SIMULOITU JÄÄHDYTYS PENETRAATIOASTEEN OPTIMOINNISSA

Juulia Laulumaa

Pro gradu -tutkielma
Elokuu 2020

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Laulumaa, Juulia: Simuloitu jäähdytys penetraatioasteen optimoinnissa
Pro gradu -tutkielma, 44 s., 3 liites.
Sovellettu matematiikka
Elokuu 2020

Selkäreppuongelma on yksi diskreeteistä kombinatorisista optimointiongelmista, joissa aika ei riitä tehtävän laajuudesta riippuen täydelliseen luettelointiin. Tällaisten tehtävien ratkaisemiseksi on kehitelty heuristisia ratkaisualgoritmeja. Tässä tutkielmassa perehdytään yhden selkäreppuongelman erikoistapaukseen penetraatioasteen laskemiseksi, sekä sen ratkaisuun kahdella approksimointimenetelmällä: parantavalla haulla sekä simuloidulla jäähdytyksellä. Lisäksi tutustutaan simuloitu jäähdytys -algoritmin konvergenssiin homogeenisessa ja epähomogeenisessa tapauksessa Markov-ketjujen avulla.

Parantavan haun teoria ja algoritmin sekä simuloitu jäähdytys -algoritmi perustuvat Ronald Rardin teokseen Optimization in Operations Research. Simuloitu jäähdytys -menetelmän tarkastelu Markov-ketjujen avulla pohjautuu Laarhovenin ja Aartsin teokseen Simulated Annealing: Theory and Applications. Samaan teokseen perustuu myös simuloidussa jäähdytyksessä käytettävä jäähdytysohjelma. Tätä jäähdytysohjelman optimaalista valintaa tarkastellaan kahdella eri aineistolla osin puhtaasti kokeilemalla ja osin teoriaan perustuen. Samalla tutkitaan simuloitu jäähdytys -algoritmin erilaisten rakenteiden, kuten pisteen naapuruston, aloituspisteen sekä Markov-ketjujen siirtymätavan valinnan vaikutuksia penetraatioasteeseen.

Simuloitu jäähdytys konvergoi teoriassa kohti globaalia optimiarvoa tietyin ehdoin, mutta käytännössä tätä konvergenssia voidaan vain approksimoida, jolloin algoritmin päätymistä globaaliin optimiarvoon ei voida taata, mutta sopivalla toteutuksella voidaan saada hyviä tuloksia. Algoritmin hyvä toteutus voi riippua käytettävästä aineistosta paljonkin. Tutkielmassa käytettyjen aineistojen valossa voidaan todeta algoritmissa käytetyn aloituspisteen tyypin vallitsevan tuloksia. Näkyvä vaikutus, mutta selvästi pienempi, oli myös jäähdytysohjelman valinnalla varsinkin suoritusaikoihin sekä naapuruston että Markov-ketjujen siirtymien valinnalla.

Asiasanat: tiivistelmäsivu, Pro gradu -tutkielma, Simuloitu jäähdytys, LaTeX-ladontajärjestelmä.

Sisältö

1 Johdanto

Kombinatorisessa optimointiongelmassa etsitään joko kohdefunktion minimiä tai maksimia, missä muuttujat ovat diskreettejä, mutta konfiguraatioiden joukko on laaja. Konfiguraatioavaruus on niin suuri, ettei sitä ole mahdollista läpikäydä yksitellen, vaikka näin vertailemalla jokaisen kombinaation kohdefunktion arvoja pystyttäisiin etsimään paras kombinaatio. Tunnettu tällainen ongelma on selkäreppuongelma, jossa reppu tulisi pakata tavaroilla sen maksimikantokyvyn puitteissa. Jokaisella tavaralla on jokin arvo ja paino. Tavoitteena on löytää paras tavarakombinaatio, joka maksimoi niiden arvon, mutta joita reppu vielä kestää. Selkäreppuongelman oletetaan olevan vaikea, eikä sille ole löydetty nopeaa ratkaisualgoritmia. Tällaisten ongelmien ratkaisemiseen on kehitetty heuristisia approksimointimenetelmiä, joista simuloitu jäähdytys on yksi.

Simuloitu jäähdytys -menetelmän nykyversiossaan on kehittänyt Kirkpatrick, Gelatt ja Vecchi [9], jotka ymmärsivät käsitteiden 'kohdefunktion minimoinnin kombinatorisessa optimointiongelmassa' ja 'hitaasti viilenevän kiinteän aineen kunnes se saavuttaa matalan energian tason' yhteyden. He myös ymmärsivät, että tämä optimointiprosessi voidaan toteuttaa soveltamalla Metropolis-kriteeriä. Korvaamalla energian kustannuksilla ja toteuttamalla Metropolis-kriteeri hitaasti laskevana lämpötila-arvojen sarjana Kirkpatrick kollegoineen saivat aikaan optimointialgoritmin, jota he kutsuivat nimellä 'simulated annealing' (suom. simuloitu jäähdytys). [1]

Tässä tutkielmassa perehdytään yhteen selkäreppuongelman erikoistapaukseen, josta käytetään nimeä kauppaesimerkki, jonka tarkempi kuvaus on esitetty kappaleessa 2. Perinteisen selkäreppuongelman tavarat on korvattu nyt tuoteryhmillä tai tuotteilla, joista tulisi valita 'reppuun' niistä parhaimmat. Repun kantokyky on tarkoittaa tässä tapauksessa mainokseen haluttujen tuotteiden lukumäärää. Tuotteiden arvo perustuu siihen, kuinka moni asiakas on niitä ostanut, jolloin mainokseen halutaan paras tuotekombinaatio joka maksimoi penetraatioasteen eli prosenttiosuuden asiakkaista, jotka ovat ostaneet vähintään yhtä tuotetta tästä kyseisestä tuotekombinaatiosta. Kun asiakas on ostanut jonkun tuotteen, oletetaan että hän on kiinnostunut ostamaan saman tuotteen jatkossakin, jolloin maksimipenetraatiolla ja siten mainokseen valittujen tuotteiden avulla houkutellaan asiakkaita liikkeeseen mahdollisimman paljon. Ongelmalle esitetään kaksi heuristista ratkaisualgoritmia; parantava hakuheuristiikka kappaleessa 3 ja simuloitu jäähdytys kappaleessa 4. Simuloitu jäähdytys -menetelmän konvergenssia tutkitaan myös kappaleessa 4 Markov-ketjujen avulla. Lopussa käydään läpi kaksi konkreettista esimerkkiä kauppaesimerkin ratkaisemisesta simuloidulla jäähdytyksellä, sekä esitetään tuloksia erilaisilla algoritmin rakenteilla sekä parametreilla,

kuten Markov-ketjujen lukumäärä ja niiden pituus. Toinen näistä esimerkeistä on aineistoltaan pieni, jotta algoritmin ymmärtäminen olisi helpompaa, ja toinen perustuu taas oikeaan olemassa olevaan aineistoon, joka on huomattavasti laajempi.

2 Kauppaesimerkki

Tarkastellaan esimerkkinä kaupan myyntiä viimeisen päivän ajalta. Kaupassa on myynnissä tuotteita kahdestakymmenestä eri tuoteryhmästä. Viimeisen päivän aikana kaupassa kävi 100 asiakasta, ja kauppiaalla on tiedot asiakkaiden ostamista tuoteryhmistä. Kauppias haluaa laatia sen päivän perusteella mainoksen, joka houkuttelisi mahdollisimman monta asiakasta liikkeeseen. Mainokseen halutaan tuotteita viidestä eri tuoteryhmästä.

Ongelma on nyt eräänlainen selkäreppuongelma (knapsack), jossa pitää valita v tuoteryhmää maksimoiden *penetraatioaste*, joka perustuu asiakkaiden viime päivän ostoksiin. Penetraatioaste on siis prosenttiosuus asiakkaista, jotka ovat ostaneet vähintään yhdestä tuoteryhmästä tietystä tuoteryhmäkombinaatiosta. Muuttuja v on siis mainokseen haluttujen tuoteryhmien lukumäärä. Merkitään asiakkaita $i = 1, 2, 3, ..., m$ ja tuoteryhmiä $j = 1, 2, 3, ..., n$. Joukko N_i sisältää asiakkaan i tuoteryhmät j, joista se on ostanut tuotteita. Otetaan käyttöön seuraavat päätösmuuttujat:

$$x_j = \begin{cases} 1, \text{ jos tuoteryhmä } j \text{ on valittu mainokseen} \\ 0, \text{ muulloin,} \end{cases}$$

$$y_i = \begin{cases} 1, \text{ jos asiakas } i \text{ on kiinnostunut} \\ 0, \text{ muulloin.} \end{cases}$$

Saadaan tehtävälle seuraava yleinen kohdefunktio:

$$\max \sum_{i=1}^{m} y_i$$

$$\begin{array}{lll} \text{st.} & \sum_{j \in N_i} x_j \geq y_i, & i = 1, ..., m \\ & \sum_{j=1}^{n} x_j = v & \\ & y_i = \{0, 1\}, & i = 1, ..., m \\ & x_j = \{0, 1\}, & j = 1, ..., n. \end{array}$$

Jos $x_j = 1$, sanotaan tuoteryhmää j *valituksi tuoteryhmäksi*. Jos taas $x_j = 0$, sanotaan tuoteryhmää *ei-valituksi tuoteryhmäksi*.

Kauppaesimerkin ostoaineisto löytyy liitteestä A. Siinä rivit kuvastavat asiakkaita ja sarakkeet tuoteryhmiä, jolloin taulukossa arvo 1 tarkoittaa 'asiakas i on ostanut tuotteen tuoteryhmästä j' ja arvo 0 'asiakas i ei ole ostanut

tuotetta tuoteryhmästä j'. Liitteestä B löytyy aineiston A pohjalta tehty frekvenssitaulukko, jossa tuoteryhmät on järjestetty niiden ostofrekvenssien perusteella. Näiden liitteiden mukaista kauppaesimerkin aineistoa tullaan käyttämään parantavassa hakuheuristiikassa sekä simuloidun jäähdytyksen ensimmäisessä esimerkissä. Simuloidun jäähdytyksen toisessa esimerkissä käytetään huomattavasti laajempaa aineistoa.

Esimerkki 2.1. Penetraatioaste
Alla olevassa taulukossa on esitetty penetraatioasteen laskeminen kolmella tuoteryhmällä ja kymmenellä asiakkaalla. Taulukossa ryhmä tarkoittaa yksittäistä tuoteryhmää tai tuoteryhmäkombinaatiota ja aste tarkoittaa penetraatioastetta.

asiakas/ryhmä	1	2	3	1 ja 2	1 ja 3	2 ja 3
1	**ostanut**	ei ostanut	**ostanut**	**ostanut**	**ostanut**	**ostanut**
2	ei ostanut	**ostanut**	ei ostanut	**ostanut**	ei ostanut	**ostanut**
3	ei ostanut	**ostanut**	ei ostanut	**ostanut**	ei ostanut	**ostanut**
4	**ostanut**	**ostanut**	ei ostanut	**ostanut**	**ostanut**	**ostanut**
5	ei ostanut	**ostanut**	**ostanut**	**ostanut**	**ostanut**	**ostanut**
6	ei ostanut	ei ostanut	**ostanut**	ei ostanut	**ostanut**	**ostanut**
7	ei ostanut	ei ostanut	**ostanut**	ei ostanut	**ostanut**	**ostanut**
8	**ostanut**	ei ostanut	**ostanut**	**ostanut**	**ostanut**	**ostanut**
9	**ostanut**	ei ostanut	ei ostanut	**ostanut**	**ostanut**	ei ostanut
10	ei ostanut	**ostanut**	ei ostanut	**ostanut**	ei ostanut	**ostanut**
aste %	40	50	50	80	70	90

Taulukosta nähdään, että paras penetraatioaste saavutettiin tuoteryhmäkombinaatiolla 2 ja 3 penetraatioasteella 90%, jos mainokseen halutaan 2 tuoteryhmää. Jos mainokseen halutaan vain yksi tuoteryhmä, on paras valinta tuoteryhmä 2 tai 3. Kaiken kaikkiaan paras tulos saadaan tietenkin kombinaatiolla 1 ja 2 ja 3, jolloin tavoitetaan kaikki asiakkaat penetraatioasteella 100%, mutta vain jos mainokseen halutaan kaikki kolme tuoteryhmää.

3 Diskreetit optimointimenetelmät

Edellinen esimerkki on yksi diskreeteistä optimointitehtävistä. Muita hyvin tunnettuja diskreettejä optimointitehtäviä selkäreppuongelman lisäksi on kauppamatkustaja- ja reititysongelmat. Nämä kokonaislukuoptimointitehtävät voitaisiin ratkaista käymällä kaikki sallitut pisteet läpi ja valitsemalla paras. Tehtävä osoittautuu kuitenkin haastavaksi, kun erilaisia kombinaatioita on todella paljon.

Jotta selkäreppuongelman vaativuutta voitaisiin ymmärtää paremmin, tutustutaan käsitteisiin P ja NP, joiden teoria on peräisin Matti Monosen Pro gradu -tutkielmasta 2012 [7]. Joukkoon P kuuluvat päätösongelmat, joilla on olemassa polynomisessa ajassa toimiva ratkaisumenetelmä. Jos ratkaisu vaatii ylipolynomisen ajan, mutta vastaus pystytään verifioimaan polynomisessa ajassa, päätösongelman sanotaan kuuluvan joukkoon NP. Huomataan, että jokainen polynomisessa ajassa ratkeava päätösongelma kuuluu myös luokkaan NP eli $P \subseteq NP$. Joukkojen yhtäsuuruutta ei ole pystytty kumoamaan tai vahvistamaan, ja tämän tilanteen selvittäminen onkin kuuluisa "$P = NP$?" -ongelma. Arvellaan kuitenkin että $P \neq NP$. Näin ollen joukon P ongelmat mielletään yleisesti helpoiksi ja sen ulkopuolelle jäävät ongelmat, kuten joukon NP, vaikeiksi.

Määritelmä 3.1. Ongelma P on NP-*vaikea*, jos ratkaisemalla ensin ongelma P voidaan jokainen luokan NP ongelma ratkaista polynomisessa ajassa.

Määritelmä 3.2. Ongelma P kuuluu joukkoon NP-*täydellinen*, jos se on NP-vaikea ja kuuluu joukkoon NP.

Selkäreppuongelman ja muiden haku- ja optimointiongelmien todistaminen vaikeaksi ongelmaksi on usein hyvin haastavaa. Formuloimalla polynomiaikainen palautus voitaisiin ongelman P kuitenkin todeta olevan NP-vaikea suhteellisen helposti. Tästä johtuen selkäreppuongelman oletetaan olevan vaikea, vaikkei sitä ole täysin pystytty todistamaan. NP-täydellisillä ongelmilla on merkittävä ominaisuus, että ne ovat joko kaikki helppoja tai kaikki vaikeita. Siksi polynomisen ratkaisumenetelmän löytyminen selkäreppuongelmalle tarkoittaisi, että kaikille joukon NP(-täydellisille) ongelmille löytyisi polynomiaikainen ratkaisualgoritmi. Tätä pidetään kuitenkin epätodennäköisenä.

Olemme siis tilanteessa, jossa emme pysty ratkaisemaan suuria kombinatorisia vaikeita optimointiongelmia, koska optimin löytäminen vaatii kohtuuttoman paljon laskenta-aikaa. Ongelman ratkaisemiseksi on kehitetty heuristisia menetelmiä, jotka eivät anna välttämättä optimiarvoa vastaukseksi, mutta antavat usein riittävän hyviä tuloksia laskenta-ajan ollessa polynomiajan rajoissa.

Tämän kappaleen teoria pohjautuu suoraan Ronald L. Rardinin teokseen Optimization in Operation Research [3] ja osin Mäkelän luentomonisteeseen [4], joka sekin pohjautuu Rardin teokseen.

Kaikkien mahdollisten kombinaatioiden läpikäymistä kutsutaan *täydelliseksi luetteloinniksi*. Jos mallissa on vain muutama diskreetti päätösmuuttuja, on tehokkain tapa yleensä käyttää täydellistä luettelointia. Menetelmä kokeilee kaikkia mahdollisia kombinaatioita diskreetin muuttujan arvoille ja

laskee kohdefunktion arvon. Kombinaatio, joka antaa parhaan ratkaisun kohdefunktiolle, on optimaalinen. Kauppaesimerkissä oli 20 tuoteryhmää, joista piti valita 5 tuoteryhmää mainokseen. Mahdollisia sallittuja kaikki rajoitukset huomioon ottavia kombinaatioita on siis $\frac{20!}{5!(20-5)!} = 15504$ kappaletta. Esimerkki voitaisiin ratkaista täydellisellä luetteloinnilla, mutta jos katsotaan todenmukaisempaa esimerkkiä, jota käsitellään myöhemmin, missä noin 9 000 tuotteesta on valittava 5 tuotetta, on sallittujen kombinaatioiden lukumäärä jo 4.9×10^{17}. Jos tietokoneella laskiessa yhden kombinaation laskemiseen kuluisi millisekunti aikaa, menisi kaikkien mahdollisuuksien läpikäyntiin noin 1.5×10^7 vuotta.

Jos tarkastellaan kauppaesimerkin koko kombinaatioavaruutta, on jokaisella tuoteryhmämuuttujalla kaksi mahdollista arvoa 0 (ei valita mainokseen) ja 1 (valitaan mainokseen). Kombinaatioiden kokonaislukumäärä on siis $2^{20} = 1048576$ kappaletta, jos tuoteryhmiä on 20. Jos tuoteryhmiä on n kappaletta, on kombinaatioita silloin 2^n. Kun tuoteryhmien lukumäärää nostetaan, lisääntyy kombinaatioiden lukumäärä eksponentiaalisesti. Eksponentiaalinen kasvu ei kuitenkaan suoraan tarkoita, että ongelma olisi vaikea. Esimerkiksi pienin virittävä puu -ongelman aikavaativuus on eksponentiaalinen täydellisellä luetteloinnilla, mutta sille on kuitenkin olemassa tehokkaita tietorakenteita hyväksi käyttävä algoritmi, jonka aikavaativuus on polynominen, tarkemmin $\mathcal{O}(n \log n)$ [8].

On huomattu, että binäärinen selkäreppuongelma olisi *heikosti NP-vaikea* olemalla kuitenkin vaikea, mutta NP-vaikeiden helpoimmasta päästä. Ongelmalle voidaan siis löytää pseudo-polynominen ratkaisumenetelmä. Tällaisen aikavaativuuden ongelma voidaan ratkaista polynomiajassa syötelukujen suuruuteen nähden, jolloin aikavaativuus suhteessa syötelukujen esityspituuteen on kuitenkin eksponentiaalinen [7].

Kun täydellinen luettelointi ei tule kyseeseen, voidaan kokonaislukuoptimointitehtävässä koittaa käyttää relaksaatiota. Siinä ideana on heikentää rajoituksia tai kohdefunktiota. Näin saadaan kasvatettua sallittujen pisteiden joukkoa. Rajoituksiin keskittyvässä relaksaatiossa saadaan aikaan yksinkertaisempi malli lieventämällä tai suoraan poistamalla rajoituksia. Malli $\tilde{P}$ on mallin P relaksaatio, jos jokainen mallin P sallittu piste on myös sallittu piste mallissa $\tilde{P}$, ja molemmissa malleissa on sama kohdefunktio. Jos relaksoidulla mallilla $\tilde{P}$ ei ole yhtään sallittua pistettä, ei ole myöskään alkuperäisellä mallilla P. Jos relaksaatiomallille löydetään optimiratkaisu, joka on sallittu myös alkuperäisessä mallissa, se on myös alkuperäisen mallin optimiratkaisu. Relaksaatio antaa maksimoitaessa ylärajan optimiratkaisulle alkuperäiseen malliin. Minimoitaessa se taas antaa alarajan alkuperäisen mallin kohdefunktion optimiarvolle. [4]

Monet kombinatoriset optimointitehtävät ovat liian suuria täydelliseen luetteloimiin eikä relaksaatiomenetelmä approksimoi tarpeeksi hyvin alkuperäistä tehtävää. Heuristisista menetelmistä löytyy kuitenkin vaihtoehtoja, jotka voivat tuottaa hyviä sallittuja ratkaisuja, vaikka niiden optimaalisuutta ei voida taata, tai edes kuinka lähellä ne ovat optimia. Tällaisia menetelmiä on esimerkiksi *parantava hakuheuristiikka, tabuhaku, simuloitu jäähdytys* ja *geneettiset algoritmit*. Tarkastellaan näistä tarkemmin parantavaa hakuheuristiikkaa ja simuloitua jäähdytystä.

3.1 Parantava hakuheuristiikka

Parantava hakuheuristiikka pyrkii löytämään nykyisen pisteen x^t *naapurustosta* parempia ratkaisuja. Algoritmi tarvitsee jonkin aloituspisteen x^0. Merkitään kaikkien pisteiden joukkoa R. Jokaisella iteraatiokierroksella t määritellään pisteen x^t naapurusto R_{x^t}, josta koitetaan löytää uusi sallittu ja parantava piste. Merkitään joukolla M siirtymiä, jotka määrittelevät pisteen x^t naapuruston. Pisteen x^t naapurustoon kuuluu siis pisteet, jotka ovat saavutettavissa siitä yhdellä siirtymällä $\Delta x \in M$, eli naapurusto $R_{x^t} = x^t + \Delta x$, missä $\Delta x \in M$.

Algoritmi 3.3. Diskreetti parantava haku

Askel 0. Valitaan jokin aloituspiste x^0 ja asetetaan indeksilaskuriksi $t = 0$.

Askel 1. Jos mikään siirroista $\Delta x \in M$ ei ole sallittu eikä parantava pisteelle x^t, niin lopetetaan. On löydetty lokaali optimi x^t.

Askel 2. Valitaan jokin sallittu ja parantava siirto $\Delta x \in M$. Merkitään tätä siirtoa Δx^{t+1}.

Askel 3. Päivitetään uudeksi pisteeksi $x^{t+1} = x^t + \Delta x^{t+1}$.

Askel 4. Päivitetään indeksilaskuri $t = t + 1$, ja mennään askeleeseen 1.

Esimerkki 3.4. Tarkastellaan seuraavaa diskreettiä optimointitehtävää:

$$\begin{aligned} \max \quad & -5x_1 + 15x_2 + 7x_3 \\ \text{s.t.} \quad & x_1 + 2x_2 + 4x_3 \leq 5 \\ & x_1, x_2, x_3 = \{0, 1\}. \end{aligned}$$

Valitaan aloituspisteeksi $x^0 = (1, 1, 0)$. Tehtävän siirtymäjoukko

$$M = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \right\}.$$

Pisteen x^0 naapurit ovat näin ollen siirtymäjoukon M nojalla

$$
\begin{array}{lcl}
(1,1,0) + (1,0,0) & = & (2,1,0) \\
(1,1,0) + (\text{-}1,0,0) & = & (0,1,0) \\
(1,1,0) + (0,1,0) & = & (1,2,0) \\
(1,1,0) + (0,\text{-}1,0) & = & (1,0,0) \\
(1,1,0) + (0,0,1) & = & (1,1,1) \\
(1,1,0) + (0,0,\text{-}1) & = & (1,1,\text{-}1).
\end{array}
$$

Näistä naapureista ainoastaan (0,1,0) ja (1,0,0) ovat sallittuja siirtoja. Pisteellä (0,1,0) kohdefunktion arvo on 15 ja pisteellä (1,0,0) se on -5. Nykyisellä pisteellä x^0 kohdefunktio saa arvon 10, joten $x = (0,1,0)$ on ainut naapuri, joka on sallittu ja parantava. Valitaan siis $x^1 = (0,1,0)$ seuraavaksi pisteeksi ja todetaan, että se on myös lokaali optimi.

Diskreetissä parantavassa haussa siirtymäjoukon valinta on ratkaiseva tekijä. Jos olisi mahdollista, asetettaisiin kaikki pisteet toistensa naapureiksi. Silloin päädyttäisiin globaaliin optimiin, koska lopetuskriteerin perusteella naapurista (joukosta R) ei löydettäisi sallittua ja parantavaa pistettä. Käytännössä kuitenkin siirtymäjoukkoa on pakko rajoittaa. Sen on oltava tarpeeksi suppea, jotta se pystytään käymään läpi jokaisella iteraatiokierroksella tehokkaasti. Toisaalta se ei saa olla liian suppea, milloin iteraatiokierroksilla käydään läpi vain muutama vaihtoehto. Silloin voidaan päätyä huonoon lokaaliin optimiin. Laajemmalla siirtymäjoukolla päädytään todennäköisemmin parempaan vastaukseen, mutta näin kasvatetaan myös algoritmin suoritusaikaa. Siirtymäjoukon valinnalla on siis merkittävä vaikutus siihen, millaiseen vastaukseen päädytään, ja kuinka tehokas ja nopea algoritmi on.

Esimerkissä 3.4 käytettiin siirtymäjoukkona yhden muuttujan arvon lisäämistä tai vähentämistä luvulla 1. Kauppaesimerkissä samantyylinen siirtymäjoukon valinta ei ole eduksi, sillä sallitussa pisteessä tulee olla täsmälleen 5 tuoteryhmää valittuna rajoitusten nojalla, joten yhden muuttujan arvon (tuotteen) lisääminen tai vähentäminen johtaisi ei sallittuihin pisteisiin. Tällöin parempi vaihtoehto on käyttää siirtymäjoukkona *parittaisia vaihtoja* (pairwise interchanges) (k, l). Tarkoituksena on vaihtaa indeksin k solun arvo indeksin l solun arvon kanssa.

Esimerkki 3.5. Tarkastellaan kauppaesimerkkiä (aineistona liittee A), jossa alkuarvona on valittuna tuotteet {1,2,3,4,5}. Merkitään siis aloituspisteeksi $x^0 = (1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$, kun tuoteryhmiä on 20 ($n = 20$). Valitaan satunnaisesti siirroksi parittainen vaihto (1,6), jolloin tuotteiden vaihto saadaan muutoksilla $\Delta x_1 = -1$ ja $\Delta x_6 = 1$. Uudeksi pisteeksi tulee x^1 $= (0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$. Joukossa M on nyt 75 mahdollista vaihdettavien suureiden k ja l kombinaatiota. Jokainen valittu tuote ($x_j = 1$) voidaan vaihtaa ei-valitun ($x_j = 0$) tuotteen kanssa. Ei-valittujen tuotteiden

määrä on $n - v = 20 - 5 = 15$, joten jokaisella valitulla tuotteella on 15 mahdollista vaihtoparia. Yhteensä vaihtopareja on siis $v * (n - v) = 75$. Näin määrittelemällä kaikki joukon M siirrot ovat sallittuja.

Esimerkki 3.6. Diskreetti parantava haku

Jatketaan kauppaesimerkin käsittelyä (aineistona liite A). Aloitetaan pisteestä $x^0 = (1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, jolloin kohdefunktio saa arvon 89. Mahdollisia siirtoja on nyt 75. Listataan näiden kaikkien siirtojen antama kohdefunktion arvo alla olevaan taulukkoon.

k/l	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	89	86	89	85	86	88	87	86	87	86	88	87	89	90	91
2	92	88	91	87	88	90	89	87	88	88	91	89	91	**94**	92
3	82	76	81	77	76	80	77	75	76	81	79	77	80	81	86
4	88	85	87	83	85	86	85	83	86	86	86	85	86	91	91
5	88	82	87	81	81	84	82	80	83	83	85	82	86	87	93

Käytetään *nopeimman laskeutumisen menetelmää*, jossa valitaan siirto $\Delta x \in M$ niin, että kohdefunktio saa pienimmän arvonsa pisteessä $x^t + \Delta x$. Nähdään, että siirrolla (2,19) saadaan paras kohdefunktion arvon parannus $\Delta x = 5$, joten valitaan x^2=(1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0). Saadaan lopulta seuraava taulukko käyttäen nopeimman laskeutumisen menetelmää.

t	valitut tuotteet	kohdefunktio	vaihto	muutos
0	{1,2,3,4,5}	89	(2,19)	$\Delta x_2 = -1, \Delta x_{19} = 1$
1	{1,3,4,5,19}	94	(4,20)	$\Delta x_4 = -1, \Delta x_{20} = 1$
2	{1,3,5,19,20}	96	-	lokaali optimi

Päädyttiin siis lokaaliin optimiarvoon 96, joka on lähtöarvosta 7% parempi. Yhteensä jouduttiin laskemaan 3*75=225 kombinaatiota.

Jos esimerkissä 3.6 hyväksyttäisiin ensimmäinen parantava siirto, olisi se (1,19), jos vaihdot aloitetaan numerojärjestyksessä. Silloin kohdefunktio saa arvoksi 90. Seuraava parantava siirto saadaan heti ensimmäisellä parilla (2,1), jolloin kohdefunktion arvo on 94. Seuraavalla parittaisella vaihdolla (4,20) päädytään jo lokaaliin maximiin 96 vain 60 kombinaation laskulla. Tuotteiksi valikoitui {1,3,5,19,20}. Päästiin siis täsmälleen samaan ratkaisuun kuin nopeimman laskeutumisen menetelmässä, mutta kombinaatioita laskettiin 225 sijaan vain 60.

Parittaisten vaihtojen sijasta voitaisiin käyttää *kolmittaisia* tai *neljittäisiä vaihtoja*. Edellisessä esimerkissä kolmittainen vaihto voitaisiin suorittaa valitsemalla kolme valittua tuotetta ja kolme ei-valittua tuotetta ja vaihtamalla niiden arvot keskenään. Tosin sama saadaan aikaan parittaisia vaihtoja

toistamalla. Kauppamatkustajaongelmassa kolmittaisilla vaihdoilla voitaisiin saavuttaa enemmän etuja.

Hakuheuristiikka voidaan aloittaa yhden aloituspisteen sijasta myös *useammasta pisteestä* (multistart). Nimensä mukaisesti haku aloitetaan useammasta eri aloituspisteestä. Näin voidaan parantaa algoritmin tuomia ratkaisuja. Aloituspisteet voidaan valita esimerkiksi satunnaisesti, jolloin liian hyvän aloituspisteen käyttäminen, ja siten haun päätyminen liian nopeasti lokaaliin optimiin, riski pienentyy. Näin tulokseksi voidaan saada useampi lokaali optimi.

Esimerkki 3.7. Useamman pisteen aloitus satunnaisista alkupisteistä kauppaesimerkissä taulukoituna käyttäen nopeimman laskeutumisen menetelmää.

t	valitut tuotteet	kohdefunktio	vaihto	muutos
		Haku 1		
0	{1,2,3,4,5}	89	(2,19)	$\Delta x_2 = -1, \Delta x_{19} = 1$
1	{1,3,4,5,19}	94	(4,20)	$\Delta x_4 = -1, \Delta x_{20} = 1$
2	{1,3,5,19,20}	96	-	lokaali optimi
		Haku 2		
0	{3,7,8,16,18}	83	(7,20)	$\Delta x_7 = -1, \Delta x_{20} = 1$
1	{3,8,16,18,20}	91	(8,4)	$\Delta x_8 = -1, \Delta x_4 = 1$
2	{3,4,16,18,20}	94	(16,8)	$\Delta x_{16} = -1, \Delta x_8 = 1$
3	{3,4,8,18,20}	95	-	lokaali optimi
		Haku 3		
0	{6,9,10,13,14}	53	(10,3)	$\Delta x_{10} = -1, \Delta x_3 = 1$
1	{3,6,9,13,14}	81	(9,20)	$\Delta x_9 = -1, \Delta x_{20} = 1$
2	{3,6,13,14,20}	91	(13,19)	$\Delta x_{13} = -1, \Delta x_{19} = 1$
3	{3,6,14,19,20}	95	(6,8)	$\Delta x_6 = -1, \Delta x_1 = 1$
4	{3,8,14,19,20}	96	-	lokaali optimi

Kuten taulukosta huomataan, erilaisilla alkuarvoilla päästään lähes samaan kohdefunktion arvoon, mutta iteraatiokierrosten ja siten kombinaatioiden laskussa on huomattava ero. Useamman pisteen haku ilmoittaisi vastaukseksi parhaan haun 1 tuotteet {1,3,5,19,20}, koska kohdefunktio sai siinä parhaan arvon ja kombinaatioita laskettiin 225. Huomataan myös, että haussa 3 päästään samaan kohdefunktion arvoon 96 eri tuotteilla {3,8,14,19,20}, mutta kombinaatioita laskettiin peräti 375.

3.2 Parantavan haun laajennukset

Vaikka parantava hakuheuristiikka voi olla tehokas monille optimointimalleille sellaisenaan, voi olla hyödyksi sallia myös ei-parantavia askelia. Täl-

löin tarkoituksena on päästä eteenpäin lokaalista optimista mahdollisesti parempaan lokaaliin optimiin. Muutama askel huonompaan suuntaan voi johtaa haun suuntaan, jossa voidaan saavuttaa parempia tuloksia, jopa globaali optimiarvo. Sallimalla huonontavia siirtoja, voidaan kuitenkin päätyä algoritmissa ikuiseen silmukkaan. Jotta sellaiselta voidaan välttyä, on otettava käyttöön ehtoja, jotka estävät saman tuloksen toistoa.

Kolme tunnetuinta tällaista huonontavia askelia sallivaa menetelmää on tabuhaku, simuloitu jäähdytys ja geneettinen algoritmi. Jokaisella on oma menetelmänsä estää toistuvia askelia. Tabuhaussa pidetään yllä tabulistaa, jossa on tieto muutamista viimeisistä askeleista. Listassa olevat askeleet ovat kiellettyjä, kunnes ne ovat poistuneet listalta. Näin estetään askelien tiheä toistuminen, vaikkakin pidemmällä ajanjaksolla se on mahdollista. Simuloidussa jäähdytyksessä toistuvaa sykliä kontrolloidaan hyväksymällä huonontava askel tietyllä todennäköisyydellä, joka pienenee algoritmin edetessä. Geneettinen algoritmi pyrkii jäljittelemään biologista periytymistä, jolloin eri ratkaisuiden hyvät ominaisuudet yritetään periyttää tulevalle sukupolvelle. Näistä kolmesta menetelmästä perehdytään tarkemmin simuloituun jäähdytykseen.

4 Simuloitu jäähdytys

Kuten aikaisemmin todettiin on simuloitu jäähdytys (Simulated Annealing) yksi kombinatorinen diskreetti optimointimenetelmä, joka hyväksyy myös huonontavia siirtymiä vähenevällä todennäköisyydellä. Se on heuristinen menetelmä, joten se ei takaa globaaliin optimiin päätymistä, mutta voi päätyä hyvään lokaaliin optimiarvoon. Kappale pohjautuu vahvasti Laarhovenin ja Aartsin esitykseen teoksessa Simulated Annealing: Theory and Applications. Myöhemmin kappaleessa esitetyt simuloidun jäähdytyksen konvergenssitodistukset puolestaan viittaavat Mitran, Romeon ja Sangiovanni-Vincentellin esitykseen [2].

Simuloitu jäähdytys perustuu yhteyteen kiinteän aineen hehkuttamisen ja suuren kombinatorisen optimointiongelman ratkaisemisen välillä, minkä mukaan algoritmi on saanut nimensäkin. Hehkutus tarkoittaa fysikaalista prosessia, jossa lämpöhauteessa olevaa kiinteää ainetta kuumennetaan lisäämällä lämpöhauteen lämpötilaa pisteeseen, jossa kaikki kiinteän aineen hiukkaset järjestäytyvät satunnaisesti nestemäiseen muotoon, minkä jälkeen ainetta jäähdytetään laskemalla hitaasti lämpöhauteen lämpötilaa. Näin kaikki hiukkaset järjestäytyvät alhaisen energian perustilaan kunkin hilarakenteen mukaisesti, kunhan lämpötila on riittävän korkea alussa ja jäähdytys tapahtuu tarpeeksi hitaasti. Kussakin lämpötilassa T kiinteän aineen tulee saavut-

taa lämpötasapaino, jota kuvataan todennäköisyydellä, että aine on tilassa energialla E annetulla *Boltzmannin jakaumalla*:

$$P(\mathbf{E} = E) = \frac{1}{\mathbf{Z}(T)} \cdot exp(-\frac{E}{k_B T}),$$

missä $\mathbf{Z}(T)$ on normalisoiva vakio (tunnettu myös *partitiofunktiona*) ja k_B on *Boltzmannin vakio*. Tekijä $exp(-\frac{E}{k_B T})$ tunnetaan *Boltzmannin tekijänä*. Kun lämpötila laskee, Boltzmannin jakauma keskittyy kohti matalan energian tilaa, ja lopulta kun lämpötila lähestyy nollaa, vain tilalla, jolla on matalin energia, on nollasta poikkeava todennäköisyys esiintyä. Jos jäähtyminen on liian nopeaa (aineen ei anneta saavuttaa lämpötasapainoa jokaisessa lämpötilassa), voi aineeseen jäätyä epävakauksia, jolloin voidaan saavuttaa metastabiili amorfinen rakenne ennemmin kuin haluttu matalaenerginen kiteinen hilarakenne. Lisäksi jos lämpötila lämpöhauteessa lasketaan välittömästi, päädytään myös hiukkasten jäätymiseen aineessa ja siten metastabiiliin amorfiseen rakenteeseen.

Kiinteän aineen lämpötasapainon kehittymisen simulointiin muuttuvissa lämpötiloissa T on käytetty *Monte Carlon -menetelmää*. Menetelmä generoi kiinteän aineen eri tiloja. Kun aineen nykyistä tilaa kuvataan hiukkasten sijainneilla, saavutetaan uusi tila aiheuttamalla pieni, satunnaisesti valittu perturbaatio, pienellä siirtymällä satunnaisesti valittuun hiukkaseen. Jos muutos energiassa ΔE nykyisen ja kevyesti perturboidun tilan välillä on negatiivinen (perturbaatio tuottaa aineelle matalamman energian tilan), niin edetään uuteen tilaan. Jos $\Delta E \geq 0$, niin perturboitu tila hyväksytään todennäköisyydellä $exp(-\frac{\Delta E}{k_B T})$. Tämä hyväksymissääntö uudelle tilalle tunnetaan *Metropolis-kriteerinä*. Noudattamalla kriteeriä päädytään lopulta lämpötasapainoon. Kun on suoritettu suuri määrä perturbaatioita hyväksymissääntöä noudattaen, tilojen todennäköisyysjakauma lähestyy Boltzmannin jakaumaa. Tilastollisessa mekaniikassa Monte Carlon -menetelmää (tunnetaan myös nimellä *Metropolis-algoritmi*) käytetään keskiarvojen tai integraalien estimointiin satunnaisten näytteenottomenetelmien avulla.

Metropolis-algoritmia voidaan käyttää myös uusien pisteiden generoimiseen kombinatorisessa optimointiongelmassa. Nyt pisteet $(x_1, x_2, x_3, ..., x_t)$ vastaavat käsitystä kiinteän aineen tiloista. Kustannusfunktio $C(x)$ ja *kontrolliparametri* c vastaavat taas energiaa ja lämpötilaa. Simuloitu jäähdytys -algoritmissa voidaan nähdä Metropolis-algoritmin ilmaisevan laskevia kontrolliparametrin arvoja. Alkuarvoksi kontrolliparametri saa suuren arvon ja askelien edetessä se pienenee. Algoritmin askeleet määritellään kuten parantavassa haussa: pisteestä x^t siirtyminen toiseen pisteeseen voidaan teh-

dä valitsemalla satunnainen pisteen x^t naapuri, eli valitsemalla jokin siirto $\Delta x \in M$. Naapuri edustaa tässä nyt pientä perturbaatiota Metropolis-algoritmissa. Kun merkitään $\Delta C_{t,t+1} = C(x^{t+1}) - C(x^t)$, silloin todennäköisyys, että siirrytään pisteeseen x^{t+1} on 1, jos $\Delta C_{t,t+1} \leq 0$, ja $exp(-\frac{\Delta C_{t,t+1}}{c})$, jos $\Delta C_{t,t+1} > 0$ (Metropolis-kriteeri). On siis nollasta poikkeava todennäköisyys, että jatkettaisiin pisteeseen, joka antaa huonomman kohdefunktion arvon kuin nykyinen piste. Askelia toistetaan, kunnes saavutetaan tasapainotila, eli kun otettujen askelien todennäköisyysjakauma lähestyy Boltzmannin jakaumaa. Se on nyt

$$P(\text{ollaan pisteessä } x_t) = q_t(c) = \frac{1}{Q(c)} \cdot exp(-\frac{C(t)}{c}), \tag{1}$$

missä Q(c) on kontrolliparametristä c riippuva normalisoitu vakio, joka on yhtä suuri aikaisemmin mainitun partitiofunktion kanssa, ja $C(t) = C(x^t)$ on kohdefunktion arvo pisteessä x^t. Kontrolliparametriä pienennetään aina kun tasapainotila on saavutettu kaavan 1 mukaan. Algoritmin suoritus lopetetaan jollain pienellä kontrolliparametrin c arvolla, käytännössä kun huonontavia pisteitä ei enää sallita. Viimeiseksi saatu piste on tehtävän ratkaisu.

Kuten parantavassa haussa, meillä on nyt yleinen sovellettavissa oleva approksimointialgoritmi. Kun on määritelty kustannusfunktio, pisteet ja naapurustorakenne, kombinatorinen optimointiongelma voidaan ratkaista simuloitu jäähdytys -algoritmin mukaisesti [3].

Algoritmi 4.1. Simuloitu jäähdytys

Askel 0. Valitaan jokin sallittu aloituspiste x^0, iteraatioiden maksimilukumäärä t_{max} ja tarpeeksi suuri kontrolliparametrin alkuarvo $c > 0$. Asetetaan aloituspiste parhaaksi ratkaisuksi $\hat{x} = x^0$, iteraatioindeksiksi $t = 0$.

Askel 1. Valitaan jokin sallittu siirto $\Delta x \in M$ ja lasketaan uusi kohdefunktion arvo $x^t + \Delta x^{t+1}$. Jos pisteen x^t ympäristöstä ei löydy yhtään sallittua siirtoa $\Delta x \in M$ sen naapuriin tai iteraatiokierrokset ovat saavuttaneet maksimin $t = t_{max}$, lopetetaan algoritmi, jolloin tehtävän ratkaisu on $\hat{x}$.

Askel 2. Lasketaan kohdefunktion arvon muutos ΔC, kun siirrytään pisteestä x^t pisteeseen x^{t+1}. Jos muutos $\Delta C > 0$ (maksimoitaessa), niin asetetaan uudeksi pisteeksi $x^{t+1} = x^t + \Delta x^{t+1}$. Jos $\Delta C <= 0$, hyväksytään siirto todennäköisyydellä $exp(\Delta C/c)$ ja asetetaan uudeksi pisteeksi $x^{t+1} = x^t + \Delta x^{t+1}$. Jos siirtoa ei hyväksytä, siirrytään askeleeseen 1.

Askel 3. Verrataan parasta ratkaisua $\hat{x}$ uuteen ratkaisuun x^{t+1}; jos kohdefunktio saa paremman arvon pisteessä x^{t+1}, niin asetetaan $\hat{x} = x^{t+1}$.

Askel 4. Jos on suoritettu riittävä määrä iteraatioita viimeisestä kontrolliparametrin laskusta, lasketaan kontrolliparametrin c arvoa.

Askel 5. Päivitetään iteraatiolaskuri $t = t + 1$ ja siirrytään askeleeseen 1.

Algoritmin kontrolliparametri c viittaa lämpötilaan (joissain lähteissä puhutaankin vain lämpötilasta, ja sen muutoksista). Jos parametri c on suuri, niin eksponenttitermi $exp(\Delta C/c)$ lähestyy nollaa, jolloin todennäköisyys hyväksyä huonontava askel lähestyy arvoa $e^0 = 1$. Siis mitä suurempi parametri c arvo on, sitä suuremmalla todennäköisyydellä huonontava askel hyväksytään. Jos taas parametri c asetetaan nollaksi, vastaa algoritmi silloin lähes parantavaa hakua, sillä erolla, että parantavassa haussa nykyisen pisteen naapurin valinta ei ole välttämättä satunnaista. Toisin sanoen simuloitu jäähdytys on parantavan haun laajennus, joka hyväksyy nollasta poikkeavalla, vähitellen laskevalla, todennäköisyydellä huonontavat askeleet. Ei ole yksiselitteistä, kumpi algoritmeista, parantava haku vai simuloitu jäähdytys, antaa paremman approksimaation esillä olevaan ongelmaan. Sopivalla toteutuksella simuloitu jäähdytys parantavan haun variaationa voi kuitenkin suuresti parantaa heuristisen vastauksen laatua. Kontrolliparametrin alkuarvo, sen laskunopeus ja siirtymien lukumäärä kussakin parametrin arvossa (lämpötilassa) ovat kaikki tärkeitä parametreja simuloidussa jäähdytyksessä, mitkä vaikuttavat algoritmin nopeuteen sekä vastauksen laatuun. Palataan näiden parametrien tarkempaan tarkasteluun myöhemmin.

4.1 Markov-ketjut simuloidussa jäähdytyksessä

Sekä parantava haku, että simuloitu jäähdytys konvergoivat asymptoottisesti kohti globaalia optimiarvoa. Parantavassa haussa konvergenssi saavutetaan kun $N \to \infty$, missä N on alkupisteiden x^0 lukumäärä. Tarkastellaan simuloidun jäähdytyksen konvergenssia Markov-ketjujen avulla.

Annetulla naapurustorakenteella simuloitu jäähdytys -algoritmi yrittää periaatteessa jatkuvasti siirtyä tietystä pisteestä eli *konfiguraatiosta* sen naapuriin. Matemaattisesti tämä voidaan kuvata *Markov-ketjun* avulla siirtymien jonona, jossa jokainen siirtymä riippuu vain edellisestä siirtymästä. Toisin sanoen siirtymä pisteestä x^t sen naapuriin riippuu ainoastaan edellisen (eli nykyisen) pisteen x^t ja siirtymän vaikutuksesta kohdefunktioon, mutta ei aikaisemmista pisteistä $x^0, x^1, ..., x^{t-1}$.

Markov-ketju esitetään ehdollisten todennäköisyyksien joukkoina $P_{ij}(k-1, k)$ jokaiselle pisteparille (i, j). Simuloidussa jäähdytyksessä $P_{ij}(k-1, k)$ on siis todennäköisyys sille, että k:s siirtymä on pisteeseen j, kun k-1:s siirtymä on pisteeseen i. Merkitään muuttujalla $a_i(k)$ todennäköisyyttä, että k:s

siirtymä on pisteeseen i, jolloin $a_i(k)$ saadaan ratkaisemalla rekursio

$$a_i(k) = \sum_{l=1}^{R} a_l(k-1) \cdot P_{li}(k-1,k), \quad k = 1, 2, ...,$$

missä summa otetaan kaikkien mahdollisten pisteiden yli, siis R on ongelman kaikkien konfiguraatioiden joukko. Kun merkitään satunnaismuuttujalla $\mathbf{X}(k)$ k:nnen siirtymän tulosta, saadaan

$$P_{ij}(k-1,k) = \mathbb{P}(\mathbf{X}(k) = j | \mathbf{X}(k-1) = i)$$

ja

$$a_i(k) = \mathbb{P}(\mathbf{X}(k) = i).$$

Nyt voidaan esittää ehto, *Markov-ehto*, jolloin prosessia voidaan kutsua Markov-ketjuksi:

$$\mathbb{P}(\mathbf{X}(k) = l_k | \mathbf{X}(k-1) = l_{k-1}, ..., \mathbf{X}(0) = l_0) = \mathbb{P}(\mathbf{X}(k) = l_k | \mathbf{X}(k-1) = l_{k-1})$$

kaikilla $k \geq 1$ ja $l_n \in R, k \geq n \geq 0$. Jos ehdollinen todennäköisyys ei riipu siirtymästä k, kutsutaan Markov-ketjua *homogeeniseksi*, muulloin *epähomogeeniseksi*. Todennäköisyyttä $P_{ij}(k-1,k)$ kutsutaan *siirtymätodennäköisyydeksi* ja $R \times R$-matriisia $P(k-1,k)$ *siirtymämatriisiksi*, missä R on optimointiongelmassa esiintyvien kaikkien mahdollisten konfiguraatioiden (pisteiden) joukko. Merkitään muuttujalla R_i pisteiden i naapureiden joukkoa. Oletetaan nyt, että $i \notin R_i$.

Siirtymätodennäköisyydet riippuvat myös kontrolliparametrin c arvosta, lämpötilan analogiasta fysikaalisessa jäähdytysprosessissa. Jos parametri c pidetään vakiona, on silloin syntyvä Markov-ketju homogeeninen ja siirtymämatriisi $P = P(c)$ määritellään

$$P_{ij}(c) = \begin{cases} G_{ij}(c)A_{ij}(c) & \forall j \neq i \\ 1 - \sum_{l=1, l\neq i}^{|R|} G_{il}(c)A_{il}(c) & j = i. \end{cases} \tag{2}$$

Jokainen siirtymätodennäköisyys on määritelty kahden ehdollisen todennäköisyyden tulona: *generointitodennäköisyyden* $G_{ij}(c)$ (todennäköisyys että pisteestä i generoidaan uusi piste j), ja *hyväksymistodennäköisyyden* $A_{ij}(c)$ (pisteen j hyväksymistodennäköisyys, kun on se ollaan generoitu pisteestä i) avulla. Matriiseja $G(c)$ ja $A(c)$ kutsutaan vastaavasti *generointi- ja hyväksymismatriiseiksi*. Kaavan 2 nojalla matriisi $P(c)$ on *stokastinen matriisi*, jolloin $\forall i : \sum_j P_{ij}(c) = 1$. Määrittelemällä siirtymämatriisin $P(c)$ yhtälön 2 mukaisesti, sivuamme hieman alkuperäistä määritelmää, jossa matriisille $G(c)$ annettiin arvoja tasaisesta jakaumasta (nykyisestä pisteestä i valitaan

satunnaisesti jokin sen naapuri j) ja matriisille $A(c)$ arvot saatiin Metropolis-kriteerin avulla. Tästä eteenpäin, ellei toisin mainittu, viitataan siirtymätodennäköisyyksillä kaavan 2 määrittelemään muotoon.

Simuloitu jäähdytys -algoritmin sovelluksesta riippuen voidaan generointitodennäköisyyksinä käyttää yksinkertaista todennäköisyyttä, missä pisteestä i siirrytään sen naapuriin todennäköisyydellä $1/|R_i|$, missä $|R_i|$ on joukon R_i alkioiden lukumäärä. Joissain tehtävissä on kuitenkin suotavaa antaa tietyille naapureille suurempi generointitodennäköisyys. Merkitään tällaista todennäköisyyttä

$$\frac{g(i,j)}{g(i)}, \tag{3}$$

missä $g(i,j)$ antaa painon jokaiselle pisteen i naapurille, ja $g(i)$ on normalisoitu funktio, joka toteuttaa yhtälön

$$\frac{1}{g(i)} \sum_{j \in R_i} g(i,j) = 1.$$

Kuten aiemmin on todettu, kontrolliparametrin c arvo laskee algoritmin edetessä. Tämän seurauksena algoritmille saadaan kaksi erilaista muotoilua. *Homogeenisessä algoritmissa* tapahtumat kuvataan homogeenisina Markov-ketjujen sarjana. Siinä jokainen Markov-ketju on generoitu vakiolla c, jolloin parametrin c arvoa on laskettu Markov-ketjujen välissä. *Epähomogeenisessa algoritmissa* tapahtumat kuvataan yhdellä epähomogeenisellä Markov-ketjulla, jossa parametrin c arvoa lasketaan siirtojen välillä.

Simuloitu jäähdytys saavuttaa globaalin optimiarvon K siirron jälkeen, kun seuraava yhtälö pätee:

$$\mathbb{P}(\mathbf{X}(K) \in R_{opt}) = 1, \tag{4}$$

missä R_{opt} on pisteiden joukko, jossa kohdefunktio saa globaalin optimiarvon. Homogeenisessä algoritmissa yhtälö 4, toisin sanoen $\lim_{K \to \infty} \mathbb{P}(\mathbf{X}(K) \in R_{opt}) = 1$, on voimassa, kun seuraavat ehdot toteutuvat: jokainen Markov-ketju on äärettömän pituinen, matriiseilla $A(c_l)$ ja $G(c_l)$ on tietyt ominaisuudet ja $\lim_{l \to \infty} c_l = 0$, missä c_l on kontrolliparametrin arvo l:nnessä Markov-ketjussa. Epähomogeenisessä algoritmissa tulee toteutua seuraavat ehdot: matriisit $A(c_k)$ ja $G(c_k)$ toteuttavat tietyt ehdot, $\lim_{k \to \infty} c_k = 0$ ja matriisissa $A(c_k)$ konvergenssiaste sarjassa $\{c_k\}$ ei saa olla nopeampi kuin $O([\log k]^{-1})$. Tarkastellaan seuraavaksi homogeenisen ja epähomogeenisen algoritmin konvergensseja minimointitapauksessa.

4.1.1 konvergenssi homogeenisessa algoritmissa

Jotta konvergenssi saavutetaan homogeenisessä algoritmissa, on homogeenisella Markov-ketjulla oltava stationaarinen jakauma q. Jakauma q on siis pisteiden todennäköisyysjakauma äärettömän siirtymän jälkeen. Laskevilla parametrin c arvoilla jakauman $q(c)$ tulee konvergoida kohti globaaleita optimiarvoja. Jotta äärelliselle homogeeniselle Markov-ketjulle on olemassa jakauma q, pitää Markov-ketjun olla *redusoimaton* ja *jaksoton*.

Määritelmä 4.2. Markov-ketju on
1. *redusoimaton*, jos ja vain jos kaikilla pisteillä (i, j) pätee: on olemassa positiivinen todennäköisyys saavuttaa piste j pisteestä i äärellisellä määrällä siirtymiä;

$$\forall i, j \quad \exists n : 1 \leq n < \infty \wedge (P^n)_{ij} > 0,$$

2. *jaksoton*, jos ja vain jos kaikilla pisteillä $i \in R$ kokonaislukujen $n \geq 1$ suurin yhteinen tekijä, siten että

$$(P^n)_{ii} > 0,$$

on yhtä suuri kuin 1.

Simuloidussa jäähdytyksessä on riittävää olettaa, että $\forall i, j, c > 0 : A_{ij}(c) > 0$, jotta Markov-ketju on redusoimaton:

$$\forall i, j \in R \quad \exists p \geq 1 \quad \exists l_0, l_1, ..., l_p \in R(l_0 = i \wedge l_p = j) :$$

$$G_{l_k l_{k+1}}(c) > 0, \quad k = 0, 1, ..., p-1. \tag{5}$$

Redusoimaton Markov-ketju on jaksoton, jos $\forall c > 0 \quad \exists i_c \in R : P_{i_c i_c}(c) > 0$. Jaksottomuuteen riittää olettaa, että

$$\forall c > 0 \quad \exists i_c, j_c \in R : A_{i_c j_c}(c) < 1 \wedge G_{i_c j_c} > 0. \tag{6}$$

Kun hyväksymistodennäköisyydet määritellään $A_{ij}(c) = \min(1, \exp(-(C(j)-C(i))/c))$, riittää jaksottomuus-ominaisuuden takaamiseen asetukset $\forall c > 0, i_c \in R_{opt}, j_c \notin R_{opt}$ Näiden ehtojen täyttyessä matriiseilla $A(c)$ ja $G(c)$ on homogeenisella Markov-ketjulla stationaarinen jakauma. Stationaarisen jakauman olemassaolon lisäksi, sen tulee konvergoida kohti globaalia optimaalia. Yhtälöiden 5 ja 6 lisäksi matriisien $A(c)$ ja $G(c)$ tulee toteuttaa *globaalin tasapainon ehto* $\forall j \in R$:

$$\sum_{i=1, i \neq j}^{|R|} \psi(C(i), c) G_{ij}(c) A_{ij}(c) = \psi(C(j), c) \sum_{i=1, i \neq j}^{|R|} G_{ji}(c) A_{ji}(c), \tag{7}$$

missä ψ on kahden muuttujan funktio, jonka tulee täyttää ehdot:

$$\begin{array}{ll} 1. & \lim_{c\downarrow 0} \psi(\gamma, c) = \begin{cases} 0 & \text{if} \quad \gamma > 0 \\ \infty & \text{if} \quad \gamma < 0 \end{cases} \\ 2. & \dfrac{\psi(\gamma_1, c)}{\psi(\gamma_2, c)} = \psi(\gamma_1 - \gamma_2, c) \\ 3. & \forall c > 0 : \psi(0, c) = 1. \end{array}$$

Näin määrittelemällä ja yhtälöt 5, 6 ja 7 toteuttamalla saavutetaan asymptoottinen konvergenssi globaaliin minimiin $\lim_{c\downarrow 0} q(c) = \pi$, missä $q_i(c)$ voidaan kirjoittaa

$$q_i(c) = \frac{\psi(C(i), c)}{\sum_j \psi(C(j), c)}.$$

4.1.2 Konvergenssi epähomogeenisessa algoritmissa

Epähomogeenisessä algoritmissa kontrolliparametrin arvo c_k muutetaan jokaisen siirron jälkeen. Tällöin siirtymämatriisi (missä $k = 0, 1, 2, ...$) on seuraavan määritelmän mukainen.

Määritelmä 4.3.

$$P_{ij}(k-1, k) = \begin{cases} G_{ij}(c_k) A_{ij}(c_k) & \forall j \neq i \\ 1 - \sum_{l=1, l\neq i}^{|R|} G_{il}(c_k) A_{il}(c_k) & j = i. \end{cases}$$

Nyt näytettäessä konvergenssia globaaliin minimiin saadaan myös rajoituksia parametrin c_k muuttuessa seuraavaan arvoon c_{k+1}. Oletetaan, että jono $\{c_k\}$, $k = 0, 1, 2, ...$, toteuttaa ehdot:

$$\begin{array}{ll} 1. & \lim_{k\to\infty} c_k = 0 \\ 2. & c_k \geq c_{k+1}, k = 0, 1, ... \end{array} \tag{8}$$

Näin määrittelemällä vältetään parametrin c_k pysymistä vakiona siirtymien välillä, jolloin saataisiin äärellisen pituinen homogeeninen Markov-ketju. Konvergenssin osoittaminen perustuu kahteen määritelmään.

Määritelmä 4.4. Epähomogeeninen Markov-ketju on *heikosti ergodinen*, jos kaikilla $m \geq 1, \quad i, j, l \in R$:

$$\lim_{k\to\infty} (P_{il}(m, k) - P_{jl}(m, k)) = 0, \tag{9}$$

missä siirtymämatriisi $P(m, k)$ on

$$P_{il}(m, k) = \mathbb{P}(\mathbf{X}(k) = l \mid \mathbf{X}(m) = i).$$

Määritelmä 4.5. Epähomogeeninen Markov-ketju on *vahvasti ergodinen*, jos on olemassa vektori π:

$$\sum_{i=1}^{|R|} \pi_i = 1, \quad \forall i : \pi_i \geq 0, \tag{10}$$

siten että kaikilla $m \geq 1, i, j \in R$:

$$\lim_{k\to\infty} P_{ij}(m,k) = \pi_j. \tag{11}$$

Heikossa ergodisuudessa muuttujan $\mathbf{X}(k)$ riippuvuus muuttujasta $\mathbf{X}(0)$ katoaa (muistinmenetys). Vahva ergodisuus taas viittaa muuttujan $\mathbf{X}(k)$ jakauman konvergenssiin. Konvergenssi osoitetaan siis vahvan ergodisuuden avulla. Huomioi, että homogeenisessä Markov-ketjussa ei ole eroa vahvan ja heikon ergodisuuden välillä. Esitetään seuraavat lauseet heikosta ja vahvasta ergodisuudesta ilman todistusta. Heikon ergodisuuden todistus löytyy Senetan esittämänä [6] ja vahvan ergodisuuden Isaacsonin ja Madsenin esittämänä [5].

Lause 4.6. Epähomogeeninen Markov-ketju on heikosti ergodinen, jos ja vain jos on olemassa positiivisten numeroiden muodostama kasvava jono $\{k_l\}$, $l = 0, 1, 2, ...$, siten että

$$\sum_{l=0}^{\infty}(1 - \tau_1(P(k_l, k_{l+1}))) = \infty, \tag{12}$$

missä $\tau_1(P)$, $n \times n$-matriisin P *ergodisuuden kerroin*, on

$$\tau_1(P) = \frac{1}{2}\max_{i,j}\sum_{l=1}^{n}|P_{il} - P_{jl}| = 1 - \min_{i,j}\sum_{l=1}^{n}\min(P_{il}, P_{jl}). \tag{13}$$

Esimerkki 4.7. Ergodisuuden kerroin
Olkoon $R = \{a, b, c\}$ ja

$$P = \begin{bmatrix} P(a,a) & P(a,b) & P(a,c) \\ P(b,a) & P(b,b) & P(b,c) \\ P(c,a) & P(c,b) & P(c,c) \end{bmatrix} = \begin{bmatrix} P(0.2) & P(0.5) & P(0.3) \\ P(0.4) & P(0.1) & P(0.5) \\ P(0.5) & P(0.3) & P(0.2) \end{bmatrix}.$$

Nyt ergodisuuden kerroin saadaan laskettua kaavasta

$$\tau_1(P) = \frac{1}{2}\max_{i,j}\sum_{l\in R}|P_{il} - P_{jl}|, \quad i, j \in R.$$

Lasketaan ensin rivien vaihteluetäisyydet:

$$\begin{aligned} |P(a,l) - P(b,l)| &= |0.2 - 0.4| + |0.5 - 0.1| + |0.3 - 0.5| = 0.8 \\ |P(a,l) - P(c,l)| &= |0.2 - 0.5| + |0.5 - 0.3| + |0.3 - 0.2| = 0.6 \\ |P(b,l) - P(c,l)| &= |0.4 - 0.5| + |0.1 - 0.3| + |0.5 - 0.2| = 0.6 \end{aligned},$$

jolloin kertoimeksi saadaan

$$\tau_1(P) = \frac{1}{2} \cdot 0.8 = 0.4.$$

Lause 4.8. Epähomogeeninen Markov-ketju on vahvasti ergodinen, jos se on heikosti ergodinen ja jos kaikilla k on olemassa vektori $\pi(k)$, siten että $\pi(k)$ on $P(k-1,k)$ ominaisvektori ominaisarvolla 1, $\sum_{i=1}^{|R|} |\pi_i(k)| = 1$ ja

$$\sum_{k=0}^{\infty} \sum_{i=1}^{|R|} |\pi_i(k) - \pi_i(k+1)| < \infty. \tag{14}$$

Lisäksi, jos $\pi = \lim_{k\to\infty} \pi(k)$, niin π on vektori määritelmässä 4.5, jolloin siis pätee

$$\lim_{k\to\infty} P_{ij}(m,k) = \pi_j.$$

Näytetään seuraavaksi simuloidu jäähdytys -algoritmin saavuttama heikko ja vahva ergodisuus lauseiden 4.6 ja 4.8 avulla. Seurataan Mitran, Romeon ja Alberton [2] todistusta, jossa käytetään aikaepähomogeenisia Markov-ketjuja. Heikon ja vahvan ergodisuuden lisäksi he käsittelivät algoritmin käyttäytymistä rajallisessa ajassa ja konvergenssin astetta. Siis poikkeamalle lähtöpisteestä optimipisteeseen annetaan raja äärellisen iteraation jälkeen. Tämä raja ilmaisee miten *jäähdytysaikataulu* (kontrolliparametrin laskeminen) tulee tasapainottaa. Määritellään ensin raja ergodisuuden kertoimelle ja jäähdytysaikataulu, jota kutsutaan tästä eteenpäin *jäähdytysfunktioksi*, jonka tulee täyttää ehto 12.

Todistuksen pohjana on suunnattu verkko G, joka oletetaan vahvasti kytketyksi; mistä tahansa solmusta on polku mihin tahansa toiseen solmuun. Markov-ketjun Yhden askeleen siirtymä kuvaa verkossa G kaarien painoja. Siirtymät verkossa G saadaan määritelmästä 4.3 (matriiseille $G(c)$ ja $A(c)$ annetaan myöhemmin tarkempi muoto). Määritellään uusi muuttuja r. Olkoon R_{max} tuttuun tapaan lokaalien minimipisteiden joukko kohdefunktiolle,

$$R_{max} = \{i \in R \mid \forall j \in R_i : C(j) \leq C(i)\}.$$

Nyt saadaan muuttujalle r seuraava esitys:

$$r = \min_{i \in R \setminus R_{max}} \max_{j \in R} d(i,j), \tag{15}$$

missä $d(i,j)$ on minimilukumäärä siirtymiä saavuttaa piste j pisteestä i. Muuttuja r on kokonaisluku siten, että on ainakin yksi piste (mistä minimi yhtälöstä 15 on saavutettu), joka ei ole lokaali maksimi, joka voidaan saavuttaa mistä tahansa pisteestä enintään siirtymien lukumäärällä r. Merkitään pisteellä $\hat{l}$ pistettä, jossa minimi saavutetaan yhtälössä 15. Annetulla verkolla G solmua $\hat{l}$ sanotaan *verkon keskustaksi* ja muuttujaa r säteeksi. Näytetään, että r edustaa nyt vaadittavaa ylärajaa siirtymien lukumäärälle Markov-ketjussa, jotta siirtymämatriisilla P on vaadittavat elementit ainakin yhdellä sarakkeella (ainakin pistettä $\hat{l}$ vastaava arvo oltava eri kuin 0). Huomaa, että generoinnissa käytettävän yhtälön $g(i,j)$ on oltava symmetrinen:

$$g(i,j) = g(j,i) \quad \forall i,j \in R$$

Tämä tarkoittaa, että kaikilla naapureilla jokaisessa pisteessä on sama paino, mikä takaa verkon G vahvan kytkeytymisen. Tällöin myös muuttuja r on hyvin määritelty.

Oletetaan, että siirtymämatriisin P generointimatriisi ja hyväksymismatriisi ovat määritelmästä 4.3 muotoa:

$$G_{ij}(c_k) = \frac{g(i,j)}{g(i)} \text{ ja } A_{i,j}(c_k) = \min\left[1, \exp\left(\frac{-(C(j)-C(i))}{c_k}\right)\right],$$

jos $j \in R_i$. Otetaan käyttöön *Lipschitz-vakiota* muistuttava vakio rajoittamaan hyväksymistodennäköisyyksiä:

$$L = \max_{i \in R} \max_{j \in R_i} |C(j) - C(i)|.$$

Määritellään generaatiofunktiolle alaraja käyttäen yhtälön 3 määritelmää hyväksi:

$$w = \min_{i \in R} \min_{j \in R_i} \frac{g(i,j)}{g(i)} > 0.$$

Jos i ja j ovat naapureita $j \in R_i$, saadaan siirtymämatriisille raja

$$P_{ij}(c_k) = \frac{g(i,j)}{g(i)} \min\left[1, \exp\left(\frac{-(C(j)-C(i))}{c_k}\right)\right] \geq w \exp\left(\frac{-L}{c_k}\right), \quad k = 0, 1, ...$$

Nyt matriisin diagonaalialkiot $P_{ii}(c_k)$, $i \in R \setminus R_{max}$ voivat olla alussa pieniä, mutta ne ovat monotonisia, kasvavia kasvavalla aikaparametrilla k. Siirtymän todennäköisyys pisteestä i sen naapuriin, joka laskee kohdefunktion arvoa, on vakio, kun taas todennäköisyys siirtyä suuremman kohdefunktion arvon pisteeseen on monotonisesti laskeva kasvavalla parametrilla k. Tällöin on olemassa $l_0 < \infty$ siten, että kaikilla $i \in R \setminus R_{max}$

$$P_{ii}(c_k) \geq w \exp\left(\frac{-L}{c_k}\right), \quad k \geq (l_0 - 1)r,$$

kun vasen puoli on monotonisesti kasvava ja oikea puoli monotonisesti laskeva kasvavalla parametrilla k. Nyt saadaan todennäköisyydelle $P_{i\hat{l}}(k-r,k)$ alaraja kaikilla $i \in R$ ja $k \geq l_0 r$:

$$P_{i\hat{l}}(k-r,k) \geq \prod_{n=k-r}^{k-1} \left(w \exp\left(\frac{-L}{c_n}\right)\right) \geq w^r \exp\left(\frac{-rL}{c_{k-1}}\right).$$

Ergodisuuden kertoimelle määritelmästä 13 pätee nyt

$$\begin{aligned} &\tau_1(P(lr-r,lr)) = 1 - \min_{i,j}(\textstyle\sum_{l=1,l\neq\hat{l}}^{n} \min(P_{il},P_{jl}) + \min\{P_{i\hat{l}},P_{j\hat{l}}\}) \\ &\leq 1 - \min_{i,j}(\min\{P_{i\hat{l}},P_{j\hat{l}}\}) \leq 1 - w^r \exp\left(\frac{-rL}{c_{lr-1}}\right), \quad l \geq l_0, \end{aligned} \tag{16}$$

ja silloin määritelmän 4.4 avulla Markov-ketjulle simuloidussa jäähdytyksessä saadaan ehto; se on heikosti ergodinen jos

$$\sum_{l=l_0}^{\infty} \exp\left(-\frac{rL}{c_{lr-1}}\right) = \infty.$$

Huomaa, että jonon $\{c_k\}$ oletetaan olevan monotonisesti laskeva ja $\lim_{k\to\infty} c_k = 0$. Kontrolliparametrin c_k riippuvuutta parametrista k ei siis ole vielä määritelty. Seuraavaksi määritellään jäähdytysaikataulu, eli kontrolliparametrin c_k jäähdytysfunktio, joka takaa Markov-ketjun olevan heikosti ergodinen.

Lause 4.9. jos Simuloidussa jäähdytyksessä Markov-ketjun jäähdytysfunktio on muotoa:

$$c_k = \frac{\gamma}{\log(k+k_0+1)}, \quad k = 0,1,2,..., \tag{17}$$

missä k_0 on mikä tahansa parametri $1 \leq k_0 < \infty$, niin se on heikosti ergodinen, jos $\gamma \geq rL$.

Todistus. Sijoitetaan jäähdytysfunktio c_k yhtälöön 16, jolloin saadaan:

$$\begin{aligned}
&\tau_1(P(lr-r,lr)) \leq 1 - w^r \exp\left(-\frac{rL}{c_{lr}-1}\right) \\
&= 1 - w^r \exp\left(-\frac{rL\ln(lr-1+k_0+1)}{\gamma}\right) \\
&= 1 - w^r \exp\left(-\frac{rL}{\gamma}\left(\ln(r) + \ln\left(l+\frac{k_o}{r}\right)\right)\right) \\
&= 1 - w^r \exp\left(\ln\left(r^{-\frac{rL}{\gamma}}\right) + \ln\left(\left(l+\frac{k_o}{r}\right)^{-\frac{rL}{\gamma}}\right)\right) \\
&= 1 - w^r \exp\left(\ln\left(r^{\frac{-rL}{\gamma}}\left(l+\frac{k_0}{r}\right)^{\frac{-rL}{\gamma}}\right)\right) \\
&= 1 - \frac{a}{\left(l+\frac{k_0}{r}\right)^{\mu}},
\end{aligned} \tag{18}$$

missä

$$\mu = \frac{rL}{\gamma},$$

ja

$$a = \frac{w^r}{r^{rL/\gamma}}.$$

Nyt siis pätee mille tahansa m:

$$\sum_{l=m}^{\infty}(1-\tau_1(lr-r,lr)) = \infty,$$

jos $\mu \leq 1$. $\square$

Jos jäähdytysaikataulu ei muuta lämpötilaa (kontrolliparametri pysyy vakiona), säilytetään heikko ergodisuus äärellisen monen askeleen aikana.

Homogeenisessa algoritmissa saatujen matriisien $A(c)$ ja $G(c)$ oletusten seurauksena, mitkä takaavat stationaarisen jakauman olemassaolon, on olemassa yksikkövektori $q(c_k)$ matriisille $P(k-1,k)$ jokaiselle $k \geq 0$. Lisäksi homogeenisen algoritmin konvergenssioletusten voimassa ollessa saadaan $\lim_{k\to\infty} q(c_k) = \pi$, kun $\lim_{k\to\infty} c_k = 0$. Simuloidun jäähdytyksen Markov-ketjun vahva ergodisuus saadaan todistettua nyt lauseesta 4.8. Täytyy siis todistaa vain seuraava väite; annetun jäähdytysfunktion, joka toteuttaa yhtälön 17, vastaavat stationaariset todennäköisyydet toteuttavat:

$$\sum_{k=0}^{\infty}\sum_{i=1}^{|R|}|\pi_i(k)-\pi_i(k+1)| \leq 2(\tilde{k}+1) < \infty, \tag{19}$$

missä $\tilde{k} = \max_{i \notin R_{opt}} \hat{k}_i$, ja $\hat{k}_i$, $0 \leq \hat{k}_i < \infty$, toteuttaa

$$\pi_i(c_{k+1}) - \pi_i(c_k) > 0, \quad 0 \leq k \leq \hat{k}_i - 1$$

$$\pi_i(c_{k+1}) - \pi_i(c_k) < 0, \quad k \geq \hat{k}_i$$

kaikilla $i \notin R_{opt}$. Merkitään nyt lyhyemmin $\pi(c_k) = \pi(k)$.

Näytetään, että yhtälö 19 pitää. Koska

$$\pi_i(c_{k+1}) - \pi_i(c_k) > 0, \quad \forall k \geq 0, \text{ jokaisella } i \in R_{opt}$$

ja

$$\pi_i(c_{k+1}) - \pi_i(c_k) < 0, \quad \forall k \geq \tilde{k}, \tilde{k} < \infty, \text{ jokaisella } i \notin R_{opt}$$

(ehdot todistettu lähdemateriaalissa [2] sivulla 755) saadaan

$$\sum_{i=1}^{|R|} |\pi_i(k) - \pi_i(k+1)| = \sum_{i \in R_{opt}} (\pi_i(k+1) - \pi_i(m)) - \sum_{i \notin R_{opt}} (\pi_i(k+1) - \pi_i(k)),$$

kun $k \geq \tilde{k}$.Kun määritelmästä 4.5

$$\sum_{i \in R_{opt}} \pi_i(k) + \sum_{i \notin R_{opt}} \pi_i(k) = \sum_{i \in R} \pi_i(k) = 1, \quad \forall m \geq 0,$$

saadaan

$$\begin{aligned} &\textstyle\sum_{i=1}^{|R|} |\pi_i(k) - \pi_i(k+1)| \\ &= \textstyle\sum_{i \in R_{opt}} \pi_i(k+1) + \sum_{i \in R_{opt}} \pi_i(k+1) - 1 - \sum_{i \in R_{opt}} \pi_i(k) - \sum_{i \in R_{opt}} \pi_i(k) + 1 \\ &= 2\left(\textstyle\sum_{i \in R_{opt}} \pi_i(k+1) - \sum_{i \in R_{opt}} \pi_i(k)\right), \quad k \geq \tilde{k} \end{aligned}$$

Nyt pätee siis

$$\sum_{k=\tilde{k}}^{\infty} \sum_{i \in R} |\pi_i(k) - \pi_i(k+1)| \leq 2. \quad \square$$

Merkitsemällä $\pi(k) = q(c_k)$, vahva ergodisuus voidaan yleisesti osoittaa, näyttämällä seuraavat kohdat todeksi :
1. Markov-ketju on heikosti ergodinen
2. $q(c_k), k = 0, 1, 2, ...,$ toteuttaa yhtälön 14.
Käyttämällä tietoja $\lim_{k \to \infty} \mathbb{P}(\mathbf{X}(k) = j) = \pi_j$ ja
$\pi_j = \begin{cases} |R_{opt}|^{-1} & \text{jos } j \in R_{opt} \\ 0 & \text{muulloin} \end{cases}$, saadaan

$$\lim_{k \to \infty} \mathbb{P}(\mathbf{X}(k) \in R_{opt}) = 1.$$

Käytännössä jäähdytysfunktio 17 ehdolla $\gamma \geq rL$ antaa vahvasti ergodisen Markov-ketjun, jossa pätee yhtälö 14.

Kuten näytettiin simuloitu jäähdytys - algoritmi tietynlaisella muotoilulla konvergoi todennäköisyydellä 1 globaaliin optimaaliin kahdella mahdollisella tavalla:
1. Jokaisella kontrolliparametrin arvolla c_k generoidaan ääretön määrä siirtymiä ja $\lim_{k\to\infty} c_k = 0$ (homogeeninen algoritmi)
2. jokaisella parametrin c_k arvolla generoidaan yksi siirtymä ja c_k lähenee nollaa nopeimmillaan $O([\log k]^{-1})$ (epähomogeeninen algoritmi).
Kaikilla algoritmin toteutuksilla asymptoottista konvergenssia voidaan vain approksimoida. Saadaan siis approksimointialgoritmi. Esimerkiksi siirtymien lukumäärä jokaisella parametrin c_k arvolla pitäisi olla ääretön homogeenisessä algoritmissa ja $\lim_{k\to\infty} c_k = 0$, jotka voidaan approksimoida vain äärellisellä lukumäärällä parametrin c_k arvoja. Approksimaatioalgoritmi ei siis enää takaa päätymistä globaaliin optimiarvoon.

4.2 Jäähdytysohjelman valinta

Edellisen luvun konvergenssituloksista ei ole juurikaan hyötyä approksimoinnissa. Markov-ketjun pituudelle L homogeenisessa tapauksessa on esitetty seuraava riittävä ehto:

$$L > K(|R|^2 + 3|R| + 3),$$

missä K on verrannollinen luvun $\log \epsilon$ kanssa (ϵ on pieni luku). Luku $|R|$ on yleensä eksponentiaalinen ongelman koosta, jonka pituinen Markov-ketjun tulisi vähintään olla. Tämä on tietysti ei-toivottu ominaisuus. Kontrolliparametrin muuttamiselle on myös saatu muoto

$$c_{k+1} = \frac{\Gamma}{\log(\exp\left(\frac{\Gamma}{c_k}\right) + 1)},$$

missä tarvittaisiin tieto vakion Γ arvosta. Sen määrittäminen on kuitenkin yleensä hyvin vaikeaa. Pienessä tehtävässä sen voi joskus selvittää täydellisellä luetteloinnilla, mutta isommassa tehtävässä sekään ei tule kysymykseen.

On mahdollista yhdistää homogeeninen ja epähomogeeninen algoritmi; kontrolliparametri c_k pidetään vakiona siirtymien välillä, ja pidetään huoli ettei parametri laske nopeammin kuin $O([\log k]^{-1})$. Tällainen lähestymistapa voisi olla seuraava:

$$c_k = \gamma_i, \quad k_i \leq k < k_{i+1},$$
$$\gamma_i = \frac{c}{\log k_i}, \quad i = 0, 1, 2...,$$

jollakin vakiolla c ja positiivisten numeroiden jonolla $\{k_i\}, i = 0, 1, 2, ...$ Tämä johtaa homogeeniseen Markov-ketjuun, jossa i:s ketju koostuu k_i siirtymästä, jolloin Markov-ketjun pituus pitäisi kasvaa aina edellisestä, jotta kontrolliparametrin arvo laskee.

Simuloitu-jäähdytys algoritmin homogeenisten Markov-ketjujen jono generoidaan yleensä laskevista kontrolliparametrin arvoista. Tällöin tulisi määrittää seuraavat parametrit:

1. kontrolliparametrin alkuarvo, c_0
2. kontrolliparametrin loppuarvo, c_f (lopetuskriteeri)
3. Markov-ketjun pituus
4. Sääntö nykyisen kontrolliparametrin c_k muuttamisesta seuraavaan c_{k+1}.

Näiden parametrien valintaa kutsutaan *jäähdytysohjelmaksi.*

Käsittelemme jäähdytysohjelmaa aikaisemmin määrittelemien hyväksymis- ja generointimatriisien

$$A_{ij}(c) = \min(1, \exp(-(C(j) - c(i))/c))$$

$$G_{ij} = \begin{cases} \dfrac{1}{|R_i|}, & \text{if } j \in R_i \\ 0, & \text{muulloin} \end{cases}$$

pohjalta. On huomattu, että erilaisten hyväksymismatriisien käyttö ei muuta merkittävästi algoritmin löytämää vastausta. Monien jäähdytysohjelmien perustana on kvasi-tasapaino. Jos L_k on k:nnen Markov-ketjun pituus, silloin jäähdytysalgoritmin sanotaan olevan kvasi-tasapainossa parametrilla c_k (k:nnen kontrolliparametrin arvo), jos $a(L_k, c_k)$ on tarpeeksi lähellä jakaumaa $q(c_k)$, missä $a(L, c)$ on todennäköisyysjakauma siirtymien L jälkeen ja $q(c)$ homogeenisen Markov-ketjun stationaarinen jakauma kontrolliparametrin arvolla c. Tämän 'tarpeeksi lähellä' -ilmaisun erilaiset tarkat määritelmät on yksi merkittävä tapa erotella jäähdytysohjelmia toisistaan. Jäähdytysohjelman todellinen rakenne perustuu usein seuraaviin väitteisiin.

1. Kontrolliparametrilla $c_k \to \infty$ stationaarinen jakauma saadaan tasajakaumasta pisteiden joukolla R. Alussa kvasi-tasapaino voidaan saavuttaa valitsemalla alkuarvo c_0, siten että käytännössä kaikki siirtymät hyväksytään. Siinä tapauksessa kaikki pisteet esiintyvät samalla todennäköisyydellä, mikä vastaa edellä mainittua tasajakaumaa ja siten $q(\infty)$.

2. Lopetuskriteeri perustuu yleensä väitteeseen, jossa algoritmin suoritus voidaan lopettaa, kun seuraavalla pisteellä kohdefunktion arvon parannus on pieni

3. K:nnen Markov-ketjun pituus L_k ja siirtymissääntö parametrista c_k seuraavaan c_{k+1} on vahvasti sidoksissa kvasi-tasapainon käsitteeseen. L_k on mää-

ritelty määrittelemällä tarkemmin '$a(L_k, c_k)$ on lähellä $q(c_k)$' -merkitys. Siirtymissäännössä ($c_k \to c_{k+1}$) on intuitiivisesti selvää, että suuri lasku parametrin c_k arvosta tarkoittaa enemmän siirtymiä parametrilla c_{k+1}, jotta palautetaan kvasi-tasapaino. Voidaan siis todeta: mitä suurempi lasku on parametrilla c_k, sitä suurempi ero on välillä $q(c_k)$ ja $q(c_{k+1})$, ja sitä kauemmin kestää saavuttaa kvasi-tasapaino parametrissa c_{k+1}. Näin ollen parametrin c_k jyrkän laskun ja pituuden L_k pienten arvojen välillä on vaihtokauppa. Voidaan valita parametrille c_k pieni muutos, jolloin vältetään pitkiä ketjuja, tai valitaan ketjuille pidempiä pituuksia L_k, jolloin parametria c_k voidaan laskea rajummin.

Kontrolliparametrin alkuarvo

Alkuarvo c_0 määritellään, siten, että kaikki siirtymät ovat sallittuja, siis $\exp\left(\frac{-\Delta C_{ij}}{c_0}\right) \sim 1$ lähes kaikille i ja j. On esitetty seuraava empiirinen sääntö: valitse suuri arvo parametrille c_0 ja suorita useita siirtymiä. Jos *hyväksymissuhde* χ (hyväksyttyjen siirtymien lukumäärä jaettuna kaikkien kokeiltujen siirtymien lukumäärällä), on pienempi kuin χ_0 (usein $\chi_0 = 0.8$), niin kaksinkertaista parametrin nykyinen arvo c_0. Jatka niin kauan, kunnes saatu hyväksymissuhde ylittää arvon χ_0. Sääntöä on myös tarkennettu seuraavasti: määritä alkuarvo c_0 laskemalla kohdefunktion keskimääräinen nousu $\overline{\Delta C}^{(+)}$ usealle satunnaiselle siirrolle ja ratkaise c_0 yhtälöstä:

$$\chi_0 = \exp\left(\frac{-\overline{\Delta C}^{(+)}}{c_0}\right).$$

Ratkaisemalla c_0 saadaan

$$c_0 = \frac{\overline{\Delta C}^{(+)}}{\ln(\chi_0^{-1})}.$$

Kontrolliparametrin loppuarvo

Kontrolliparametrin loppuarvo on määritelty joko kiinnittämällä jokin maksimilukumäärä parametrin c_k arvoille, jolloin algoritmin suoritus lopetetaan, tai lopettamalla suoritus, kun viimeiset peräkkäiset Markov-ketjut ovat yhtä pitkiä.

Markov-ketjun pituus

Yksinkertaisin valinta k:nnen Markov-ketjun pituudelle L_k on polynomisesti ongelman koosta riippuva arvo. Silloin L_k ei riipu parametrista k. Toinen tarkempi vaihtoehto Markov-ketjun pituudelle perustuu hyväksyttävien siirtojen minimimäärään η_{min} jokaisella kontrolliparametrin c_k arvolla. Siis $L_k \geq \eta_{min}$. Kuitenkin kun c_k lähestyy nollaa, siirtymät hyväksytään vähenevällä todennäköisyydellä, jolloin lopulta saavutetaan $L_k \to \infty$. Näin ollen

pituus L_k on käytännössä rajoitettava jollakin vakiolla $\overline{L}$, jotta vältetään erittäin pitkät Markov-ketjut pienillä parametrin c_k arvoilla. Esimerkiksi $\overline{L} = n$, missä n on muuttujien lukumäärä ongelmassa tai $\overline{L} = m \cdot |R_i|$, missä m on jokin kerroin.

L_k voidaan määritellä rajoittamalla se myös hylättyjen siirtojen minimimäärällä. Näin määrittelemällä Markov-ketjujen pituus vähitellen pienenee algoritmin edetessä. Ketjujen pituudelle tapahtuu siis päinvastoin kuin aikaisemmassa määritelmässä, jossa niiden pituus kasvoi.

Lähimpänä kvasi-tasapainon konseptia on määritellä ketjun pituus seuraavasti: määritellään *jakso* siirtymien lukumääräksi kiinnitetyllä hyväksymisien lukumäärällä ja jakson kustannus jakson viimeisen konfiguraation (pisteen) kustannuksen(kohdefunktion) arvona. Kun jakson kustannus on tietyllä välillä edeltävän jakson kustannuksesta, Markov-ketju päätetään. Siis Markov-ketjun päättäminen ja sen pituus on sidoksissa kohdefunktion arvon vaihteluihin, jotka on havaittu kyseiseltä ketjulta.

Kontrolliparametrin arvon laskeminen

Kuten aikaisemmin todettiin, parametrin laskeminen pienellä muutoksella voidaan Markov-ketjujen pituus pitää lyhyempänä, kuin jos kontrolliparametrin muutos olisi iso. Usein käytetty sääntö on

$$c_{k+1} = \alpha \cdot c_k, \quad k = 0, 1, 2, ...,$$

missä α on vakio lähellä arvoa 1, mutta $\alpha < 1$. Usein käytetty arvo vakiolle on $0.5 \leq \alpha \leq 0.95$. Toinen käytetty määritelmä tämän suhteen $\frac{c_{k+1}}{c_k}$ vakiona pitämisen lisäksi on pitää peräkkäisten parametrien arvojen välinen erotus vakiona. Esimerkiksi jakamalla väli $[0, c_0]$ K osaväliin ja c_k, $k = 1, ..., K$, on valittu

$$c_k = \frac{K - k}{K} \cdot c_0, \quad k = 1, ..., K.$$

Tarkastellaan seuraavaksi yksinkertaisia jäähdytysohjelmia, jotka perustuvat empiirisesti löydettyihin sääntöihin eikä niinkään teoreettiseen tarkasteluun.

4.3 Esimerkki 1: pieni aineisto

Tarkastellaan kauppaesimerkkiä käyttäen simuloitua jäähdytystä homogeenisilla Markov-ketjuilla. Nyt aineisto voidaan esittää $m \times n$-binäärimatriisin K avulla (liite A). Sarakkeet kuvastavat eri tuoteryhmiä ja rivit asiakkaita. Merkitään

$$K_{ij} = \begin{cases} 1, \text{ jos asiakas } i \text{ on ostanut tuotteen } j \\ 0, \text{ muulloin.} \end{cases}$$

Jotta simuloitua jäähdytystä voidaan käyttää, tulee meidän ensin määritellä naapurustorakenne. Parantavan haun naapurustoa, missä siirtymäjoukkona M käytettiin parittaisia vaihtoja (k, l) (k jokin valittu tuoteryhmä ja l jokin ei-valittu tuoteryhmä), voitaisiin käyttää myös simuloidussa jäähdytyksessä. Jos naapurustoa halutaan pienentää, voidaan siirtymäjoukkoa M pienentää rajoittamalla vaihdettavien tuoteryhmien l määrää. Tämä voi olla tarpeellista, jos tuoteryhmiä on paljon (>1000). Tätä rajoitusta varten muodostetaan saadusta ostoaineistosta frekvenssimatriisi, johon merkitään kunkin tuoteryhmän ostofrekvenssit. Tämä matriisi järjestetään suurimmasta tuoteryhmän frekvenssistä pienimpään. Suurin penetraatioaste saavutetaan todennäköisimmin tuoteryhmillä, joilla on suuri frekvenssi. Liitteessä B on kauppaesimerkin järjestetty frekvenssimatriisi. Nyt aloituspiste x^0 saadaan tuoteryhmistä, joissa on suurin frekvenssi. Tuoteryhmiä haluttiin 5, joten aloituspiste $x^0 = \{0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1\}$. Jotta siirtymäjoukkoa M saadaan supistettua, asetetaan parittaisen vaihdon parametrille l rajoitus: l on oltava jokin ei-valittu tuoteryhmä ($x_j = 0$), joka on jollakin maksimietäisyydellä frekvenssimatriisissa valitusta tuoteryhmästä k. Käytetään maksimietäisyytenä $v * 2 = 10$ (haluttu tuoteryhmien lukumäärä x 2). Näin ollaan määritelty pisteen x naapurusto parittaisten vaihtojen ja frekvenssimatriisin avulla. Kutsutaan parittaisen vaihdon (k, l) tuoteryhmää l tuoteryhmän k naapuriksi. Nyt siis tuoteryhmä l voi olla maksimissaan etäisyydellä 10 tuoteryhmästä k frekvenssitaulussa. Esimerkiksi aiemmin annetulla aloituspisteellä $x^0 = \{0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1\}$ tuoteryhmän 5 sallittuja naapureita ovat tuoteryhmät {18,6,16,19,15,14,17,1}. Huomaa, että jo valitut tuotteet {3,4,8,20} eivät ole tuotteen 5 sallittuja naapureita, sillä tällainen vaihto ei muuttaisi kohdefunktion arvoa. Sallittu siirto on nyt siirto, jossa jokin valittu tuoteryhmä k vaihdetaan sen sallittuun naapuriin l. Näin ollen pisteen x naapurusto koostuu pisteistä, joissa on suoritettu parittainen vaihto valitun tuoteryhmän ($x_j = 1$) ja sen naapurituoteryhmän välillä. Aloituspisteen sallittu naapuri voisi olla esimerkiksi piste {0,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1} tai {0,0,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,1}. Yhteensä sillä on kuitenkin 40 eri sallittua naapuripistettä (huomaa ero parantavaan hakuun, jossa pisteellä oli 75 naapuria).

Ennen algoritmia tulee myös määrittää kontrolliparametrin alku- ja loppuarvo ja sen jäähdytysaikataulu sekä Markov-ketjun pituus. Empiirisellä säännöllä kontrolliparametrin alkuarvoksi saadaan $c_0 = 8$, kun otetaan sata siirtymäyritystä, $\chi_0 = 0.8$ ja alkuarvoksi kokeillaan ensin $c_0 = 2$. Tarkennetulla säännöllä saadaan laskettua alkuarvoksi

$$c_0 = \frac{\overline{\Delta C}^{(+)}}{\ln(\chi_0^{-1})} = \frac{5.63}{\ln(1/0.8)} \approx 25.$$

Käytetään näistä kahdesta vaihtoehdosta ensimmäistä $c_0 = 8$. Päätetään, että kontrolliparametrilla voi olla maksimissaan 20 eri arvoa. Päätetään myös, että Markov-ketjun pituus on maksimissaan $\overline{L} = 200$, koska asiakkaita on nyt 100 kappaletta, ja hyväksyttävien siirtojen minimimäärä $\eta_{min} = 10$. Nyt siis $10 = \eta_{min} \leq L_k \leq \overline{L} = 200$. Käytetään yleistä jäähdytystä $c_{k+1} = \alpha \cdot c_k$, missä $\alpha = 0.8$.

Edellä esitettyjen parametrien valinnoilla simuloitu jäähdytys algoritmi päätyy lokaaliin optimiin kohdefunktion arvolla 96. Hyväksyttyjä siirtoja oli 182 kappaletta ja hylättyjä 1469 kappaletta. Yhteensä kohdefunktio laskettiin siis 1651. Koska algoritmissa on satunnaisuutta, vaihtelee luvut hieman eri ajokerroilla. Kymmenen ajokerran jälkeen saadaan seuraavat tulokset: hyväksyttyjen siirtojen lukumäärän vaihteluväli on [177,195] keskiarvolla 187,7 ja hylättyjen siirtymien lukumäärän vaihteluväli on [1246,1600] keskiarvolla 1475,5. Kaikilla kerroilla päädyttiin lokaaliin optimiin kohdefunktion arvolla 96. Täydellisellä luetteloinnilla nähdään, että se on myös globaali optimi. Tehtävällä on 4 globaalia optimia tuoteryhmien valinnoilla {1,3,5,19,20},{1,3,4,19,20},{3,8,14,19,20} tai {1,3,14,19,20}.

Jos naapurustorakennetta muutetaan siten, että suuremmalla todennäköisyydellä 0,7 valitaan parittaisessa vaihdossa tuotteen naapuri pienemmältä etäisyydeltä $v = 5$ ja pienemmällä todennäköisyydellä 0,3 naapuri valitaan etäisyydeltä $v * 2 = 10$, saadaan hyväksyttyjen siirtymien lukumäärälle keskiarvo 176,7 ja hylättyjen 1526,1. Kaikilla ajokerroilla päädyttiin optimiarvoon 96. Nyt siis hyväksyttyjen siirtojen suhde kokonaisyritysten määrään pieneni arvosta 0,1128 arvoon 0,1028.

Jos naapurustoa pienennetään alkuperäiseen, jolloin päädytään suoraan parantavaan hakuun, joka myös sallii huonontavat siirtymät simuloitu jäähdytys -algoritmin tapaan, ja aloituspisteeseen valitaan satunnaisesti mitkä tahansa 5 tuotetta saadaan keskiarvoille arvot 177 ja 1617,7. Optimiarvoon 96 päädyttiin taas kaikilla kerroilla. Kuten nähtiin, keskiarvoluvut eikä vastauksen laatu juurikaan muutu tässä esimerkissä, vaikka tehdään muutoksia naapurustorakenteessa ja aloituspisteessä.

Kun muutetaan parametrien arvoja (kontrolliparametrin alku-, loppu- ja laskuarvoja sekä Markov-ketjun pituutta) saadaan aikaan suuria muutoksia tuloksiin. Taulukoidaan eri parametreilla, naapurustorakenteella ja alkuarvoilla saatuja tuloksia. Ensimmäisessä taulukossa on esitelty algoritmin rakenteista erilaisia vaihtoehtoja. Markov-ketjun siirtymä vaihtoehto 'paras' tarkoittaa Markov-ketjun päättyessä algoritmin jatkamista seuraavaan ketjuun parhaasta edellisen Markov-ketjun alkiosta, missä saavutetaan paras kohdefunktion arvo. Eli algoritmia jatketaan kontrolliparametrin laskemisen jälkeen niillä valituilla tuoteryhmillä, missä se sai parhaan kohdefunktion arvon. Vaihtoehto 'viimeinen', tarkoittaa algoritmin jatkamista seuraavaan

Markov-ketjuun edellisen ketjun viimeisestä alkiosta (eli jatkamista valituilla tuoteryhmillä, mihin viimeksi jäätiin). Naapuruston valinnassa on kolme vaihtoehtoa, joista vaihtoehto 'normaali' määrittelee pisteen x naapuruston parittaisilla vaihdoilla, jossa valitun tuoteryhmän k naapurituote l on maksimissaan etäisyydellä $2 \cdot v = 10$ frekvenssitaulusta. Puolestaan vaihtoehto 'jaettu todennäköisyys' tarkoittaa, että todennäköisyydellä 0,7 parittaisen vaihdon valittu tuote k saa olla ei-valitusta tuotteesta l maksimietäisyydellä $v = 5$ frekvenssitaulusta ja todennäköisyydellä 0,3 maksimietäisyydellä $2 \cdot v = 10$. Valinta 'laaja' tarkoittaa, että parittainen vaihto voidaan tehdä minkä tahansa valitun ja ei-valitun tuotteen välillä. Aloituspisteen valitut tuoteryhmät voidaan valita joko täysin satunnaisesti 'satunnainen' tai suurimpien frekvenssien mukaan 'frekvenssi'.

alg-asetus	naapurusto	aloituspiste	Markov-ketjun siirtymä
1	jaettu todennäköisyys	frekvenssi	paras
2	normaali	frekvenssi	paras
3	laaja	frekvenssi	paras
4	laaja	satunnainen	paras
5	laaja	satunnainen	viimeinen
6	normaali	satunnainen	paras
7	normaali	satunnainen	viimeinen

Esitellään seuraavassa taulukossa eri parametrien arvot, joilla algoritmia ajettiin. Parametri 'ketjujen lkm' tarkoittaa Markov-ketjujen lukumäärää eli toisin sanoen kuinka monta eri kontrolliparametrin arvoa voidaan käyttää (vastaa algoritmin lopetuskriteeriä). Parametri 'Markov-max' kertoo Markov-ketjun maksimipituuden, eli kuinka monta siirtymäyritystä voidaan maksimissaan tehdä samalla kontrolliparametrin arvolla (sisältää myös hylätyt siirtymät). 'Hyv-min' puolestaan on haluttujen hyväksyttävien siirtymien lukumäärä tietyllä kontrolliparametrin arvolla. Nimestä huolimatta Markov-ketjulla voi olla maksimissaan parametrin 'Hyv-min' mukainen lukumäärä hyväksyttyjä siirtymiä, sillä parametrit 'Markov-max' ja 'Hyv-min' yhdessä rajoittavat yksittäisen Markov-ketjun pituutta. Kontrolliparametrin alkuarvo on parametri 'c_0' ja 'jääh.kerroin' vastaa kontrolliparametrin jäähdytyskerrointa α.

param-id	ketjujen lkm	Markov-max	Hyv-min	c_0	jääh.kerroin
1	20	200	10	8	0.8
2	10	100	5	8	0.8
3	10	100	7	8	0.9
4	30	80	20	8	0.8
5	60	20	4	8	0.8
6	60	20	4	25	0.8

Seuraavassa taulukossa on esitetty 10 ajokerran keskiarvot (hyväksyttyjen siirtojen lukumäärä/hylättyjen siirtymien lukumäärä) edellisten taulukoiden mukaisten asetusten nojalla. Parametri 'a/p' on lyhennys alg-asetus/param-id.

Siirtymien lukumäärät (hyväksytty/hylätty)						
a/p	1	2	3	4	5	6
1	176.7/1526.1	50/62.8	70/38.2	267.3/1748	81.9/971.1	94.9/892.1
2	187.7/1475.5	50/66	70 /39.4	289.1/1774.1	84.8/986.6	103.8/900
3	176.7/1546.6	50/82.9	70/51.8	262.9/1824.3	72.8/1014.4	91.3/922.1
4	177/1617.7	50/69.5	70/48.4	259.1/1823.4	70.8/1010	94.1/922.6
5	177.9/1584.3	50/65.8	70/49.5	269.8/1813.5	78.1/995.2	91.5/911.7
6	189.3/1419.9	50/75.2	70/44.4	281.6/1778.5	93.5/973.4	100.2/908.4
7	187.4/1505.4	50/61.2	70/35.8	288.5/1745.3	89/897.3	102.8/892.8

Vastaavasti saadaan lokaaleille maksimeille keskiarvot eli penetraatioasteet.

Penetraatioaste %						
a/p	1	2	3	4	5	6
1	96	95.7	95.3	96	96	96
2	96	95.8	95.3	96	96	96
3	96	95.8	95.1	96	96	96
4	96	95.2	95.1	96	96	96
5	96	95.3	94.9	96	96	96
6	96	95.8	95.3	96	96	96
7	96	94.7	94	96	96	96

Nähdään, että param-id asetuksilla 2 ja 3 päädytään todennäköisimmin lähelle globaalia optimiarvoa 96, mutta ei kuitenkaan aina, kun keskiarvo jää välille 94-95,8. Muilla param-id asetuksilla 1,4,5 ja 6 päädyttiin jokaisella kerralla globaaliin maksimiin 96. Tästä voidaan siis päätellä, että ketjujen lukumäärällä on merkittävä vaikutus tuloksen laatuun, koska se vaikuttaa suoraan algoritmin suoritukseen; kuinka monta kertaa se toistetaan, ja siten kuinka monta kertaa kontrolliparametria lasketaan. Näin ketjujen lukumäärä

vaikuttaa suoraan myös läpikäytävien pisteiden ja eri kombinaatioiden määrään. Tämän myötä hyväksyttyjen ja hylättyjen siirtymien lukumäärä on myös alhaisempi kuin muilla param-id asetuksilla. Huomioi kuitenkin, että algoritmi estää arpomasta samaa pistettä, missä se sillä hetkellä on, mutta ei estä myöhemmin palaamasta samaan pisteeseen (vaikkakin tämä tapahtuu vähenevällä todennäköisyydellä). Algoritmi voi kuitenkin yrittää siirtyä samaan huonontavaan pisteen useamman kerran peräkkäin (vaikka se ei ole kovin todennäköistä), niin kauan kunnes sitä ei hyväksytä siirtymäksi. Siksi hylättyjen ja hyväksyttyjen siirtymien kokonaismäärä ei kerro suoraan, kuinka monta erilaista kombinaatiota käytiin läpi, mutta antaa kuitenkin suuruusluokan.

Jos vertaillaan param-id asetuksia 2 ja 3, huomataan asetuksen 2 antavan keskimäärin hieman parempia tuloksia. Koska 'Hyv-min' parametri on asetuksella 3 hieman suurempi 7 kuin asetuksella 2 se on 5, voidaan paremman tuloksen olettaa johtuvan tästä. Huomataan myös, että yksittäisen Markov-ketjun pituutta rajoittavista parametreista 'Markov-max' ja 'Hyv-min' vain 'Hyv-min' itse asiassa rajoittaa ketjujen pituutta, koska asetuksella 2 hyväksyttyjen siirtymien lukumäärä on maksimi $10 * 5 = 50$, ja samoin asetuksella 3 maksimi $10 * 7 = 70$, jolloin Markov-ketju katkaistaan heti kun hyväksyttyjä siirtoja on minimimäärä 5/7, jonka jälkeen kontrolliparametria lasketaan ja aloitetaan uusi ketju. Jos parametri 'Markov-max' rajoittaisi asetuksilla 2 ja 3 Markov-ketjun pituutta, tulisi jonkin Markov-ketjun pituus olla 100. Tämä ei ole mahdollista, sillä keskimäärin hylätyt siirtymät on asetuksella 2 välillä 61,2-82,9, jolloin yksittäisen Markov-ketjun hylättyjen ja hyväksyttyjen siirtymien maksimimäärä on $82,9 + 5 = 87,9 < 100$. Todennäköisesti hylättyjen siirtymien kokonaislukumäärä 82,9 jakautuu nousevasti kaikille kymmenelle ketjulle, eikä suinkaan kaikki vain yhdelle ketjulle. Sama päättely pätee asetuksella 3. Yleisesti voidaan todeta, että jos hyväksyttyjen siirtymien lukumäärä = 'ketjujen lkm' * 'Hyv-min', niin Markov-ketjun pituutta rajoittavista parametreista parametri 'Hyv-min' on rajoittavampi, kuin 'Markov-max'. Tällöin parametri 'Markov-max' on niin suuri, että hyväksyttyjä siirtymiä ehditään ottamaan niin monta kuin halutaan ('Hyv-min') ennen kuin saavutetaan hylättyjen siirtymien maksimimäärä. Hylättyjen siirtymien maksimimäärä on siis yksittäisellä Markov-ketjulla 'Markov-max' - 'Hyv-min'. Param-id asetukset 2 ja 3 eroavat myös parametrilla jääh.kerroin toisistaan. Tämän vaikutus oletetaan kuitenkin olevan melko mitätön tulosten perusteella.

Tarkastellaan param-id asetuksia 1,4,5 ja 6 tarkemmin. Kuten aikaisemmin todettiin, näillä asetuksilla päädyttiin aina optimiarvoon 96. Tämä johtuu todennäköisimmin Markov-ketjujen suuremmasta lukumäärästä. Näillä asetuksilla myös Markov-ketjun pituutta rajoittava parametri 'Markov-max'

saa enemmän roolia. Esimerkiksi asetuksella 1 hyväksyttyjen siirtymien lukumäärä on välillä 176,7-189,3, jolloin suurinkin luku 189,3 on pienempi kuin 'ketjujen lkm' * 'Hyv-min' = $20 * 10 = 200$. Nyt siis parametri 'Markov-max' rajoittaa vähintään yhden Markov-ketjun pituutta. Toisin sanoen kaikki Markov-ketjut eivät ehdi ottamaan kymmentä hyväksyttävää siirtoa ennen kuin siirtymäyrityksiä on ollut 200 kappaletta. Eniten parametri 'Markov-max' on rajoittanut Markov-ketjujen pituutta asetuksessa 5, jossa hyväksyttyjen siirtymien keskiarvon 81,6 suhde hyväksyttyjen siirtymien maksimimäärään 60*4=240 on pienin 0,34. Myös keskimääräinen hyväksyttyjen siirtymien lukumäärän suhde hylättyihin on tällä asetuksella pienin. Vähiten hyväksyttyjä ja hylättyjä siirtymiä keskimäärin yhteensä oli asetuksella 6, joten se on myös todennäköisesti nopein asetuksien joukosta 1,4,5 ja 6. Asetuksien 5 ja 6 välillä ei ollut muuta eroa kuin kontrolliparametrin alkuarvo. Huomataan, että alkuarvolla 8 saadaan vähemmän hyväksyttyjä ja enemmän hylättyjä siirtoja kun asetuksella 6 alkuarvolla 25. Eroa kohdefunktion arvoissa ei kuitenkaan ole.

Seuraavaksi katsotaan algoritmin erilaisten asetuksien vaikutusta tuloksiin. Näitä asetuksia oli yhteensä 7 ja niiden eri muunnokset nähtiin alg-asetus-taulukosta. Ihan ensin huomataan, että parhaat lokaalit maksimit löydetään algoritmin asetuksilla 2 ja 6. Puolestaan huonoin tulos saadaan algoritmin asetuksella 7. Näin ollen voidaan olettaa, että Markov-ketjun siirtymä valinnalla 'paras' antaa paremman tuloksen. Kun taas valinnalla 'viimeinen' päädytään hieman huonompiin tuloksiin. Markov-ketjun siirtymän valinnalla näyttääkin olevan melko voimakas vaikutus tuloksiin. Toisin kun naapuruston valinnalla ei näyttäisi olevan niin suurta vaikutusta, mutta valinta 'normaali' vaikuttaisi parhaimmalta vaihtoehdolta. Aloituspisteen valinta ei näy juuri lainkaan tuloksissa, mutta myöhemmin nähdään, että sillä voi olla hyvinkin suuri vaikutus erilaista aineistoa tutkiessa.

Parhaat tulokset saatiin siis param-id asetuksilla 1,4,5 ja 6 ja algoritmin asetuksilla 2 ja 6. Seuraavassa esimerkissä käytetään todellista, paljon suurempaa, aineistoa kaupan tuotteista ja asiakkaista kuukauden ajalta. Tällöin penetraatioaste tulee olemaan paljon alhaisempi kuin tässä esimerkissä, ja vaihtelu vastauksissa on suurempaa. Silloin on aiheellista tulostaa myös algoritmin suoritusajat, joka osaltaan myös vaikuttaa param-id asetusten valintaan.

4.4 Esimerkki 2: suuri aineisto

Tarkastellaan oikeaa aineistoa asiakkaiden ostoksista erään ketjuliikkeen liikkeistä yhdestä kaupungista muutaman vuoden ajalta. Tehtävänä on jälleen valita muutama tuote, joita on ostettu eniten. Tässä osa-aineistossa erilaisia

tuotteita on 9074, ja kuten aikaisemmin todettiin, menisi täydellisellä luetteloinnilla kombinaatioiden laskemiseen tietokoneella noin 1.5×10^7 vuotta, jos yhden kombinaation laskemiseen kuluisi millisekunti aikaa. Eri asiakkaita aineistossa on 23968, joten edellisen esimerkin tapaan merkittynä olisi binäärimatriisimatriisi K nyt kooltaan 23968×9074.

Parhaan penetraatioasteen etsimiseen käytetään samaa simuloidun jäähdytyksen algoritmia kuin edellisessä esimerkissä. Käytetään myös samoja algoritmin asetuksia.

alg-asetus	naapurusto	aloituspiste	Markov-ketjun siirtymä
1	jaettu todennäköisyys	frekvenssi	paras
2	normaali	frekvenssi	paras
3	laaja	frekvenssi	paras
4	laaja	satunnainen	paras
5	laaja	satunnainen	viimeinen
6	normaali	satunnainen	paras
7	normaali	satunnainen	viimeinen

Param-id asetuksia varten pitää laskea kontrolliparametrille alkuarvo. Empiirisellä tarkastelulla saadaan alkuarvoksi $c_0 = 0.0325$, jolloin sadalla siirtymäyrityksellä noin 80% siirtymistä hyväksytään ($\chi_0 = 0.8$) (pätee vain aloituspisteen ollessa 'frekvenssi'). Toisaalta laskemalla saadaan alkuarvoksi

$$c_0 = \frac{\overline{\Delta C}^{(+)}}{\ln(\chi_0^{-1})} = \frac{0.03949}{\ln(1/0.8)} \approx 0.177.$$

Huomaa, että kontrolliparametrin alkuarvon laskeminen perustuu satunnaisiin siirtymäyrityksiin, jolloin sen arvon suuruus voi hieman vaihdella eri kerroilla laskettaessa. Myös algoritmin aloituspiste ja valittu naapurusto vaikuttaa kontrolliparametrin alkuarvon laskemiseen. Aloituspisteellä 'frekvenssi' päädytään kuitenkin aina melkein samaan tulokseen kontrolliparametrin alkuarvolla 0.0325, jolloin noin 80% siirtymistä hyväksytään sadalla siirtymäyrityksellä laskettaessa, kun naapurustoksi valitaan joko 'jaettu todennäköisyys' tai 'normaali'. Naapuruston ollessa 'laaja' vaihtelee hyväksyttyjen siirtymien lukumäärä hieman enemmän. Kun aloituspiste valinnassa 'frekvenssi' on aina sama, voidaan kontrolliparametrin alkuarvolle etsiä optimaalisin vaihtoehto, kuten empiirisesti etsimällä. Näin ei ole kuitenkaan algoritmin aloituspiteen ollessa satunnainen, mutta tarkastellaan sitä myöhemmässä vaiheessa. Parametrien 'ketjujen lkm', 'Markov-max' ja 'Hyv-min' arvoja on säädelty, jotta suoritusajat eivät olisi liian suuria. Parametrin 'jääh.kerroin' arvot pidetään samoina. Näin ollen saadaan seuraavat param-id asetukset.

param-id	ketjujen lkm	Markov-max	Hyv-min	c_0	jääh.kerroin
1	20	70	7	0.0325	0.8
2	10	20	4	0.0325	0.8
3	10	20	6	0.0325	0.9
4	30	10	8	0.0325	0.8
5	60	20	4	0.0325	0.8
6	50	30	4	0.177	0.8

Koska suoritusajat ovat sen verran pitkiä joillakin tapauksilla, ajetaan algoritmi joka kohdassa vain kerran. Arvot eivät siis ole 10 kerran keskiarvoja niin kuin edellisessä esimerkissä. Etsitään ensin 5 eniten ostettua tuotetta kaikkien tuotteiden joukosta. Taulukoidaan saadut vastaukset. Seuraavassa taulukossa on hyväksyttyjen ja hylättyjen siirtymien kokonaislukumäärät.

Siirtymien lukumäärät (hyväksytty/hylätty)						
a/p	1	2	3	4	5	6
1	117/583	40/35	60/23	68/232	71/969	104/891
2	115/571	39/57	60/19	63/236	59/994	96/956
3	119/587	40/56	60/35	58/241	79/980	105/923
4	115/395	40/3	60/3	65/235	87/839	126/706
5	140/63	40/2	60/1	122/161	94/839	113/758
6	140/2	40/0	60/0	221/49	237/336	200/60
7	140/1	40/0	60/0	196/82	240/25	200/61

Hyväksyttyjen ja hylättyjen siirtymien lukumäärissä nähdään sama ilmiö kuin edellisessä esimerkissä. Param-id asetuksilla 2 ja 3 on selkeästi vähemmän siirtymiä ja niillä parametri 'Markov-max' ei juurikaan rajoita algoritmia saavuttamasta haluttu määrä hyväksyttyjä siirtymiä kussakin ketjussa. Poikkeuksena param-id asetuksen 2 ja algoritmin asetus 2, jossa hyväksyttyjen siirtymien lukumäärä $39 < 40$. Nyt myös algoritmin asetuksilla 6 ja 7 on usein päästy hyväksyttyjen siirtojen maksimimäärään. Tämä voi johtua aloituspisteen valinnasta 'satunnainen', jolloin huonontavan siirron hyväksymistodennäköisyys ei välttämättä ole aivan optimaalinen, koska peräkkäisten pisteiden antama kohdefunktion erotus voi olla hyvinkin vaihteleva. Näin ollen huonontavan siirron hylkäämistodennäköisyys voi olla liian suuri tai liian pieni. Kuten param-id asetuksilla 1, 2 ja 3 ja algoritmin asetuksilla 6 ja 7 hylättyjen siirtymien lukumäärä on hyvin pieni tai jopa nolla. Tällöin huonontavan siirron hyväksymistodennäköisyys on ollut liian suuri, ja/tai aloituspisteeksi on valittu hyvin huono piste.

Lokaaleille maksimeille saatiin alla olevan taulukon mukaiset arvot (penetraatioaste). Penetraatioasteet on nyt merkitty suhdelukuna (tuotekombinaatiosta ostaneiden asiakkaiden lkm/kaikkien asiakkaiden lkm), toisin kuin aikaisemmin ne oli merkitty prosentteina.

Penetraatioaste (suhdeluku)						
a/p	1	2	3	4	5	6
1	0.1867591	0.1867591	0.1867591	0.1867591	0.1867591	0.1867591
2	0.1867591	0.1823134	0.1809516	0.1867591	0.1867591	0.1867591
3	0.1706985	0.1660926	0.1660926	0.1660926	0.1797901	0.1660926
4	0.07293335	0.01269625	0.01970522	0.1766261	0.07705864	0.08743191
5	0.03832906	0.02599327	0.0157001	0.06195931	0.07057033	0.08170458
6	0.001321692	0.0008410766	0.001441846	0.005126562	0.01598045	0.002563281
7	0.001441846	0.004405639	0.00220282	0.02266902	0.001602051	0.003364306

Parhaat tulokset saatiin selkeästi algoritmin asetuksilla 1 ja 2. Näyttääkin siltä, että algoritmin asetuksilla on tässä kohtaan suurempi vaikutus tuloksiin kuin param-id asetuksilla. Param-id asetuksilla 2 ja 3 saadut arvot ovat hieman huonompia, kuin muilla asetuksilla. Keskimäärin parhaat tulokset saatiin param-id asetuksella 4, mutta erot ovat pieniä ja tulee myös huomioida sattuman osuus (arvot eivät ole keskiarvoja, vaan yhden ajokerran tuloksia). Algoritmin asetuksilla 1,2 ja 3 vastaukset ovat samaa suuruusluokkaa, mutta asetuksilla 4,5,6 ja 7 saadaan varsin huonoja tuloksia. Yhteistä näille on satunnainen aloituspiste, kun taas asetuksilla 1,2 ja 3 aloituspiste valittiin suurimman frekvenssin perusteella. Tietyn kontrolliparametrin alkuarvon käyttö ei ole kovin käytännöllistä käytettäessä satunnaista aloituspistettä. Saman alkuarvon 0,0325 käyttäminen satunnaisessa aloituspisteessä johtaa heikompiin tuloksiin, koska on hyvin epätodennäköistä, että aloituspisteeksi arvottaisiin yhtä hyvä piste kuin valinta frekvenssien perusteella. Näin ollen satunnaista aloituspistettä käytettäessä tulisi kontrolliparametrin alkuarvoa laskea vastaamaan paremmin haluttua hylättyjen siirtymien hyväksymistodennäköisyyttä. Jos aloituspisteeksi valikoituu satunnaisesti erittäin huono piste, voi olla, ettei sen naapurista löydy yhtään parempaa pistettä ja näin ollen algoritmin jää etsimään parempia pisteitä vain huonojen pisteiden joukosta. Näin voi tapahtua naapuruston ollessa 'normaali'. Selkeästi parempia tuloksia saadaankin, kun naapurusto valitaan laajaksi, jolloin aloituspisteestä voidaan siirtyä mihin tahansa pisteeseen (algoritmin asetukset 4 ja 5). Nämäkään tulokset eivät ole kovin hyviä verrattuna algoritmin asetuksien 1 ja 2 tuloksiin, joten voidaan yleisesti todeta satunnaisen aloituspisteen tuottavan heikohkoja tuloksia. Aloituspisteen valinta yhdessä valitun naapuruston kanssa näyttäisikin kontrolloivan tuloksia eniten.

Koska asiakkaita ja tuotteita on suuri määrä, vaikuttaa se jo algoritmin suoritusaikaan merkittävästi. Tehtävälle saatiin seuraavat suoritusajat (luvut minuutteina).

Suoritusajat (min)						
a/p	1	2	3	4	5	6
1	30	4.0	3.1	10	48	46
2	23	3.6	2.8	10	48	63
3	24	3.4	3.3	10	50	47
4	17	1.7	2.3	10	31	28
5	9.0	1.6	2.2	10	31	62
6	6.7	1.6	2.2	9	19	12
7	6.7	1.5	2.1	9	8.9	12

Parhaat tulokset saatiin algoritmin asetuksilla 1 ja 2 ja param-id asetuksilla 1,4,5 ja 6. Juuri näillä asetuksilla myös suoritusajat ovat pisimmät. Kuten olettaa saattaa, on satunnaista aloituspistettä käytettäessä myös suoritusajat huomattavasti nopeammat. Siirtymäyritysten lukumäärä ja näin ollen Markov-ketjujen lukumäärä ja niiden pituutta rajoittavat parametrit 'Markov-max' ja 'Hyv-min' vaikuttavat suoraan algoritmin suoritusaikoihin. Param-id asetuksella 4 suoritusaika on selvästi nopeampi kuin asetuksilla 1,5 ja 6. Vaikka asetuksella 4 onkin enemmän Markov-ketjuja kuin asetuksella 1, on niiden maksimipituutta rajoittava parametri 'Markov-max' paljon pienempi. Siksi ketjut ovat asetuksella 4 lyhyempiä ja asetuksella 1 pidempiä. Tulos on sitä luotettavampi, mitä enemmän Markov-ketjuja otetaan ja mitä pidemmäksi niitä kasvatetaan. Laskenta-ajan tullessa vastaan on niiden määrää ja pituutta kuitenkin rajoitettava. Jos tulos halutaan nopeasti on asetus a=1 ja p=4 hyvä vaihtoehto. Jos tulos on saatava vielä merkittävästi nopeammin, on asetus a=1 ja p=3 paras vaihtoehto, mutta silloin vastauksen luotettavuuteen tulee suhtautua kriittisesti. Jos taas suoritusaika voi olla enemmänkin, on luotettavinta valita asetus a=1 ja p=1/5/6.

Koska algoritmin asetuksilla 1 ja 2 päädyttiin eri param-id asetuksilla lähes aina (10/12) samaan lokaaliin maksimiin 0,1867591, voitaisiin tämän olettaa olevan myös globaali maksimi. Täysin varmoja tästä ei kuitenkaan voida olla. Näillä kymmenellä eri suorituskerralla päädyttiin aina samaan tuotteiden kombinaatioon, joten voidaan arvella tehtävällä olevan yksi globaali maksimi. Toiseksi parhaalla lokaalilla maksimilla 0,1823134 (asetus a=2, p=2) tuotteiden kombinaatio erosi kahdella tuotteella parhaasta saadusta kombinaatiosta.

Tarkastellaan seuraavaksi tuloksia, kun viiden tuotteen valinnan sijasta tuotteita halutaankin kuusi. Param-id ja algoritmin asetukset pidetään samoina.

Siirtymien lukumäärät (hyväksytty/hylätty)						
a/p	1	2	3	4	5	6
1	122/521	40/25	60/18	67/233	71/989	106/876
2	122/547	40/28	60/31	63/236	77/973	100/904
3	112/561	40/34	60/46	57/241	63/1031	95/953
4	119/469	40/5	60/2	129/150	72/909	114/779
5	140/106	40/2	60/5	160/118	93/836	112/733
6	140/9	40/3	60/0	228/31	240/18	200/34
7	140/0	40/0	60/0	229/32	239/169	200/82

Penetraatioaste (suhdeluku)						
a/p	1	2	3	4	5	6
1	0.2056632	0.2056632	0.2037007	0.2056632	0.2056632	0.2056632
2	0.2056632	0.2029398	0.2037808	0.2056632	0.2056632	0.2056632
3	0.1805111	0.1735421	0.1742631	0.1735421	0.1910846	0.1967719
4	0.09115668	0.02190804	0.0159805	0.03424383	0.09772509	0.08254566
5	0.03748798	0.00877122	0.0329621	0.02647389	0.1092198	0.08959468
6	0.006688561	0.0128164	0.0020026	0.0036847	0.00096123	0.008090356
7	0.001241589	0.0132169	0.0032842	0.00772989	0.01397789	0.003884973

Siirtymien lukumäärät ovat hyvin samaa suuruusluokkaa kuin viiden tuotteen tapauksessa. Joillakin satunnaisen aloituspisteen asetuksilla siirtymien lukumäärä saattaa vaihdella enemmän, mutta se johtunee juurikin satunnaisesta aloituspisteestä. Tällainen vaihtelu huomataan asetuksissa a=6/7 ja p=5 hylättyjen siirtymien lukumäärissä. Ja kuten saattaa olettaa, on lokaalien maksimien arvot nousseet, kun tuotteita on lisätty yksi. Näidenkin arvojen vaihtelu on hyvin samanlaista kuin viidellä tuotteella kokeiltaessa. Selkeästi parhaat tulokset saatiin taas algoritmin asetuksilla 1 ja 2. Asetuksella 3 saatiin vielä käyttökelpoisia tuloksia, mutta asetuksilla 4 ja 5 saadut arvot ovat jo eri suuruusluokkaa ja siten huonoja. Huonoimmat tulokset saatiin taas asetuksilla 6 ja 7. Algoritmin asetuksilla näyttää taas olevan kontrolloiva merkitys penetraatioasteisiin. Huomataan kuitenkin param-id asetuksilla 2 ja 3 saadut hieman pienemmät lokaalit maksimit. Parhaat tulokset saatiin siis asetuksilla a=1/2 ja p=1/4/5/6, siis samoilla asetuksilla kuin viiden tuotteen tapauksessa. Merkittävä rooli on siis suoritusajoilla, joille saatiin seuraavat arvot (minuutteina).

Suoritusajat (min)						
a/p	1	2	3	4	5	6
1	30	2.8	2.9	10	33	50
2	29	2.5	3.2	9.7	33	40
3	35	2.5	3.5	9.5	34	36
4	25	1.6	2.1	9.0	30	28
5	8.9	1.5	2.2	8.8	40	27
6	5.6	1.5	2.2	8.6	11	7.5
7	5.6	1.4	2.9	8.4	18	9.0

Suoritusajat ovat hyvin samansuuruisia kuin viidellä tuotteella. Vaikka kombinaatioita kaiken kaikkiaan onkin nyt enemmän, niin param-id asetukset ovat samoja, jolloin Markov-ketjujen lukumäärä ja niiden maksimipituudet ovat samoja. Siksi suoritusajatkin ovat melko lähellä toisiaan. Parhaan vastauksen löytyminen voisi tosin olla työläämpää, kun mahdollisia kombinaatioita on enemmän. Näyttää kuitenkin siltä, että paras vastaus 0,2056632 voisi olla myös globaali maksimi, koska tähän samaan lokaaliin maksimikohtaan, samalla tuotekombinaatiolla ja penetraatioasteella, on päädytty yhdeksän kertaa. Nopeiten tähän tulokseen päästiin asetuksilla a=1 ja p=2 muutamalla minuutilla. Toiseksi nopein oli asetus a=1/2 ja p=4 noin kymmenellä minuutilla. Selkeästi aikaavievin asetus oli a=1/2 ja p=6. Yhteenvetona voidaan todeta param-id asetusten vaikuttavan suoraan siirtymäyritysten lukumäärään ja siten algoritmin suoritusaikoihin ja algoritmin asetusten vaikuttavan penetraation suhdelukuihin, aivan kuten aikaisemmin viiden tuotteen tapauksessa.

Koska satunnaisella aloituspisteellä päädyttiin viiden ja kuuden tuotteen tapauksessa selvästi huonompiin lokaaleihin maksimeihin, pudotetaan algoritmin asetuksista nämä vaihtoehdot 4, 5, 6 ja 7 pois ja tarkastellaan vain tapauksia 1, 2 ja 3. Seitsemän tuotteen valinnalla saatiin seuraavat tulokset (käyttäen samoja param-id asetuksia), joissa ensimmäinen taulukko kertoo hyväksyttyjen/hylättyjen siirtymien lukumäärän, toinen taulukko löydetyn lokaalin maksimipisteen penetraatioasteen ja kolmas taulukko kunkin asetuksen suoritusajan minuutteina.

Siirtymien lukumäärät (hyväksytty/hylätty)						
a/p	1	2	3	4	5	6
1	136/461	40/30	60/16	87/211	78/958	103/861
2	112/605	40/27	60/17	79/221	66/984	98/909
3	138/505	40/39	60/36	61/238	70/988	100/952

Penetraatioaste (suhdeluku)						
a/p	1	2	3	4	5	6
1	0.2222845	0.2173983	0.217158	0.221083	0.2224848	0.2224848
2	0.2224848	0.2138738	0.217158	0.2212832	0.2224848	0.2224848
3	0.1797501	0.1797501	0.1797501	0.1918055	0.1990548	0.2012977

Suoritusajat (min)						
a/p	1	2	3	4	5	6
1	26	2.8	4.0	13	36	31
2	31	2.4	4.0	14	65	32
3	28	2.7	4.5	13	33	33

Parhaaseen lokaaliin maksimiin 0,2224848 päästiin nyt viisi kertaa asetuksilla a=2, p=1 ja a=1/2, p=5/6. Markov-ketjujen lukumäärien ja niiden pituuksien nähdään tässä kohtaan vaikuttavan jo sen verran, että param-id asetuksella 4 ei enää päästä parhaaseen tulokseen. Myöskään asetuksella a=1 ja p=1 ei siihen päästä, vaikkakin saatu arvo on hyvin lähellä parasta (erotus 0,0002003). Seitsemän tuotteen valinnassa paras ja nopein tulos saatiin asetuksella a=2, p=1 ja a=1, p=6. Koska hajontaa parhaidenkin asetusten välillä alkaa näkyä, tarkastellaan vielä 10 tuotteen valintaa samoilla algoritmin asetuksilla ja param-id arvoilla. Saatiin seuraavat tulokset.

Siirtymien lukumäärät (hyväksytty/hylätty)						
a/p	1	2	3	4	5	6
1	131/491	40/21	60/17	83/215	80/941	106/860
2	131/403	40/15	60/19	91/205	82/921	109/864
3	122/466	39/48	60/32	63/237	76/966	108/925

Penetraatioaste (suhdeluku)						
a/p	1	2	3	4	5	6
1	0.2648991	0.2552868	0.253885	0.2610541	0.2630968	0.2423502
2	0.2646588	0.2467558	0.2536447	0.2611343	0.2648991	0.2648991
3	0.2391862	0.214875	0.214875	0.2149151	0.2423502	0.2314563

Suoritusajat (min)						
a/p	1	2	3	4	5	6
1	22	2.6	3.1	11	48	49
2	19	2.4	3.0	12	34	44
3	20	3.2	3.7	11	48	38

Siirtymien lukumäärät ovat hyvin samansuuruisia kuin muissakin tapauksissa. Penetraatioasteille saadaan kuitenkin nyt paljon enemmän vaihtelua. Yhteensä kombinaatioita kymmenen tuotteen valinnalla on noin 9.6×10^{32}

(9000 tuotteella laskettuna), jolloin tietokoneella menisi kaikkien kombinaatioiden läpikäymiseen aikaa 3.0×10^{22} vuotta, jos yhden kombinaation laskemiseen kuluisi millisekunti. On siis selvää, että tulosten laatu voi hieman laskea, kun valittujen tuotteiden määrää kasvatetaan (< 4500), jolloin myös kombinaatioiden määrä kasvaa, kun Markov-ketjujen lukumäärä ja maksimipituus pidetään samoina. Parhaaseen maksimikohtaan 0,2648991 päädyttiin nyt vain kolmella eri asetuksella: a=1 p=1, a=2 p=5 ja a=2 p=6. Hieman huonompia asteita saatiin jälleen param-id asetuksilla 2 ja 3 ja algoritmin asetuksella 3, mutta toisaalta ne olivat myös suoritusajaltaan nopeimpia. Huomionarvoista on, että algoritmin pisin suoritusaika 49 minuuttia asetuksella a=1 p=6 ei tuottanut parasta vastausta. Myöskään toiseksi pisin suoritusaika 48 minuuttia ei tähän päässyt. Algoritmin asetus 2 näyttäisi antavan keskimäärin hieman parempia penetraatioasteita kuin asetus 1. Keskimääräinen erotus on kuitenkin pieni noin 0,00257, joka voi johtua myös sattumasta. Ajetaan algoritmi vielä uudestaan algoritmin asetuksilla 1 ja 2 ja param-id asetuksilla 1,4,5 ja 6, jotta voidaan tehdä luotettavampia johtopäätöksiä.

Siirtymien lukumäärät (hyväksytty/hylätty)				
a/p	1	4	5	6
1	130/455	86/214	75/916	106/881
2	120/534	81/216	73/959	110/858

Penetraatioaste (suhdeluku)				
a/p	1	4	5	6
1	0.2636975	0.2647789	0.2630968	0.2630968
2	0.2646588	0.2621756	0.2648991	0.2648991

Suoritusajat (min)				
a/p	1	4	5	6
1	28	13	34	35
2	31	10	39	35

Tulosten perusteella arvon 0,2648991 voisi uskoa olevan myös globaali maksimi, koska siihen päädyttiin taas asetuksilla a=2 ja p=5/6 ja koska tällä arvolla päädyttiin myös samaan pisteeseen eli niillä oli sama tuotekombinaatio. Jälleen algoritmin asetus 2 antoi keskimäärin hieman parempia arvoja erotuksella 0,000421275, joten voidaan tehdä varovainen johtopäätös, että se olisi parempi. Toisin sanoen naapuruston valinta 'normaali' olisi parempi kuin valinta 'jaettu todennäköisyys'. Naapurustot 'normaali' ja 'jaettu todennäköisyys' olivat hyvin samankaltaisia, vain sillä erotuksella, että valinnassa 'jaettu todennäköisyys' pisteen naapuri, uusi piste, valittiin todennäköisemmin (0,7) hyvin läheltä nykyistä pistettä frekvenssitaulusta katsottuna.

Asetuksella a=2 ja p=1 päädyttiin nyt toisella algoritmin suorituskerralla samaan pisteeseen kuin ensimmäisellä kerralla, joka oli arvoltaan 0,2646588, siis hyvin lähellä parasta saatua arvoa. Suoritusajoissa nähdään yllättävän paljon heittoa ensimmäisen ja toisen suorituskerran välillä. Kuitenkin param-id asetukset 5 ja 6 olivat jälleen eniten aikaavieviä.

5 Loppupäätelmät

Tutkielmassa penetraatioasteen optimointi- eli parhaan penetraatioasteen tuottava tuotekombinaatio (rajatulla tuotteiden määrällä) -ongelmaa tarkasteltiin simuloitu jäähdytys -menetelmän avulla. Teoriaosuudessa tämän menetelmän todettiin konvergoivan kohti globaalia optimia, mutta ehtoja sen saavuttamiseksi ei voitu todellisuudessa noudattaa, jolloin päädyttiin konvergenssin approksimaatioon. Penetraatioasteen laskemiseen ja optimointiin käytettiin simuloitu jäähdytys -approksimointialgoritmia, joka toteutettiin R kielen koodina. Koodin perusrakenteena esiintyi valinnat pisteen naapurustosta, aloituspisteestä ja Markov-ketjun siirtymästä. Tämän vuoksi koodista tehtiin seitsemän erilaista versiota, jotta voitiin tutkia näiden rakenteiden vaikutusta tulosten laatuun ja valita näin ollen paras approksimaatio. Toisena mielenkiinnon kohteena oli parametrit, joilla koodia ajettiin, kuten ketjujen lukumäärä (ketjujen lkm), Markov-ketjujen maksimipituus (Markov-max), hyväksyttyjen siirtymien minimimäärä (Hyv-min), kontrolliparametrin alkuarvo (c_0) ja jäähdytyskerroin (jääh.kerroin).

Kaikki simuloidun jäähdytyksen tulokset esimerkeissä 1 ja 2 huomioon ottaen parhaat penetraatioasteet saatiin asetuksilla a=2 ja p=1/5/6. Mitä enemmän tuotteita haluttiin mainokseen, myös sitä epävarmemmaksi globaalin tai hyvän lokaalin maksimikohdan löytyminen kävi, koska kombinaatioiden määrä kasvoi. Huomion arvoista oli myös, että algoritmin antaman vastauksen laadun ja luotettavuuden parantamiseksi voitiin Markov-ketjujen lukumäärää ja pituutta kasvattaa, mutta tämä lisäsi silloin myös algoritmin suoritusaikaa. Tulosten luotettavuutta saatiin myös kohennettua ajamalla samoilla asetuksilla useamman kerran, mutta tällaisen lähestymistavan huomataan olevan joskus hyvin aikaavievä. Tuloksia analysoitaessa voitiin vetää seuraava johtopäätös; mitä useammin päädytään samaan parhaaseen maksimikohtaan, sitä todennäköisimmin se on myös globaali maksimi.

Simuloidun jäähdytyksen tuottamien tulosten optimaalisuutta ja luotettavuutta on vaikea tarkastella, jos ei tiedetä globaalia optimiarvoa. Tällaisissa vaikeissa kombinatorisissa tehtävissä globaalin optimin selvittäminen voi olla usein nykyteknologialla lähes mahdotonta. Otanta esimerkin 2 aineistossa oli kuitenkin sen verran suppea, että jälkeenpäin onnistuttiin tarkas-

tamaan CPLEX-ohjelmalla saatujen vastauksien laatu esimerkissä 2. Kuten arveltiin, simuloitu jäähdytys -algoritmin tuottamat parhaat ratkaisut olivat myös globaaleja optimeja. Toisin sanoen 5 tuotteen tapauksessa penetraatioaste 0,1867591 on myös globaali optimiarvo, kuten myös tapauksessa 6 penetraatioaste 0,2056632, tapauksessa 7 penetraatioaste 0,2224848 ja tapauksessa 10 penetraatioaste 0,2648991. Valitut parhaat asetukset algoritmin rakenteelle ja parametrien arvoille näyttäisivät tuottavan siis hyviä tuloksia tälle kyseiselle aineistolle. On kuitenkin mahdollista, että joillakin erilaisilla param-id asetuksilla (jota ei tutkielmassa käytetty), siis parametrin arvoilla, joilla koodia ajetaan, saavutettaisiin vielä optimaalisempia tuloksia, tai lähinnä saavutettaisiin lokaali (ja käsitellyissä esimerkeissä myös globaali) maksimipiste nopeammin, sillä esitellyt param-id asetukset olivat empiirisen kokeilun kautta valittuja ja siten lähinnä suuntaa antavia. Asetuksien (a=2 ja p=1/5/6) ei voida myöskään yleistää suoraan olevan parhaita mille tahansa aineistolle. Varsinkin param-id arvoista kontrolliparametrin alkuarvo c_0 tulisi aina laskea käytettävissä olevan aineiston pohjalta ja suhteuttaa parametrit ketjujen lkm, Markov-max ja Hyv-min aineiston laatuun ja laajuuteen. Algoritmin asetukset on suunniteltu juuri tämän tutkielman tapaiseen aineistoon ja tehtävään, jolloin naapurustorakenne on myös mietitty tämän pohjalta. Jos naapurustorakenne halutaan pitää samanlaisena jossain toisessa aineistossa, voidaan algoritmin asetuksia 1, 2 ja 3 pitää käyttökelpoisina. Näistä kolmesta paras vaihtoehto voi riippua hieman aineiston laadusta (toisin sanoen millainen frekvenssimatriisi aineiston pohjalta saadaan).

Analyysissä tuotteiden valinnalla on yksi merkittävä rajaus, sillä liikkeeseen saapuvilla uusilla tuotteilla ei ole mahdollisuutta tulla valituksi ainakaan heti, sillä niillä ei ole olemassa ostohistoriaa. Toisaalta sama pätee asiakkaiden houkutteluun, sillä mainos on tehty jo olemassa olevien asiakkaiden ostohistorian perusteella. Voi olla, että uusien asiakkaiden houkuttelu onnistuisi mainostamalla nimenomaan uusia tuotteita. Yhtenä vaihtoehtona voisi ajatella, että yksi tai useampi valituista tuotteista korvattaisiin jollakin uudella tuotteella, joka ei ole samankaltainen muiden mainosten tuotteiden kanssa. Näin mainos voisi vetää puoleensa niin vanhoja kuin uusia asiakkaita.

Simuloitu jäähdytys -algoritmi ei päädy luotettavasti aina globaaliin optimiin (vaikkakin voi tarjota hyvän lokaalin optimin), mikä on menetelmän yksi huonoista puolista. Ja vaikka simuloitu jäähdytys on täydellistä luettelointia huomattavasti nopeampi, se ei kuitenkaan toteutetulla koodilla ole kovin nopea. Osaksi se johtui esimerkissä 2 käytetyn aineiston muodosta, joka ei ollut sama kuin muoto esimerkin 1 aineistossa. Siispä aineiston muotoa muokkaamalla voitaisiin päästä nopeampaan algoritmin toteutukseen. Tätä ei kuitenkaan onnistuttu tekemään aineiston suuresta koosta johtuen ja tietokoneen muistikapasiteetin tullessa vastaan. Näin ollen suurempaa osa-

aineistoa käyttämällä algoritmilla voi mennä hyvinkin kauan aikaa suoriutua. Toisena vaihtoehtona olisi penetraatioasteen optimointiongelmaa voinut tutkia geneettisten algoritmien avulla, joka sekin on simuloidun jäähdytyksen tapaan heuristinen menetelmä, jolloin globaalia optimia ei voida taata.

Esimerkissä 2 ei otettu kantaa millaisista tuoteryhmistä valitut tuotteet valittiin. Valitut tuotteet voivat siis olla hyvinkin samanlaisia ja samasta tuoteryhmästä, esimerkiksi rasvaton maito 2 dl ja rasvaton maito 1l. Jos tuotteita haluttaisiin valita eri tuoteryhmistä, tulisi aineistossa olla tieto mihin tuoteryhmään kukin tuote kuuluu. Tällaista analyysia ei pystytty tekemään osin puutteellisen aineiston takia. Kyse on kuitenkin vain pienestä muutoksesta koodissa, missä tuotteiden valintaa rajoitetaan. Rajoittamalla tuotteiden valintaa voi sallittujen tuotekombinaatioiden lukumäärä vähentyä merkittävästi.

Tutkielmassa ei myöskään selvitetty mainoksen tuotto-odotuksia, sillä tavoitteena oli saada mainokseen paras tuotekombinaatio asiakkaiden ostohistorian perusteella, jolloin mainos houkuttelisi mahdollisimman monta asiakasta. Ajatuksena on myös, että asiakas ostaa muitakin tuotteita kuin vain mainokseen päätyneitä. Mahdollista tuotto-odotusta arvioidessa voi olla myös hankalaa arvioida ketkä asiakkaista ovat saapuneet liikkeeseen mainoksen perusteella, ja ketkä olisi tulleet ilman mainostakin. Tuoton suurentuessa voidaan toki olettaa mainoksen saavuttaneen asiakkaita, mutta tutkielman tehtävänasettelun puitteissa olisi mielenkiintoisempaa seurata asiakasvirtaa erilaisten mainoksien tuotekombinaatioiden jälkeen. Jos tuottoa haluttaisiin optimoida asiakasmäärän sijaan mainoksella, olisi tehtävänasettelu hieman erilainen. Vaikkakin tavoittamalla mahdollisimman paljon asiakkaita niiden ostoshistorian perusteella, voidaan parantaa tuottoa, tulisi tuoton optimoinnissa kiinnittää huomiota muuhunkin kuin mainoksesta kiinnostuneiden asiakkaiden lukumäärään. Esimerkiksi tällöin tuotteen hinnalla sekä katteella on myös vaikutusta.

Viitteet

[1] P.J.M van Laarhoven ja E.H.L. Aarts (1987) Simulated Annealing: Theory and Applications

[2] Mitra D.,Romeo F., ja A.L. Sangiovanni-Vincentelli, Converge and Finite-Time Behavior of Simulated Annealing, Advances in Applied Probability Vol. 18 No. 3 (Sep., 1986), pp. 747-771

[3] Ronald L. Rardin (1998) Optimization in Operations Research, pp.680-694

[4] Marko M. Mäkelä (2015) Matemaattinen optimointi II

[5] Isaacson Dean and Madsen Richard, Positive Columns for Stochastic Matrices, Jornal of Applies Probability (1974), pp.829-835

[6] E. Seneta, Non-negative Matrices and Markov Chains, (Springer Verlag, New York, 2nd ed, 1981)

[7] Matti Mononen Pro gradu -tutkielma (2012) Kuinka pakata matkatavarat ennen kuin aurinko sammuu eli 0/1-reppuongelman ratkaisumenetelmät teoriassa ja käytönnössä

[8] http://users.jyu.fi/ mannikko/algoritmit1/luennot/luento13.pdf

[9] Kirkpatrick S., C.D. Gelatt Jr., M.P. Vecchi, Optimization by Simulated Annealing, Science, 220(1983) 671-680

A Ostoaineisto

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
2	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
3	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
4	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1
5	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
6	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
7	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	1	1	0
8	0	0	1	0	1	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
10	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0
11	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
13	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1
14	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1
15	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0
16	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0
17	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
18	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
19	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1
20	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
21	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
22	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1
24	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
25	1	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	1	0	0	0
26	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
27	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1
28	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1
29	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
30	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
31	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1
32	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
33	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1
34	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
35	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1
36	0	0	1	0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	0	1
37	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
40	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	1	1	0
41	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
42	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
43	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
44	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0
45	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
46	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
47	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
49	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
50	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1

51	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	1	1	0	1
52	0	0	0	0	1	1	0	1	0	0	1	0	0	0	1	0	0	1	0	1
53	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1
55	0	0	1	1	1	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0
56	0	0	1	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1
57	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
58	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
59	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1
60	0	0	1	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
61	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
62	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
63	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
64	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
65	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0
66	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
67	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
68	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	1
69	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
70	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
71	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0
72	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	1	0	1	0	1
73	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
74	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
75	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
76	1	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0
77	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0
78	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
79	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
80	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0
81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
82	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1
83	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
84	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
85	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
86	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
87	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
88	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
89	0	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0
90	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	1
91	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
92	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0
93	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1
94	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
95	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
96	0	0	1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0
97	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
98	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
99	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1
100	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

B Frekvenssimatriisi

tuoteryhmä	frek
3	55
20	46
5	44
4	38
8	33
18	31
6	28
16	24
19	22
15	21
14	20
17	18
1	16
11	13
13	11
7	10
9	9
10	6
12	5
2	4