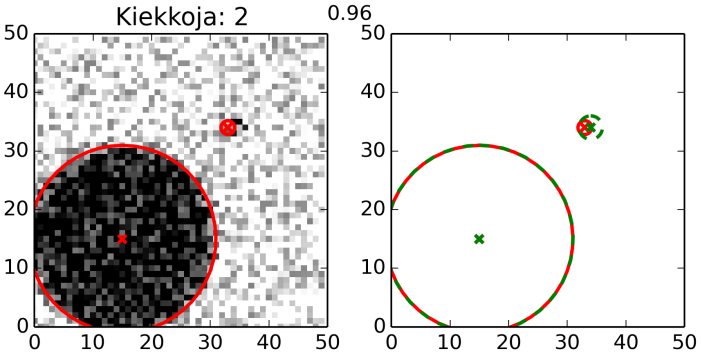
Aineisto	α	Markov-ketjuja	Energia	m_{sov}	m_{symm}
d	0.99	582	112.70531642	1.34903737042	12
	0.98	292	115.799173967	17.1400032558	20
	0.96	188	113.108609169	1.33302834307	12
	0.94	103	112.772445287	1.3089734381	12
	0.90	70	115.694414615	33.8856144998	22



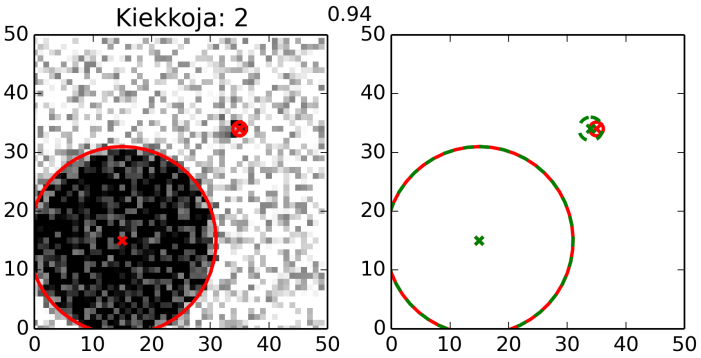
Kuva 5.6: Testikuva d, loppuenergialtaan paras kulkija 100 kulkijan kokoelmasta, kun $\alpha = 0.99$.



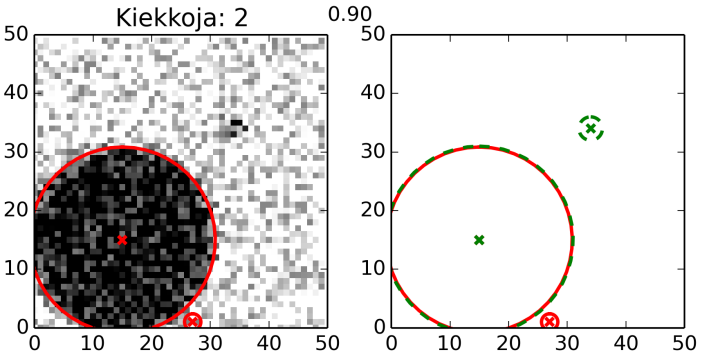
Kuva 5.7: Testikuva d. $\alpha = 0.98$.



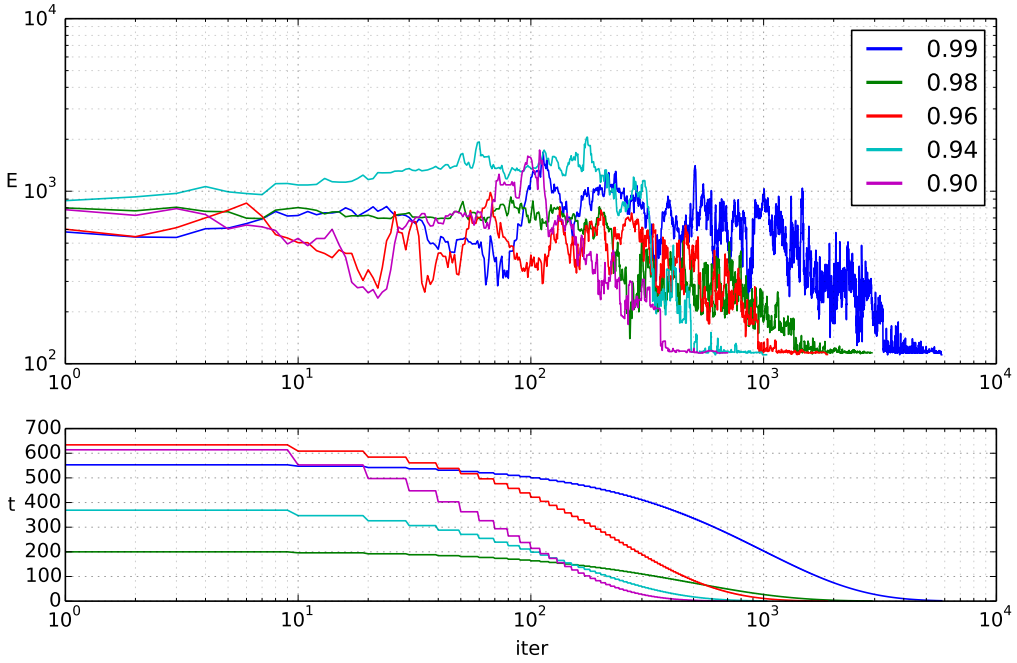
Kuva 5.8: Testikuva d. $\alpha = 0.96$.



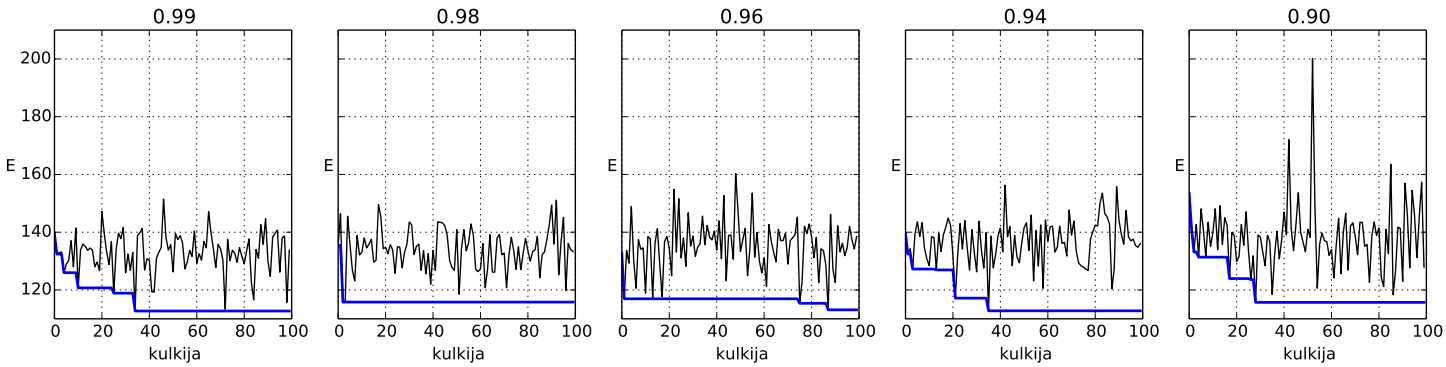
Kuva 5.9: Testikuva d. $\alpha = 0.94$.



Kuva 5.10: Testikuva d. $\alpha = 0.90$.



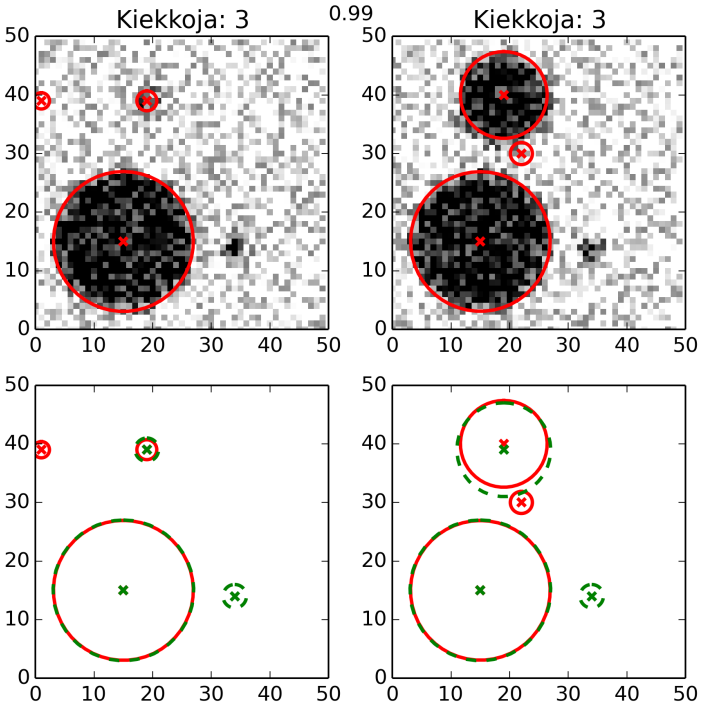
Kuva 5.11: Kuvien 5.6 – 5.10 kulkijoiden energioiden ja lämpötilojen kehitys. Stokastisen alkuarvon asetuksen takia alkulämpötilassa on huomattavaa vaihtelua ($t_0 \approx 200 \ldots 700$).



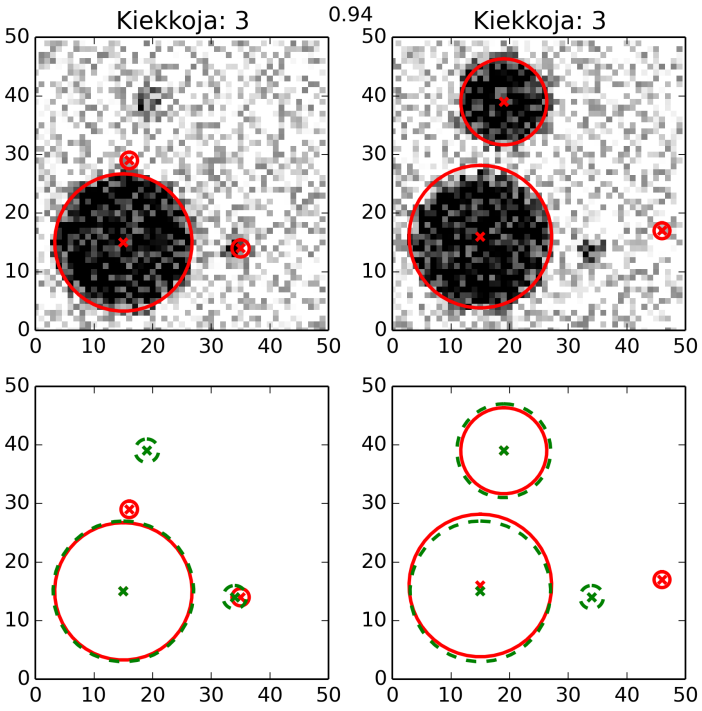
Kuva 5.12: Kuvien 5.6 – 5.10 vastaavien kokoelmien parhaan energian kehitys kokoelman koon funktiona.

Taulukko 5.3: Testiaineiston kuvat e,f. Kuvien 5.13 ja 5.15 energiafunktion suhteen parhaiden ratkaisujen lopulliset energiat, virheet ja kulkijoiden pituus.

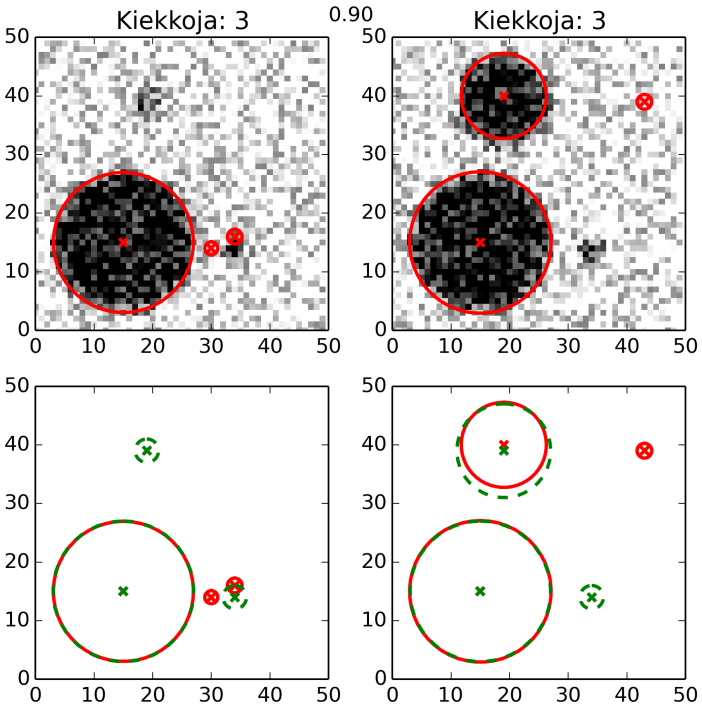
α	Aineisto	Markov-ketjuja	Energia	m_{sov}	m_{symm}
0.99	e	636	126.820768777	41.7710820974	26
	f	483	131.350277715	21.2525820166	58
0.94	e	89	126.912392409	29.5102365738	36
	f	102	134.677587474	27.8990164056	70
0.90	e	74	124.101054568	11.8660423496	30
	f	65	139.955257356	14.0600416833	92



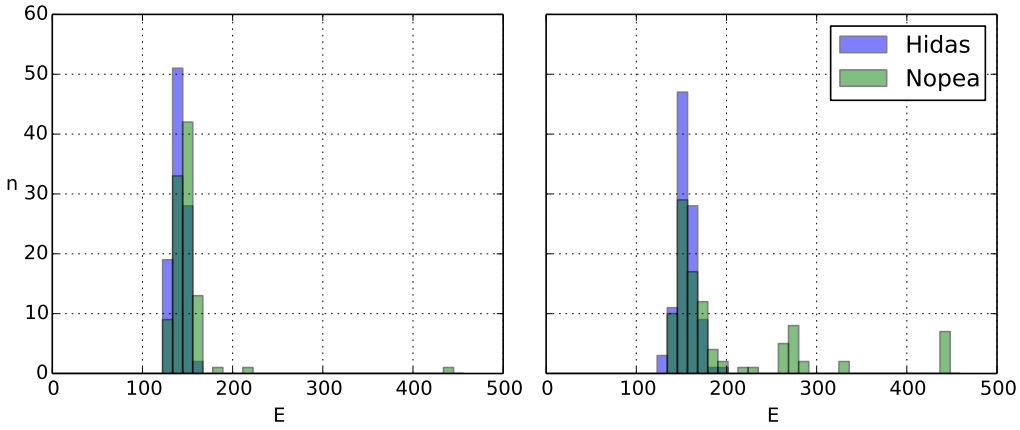
Kuva 5.13: Kuvat e, f: Parhaat ratkaisut (punaiset ympyrät) 100 kulkijan kokoelmasta jäähdytysskenaariolla $\alpha = 0.99$.



Kuva 5.14: Testikuvat e, f: Parhaat ratkaisut (punaiset ympyrät) 100 kulkijan kokoelmasta jäähdytysskenaariolla $\alpha = 0.94$.



Kuva 5.15: Testikuvat e, f: Parhaat ratkaisut (punaiset ympyrät) 100 kulkijan kokoelmasta jäähdytysskenaariolla $\alpha = 0.90$.



Kuva 5.16: Testikuvien e ja f kulkijakokoelmien loppuenergiajakaumien histogrammit jäähdytysskenaarioilla $\alpha = 0.99$ (hidas) ja $\alpha = 0.90$ (nopea)

Luku 6

Pohdinta ja johtopäätökset

6.1 Tulosten analyysi

6.1.1 Energiafunktio

Luvussa 5.3 esitettyjen tulosten perusteella voidaan todeta että algoritmi toimii yleensä kelvollisesti. Silmämääräisesti energiafunktioltaan parhaat ratkaisukiekot ovat usein varsin lähellä oikeaa. Energiafunktion teoreettista globaalia minimiä voidaan arvioida laskemalla energiafunktion E_{naiivi} odotusarvo E 50x50-kokoiselle kuvalle, jossa teoreettisen ihanteellisten ratkaisuympyröiden $P * A_W$ peittäessä datan ympyrät on jäljellä pelkkää mallin mukaista satunnaiskohinaa. Odotusarvoksi saadaan

$$\mathrm{E}\Big[E_{\text{naiivi}}([E]_{>0})\Big] = \mathrm{E}\Big[\sum_{i,j}[E(i,j)]^2_{>0}\Big], \tag{6.1}$$

missä kukin matriisin $[E]_{>0}$ alkio $[E(i,j)]_{>0}$ on yhtälön (3.17) mukainen satunnaismuuttuja, joka saadaan 'leikkaamalla' $U(-0.5, 0.5)$-jakautunut muuttuja $E(i,j)$ välille $[0,1]$, jolloin kun satunnaismuuttujan $E(i,j)$ tiheysfunktiota merkitään $p_{E(i,j)}(x)$

$$= \sum_{i,j} \mathrm{E}\Big[[E(i,j)]^2_{>0}\Big] \tag{6.2}$$

$$= \sum_{i,j} \left(\int_0^{0.5} x^2 p_{E(i,j)}(x)dx + \int_{-0.5}^{0} 0 \cdot p_{E(i,j)}(x)dx\right) \tag{6.3}$$

$$= \sum_{i,j} \left[\frac{x^3}{3}\right]_0^{0.5} = \sum_{i,j} \frac{1}{3 \cdot 8} = \sum_{i,j} \frac{1}{24} \tag{6.4}$$

$$= \frac{50 \cdot 50}{24} \approx 104.17, \tag{6.5}$$

mikä on hyvin lähellä muutamia myös silmämääräisesti erikoisen hyvin onnistuneiden ratkaisujen lopullisia energiatasoja, esimerkkinä kuvan 5.6 ratkaisu, jonka

loppunenergia ≈ 112.71 (taulukossa 5.2). Toisaalta pitää myös huomioida, että simuloidun datan ja mallin erojen ynnä muiden virheiden vuoksi on epärealistista odottaa tällaista ideaalista ratkaisua, jossa energiafunktiolla arvioitaisiin kuvaa josta kiekot olisi poistettu täydellisesti ja jäljellä olisi vain kohinaa.

Yllä vaihe (6.2) seuraa odotusarvon lineaarisuudesta ja (6.3) yleisestä kaavasta muunnetun (myös sekatyyppisen) satunnaismuuttujan odotusarvolle,

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)p_X(x)dx,$$

joiden tarkemmat perustelut löytyvät esimerkiksi luentomonisteessa [Koi13].

Toisaalta kuten jaksoissa 4.2.2 ja 4.2.3 oletettiin, paranneltu energiafunktio E_{par} ei ole täydellinen: ihmissilmään suunnilleen oikeissa paikoissa sijaitsevat mutta hieman epätarkasti A_{data}:n peittävät ratkaisukiekot, kuten kuvassa 5.2 viiden kiekon testidatan c ratkaisu ($\alpha = 0.90$, loppuenergia ≈ 159.5), on parempi ratkaisu kuin esimerkiksi saman datan toinen ratkaisu (kuvassa 5.2, $\alpha = 0.99$, energia ≈ 149.5) jossa suuret kiekot ovat sijoittuneet tarkemmin kuin $\alpha = 0.90$ -skenaariossa mutta yksi pieni kiekko täysin väärässä paikassa. Energiafunktio ja m_{symm} ovat arvoltaan kuitenkin kyseisellä silminnähden huonommalla ratkaisulla parempia, toisin kuin intuitiota paremmin vastaava mitta m_{sov}. Suurten kiekkojen pienetkin virheet näkyvät suuren pinta-alansa vuoksi energiafunktiossa ja virhemitasssa m_{symm} herkästi. Näin ollen ylipäätään pienet kiekot olivat algoritmille vaikeita, kuvat 5.13 – 5.15.

6.1.2 Jäähdytysstrategiat

Samoin kuten jäähdytysmenetelmän kirjallisuus antaa olettaa, hitaat jäähdytysskenaariot saavuttavat jossain määrin parempia tuloksia kuin hitaat. Vaikka sadan kulkijan joukosta valikoidut kaikkein parhaat ratkaisut saavat kaikki melko hyviä ja varsin samanlaisia tuloksia kaikilla α (kuvat 5.6 – 5.10, 5.13 – 5.15), hitaan ja hieman nopeamman jäähdytyksen ero näkyy kaikkien kulkijoiden loppuenergioiden histogrammissa 5.16: Nopeilla skenaarioilla histogrammin massa on oikeammalla.

Lisäksi havaitaan että useiden kiekkojen ongelmissa jakauma ei ole aina unimodaalinen, vaan erityisesti nopeilla strategioilla huomattavan moni kulkija voi 'jäähtyä' epäedulliseen konfiguraatioon jossa vain osa kiekoista löydetään. (Havainnollistava kuva 5.5, vastaavia piikkejä myös 5.16.)

Toisaalta kuitenkin vastapainona nopealle jäähdytysstrategialla kulkijakokoelman kokoa kasvattamalla todennäköisyyttä että nopeakin jäähdytys löytää hyvän tuloksen (kuva 5.12). Yleisesti jäähdytysmenetelmän kaltaisten algoritmien tehokkaassa soveltamisessa numeerisesti raskaissa ongelmissa usein onkin hyödyllistä yrittää löytää optimaalinen suhde jäähdytysskenaarion pituuden ja kulkijakokoelman koon välillä, ja etenkin sen dynaaminen arvioiminen ongelmanratkaisun aikana

[SSF02].

6.1.3 Ulottuvuuskirous

Kiekkojen lukumäärän kasvaessa yhdellä ongelman optimoitavien parametrien avaruuden ulottuvuus kasvaa kolmella. Tunnetusti optimointiongelmissa tämä avaruuden koon kasvu vaikeuttaa optimointiongelmia, ja tästä syystä kiekkojen määrä oletettiin tunnetuksi vakioksi. Näin ollen vielä tutkielman pienillä kiekkomäärillä $k = 1, \ldots, 5$ ulottuvuuksien kasvu ei näyttänyt olevan jäähdytysalgoritmille suuri ongelma.

6.2 Yhteenveto ja pohdinta

6.2.1 Algoritmi

Jäähdytysalgoritmi löysi esitettyyn kuvaongelmaan hyvän ratkaisun melko luotettavasti, joten tässä mielessä metodia voi pitää toimivana. SA on vakiintunut menetelmä, josta on laajalti monenlaisia sovelluksia. Tässä tutkielmassa sovellettiin perinteistä jäähdytysalgoritmin muotoilua hyvin yksinkertaisella jäähdytysskenaariolla. Myös toiminnan analysointiin ja parantelemiseen kuhunkin ongelmaan sopivaksi kirjallisuus esittelee runsaasti termodynamiisesta analogiasta inspiroituneita työkaluja, joiden käsittely jää tämän tutkielman ulkopuolelle. (Esimerkiksi tilastollisen mekaniikan käsitteiden, kuten jäähdytettevän ongelman 'ominaislämmön' tai 'entropian', hyödyntäminen ongelman rakenteen tutkimisessa, tai jäähdytysskenaarion tai siirtojen optimoinnissa , [ks. SSF02]. Nykyisenlaajuisessa tarkastelussa soveltamalla algoritmia erilaisiin ympyräkuviin erilaisilla jäähdytysskenaarioilla havaittiin SA:lle tyypillistä käytöstä.

6.2.2 Konenäkötehtävä

Kuten luvussa 1. "Johdanto" todettiin, oletettu kuvantunnistusongelma on melko yksinkertainen:

Esimerkiksi kiekkojen määrän olettaminen tunnetuksi on rajoittava yksinkertaistus. SA-algoritmia voisi laajentaa kasvattamalla parametriavaruuden kokoa oletettuun ympyröiden maksimimäärään, mutta vastavuoroisesti ulottuvuuksien määrä kasvattaisi ongelman kokoa. Eräs mielenkiintoinen vaihtoehto voisi olla soveltaa jotain perinteistä erillisten yhtenäisten komponenttien laskemiseen (*connected-component labeling*) tunnettua algoritmia [Cor+09] objektien lukumäärän selvittämiseen, ja sitten soveltaa tässä esitettyä SA-algoritmia. (Kappaleiden laskemista kuvassa voidaan myös lähestyä yleisenä ongelmana koneoppimisen menetelmin, kuten esimerkiksi [LZ10].)

Toisaalta myös erilliset ympyrät ovat myös melko rajoittunut malli monimutkaisempien muotojen mallintamiseen. Tällaisten vaikeuksien vuoksi konenäön alalla muotojen mallintamisessa tässä tutkielmassa käytetyn kaltaisten geometristen mallia sijasta suositumpia ovat erilaisiin merkitseviin pisteisiin perustuvat mallit (esimerkiksi *active contour model, snakes* [Pri12]).

Yhteenvetona, yleisesti reaalimaailman dekonvoluutiosovelluksia ajatellen tässä tutkielmassa käsitelty ongelma on rajoittunut. Usein tutkittavan kuvan rakenteesta ei voida tehdä näin yleisiä ja joudutaan käyttämään paljon yleisempiä malleja. Mutta ”oikeiden” konenäön ongelmien ratkaisemiseen käytetään yleisellä tasolla hyvin samankaltaisia metodeja: Sumennosta kuitenkin mallinnetaan konvoluutio-operaationa, ja ratkaisun lähtökohtana on usein muodostetaan jokin sopiva energiafunktio, jota optimoidaan.

Kirjallisuutta

[BB09] Wilhelm Burger ja Mark James Burge. *Principles of Digital Image Processing. Fundamental Techniques*. Undergraduate Topics in Computer Science. Lontoo: Springer, 2009.

[Bon05] Charles Boncelet. "Image Noise Models". Teoksessa: *Handbook of Image and Video Processing*. Toim. Alan C. Bovik. Amsterdam: Academic Press, 2005.

[CG95] Siddhartha Chib ja Edward Greenberg. "Understanding the Metropolis-Hastings Algorithm". *The American Statistician* 49 (1995).

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms*. 3. painos. Cambridge, Massachusetts: MIT Press, 2009.

[CVR14] Subhasis Chaudhuri, Rajbabu Velmurugan ja Renu Rameshan. *Blind Image Deconvolution*. Springer, 2014.

[Den+15] Vasil S. Denchev et al. "What is the Computational Value of Finite Range Tunneling?" (2015). arXiv:1512.02206. URL: `http://arxiv.org/abs/1512.02206`.

[GG84] Stuart Geman ja Donald Geman. "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1984), s. 721–741.

[GK03] Fred Glover ja Gary A. Kochenberger, toim. *Handbook of Metaheurestics*. Boston, Dordrecht, London: Kluwer Academic Publishers, 2003. ISBN: 1-4020-7263-5.

[HKR93] Juha Haataja, Juhani Käpyaho ja Jussi Rahola. *Numeeriset menetelmät*. CSC - Tieteellinen laskenta, 1993.

[Hun07] John D. Hunter. "Matplotlib: A 2D Graphics Environment". *Computing in Science & Engineering* 9 (2007), s. 90–95.

[J+01] Eric Jones, Travis Oliphant, Pearu Peterson et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 2015-11-24]. 2001–. URL: `http://www.scipy.org/`.

[Jäh02] Bernd Jähne. *Digital Image Processing*. 5. painos. Berliini ja Heidelberg: Springer, 2002.

[Joh+11] M. W. Johnson et al. "Quantum annealing with manufactured spins". *Nature* 473 (2011), s. 194–198.

[KGV83] S. Kirkpatrick, C. D. Gelatt Jr. ja M. P. Vecchi. "Optimization by Simulated Annealing". *Science* 220 (1983).

[Koi13] Petri Koistinen. *Todennäköisyyslaskenta*. Luentomoniste. 2013.

[LA87] P. J. M. van Laarhoven ja E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Dordrecht: Reidel, 1987.

[Lea92] V. F. Leavers. *Shape Detection in Computer Vision Using the Hough Transform*. Lontoo: Springer-Verlag, 1992.

[LZ10] Victor Lempitsky ja Andrew Zisserman. "Learning to Count Objects in Images". Teoksessa: *Advances in Neural Information Processing Systems 23*. Toim. J.D. Lafferty et al. Curran Associates, Inc., 2010, s. 1324–1332. URL: `http://papers.nips.cc/paper/4043-learning-to-count-objects-in-images.pdf`.

[Matlab14] *MATLAB*. Versio 8.3.0 (R2014a). Natick, Massachusetts: The MathWorks Inc., 2014.

[MS12] Jennifer Müeller ja Samuli Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. Computational Science & Engineering Series. Philadelphia: SIAM, 2012.

[Pre+07] William H. Press et al. *Numerical Recipes. Art of Scientific Programming*. 3. painos. New York: Cambridge University Press, 2007.

[Pri12] Simon J.D. Prince. *Computer vision: models, learning and inference*. Cambridge University Press, 2012.

[SSF02] Peter Salamon, Paolo Sibani ja Richard Frost. *Facts, Conjectures, and Improvements for Simulated Annealing*. SIAM Monographs on mathematical modeling and computation. Philadelphia: SIAM, 2002.

[ZBT15] Ilia Zintchenko, Ethan Brown ja Matthias Troyer. *Recent developments in quantum annealing*. 2015. URL: `http://www.scottaaronson.com/troyer.pdf`.