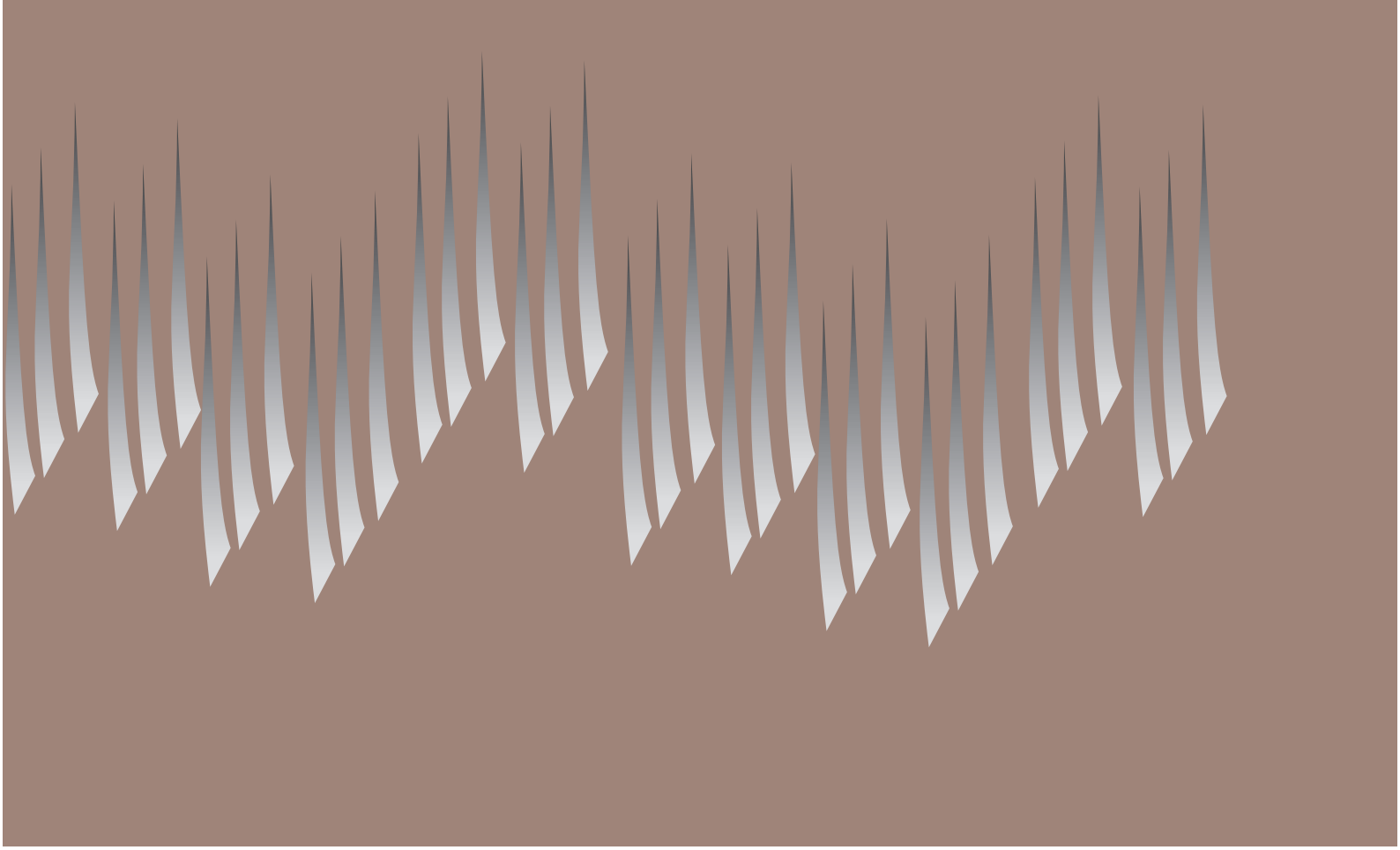


Juha Laakko

LUMIKITEIDEN VISUALISOINTI SOLUAUTOMAATEILLA

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences

Rovaniemen ammattikorkeakoulu

julkaisusarja B 13

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences

LUMIKITEIDEN VISUALISOINTI SOLUAUTOMAATEILLA

Juha Laakko

2008

Rovaniemen ammattikorkeakoulu
Julkaisutoiminta
Kehitys- ja palveluyksikkö InnoRamk
Jokiväylä 11 C
96300 Rovaniemi
p. 020 798 4000
www.ramk.fi/julkaisutoiminta
julkaisut@ramk.fi

ISSN 1239-7733
ISBN 978-952-5153-82-8 (nid.)
ISBN 978-952-5153-83-5 (PDF)

Rovaniemen ammattikorkeakoulun julkaisusarja B nro 13

Rovaniemi 2009
Kopijyvä Oy, Jyväskylä

Tiivistelmä

Tekijä	Juha Laakko	Vuosi	2008
Toimeksiantaja	Arktinen tiedekeskus, Rovaniemi		
Työn nimi	Lumikiteiden visualisointi soluautomaateilla		
Sivu- ja liitemäärä	76 + 6		

Lumikiteet ovat erinomainen esimerkki spontaanista itseorganisoitumisesta. Kiteissä yhdistyy yksinkertainen symmetria sisäiseen monimutkaisuuteen, joka on askarruttanut vuosisatojen ajan tutkijoita. Niiden kehittymistä täysin kuvaavaa mallia ei ole vielä pystytty rakentamaan. Tämän opinnäytetyön tarkoituksena oli tutkia lumikiteitä ja niitä simuloivia malleja. Tavoitteena oli muodostaa tietoja hyödyntäen kiinnostava visuaalinen esitys lumikiteiden kehittymisestä eri olosuhteissa.

Työn tilaajana toimi Rovaniemenellä sijaitseva Arktinen tiedekeskus, joka uusi ja avasi päänäyttelynsä 18.4.2007. Luotavan esityksen ja mallin oli tarkoitus tulla osaksi tätä näyttelyä.

Työssä kiteiden kehittymisen simulointiin käytettiin soluautomaatteja, jotka ovat osoittaneet monipuolisuutensa erilaisten ongelmien mallintamisessa. Työ käsittelee soluautomaattien teoriaa yleisellä tasolla sekä esittelee miksi ne soveltuvat monien kaoottisilta vaikuttavien prosessien mallintamiseen, missä tavalliset matemaattiset menetelmät ovat epäonnistuneet.

Työn toisen osuuden muodostaa soluautomaatin soveltamisen ja siihen liittyvien ongelmien ratkaisemisen. Tässä osuudessa hyödynnettiin useita tekniikoita kuten geneettisiä algoritmeja, neuroverkkoja sekä sumeaa logiikkaa. Lopuksi käsitellään lyhyesti työssä toteutettua visualisointia.

Mallin sovellus onnistui ja visualisointi otettiin käyttöön planaaristen kiteiden synnyn kuvaamisessa. Se on nykyään osana tiedekeskuksen päänäyttelyä, muuttuva arktis.

Avainsanat: lumikide, lumihiutale, soluautomaatti, tietokonegrafiikka, geneettiset algoritmit, neuroverkot

Abstract

Author	Juha Laakko	Year	2008
Commissioned by	Arctic centre, Rovaniemi		
Subject of thesis	Visualization of Snow Crystals Using Cellular Automata		
Number of pages	76 + 6		

Snow crystals are a wonderful example of spontaneous self organization. Crystals exhibit simple symmetries while still showing remarkable inner complexity which has puzzled many scientists and researchers through centuries. No method so far has been able to model or describe all features of their formation process. The purpose of this thesis was to examine snow crystal growth and different models for snow crystal formation. The aim was to produce an interesting visual representation of snow crystal formation in various conditions.

The visualization was commissioned by the Artic Centre located in Rovaniemi. Arctic centre renewed its main exhibition “Arctic in change” in 18 April 2007. The aim was that the produced model and visualization would be taken as part of that event.

Cellular automata were chosen as the modelling tool, as they have shown versatility in modelling various physical processes. This thesis covers basic cellular automata theory and their use in the modeling of chaotic processes where traditional mathematical models have proven insufficient.

Second part of the thesis describes the cellular automaton model that was chosen as the basis of the visualization and solving the related problems. This consisted of using various methods such as genetic algorithms, neural networks and fuzzy logic. Last part of thesis describes the built system briefly.

Applying the model was successful and it was introduced in the “Arctic in change” exhibition for planar snowflake growth. The visualization is currently located in the main event ice cave.

Key words: snow crystal, cellular automata, computer graphics, genetic algorithms, neural networks

Esipuhe

Lumi on meille lappilaisille tuttu elementti. Olemme tottuneet kokemaan lumen läsnäolon massiivisena ilmiönä, emmekä välttämättä ole kiinnittäneet yksittäiseen lumikiteeseen muuta huomiota kuin ohimenevän ”sinisen ajatuksen” syksyisin talven ensilumen leijaillessa maahan. Lumi on meille innoituksen, harrastusten ja toimeentulon lähde mutta myös riesa rakennustemme katoilla ja esteinä kulkuteillämme.

Lumihiutale koostuu lumikiteistä. Lumikide edustaa muodoltaan sellaista geometrista rakennetta, johon liittyviä muistiinpanoja löytyy aineen rakenteen teorioista aina Platonin ajoilta. Tiettävästi ensimmäiset lumikiteen muotoon liittyvät muistiinpanot löytyvät noin vuodelta 1260 Albertus Magnukselta, seuraavat 1500-luvulta Olaus Magnukselta. Näissä muistiinpanoissa korostetaan enemmän lumikiteiden symmetristä kauneutta ja erilaisuutta kuin rakennetta.

1600-luvulla Kepler kirjoitti lumikiteitä tutkittuaan seuraavasti: ”*Koska aina lumisateen alkaessa ensimmäiset lumihiutaleet ovat muodoltaan kuusisäteisiä tähtiä, täytyy tähän olla jokin määrätty syy. Jos asia nimittäin olisi sattumanvaraista, miksi ne eivät leijaile viisisäteisinä tai seitsensäteisinä …”.* Näin Kepler muotoili ongelman, johon ihmiskunnalla ei ole vastausta. Lumikiteiden rakenteen muodostumisen syy–seuraus-suhteita ei tunneta tarkasti vieläkään.

Juha Laakko rakensi opinnäytetyönään Lapin yliopiston Arktisen keskuksen pysyvään näyttelyyn ”Muuttuva Arktis” ohjelmiston, joka simuloi jääkiteen muodostumista erilaisissa ilman lämpötiloissa ja kosteuksissa. Tarkan fysikaalisen mallin puuttuminen johtaa ohjelmistosuunnittelussa haasteisiin. Tulee kyetä luomaan riittävän tarkka matemaattinen malli ja toteuttaa se ohjelmistossa siten, että lopputulos ainakin näyttää oikealta.

Laakko käytti lumikiteiden kehittymisen mallintamisessa ratkaisuna soluautomaattiin perustuvaa mallia. Soluautomaatteja on ohjelmistotekniikassa ja diskreetissä matematiikassa tutkittu ja kehitetty 1960-luvulta lähtien.

Parhaiten soluautomaatin idea avautuu kuvittelemalla ruutupaperia, jossa on erivärisiä ruutuja. Kunkin ruudun väri määräytyy sen ympärillä olevien ruutujen väristä tietyllä säännöstöllä. Kun vielä kuvittelee, että tuo ruutupaperi alkaa elää eli ruutujen värit alkavat muuttua noiden laadittujen sääntöjen mukaisesti, ymmärtää soluautomaatin perusrakenteen. Kuuluisin toteutus soluautomaateista lienee John Convayn ”The Game of Life” vuodelta 1970. Siinä soluille muodostettu säännöstö aiheuttaa nimensä mukaisesti elävien solujen kaltaisen

visuaalisen käytöksen. Soluautomaattien tutkimus jatkuu edelleen esimerkiksi ”tekoelämää” käsittelevien hankkeiden yhteydessä.

Juha Laakko tuotti soluautomaattiin perustuvan matemaattisen mallin lumikiteiden kehittymisestä ja toteutti siitä sekä ohjelmistoteknisen ratkaisun että sen visuaalisen esityksen. Tuotettu kokonaisuus on parhaillaan Arktisessa keskuksessa vierailevien katsottavana ja kokeiltavana.

Samalla Laakko kiteytti työssään Lapille tärkeitä elementtejä – kylmän, matkailun ja teknisen osaaminen.

Rovaniemellä 13.4.2009
Markku Taipale
opinnäytetyön ohjaaja

SISÄLLYS

KUVA- JA KUVIOLUETTELO 8

1 JOHDANTO 9

2 LUMIKITEIDEN MUODOSTUMINEN 10

2.1 LUMIKITEIDEN TUTKIMUS 10
2.2 LUMIKITEIDEN OMINAISUUDET 10
2.3 SIMULAATIOMALLIEN KRITEERIT 16
2.4 SIMULAATIOMALLIEN ESITTELY JA VERTAILU 17
2.5 PERUSTELUT SOLUAUTOMAATIN VALINNALLE 19

3 SOLUAUTOMAATTI KÄSITTEENÄ 21

3.1 SOLUAUTOMAATTIEN PERUSTEET 21
3.2 SOLUAUTOMAATIN OMINAISUUDET 22

4 SOLUAUTOMAATTIEN SOVELTAMINEN 31

4.1 SIMULAATION KUVAUS JA RAJAUS 31
4.2 SOLUAUTOMAATTIEN TOTEUTUS 32
4.3 LUMIKITEIDEN GENEROINTI 38

5 LUMIKITEIDEN MUODOSTUMISEN SIMULOINTI 42

5.1 YLEISTÄ ONGELMASTA 42
5.2 NEUROVERKOT 43
5.3 GENEETTISET ALGORITMIT 48
5.4 SUMEA LOGIIKKA 57
5.5 MUODOSTUMISONGELMAN RATKAISEMINEN 59

6 VISUALISOINNIN TOTEUTTAMINEN 63

6.1 JÄRJESTELMÄN YLEINEN KUVAUS 63
6.2 VISUALISOINNIN OSAT 64
6.3 SIMULAATION VISUALISOINTI 68

7 JOHTOPÄÄTÖKSET 71

LÄHTEET 74

LIITTEET 76

KUVA- JA KUVIOLUETTELO

Kuva 1 Yksi testisarjan kiteistä vertailtuna aitoon. 40
Kuva 2 Simuloidun kiteen kasvu. 41
Kuva 3 Käyttöliittymänkonsepti 66
Kuva 4 Varsinainen käyttöliittymä 67

Kuvio 1 Lumikiteiden kategorisointi. 12
Kuvio 2 Tahkoesitys. 13
Kuvio 3 Yhteenliittyneiden vesimolekyylien kuusikulmainenen rakenne. 14
Kuvio 4 Erityyppisten lumikiteiden esiintyminen lämpötilan ja kosteuden mukaan..... 16
Kuvio 5 Kochin lumihiutale, ensimmäiset iteraatiot. 18
Kuvio 6 Wolfram-sääntö 90. 22
Kuvio 7 Automaatin 90-säännöstö 22
Kuvio 8 Game Of Life-solukon rakenne. 23
Kuvio 9 Game Of Life, esimerkki iteraatioita. 24
Kuvio 10 R-pentomino. 25
Kuvio 11 R-pentomino 153 iteraatio. 26
Kuvio 12 Erilaisia soluautomaatteja 28
Kuvio 13 Boolean kiteen solukonrakenne. 29
Kuvio 14 Wolframin lumikiteen kehitys 30
Kuvio 15 Reiterin automaatin rakenne. 33
Kuvio 16 Reiterin automaatin tuloksia 39
Kuvio 17 Yksittäisen neuronin kuvaus. 44
Kuvio 18 MultilayerPerceptron-verkko. 46
Kuvio 19 Geneettisen haun prosessi. 49
Kuvio 20 Risteytys yhden pisteen kohdalta 52
Kuvio 21 Pistemutaatio 53
Kuvio 22 Yksiparametrinen hakuavaruus 54
Kuvio 23 Kaksiparametrinen hakuavaruus 56
Kuvio 24. Jäsenyysfunktioita. 58
Kuvio 25 Järjestelmän ympäristön fyysinen kuvaus. 63
Kuvio 26 Järjestelmän organisaatio karkeasti. 65
Kuvio 27 Järjestelmän tilat 65
Kuvio 28 Esimerkki automaatin arvojen kartoituksesta. 68
Kuvio 29 Kuusikulmaisuuden saavuttaminen tavallisella näytöllä. 69

1 JOHDANTO

Rovaniemellä sijaitseva tiedekeskus Arktikum ja siihen kuuluva Arktinen keskus uusi perusnäyttelynsä ja avasi sen 18.4.2007. Yhtenä näyttelyn osana on talvea vuodenaikana kuvaava jääluola. Tähän luolaan Arktinen keskus tarvitsi lumikiteiden monimuotoisuutta ja kasvua kuvaavan interaktiivisen esityksen. Tämän opinnäytetyön tarkoituksena oli tuottaa tämän esityksen ohjelmisto sekä luoda malli kiteen kasvusta, jotta se voidaan esittää katsojalle.

Työ jakaantuu kahteen selkeään kokonaisuuteen, lumikiteiden muodostumisen ja fysiologian teoreettisen pohjan selvittämiseen sekä kiteiden muodostuksen simulointiin. Toisena työn osana on varsinaisen ohjelmiston sekä visuaalisen esityksen toteuttaminen.

Tavoitteena on siis tutustua lumikiteiden fysiikkaan ja aiempiin tutkimuksiin niiden muodostumisesta sekä jo valmiiden simulointimallien tutkimiseen. Näiden mallien pohjalta on tarkoitus muodostaa oma ohjelmistokomponentti, jonka antama syöte muokattaisiin reaaliajassa visuaaliseksi esitykseksi.

Visualisoinnin pääasiallisena tavoitteena on tuoda näyttelyssä kävijälle esille lumikiteiden monimuotoisuus sekä havainnollistaa visuaalisesti mielenkiintoisesti ja selkeästi lumikiteiden kasvu pienestä alkukiteestä lähtien.

Työn tavoitteena on tiivistetysti
- selvittää lumikiteiden muodostuminen
- tutkia valmiita malleja sekä muodostaa tarvittaessa näistä oma
- tuottaa visuaalisesti miellyttävä ja kiinnostava esitys.

2 LUMIKITEIDEN MUODOSTUMINEN

2.1 Lumikiteiden tutkimus

Lumikiteiden monimuotoisuutta on tutkittu ja ihmetelty jo useiden vuosisatojen ajan. Ensimmäiset kirjatut tutkimukset lumikiteiden rakenteesta teki René Descartes vuonna 1637, joka kiinnitti erityistä huomiota kiteiden symmetrisyyteen. Myöhemmin mikroskoopin keksintä johti ensimmäisiin piirustuksiin kiteistä, jotka valottivat toista lumikiteiden erikoista ominaisuutta, kompleksista sisäistä rakennetta. (Libbrecht 2004, 27–28.)

Valtaosa lumikiteistä ja niiden muodostumisesta koskevasta tiedosta on tullut Ukichiro Nakayan 1930–1950 suorittamista laboratoriokokeista, joissa hän kasvatti suuria määriä tekolumikiteitä vaihtelevissa olosuhteissa. Näiden kokeiden perusteella Nakaya päätteli, kuinka kosteus ja lämpötila sekä niiden muutokset vaikuttavat kiteen kehitykseen (Libbrecht 2004, 43–45; 2007.) Nykyisintä tutkimusta edustaa Kenneth G. Libbrect, joka on jatkanut Nakayan kokeellista tutkimusta sekä koostanut kerääntynyttä tutkimustietoa.

2.2 Lumikiteiden ominaisuudet

Lumikiteet ovat erinomainen esimerkki spontaanista itseorganisoitumisesta, jossa pienten osasten mikrotason vaikutussuhteet ja ympäristön vaikutukset yhdistyvät makrotason järjestelmäksi. Tuloksena itseorganisoitumisessa on monimutkainen järjestelmä, jonka kasvua ei voida tarkkaan ennustaa tai selittää pienimpien osiensa avulla.

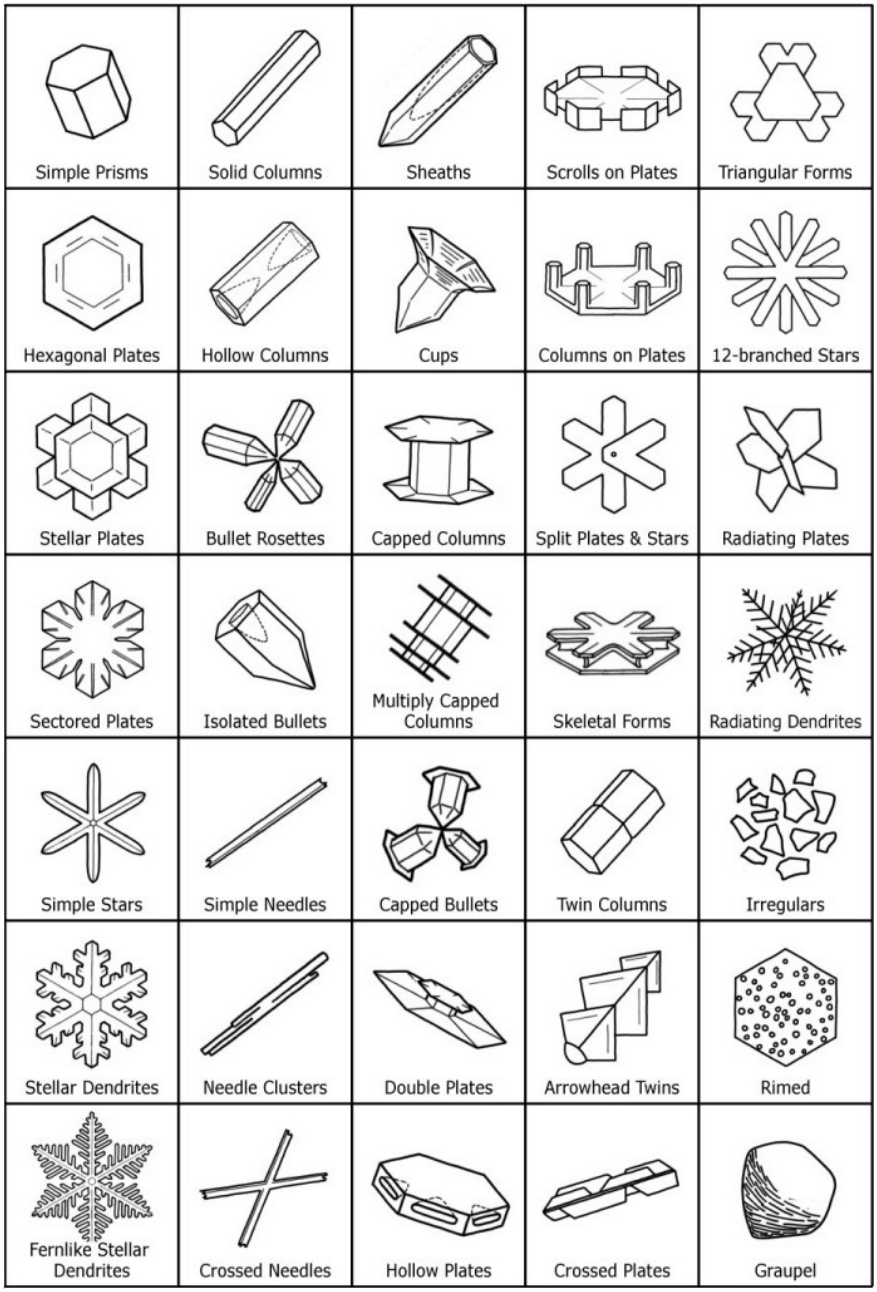
Tämä ilmenee kiteissä siten, että vaikka fysiikan ja kemian lait, jotka kontrolloivat kiteen muodostumista, ovat suhteellisen yksinkertaiset, on tuloksena usein erittäin monimutkainen järjestelmä, jonka kasvu on erittäin herkkä pienille kasvuolosuhteiden muutoksille (Libbrecht 2004, 21,41). Lumikiteiden huomattavampiin ominaisuuksiin kuuluu tarkka kuusikulmaisen symmetrian noudattaminen yhdistettynä sisäiseen monimutkaisuuteen. Tästä hyvän esimerkin antaa liite 1.

Lumikiteet muodostuvat alijäähtyneen vesihöyryn härmistyessä kiinteäksi jääksi. Prosessin käynnistää jokin ilmassa vapaana oleva kappa-

le, joka useimmiten on pölyhiukkanen (Oksanen 1999, 14–15). Tällä tavalla muodostunutta kappaletta voidaan kutsua ”siemenkiteeksi”, jonka ympärille varsinainen lumikide rakentuu. Huomionarvoista on, että lumikiteiden muodostuminen ei ole samanlainen prosessi kuin esimerkiksi lammen jäätyminen, jota tavallisimmin ajatellaan puhuttaessa veden jäätymisestä.

Kiteiden lopullinen koko vaihtelee kasvuolosuhteista riippuen hyvin pienistä 0,2 mm:n laatoista n. 7,5 mm:n kokoisiin monisakaraisiin tähtiin. Kiteiden tyypittämistä varten on esitetty useita luokittelumenetelmiä, joista tunnetuin on Ukichiro Nakayan kehittämä Nakayan luokittelu ja siitä edelleen kehitetty Magono - Lee -luokittelu. (Libbrecht 2004, 3–50.)

Magono - Lee -luokittelu sisältää 80 erityyppistä kidettä (ks. liite 2) ja on laajuutensa vuoksi sellaisenaan tässä työssä hyödytön. Suurin osa tässä luokittelussa esitellyistä lumikiteen tyypeistä on harvinaisia muotoja tai perusmuotojen kasaumia. Tämän luokittelun sijaan tässä työssä käytetään selkeämpää ja havainnollisempaa supistettua Nakayan luokittelua (kuvio 1).



Kuvio 1 Lumikiteiden kategorisointi (Libbrecht 2007).

Huomio kiinnittyy tässä luokittelussa kahden kovin erilaisen perustyypin ilmenemiseen. Kuviosta voidaan erotella tasoon muodostuvat planaariset kiteet (kuvion 1 plates, stars, dendrites) sekä huomattavasti kolmiulotteisemmat kuusikulmaiset pylväät (colums, bullets, needles). Kuvion viimeisen sarakkeen kiteet taas ovat erikoisempia kiteitä, jotka

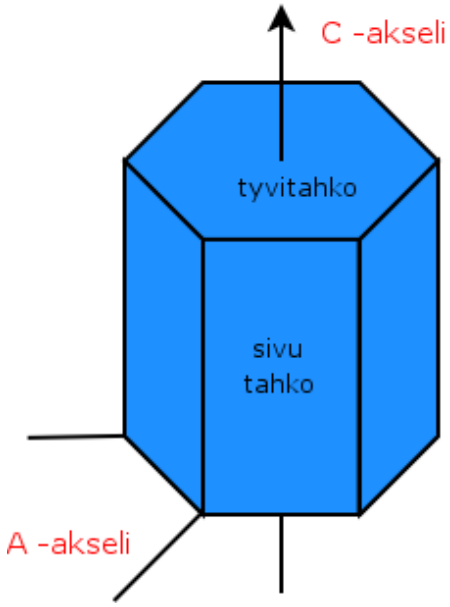
jätetään myöhemmin huomioitta. Näistä esimerkiksi 12-sakaraiset tähdet syntyvät kahden tavallisen 6 sakaraisen tähden yhteenliittymisestä kasvun alkuvaiheessa (Libbrecht 2004, 89).

Lumikiteen kasvu

Kiteen kasvu voidaan jakaa karkeasti kahteen vaiheeseen:

1. tahkojen suuntaiseen (kuvio 2)
2. haarautuvaan kasvuun.

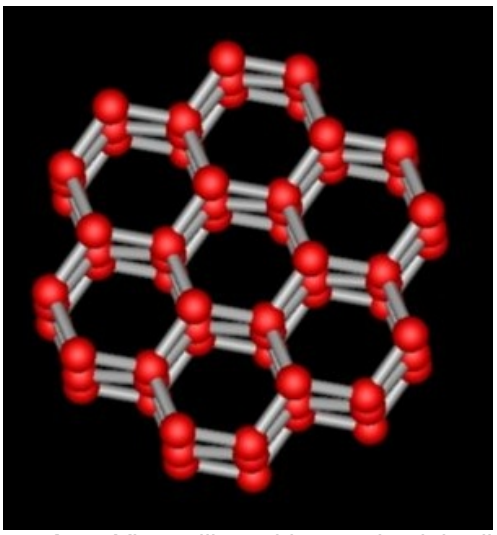
Näiden kasvuvaiheiden vuorottelu ja tasapainottelu olosuhteista riippuen vastaa suurimmaksi osaksi kiteiden kuusikulmaisen rakenteen monimuotoiseen vaihteluun. (Libbrecht 2005, 862.)



Kuvio 2 Tahkoesitys.

Kiteiden kuusikulmaisen symmetrian perustana on jäätyvän veden vesimolekyylien kemiallisten sidosten luonne. Jäätyessään molekyylit muodostavat kuusikulmaisen hilarakenteen (kuvio 3; Oksanen 1999, 11–12). Kasvun alussa siemenkiteeseen alkaa liittyä uusia molekyylejä nk. partikkelidiffuusion vaikutuksesta. Edellä mainitut sidokset määräävät sen, kuinka nopeasti ja millä tavalla uudet molekyylit voivat liittyä jo muodostuneeseen kiteeseen. Kasvun alkuvaiheessa tämä hilarakenne yhdessä liittymiskinetiikan kanssa pakottaa kiteen muo-

doksi yksinkertaisen kuusikulmaisen laatan. Näin syntynyt laatta jatkaa kasvuaan lähinnä lämpötilasta riippuen joko a- tai c-akselin suhteen, jotka havainnollistetaan kuviossa 2. (Libbrecht 2004, 36–41; 2005, 860–863.)



Kuvio 3 Yhteenliittyneiden vesimolekyylien kuusikulmainenen rakenne (Libbrecht 2007).

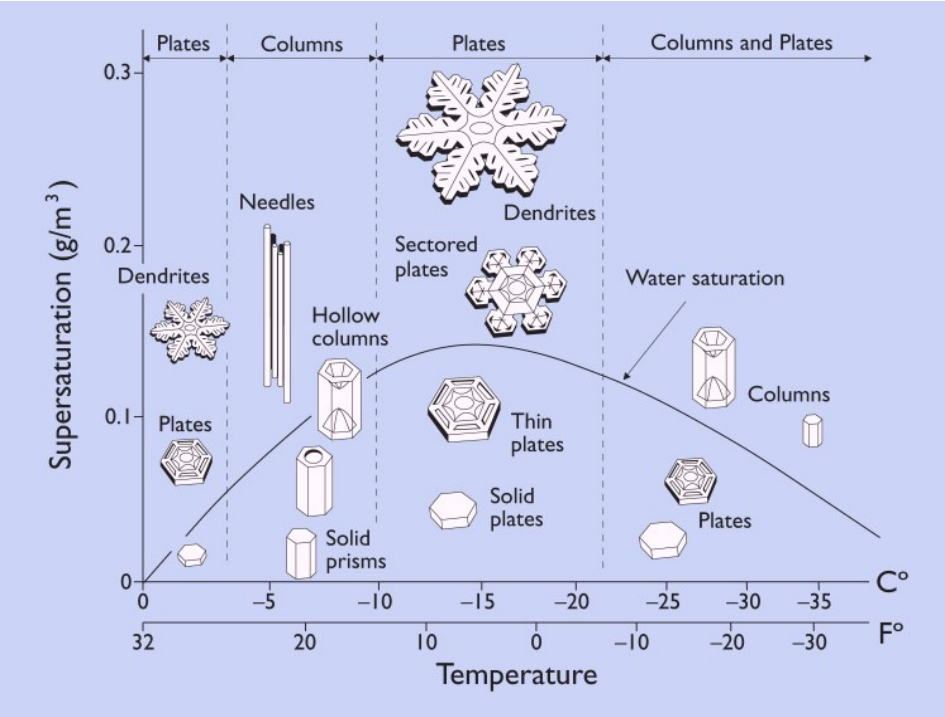
Tämä selittää kuitenkin vain pienen osan kuviossa yksi näkyvistä muodoista. Jos pelkkä liittymiskinetiikka akseleiden suhteen olisi määräävä tekijä, kiteet muodostuisivat aina joko yksinkertaisiksi laatoiksi tai pylväiksi. Tahkojen suuntaisen kasvun lisäksi tarvitaan haarautuvaa kasvua, joka synnyttää laattoihin liittyvät varret. (Libbrecth 2004, 40–56.)

Haarautuvan kasvun synnyttää ja sitä kontrolloi toinen diffuusion tyyppi, lämpödiffuusio. Aina kun vapaa molekyyli liittyy partikkelidiffuusiosta johtuen siemenkiteeseen, vapautuu prosessissa lämpöä. Tämä lämpö puolestaan ”työntää” ympärillään olevia vapaita molekyylejä kauemmas muodostuneesta kiderakenteesta. Koska molekyylien liittyminen on nopeinta kuusikulmaisen laatan kulmissa, nämä keräävät enemmän vapaana liikkuvia molekyylejä, samalla tyhjentäen lähialueensa ja kasvattavat pienet haarakkeet. Nämä haarakkeen alut puolestaan sisältävät enemmän valmiita vapaita kohtia uusille sidoksille. Tämä johtaa tilanteeseen, jossa haarat kasvavat nopeasti mutta muut tasaiset pinnat huomattavasti hitaammin vapaiden molekyylien puuttuessa. (Gravner – Griffeath 2006, 5,6; Libbrecht 2005, 863–865.)

Tasapainotus näiden kahden eri kasvuvaiheen välillä vaikuttaa huomattavasti lumikiteen saavuttamaan lopulliseen muotoon. Kasvuvaiheiden vaihteluun on syynä lumikiteen synnyn aikana tapahtuvat ympäristön muutokset, sillä olosuhteet pilvimassassa voivat vaihdella huomattavasti ajan myötä. Kiteet myös kulkeutuvat kasvunsa aikana yleensä pitkiä matkoja ennen, kuin ne saavuttavat nk. kriittisen massan ja satavat maahan asti. Kiteen kasvu voi myös hetkellisesti loppua, tai kide voi jopa osittain sulaa. (Libbrecht 2004, 3–50.)

Kuinka voidaan selittää kiteiden kuusikulmainen symmetria, jos olosuhteet ovat sitten näin vaihtelevia? On otettava huomioon, että vaikka olosuhteet vaihtelevat pilven eri osissa, ovat paikalliset kasvavan kiteen kokoisen kappaleen (0,01 mm – 1 mm) olosuhteet suhteellisen vakaat (Libbrecht 2004, 47).

Kiteiden kasvuun ja muotoutumiseen vaikuttavat eniten ympäristön lämpötila ja suhteellinen ilmankosteus (Libbrecht 2004, 42–45). Kokeiden perusteella on pystytty muodostamaan kaavio, joka ilmaisee, minkälaisissa olosuhteissa kunkin tyypin lumikiteitä muodostuu (Libbrecht 2004, 42–45). Tämä kaavio muodostaa työn perustan: se antaa osviittaa siitä, kuinka tulevan simulaation on kullakin syötteellä toimittava. Kuvio 4 osoittaa erityyppisten kiteiden muodostumisen.



Kuvio 4 Erityyppisten lumikiteiden esiintyminen lämpötilan ja kosteuden mukaan (Libbrecht 2007).

2.3 Simulaatiomallien kriteerit

Lumikiteitä on yritetty luoda tietokonemallien avulla vuosikymmenien aikana useilla eri menetelmillä ja tekniikoilla. Tulos on kuitenkin ollut hyvin vaihteleva, ja usein vain osa kiteen muodostuksesta on pystytty esittämään. Lähtökohdat eri mallien tarkastelulle antaa luvussa 1 esitetty työnrajaus. Nämä kriteerit asettavat myös rajat myöhemmin rakennettavalle visualisoinnille. Kriteerejä tarkennettiin työn edistymisen aikana työnantajan kanssa käydyissä keskusteluissa. Alla on esitettynä kriteereitä, joiden avulla sopivaa simulaatiomallia haetaan, ensimmäisenä olevalla on suurin painoarvo.

1. Kyky tuottaa erilaisia kiteitä eri kidekategorioista.
Visualisoinnin pääasiallisena tarkoituksena on esitellä lumikiteiden monimuotoisuutta. Tästä johtuen on tärkeää, että valittu malli pystyy tuottamaan kiteitä mahdollisimman monesta luvussa 2.2 esitellystä kidekategoriasta.

2. Selkeä kuusikulmainen symmetria sekä sisäisen rakenteen monimutkaisuus.
Tämän kriteerin täyttäminen tukee edellistä, sillä kiteet erottuvat toisistaan juuri poikkeamilla symmetriasta. Sisäisen rakenteen monimutkaisuuden säilyttäminen taas luo yksilöllisemmän/vaihtelevamman tuntuisia kiteitä ja luo illuusiota oikeasta kiteestä.

3. Reaaliaikaisuus simulaation ja visualisoinnin välillä.
Simulaation tulee olla ajettavissa lähes samanaikaisesti kuin sitä esitetään, ts. katsojan antaman syötteen ja vasteen välillä ei saa olla pitkiä taukoja. Tämä vaatimus myös rajoittaa ennalta laskettavien mallien käyttämistä. Varsinainen reaaliaikaisuus (1:1) ei ole välttämätöntä, sillä joidenkin mallien kohdalla aikaskaalaus on välttämätön miellyttävän tuloksen saavuttamiseksi.

4. Visuaalisesti mielenkiintoiset tulokset.
Päätavoite ei ole luoda pelkästään fyysisesti tarkkaa mallia, vaan visuaalisesti mielenkiintoista esitystä. Tarkoituksena on, että tavallinen näyttelyssä kävijä saa nopeasti selkeän kuvan lumikiteen muodostumiseen vaikuttavista seikoista. Mallin tuottaman datan on oltava suhteellisen helposti muokattavissa joko 3D-malliksi tai muuksi graafiseksi komponentiksi, joka voidaan esittää katsojalle.

5. Prosessi jäljitettävissä sekä mahdollisuus satunnaisuuteen.
Prosessin jäljitettävyydellä tarkoitetaan yksinkertaisesti sitä, että mallille annettu syöte tuottaa ennustettavissa olevia tuloksia. Sama syöte antaa aina yhtäläisen ulostulon ja prosessi on käännettävissä tuloksesta alkuun. Kyseisen ehdon toteuttaminen ei ole välttämätöntä mutta helpottaa mallin toiminnan tarkastelua. Mahdollisuus varioida prosessia satunnaisuudella on taas tärkeää monipuolisuuden ja luonnonmukaisuuden kuvaamiseksi, sillä juuri pienet paikalliset muutokset tekevät jokaisesta lumikiteestä ainutlaatuisen. Tällä siis kuvataan muiden fyysisten elementtien vaikutusta kuten tuuli, pienet variaatiot kosteudessa ja epäpuhtaudet.

2.4 Simulaatiomallien esittely ja vertailu

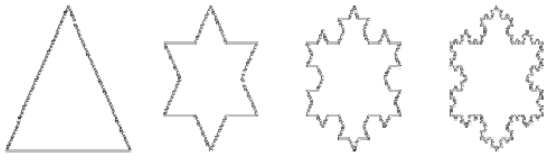
Edelliset luvut muodostavat kriteerit eri simulaatiomallien vertailulle, joista huomio työssä kiinnitettiin kolmeen lupaavimpaan ja eniten käytettyyn malliin lumikiteiden simuloinnissa.

DLA-algoritmi

DLA-algoritmi (Diffusion Limited Aggregation) on suosittu tapa simuloida kiteiden kasvua. DLA:ta on käytetty yleisesti muiden luonnonprosessien mallintamiseen salamaniskuista kasvien kasvuun. DLA:n perustana on n-ulotteinen solukko, jossa solukon keskellä on yksittäinen partikkeli. Tämän jälkeen jostain solukon reuna-alueelta lähetetään uusia partikkeleita, jotka kulkevat solukon läpi satunnaiskävelyä noudattaen. Osuessaan keskelle muodostuneeseen partikkeliin tai ryhmittymään ne jäävät tietyllä todennäköisyydellä tähän kiinni (Kim – Henson – Lin 2007). Tällä tavalla muodostuneet kokonaisuudet (aggregate) ovat yleensä erittäin haarakkeisia. Kasvua voidaan ohjata erilaisilla rajoitteilla tai säätelemällä partikkeleiden satunnaisliikettä ja tarttumistodennäköisyyksiä (Kim ym. 2007).
DLA:n suurin ongelma on sen vaatima prosessointiaika erityisesti simulaation edetessä pitemmälle (partikkeleiden määrä kasvaa nopeasti). Esimerkkinä tästä toimii Kim 2007, jossa esitellään hybridialgoritmi, jonka yhtenä osana on DLA-algoritmi. Yksittäisen lumikiteen simulointi tällä menetelmällä vie 2 tuntia tehokkaalla PC:llä (Xeon 2,66Ghz) (Kim ym. 2007). Prosessia voitaisiin huomattavasti nopeuttaa pienentämällä simulaation tarkkuutta, mutta on selvää, ettei tämä malli kelpaa reaaliaikaiseksi tarkoitetun esityksen pohjalle. On myös jossain määrin kyseenalaista, kuinka hyvin DLA-algoritmin synnyttämät partikkelikasaumat ovat selkeästi visualisoitavissa.

Fraktaalit ja L-systeemit

Lumikiteet sisältävät monia fraktaalimaisia piirteitä ja on useita esimerkkejä yrityksistä muodostaa lumikiteitä fraktaaleilla. Yksi ensimmäisiä on vuonna 1904 esitelty nk. Kochin lumihiutale, jossa lumihiutaleen kaltainen muoto muodostetaan jakamalla tasasivuista kolmiota rekursiivisesti lisäämällä uusi kolmio jokaiselle kyljelle.



Kuvio 5 Kochin lumihiutale, ensimmäiset iteraatiot.

Varsinaisen lumikidemallin luomiseen fraktaalit eivät sovellu hyvin, sillä hyvien tulosten aikaansaaminen on matemaattisilla funktioilla erittäin hankalaa. Yleisesti ottaen niillä voidaan kuvata vain yhtä kehittymisen osa-aluetta kerrallaan. Työtä voidaan tässä tapauksessa kuvailla satunnaishakuna funktion muodostamiseksi. L-systeemit (nimitys Aristid Lindemayerin mukaan) ovat eräänlaisia fraktaaleja, joissa imitoidaan esimerkiksi kasvin kasvuprosessia käyttämällä erilaisia korvaussääntöjä (Emmeche 1991, 81-82).

Soluautomaatit

Soluautomaatit yrittävät lähestyä ongelmia luonnollisella tavalla, jossa kompleksinen kokonaisuus muodostuu pienistä yksinkertaisista, lähes atomaarisista kappaleista, joita käsitellään rinnakkain. Soluautomaatteja on sovellettu menestyksellisesti useisiin kompleksisiin ongelmiin eri aloilta, mm. fysiikassa nesteiden dynamiikasta, tähtien synnystä, ekonomiasta aina biologisiin prosesseihin. (Kari 1990 13,14; Wolfram 2002 1–22.)

Soluautomaattien suurimpana etuna on niiden selkeys ja helppous toteuttamisessa. Automaatit rakentuvat yksinkertaisten sääntöjen pohjalta, eikä monimutkaisia laskentamenetelmiä vaadita niiden toteuttamiseen.

2.5 Perustelut soluautomaatin valinnalle

Huomioitavaa on, että jokainen edellä mainittu malli sisältää elementtejä, jotka muistuttavat toisia malleja tai käyttävät osittain samoja menetelmiä. Kummatkin edellä mainituista, DLA ja soluautomaatit, sisältävät fraktaalit. Toisaalta voidaan myös ajatella DLA:n ja soluautomaattien olevan jossain määrin fraktaalien erikoistapauksia.

Edellisistä malleista lähempää tutkimusta ja mallin toteutusta varten valitaan soluautomaatit. Soluautomaattien luonnollinen tapa lähestyä ongelmia ja melko yksinkertaisten laskennallistenmenetelmien käyttö vaikuttivat parhaalta vaihtoehdolta ratkaista ongelma. Automaateista oli myös saatavilla hyvin valmiita esimerkkejä ja kokeellisia tuloksia.

Automaatit ovat suhteellisen yksinkertaisia toteuttaa, ja niitä voidaan myös varioida melko helposti, joten ne soveltuvat hyvin laajempaan

testaukseen. Tämä on tärkeää sopivan toteutustavan valinnalle. Ne ovat myös henkilökohtaisesti kaikkein kiinnostavimpia laajan sovellettavuuden vuoksi ja niiden tapa lähestyä ongelmaa vaikutti luonnollisimmalta.

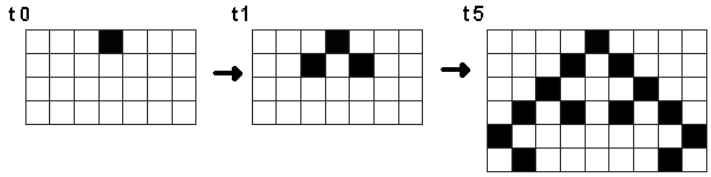
3 SOLUAUTOMAATTI KÄSITTEENÄ

3.1 Soluautomaattien perusteet

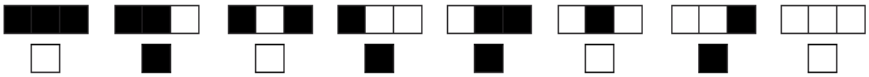
Soluautomaatti (engl. cellular automaton), on alun perin Von Neumanin ja Stanislav Ulamin n. 1950 muotoilema diskreettimalli, jota aluksi sovellettiin biologisten järjestelmien kuvaukseen ja tutkimukseen. Tästä biologisesta alkukohdasta johtuen mallia nimetään soluautomaatiksi, ja se on vakiintunut kuvaamaan ominaisuuksiltaan samankaltaisia malleja. Myöhemmin 1970-luvulla englantilainen matemaatikko John Conway kehitti ja teki tunnetuksi yhden tunnetuimmista soluautomaateista, Game Of Lifen. Tämä malli (GoL) herätti yleisesti kiinnostusta, koska yksinkertaisuudestaan huolimatta siinä esiintyi suppeasti erittäin monimutkaisia prosesseja. Tätä ennen oli yleisesti ajateltu, että kompleksisten ilmiöiden tarkka kuvaus, jos suurimmassa osassa mahdollistakaan, vaatisi yhtäläisten kompleksisten säännöstöjen luomista ja ratkaisemista. (Emmeche 1991, 59–64, 72–76; Wolfram 2002, 872.)

Varsinainen kultainen aikakausi soluautomaateille alkoi vuodesta 1981 lähtien, kun matemaatikko Stephen Wolfram aloitti laajan tutkimustyönsä. Hänen perusteelliset tutkimuksensa yksiulotteisista automaateista johtivat automaattien laajempaan tutkimukseen ja myöhemmin myös niiden sovellusten käyttämiseen hyvinkin toisistaan poikkeavien ongelmien ratkaisuun. Nämä ovat monessa tapauksessa kaukana alkuperäisestä alastaan, biologiasta. (Emmeche 1991, 72–76; Wolfram 2002, 880–882.)

Kuviossa 6 esitetään yksinkertaisin mahdollinen yksiulotteinen, kaksitilainen soluautomaatti ja sen kehittyminen ajan myötä. Automaattia kutsutaan yksiulotteiseksi, koska sen säännöstö (kuvio 6) koostuu vain samalla rivillä vierekkäin sijaitsevista soluista. Ajan etenemistä kuvataan esittämällä edellisestä rivistä saadut tulokset sen alla sijaitsevalla rivillä siten, että kuviosta muodostuu kaksiulotteinen kuva, jota on helppo tulkita. Solun tila seuraavalla ajan hetkellä eli alemmalla rivillä määräytyy sen yläpuolella sijaitsevien kolmen solun mukaan (kuvio 7).



Kuvio 6 Wolfram-sääntö 90.



Kuvio 7 Automaatin 90-säännöstö (Wolfram 2002.)[1]

Esimerkki sisältää kaikki automaattien perusominaisuudet, solukon (rivi), säännöstön sekä muutoksen ajankuluessa. Seuraavassa luvussa näitä tärkeitä ominaisuuksia ja niiden merkitystä käsitellään tarkemmin. Kannattaa huomioida, että automaateissa kiinnostavinta ei välttämättä ole itse senhetkinen tila, vaan automaatin läpikäymä historia, joka ilmenee, kun tarkastellaan automaatin läpikäymiä muutoksia.

3.2 Soluautomaatin ominaisuudet

Edellisessä luvussa esitelty yksiulotteinen automaatti sisältää kaikki automaattien perusominaisuudet, jotka voidaan myös formalisoida seuraavalla tavalla:

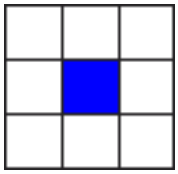
1. Solukko, automaatti muodostuu n–ulotteisesta äärettömästä hilasta, jossa hilan osina ovat solut.
2. Diskreetti ajallisesti sekä solullisesti, jokaisella solulla on ääreellinen tila ja tilamuutokset tapahtuvat yhtäaikaisesti rinnakkain jokaisen solun kohdalla.
3. Soluntila hetkellä t+1, määräytyy sihen liittyvien solujen tilasta hetkellä t.

Tarkastellaan lähemmin edellisiä ominaisuuksia ja käytetään niiden havainnollistamiseen esimerkkinä Game of Lifeä.

[1]Automaatit nimetään lukemalla säännöstöstä muodostuva binäärinen ”viivakoodi” vasemmalta oikealle (01011010 eli 90).

Solukko

Soluautomaattien solukon rakennetta tai ulottuvuuksien määrää ei ole tarkkaan määritetty. Ainoana vaatimuksena on, että jokainen solukon osa on samanarvoinen ja samalla tavalla liitettynä muihin soluihin. Tavallisimmin käsitellään yksi- tai kaksiulotteisia automaatteja niiden helpon tulkittavuuden ja hahmotettavuuden takia. Game of Lifessa solukko koostuu kaksiulotteisesta hilasta, jossa jokainen solu on liitettynä 8 viereiseensä soluun (Emmeche 1991, 18; kuvio 8).



Kuvio 8 Game Of Life-solukon rakenne.

Diskreetti

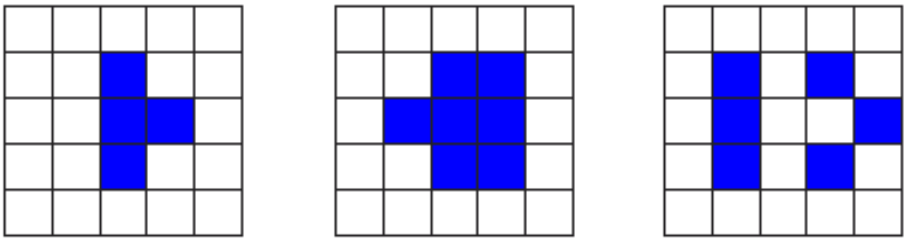
Toisena vaatimuksena on, että jokaisen yksittäisen solun kohdalla on selkeästi määriteltävissä, missä tilassa tietty solu on. Soluntila ilmaistaan yleensä kokonaislukuina ja esitetään tietyn värisenä havainnoinnin helpottamiseksi. Yleisimmin käytetty on yksinkertainen boolean jako, jossa solu saa tilakseen 0 tai 1. Tähän vaatimukseen liitetään myös vaatimus siitä, että jokaisen solun kohdalla tilamuutokset tapahtuvat yhtäaikaisesti. Automaattia ei päivitetä siten, että seuraavaan vaiheeseen siirryttäessä jo lasketut tilat vaikuttaisivat vielä päivittämättömään automaatin soluun (Emmeche 1991, 19). GoL:ssa noudatetaan aiemmin mainittua boolean jakoa (Emmeche 1991, 18), ’1’ merkitsee solun olevan elossa värinä musta, ’0’ taas vastaavasti kuollutta ja sen värinä on valkoinen. Edellisen mukaan voidaan todeta, että kuviossa 8 ainoastaan solukon keskellä oleva solu on elossa.

Säännöstö

Viimeinen kaikkien automaattien yhteinen ominaisuus on nk. säännöstö. Se määrittää, millä tavalla solukon muut osat vaikuttavat tietyn solun tilaan. Näistä muista soluista voidaan puhua myös solun naapurustona, soluista jotka ovat jollain tapaa sidoksissa siihen soluun, johon kiinnitetään huomiota (Emmeche 1991, 18). Kannattaa huomioida, että vaikka puhutaan naapurustosta, tämä sisältää usein myös itse

tarkasteltavan solun. Game of Lifessä säännöstö määritellään seuraavalla tavalla (kuvio 9):

1. Jos solulla on tarkalleen ottaen kolme elävää naapuria hetkellä t, sen tila on hetkellä t+1 elävä.
2. Jos solulla on hetkellä t kaksi elävää naapuria, se säilyttää nykyisen tilansa.
3. Kaikissa muissa tapauksissa solu ’kuolee’.
   (Emmeche 1991, 18.)



Kuvio 9 Game Of Life, esimerkki iteraatioita.

Säännöstön monimutkaisuus

Säännöstön monimutkaisuudella tarkoitetaan sitä, kuinka monta erilaista sääntöyhdistelmää pystytään muodostamaan naapuruston ja tilojen suhteesta. Alla on esitettynä kaava kokonaisyhdistelmien määrän laskemiseksi.

$$b^{b^k} \qquad (1)$$

jossa b = tilojen lukumäärä
K = naapuruston laajuus

Kyseisen esityksen mukaan edellisessä luvussa esitettyjä yksiulotteisia automaatteja voidaan muodostaa 256 erilasta, GoL:n tapauksessa (kaksitilainen automaatti, jossa 9 solua) mahdollisia yhdistelmiä on jo 2^{512}. (Wolfram 2002, 928.)

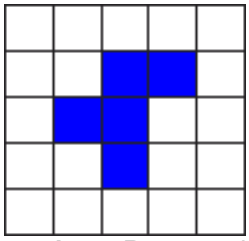
Kuvatut ominaisuudet määrittävät täysin kunkin automaatin toiminnan. Jokainen tässä työssä esitelty automaatti noudattaa edellä mainittua rakennetta.

Konfiguraatio

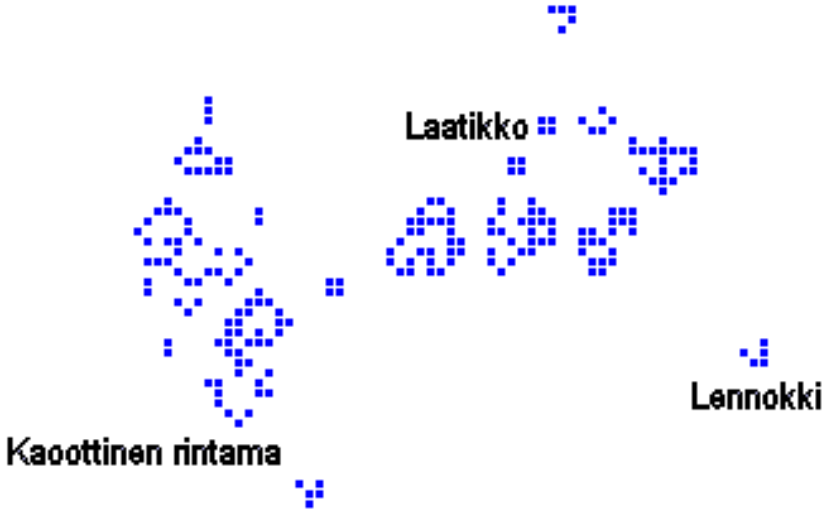
Konfiguraatioksi sanotaan sitä solukon alustavaa tilaa, josta automaatin laskeminen aloitetaan. Konfiguraatiota voidaan kuvata siten, että se käsittää ääreellisen määrän soluntiloja, jotka poikkeavat äärettömän solukon perustilasta. Esimerkiksi aiemmin esitellyn yksiulotteisen automaatin konfiguraatioksi voidaan määritellä "yksi musta solu kaikkien muiden ollessa valkoisia".

Solukon konfiguraatiolla on suora vaikutus siihen millaiseksi solukon kehitys ajan suhteen muodostuu (Wolfram 2002, 24–31). Kannattaa huomioida, ettei mikään pakota ottamaan lähtökohdaksi yhden solun kongifuraatiota, vaan yhtä hyvin voidaan ottaa tilanne, jossa kaikkien solujen tilat arvotaan satunnaisesti (Wolfram 2002, 223) tai valitaan tietyt solut. Lumikiteissä siemenkiteen ja sitä ympäröivän ilmankosteuden voidaan ajatella olevan eräänlainen konfiguraatio.

Edellä annettujen esimerkkien perusteella voisi intuitiivisesti päätellä, ettei näin yksinkertainen järjestelmä voi tuottaa mitään monimutkaista tai kiinnostavaa. Tämä osoittautuu nopeasti harhaksi, kun tarkastellaan, kuinka paljon GoL:ia ja sen erilaisia lähtökohtia on tutkittu. GoL:ssa on havaittu monia alkukonfiguraatioita, jotka säilyvät, muuntuvat tai luovat uusia rakenteita. Kuviossa 10 esitellään yksi mielenkiintoisemmista GoL:n konfiguraatioista, nk. r-pentomino. (Emmech 1991, 70–75; Wolfram 2002, 50, 964.)



Kuvio 10 R-pentomino.



Kuvio 11 R-pentomino 153 iteraatio.

Kuvio 11 esittää saman pentominon 153 iteraation jälkeen, jolloin se on kasvanut huomattavasti monimutkaisemmaksi. Kuviossa esiintyy useita erityyppisiä rakenteita, lennokkeja, jotka ovat erottautuneet muista kehittyvistä soluista ja matkaavat kohti solukon reunaa ja laatikoita, jotka ovat pienimpiä vakaatilaisia rakenteita (4 solua neliönä). Jos yksinkertaisilla säännöillä varustettu järjestelmä voi tuottaa näin monimutkaisia kokonaisuuksia, voidaan ajatella, että muidenkin monimutkaisten ilmiöiden taustalta löytyy yksinkertaisia sääntöjä. R-pentomino on hyvä esimerkki herkkyydestä alkuarvoille, sillä mikään muu pentomino muodostelma ei muodosta samalla tavalla kasvavaa järjestelmää Game of Lifessä. Kaikki muut siirtyvät muutaman iteraation jälkeen sykliseen toistoon tai kuolevat kokonaan. (Emmeche 1991, 19,20, 70–75; Wolfram 2002, 965.)

Soluautomaatit osoittavat monimuotoisuutta ja kompleksisia rakenteita tiettyjen ehtojen välissä. On olemassa piste säännöstöissä, joka määrää kompleksisen rakenteen syntymisen minimalistisista säännöistä, eikä säännöstöjen monimutkaisuuden ja lisäehtojen tuottaminen välttämättä muodosta suoraan kompleksisempia rakenteita (Wolfram 2002, 351–361). Päinvastoin tuloksena voi olla, ettei rakenteita pysty syntymään. Tätä voidaan pitää eräänlaisena osoituksena siitä, että

mielenkiintoisten tulosten synnyttämiseksi tarvitaan tietynasteista kaaosta, kun taas liiallinen satunnaisuus rikkoo rakenteet, eikä prosessista synny kuin kohinaa tai täydellinen monotonia. Oleellista on, että säännöstö sisältää riittävästi elementtejä monimutkaisuuden synnyttämiseen, automaatin solukon muodolla (1,2..n-ulotteinen) ei siihen ole juurikaan vaikutusta (Wolfram 2002, 170).

Soluautomaatit voidaan kategorisoida neljään eri luokkaan niiden käyttäytymisen perusteella (Wolfram 2002, 231).

1. Triviaali
   Riippumatta automaatin saamasta konfiguraatiosta nämä automaatit muodostavat hyvin nopeasti yhtäläisen, monotonisen tilan (uniform state).

2. Rekursiivinen
   Rekursiiviseen ryhmään kuuluvat kehittyvät itsesimilaarisiksi, toistuviksi rakenteiksi. Aiemmin esitelty yksiulotteinen Wolframin automaatti kuuluu tähän ryhmään.

3. Satunnainen
   Ajanmyötä näissä ei muodostu minkäänlaista selkeää rakennetta lukuunottatta pieniä satunnaisesti ilmeneviä muotoja.

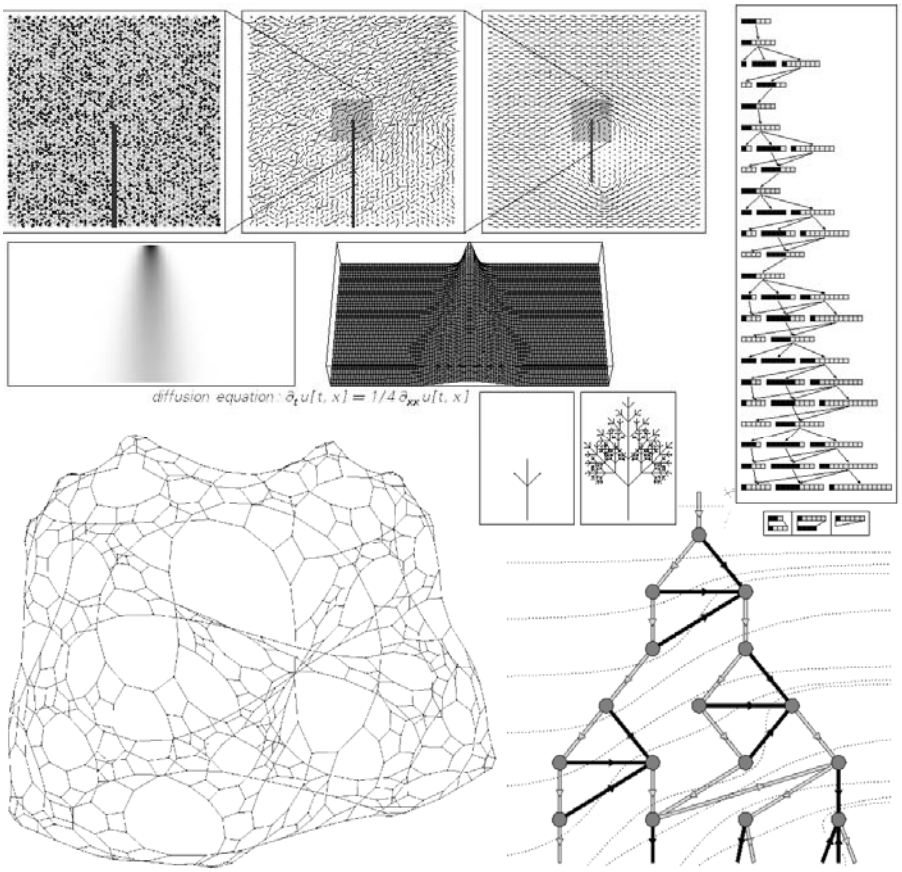
4. Kompleksinen
   Monimutkaisin automaattien tyyppi, jotka yhdistävät kahden edellisen automaatin käyttäytymisen. Tuloksena on kaoottinen järjestelmä, jossa pieniä selkeitä rakenteita syntyy ja ne ovat ajan edetessä vuorovaikutuksessa toistensa kanssa. Game Of Life kuuluu tähän luokkaan.
   (Wolfram 2002, 231–250.)

Universaalius

Universaaliudella (engl. universality) kuvataan ilmiötä, jossa systeemi pystyy periaattessa simuloimaan minkä tahansa muun systeemin toimintaa. Parhaiten tunnettu esimerkki universaaliudesta ovat tietokoneet, jotka pystyvät ohjelmoinnin kautta suorittamaan erilaisia, toisistaan poikkeavia tehtäviä (Wolfram 2002, 642). Myös joidenkin soluautomaattien on osoitettu omaavan kyvyn universaaliin käyttäytymiseen.

Kaikki näistä ovat edellä esitellyn luokittelun mukaan luokan neljä automaatteja. Esimerkiksi Game of Life ja yksiulotteisista automaateista sääntö 110 ovat universaaleja. Tämä universaalius antaa perusteet olettaa, että soluautomaateilla olisi mahdollista simuloida lumikiteiden syntyprosessia. Tämä on toki tehottomampaa kuin itse prosessi luontaisesti, mutta tässä yhteydessä riittää tieto siitä, että prosessia pystytään myös matkimaan yksinkertaisilla säännöillä, kuten tämän luvun lopussa oleva boolean kideautomaatti osoittaa. (Wolfram 2002, 642–644, 691–694.)

Kuvio 12 antaa paremman kuvan automaattien monimuotoisuudesta ja tuo esiin, etteivät automaatit aina välttämättä ole pelkästään moniulotteisia taulukoita, vaan niiden sovellusalueet ovat laajempia.



Kuvio 12 Erilaisia soluautomaatteja (Wolfram 2002).

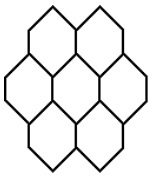
Reunaehdot

Soluautomaattien ajatellaan teoriassa koostuvan äärettömästä solukosta, jossa ei ole rajoja (Wolfram 2002, 20–37). Käytännön toteutuksissa tästä joudutaan ymmärrettävästi tinkimään, ja on olemassa useita tapoja käsitellä tätä ongelmaa. Näistä yleisin tapa ratkaista ongelma on yksinkertaisesti määrittää ”riittävän suuri” solukko ja katkaista automaatin laskeminen tietyn aikarajan perusteella. Riittävä koko selvitetään suorittamalla testauksia tai laskemalla ja yrittämällä löytää mahdollisimman hyvä tasapaino suorituskyvyn ja halutun kaltaisten tulosten osalta.

Toinen tapa on määrittää aluksi rajallinen solukko, jota kasvatetaan dynaamisesti. Kun kasvulle "merkitykselliset" solut lähenevät solukon rajoja, laajennetaan solukkoa tietyn verran. Tässä lähestymistavassa ongelmalliseksi voi muodostua merkityksellisten solujen määrittäminen, jos kyseessä ei ole yksinkertainen binääriautomaatti. Toinen selkeä ongelma syntyy ilman kasvun rajoittamista. Jos laajenemisnopeus on suuri ja automaatin solukko moniulotteinen, tulee automaatin laskeminen järkevässä ajassa hyvin nopeasti mahdottomaksi. Kolmantena vaihtoehtona on solujen liittäminen toisiinsa siten, että solukon reunalla sijaitsevat solut muodostavat toistensa naapuruston.

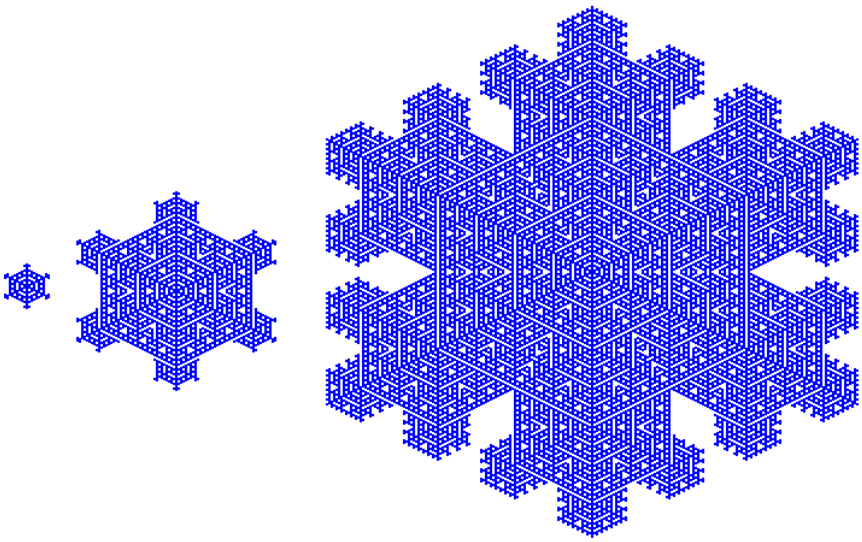
Yhteys lumikiteisiin

Wolfram esittelee kirjassaan New Kind of Science yksinkertaisen boolean automaatin lumikiteiden mallinnukseen. Tämä Packardin jo aiemmin esittelemä automaatti tuottaa muotoja, jotka muistuttavat aitoja kiteitä erityisesti symmetriansa ja rajoitetun kasvunsa takia.



Kuvio 13 Boolean kiteen solukonrakenne.

Automaatin solukko (kuvio 13) muodostuu toisiinsa liittyneistä kuusikulmaisista laatoista, joista jokaisella solulla on kuusi naapurisolua. Tilojen kuvaamiseen käytetään boolean jakoa (Wolfram 2002, 369–371). Tämä hunajakennomainen rakenne muistuttaa aiemmin luvussa 2 esiteltyä vesimolekyylien järjestymistä niiden jäätyessä. Päivityksessä noudatetaan yksinkertaista sääntöä: jos yksi ja vain yksi naapuriston soluista on edellisellä iteraatiolla sininen, tämä solu muuttuu siniseksi (kuvio 14). On olemassa 16 vastaavaa säännöstöä samalla rakenteella, jotka tuottavat lumikidemäisiä muotoja (Gravner – Griffheath 2006).



Kuvio 14 Wolframin lumikiteen kehitys (vrt. liite 1).

4 SOLUAUTOMAATTIEN SOVELTAMINEN

4.1 Simulaation kuvaus ja rajaus

Boolean automaatti pystyy esittämään vain erittäin rajatun määrän lumikidemäisiä rakennelmia. Se antaa osviittaa siitä, että automaatit pystyvät simuloimaan kiteitä, mutta kyseinen toteutustapa on liian rajoittunut. Tavoitteena on löytää sellainen automaatti ja se säännöstöjen joukko, joka jäljittelee mahdollisimman monia erilaisia kiteitä. Säännöstön on myös pystyttävä luomaan stabiileja konstruktioita, jotka kasvavat ajan myötä, sillä ilman näitä on kasvusta visuaalisen esityksen luominen mahdotonta.

Simulaatiota suunniteltaessa ja automaatti valittaessa on otettava huomioon vaatimus reaaliaikaisuudesta. Tämä asettaa selkeän rajan sille kuinka monimutkainen käytettävä automaatti voi olla, jotta sitä voidaan laskea lähes reaaliajassa tavallisella PC-kokoonpanolla.

Monimutkaisuuteen liittyvät automaatin solukon koko, naapuruston muodostava säännöstö sekä automaatin tarvitsema iterointien määrä. Jo työn alkuvaiheessa tuli selväksi, että varsinaisten 3D-automaattien ja mallien toteuttaminen tällä tavalla oli lähes mahdotonta omilla resursseilla ja käytettävissä olevalla ajalla. Jo pelkästään automaatin jokaisen solun käsittely voi osoittautua 3-ulotteisessa automaatissa liian raskaaksi tehtäväksi ilman erityisiä metodeja.

Suhteellisen pieniresoluutioisella 128x128x128 solukolla pelkästään solujen määrä ylittää kahden miljoonan rajan, naapuruston koko taas kahdeksansoluisessa rakenteessa on yli 16 miljoonaa. Varsinaisen visualisoinnin toteutukseen ratkaisuksi olisi saattanut kelvata vokseligrafiikkaan perustuva visualisointi, mutta esimerkkien ja kokemuksen puute tällä alueella sai hylkäämään tämän vaihtoehdon. Liian laajasta työalueesta ja riittämättömästä tiedosta 3D-automaattien soveltamisesta lumikiteiden mallinnukseen, huomio suunnattiin kaksiulotteisiin tapauksiin.

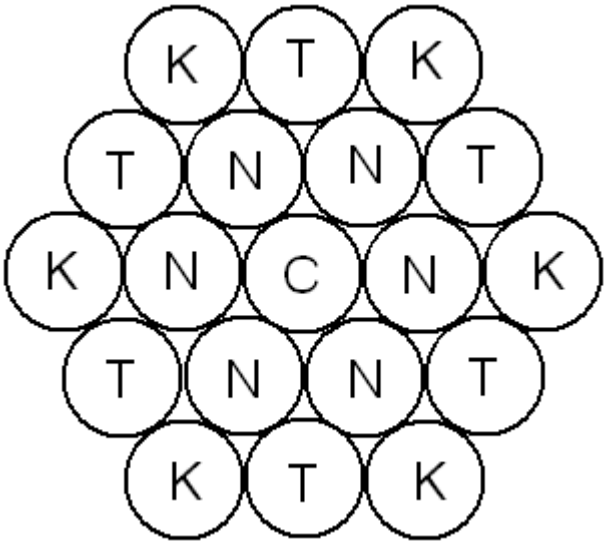
Eräs mahdollisuus ongelman ratkaisuun olisi käyttää kaksiulotteista automaattia ja luoda tämän perusteella 3D-malli. Tämän lähestymistavan ongelmana kuitenkin on se, ettei tällä tavalla saada aikaan monien pylväsmäisten kiteiden sisäisiä rakenteita, jotka ovat oleellinen osa niiden mielenkiintoisuudesta. Toinen ongelma on edelleenkin 3D-mallin reaaliaikainen luominen. Näistä ongelmista johtuen päätettiin työantajan kanssa, että työssä kannattaa keskittyä tasomaisten kiteiden mahdollisimman tarkkaan simulointiin, koska nämä ovat kaikkein monimuotoisimpia.

4.2 Soluautomaattien toteutus

Koska aiemmin esitellyt boolean mallin automaatit eivät tuota riittävää monimuotoisuutta, on etsittävä muun tyyppisiä automaatteja ratkaisuksi. Yhden tällaisen esittelee Clifford Reiter tekstissään Fuzzy Hexagonal Automata. Nk. sumea automaatti perustuu reaalilukujen käyttämiseen solun tilan ilmaisemiseksi kokonaislukujen sijasta.

Automaatin kuvaus

Sumean soluautomaatin solukon rakenne on samanlainen kuin aiemmin esitellyssä boolean automaatin tapauksessakin, eroa on kuitenkin naapuruston määrittelyssä. Naapurusto on sumeassa soluautomaatissa huomattavasti monimutkaisempi. Kuvio 15 havainnollistaa kuusikulmaisen automaatin rakennetta ja naapurustosuhteita. Lähimpien naapurisolujen (N) lisäksi automaatissa otetaan huomioon myös seuraavan tason solut jakaen ne kahteen ryhmään. Automaatissa erotellaan kärkipisteet (K) sekä tahkot (T) (Reiter – Coxe 2007), joille löytyy suora analogia luvussa 2 esitellyssä kiteen rakenteessa.



Kuvio 15 Reiterin automaatin rakenne.

Säännöstö

Solun tila ilmaistaan arvovälillä 0–1, jossa luvun ajatellaan ilmaisevan jäsenyysastetta sumeasta joukosta. Tässä joukossa 1 tarkoittaa täysin jäätynyttä solua, joka kuuluu kiteeseen ja 0 vastaavasti täysin höyrymäistä (Reiter – Coxe 2007). Tässä tapauksessa automaatin säännöstöjen mahdollinen lukumäärä on riippuvainen siitä, millä tarkkuudella liukuluvut ilmaistaan (32 bittisessä $1.128430269*10^{366}$).

Konfiguraatio ja päivittäminen

Automaatin konfiguraatioksi perustapauksissa määritellään yksi täysin jäätynyt solu (S = 1), muiden saadessa yhteisesti arvokseen 0–0,5 näin kuvaten taustakosteutta (Reiter – Coxe 2007). Päivitysvaiheessa jokaisen solun kohdalla määritellään ensin naapuruston sisältämien jäätyneiden solujen määrä (ts. S > 0,5), jossa otetaan huomioon myös naapurisolujen suhde kyseiseen soluun (Reiter–Coxe 2007). Jäätyneiden solujen perusteella määritetään painokertoimet, jotka vaikuttavat solun tilan laskentaan. Taulukko 1 sisältää esimerkkijaon. Kertoimien määrityksen jälkeen solun tila lasketaan käyttäen lineaarista yhtälöä:

$$S = \alpha x + \sum_{y \in Nx} (\beta y) + \sum_{z \in Kx} (\gamma z) + \sum_{w \in Tx} (\delta w) \qquad (2)$$

jossa x = päivitettävän solun arvo
Nx, Kx, Tx = solun naapurusto (kuvio 15)
α, β, γ, δ = painoarvo (taulukko 1)
(Reiter – Coxe 2007.)

Taulukko 1 esimerkki painoarvot (Reiter – Coxe 2007).

(C, N, K)	(α, β, γ, δ)
(C=1, 0≤N≤6, 0≤ K ≤ N)	(1, 1/300, 0, 0)
(C=0, N=0, K=0)	(1, 0, 0, 0)
(C=0, N=1, K =0)	(1, 1/6, -1/12, -1/12)
(C=0, N=1, K =1)	(1/5, 3/10, 1/10, -1/10)
(C=0, N=2, K =0)	(1/20, 2/5, -3/20, -3/20)
(C=0, N=2, K =1)	(1/10, 3/10, -1/10, -1/10)
(C=0, N=2, K =2)	(1/22, 4/11, -3/22, -3/22)
(C=0, N≥3, K =0)	(1/18, 4/9, -1/6, -1/6)
(C=0, N≥3, K =1)	(1/18, 4/9, -1/6, -1/6)
(C=0, N≥3, K =2)	(1/22, 4/22, -3/22, -3/22)
(C=0, N≥3, K ≥3)	(1/18, 4/9, -1/6, -1/6)

Haarautuvaa kasvua edistetään automaatissa suosimalla päivitysvaiheessa kärkipisteiden vaikutusta. Vastaavasti solut joiden naapurustoon ei kuulu jäätyneitä kiteitä (C = 0, N = 0, K = 0) jätetään kokonaan huomiotta, eli taustakosteudessa ei tapahdu virtausta. Automaatti pystyy toteuttamaan myös aiemmin mainitun boolean kidekasvun määrittelemällä painoarvot sopivasti. (Reiter – Coxe 2007.)

Vaikka Reiterin esittelemä sumea soluautomaatti tuottaakin lumikiteitä muistuttavia muotoja, on kyse kuitenkin harhasta sillä tausta-arvojen muuntelu ei sinänsä ajanmyötä synnytä erilaisia tuloksia. Tuloksia tarkastellaan lähemmin seuraavassa alaluvussa.

Gravner/Griffeath (GG)

Hiukan erilaisen lähestymistavan antaa Janko Gravnerin ja David Griffeathin kehittämä menetelmä lumikiteiden mallinnukseen. Gravner/Griffeath eivät itse puhu menetelmästään soluautomaattina, vaikka se sisältääkin hyvin monia automaatille tyypillisiä ominaisuuksia.
GG:n toiminta perustuu edellisenä esiteltyyn Reiterin automaattiin, tehden kuitenkin selkeän eron jäätymisen erivaiheille ja siinä kiinnite-

tään erityistä huomiota jään materiaaliominaisuuksien mallintamiseen (Gravner – Griffheath 2006). Kuten aiemminkin automaatti noudattaa solukon rakenteen puolesta kuusikulmaista symmetriaa. Naapurusto määritellään samoin, kuin aiemmin boolean kiteen tapauksessa, vain kuusi lähintä solua ovat merkitseviä.

Automaatissa noudatetaan boolean jakoa kiteeseen kuuluvien ja kiteeseen kuulumattomien solujen välillä. Tämän lisäksi jokaiseen soluun liitetään kolmea eri "olomuotoa" kuvaava painoarvo, jotka ovat vesihöyry, kvasineste sekä jää, joiden suhdetta solussa kuvataan reaaliluvuilla. Automaatissa siis yhdistetään aiemmin esitelty boolean automaatti, sekä reaalilukuihin perustuva automaatti. Mallissa erotellaan rakenne itse kiteen, sen reuna-alueen sekä ympäröivän taustakosteuden kohdalla. Kide sisältää ainoastaan aiemmin mainituista muuttujista ainoastaan jäätä, reuna-alue kaikkia kolmea, sekä taustakosteus vain vesihöyryä. (Gravner – Griffheath 2006.)

Automaatin konfiguraatio muodostuu yhdestä solukon keskellä sijaitsevasta jäätyneestä solusta. Kaikki muut solut ovat samanarvoisesti höyrymäisessä tilassa, eli kuten aiemminkin tällä kuvataan taustakosteutta, saturaatiota ρ. (Gravner –Griffheath 2006.)

Algoritmi

Jokaisella iteraatiolla automaatti käy läpi 4 vaihetta (Gravner – Griffheath 2006), jotka noudattelevat aiemmin luvussa 2 esitettyä lumikiteen muodostumisprosessia. Päivityksen erivaiheet on nimetty muodostuksessa tapahtuvien prosessien mukaisesti.

Algoritmissa reaaliluvuilla ilmaistavat ominaisuudet jaetaan seuraavalla tavalla:

Diffuusivinen massa (algoritmissa symboli d)
Diffuusiivisella massalla tarkoitetaan vesihöyryä, ts. vapaasti liikkuvia vesimolekyylejä.

Kvasineste (symbolina n)
Kvasineste kuvaa sitä veden tilaa, jossa jäätynyt vesi ei vielä ole kiderakenteeltaan jähmeä, vaan ilmenee eräänlaisena nestämäisenä jää-

nä jolla on omat ominaisuutensa. Ilmiö esiintyy yleisesti sulavan tai jähmettyvän jään pinnalla muodostaen erittäin liukkaan kerroksen (faasin).

Jäätynyt massa (symbolina k)
Kuvaa kiteeseen liittynyttä jähmeän kuusikulmaisen rakenteen saavuttanutta massaa, joka reagoi erittäin huonosti ympäristön kanssa.

Lisäksi voidaan erotella rakenteeseen liittyen,

at(x) = kuuluuko solu kiteeseen (0,1)
Nx = naapurusto
Kt = lumikiteeseen kuuluvat solut
δKt = reuna-alueeseen kuuluvat solut
K_t^c = kiteeseen kuulumattomat solut
$\overline{K}_t^c$ = kiteeseen ja reuna-alueeseen kuulumattomat solut
(Gravner – Griffheath 2006.)

1. Diffuusio

Jokaisella päivityksellä lumikiteeseen ja reuna-alueeseen kuulumattomien solujen diffuusivinen massa muuttuu kaavan 3 mukaisesti.

$$x \in \overline{K}_t^c$$

$$d'_t(x) = \frac{1}{7} \sum_{y \in Nx} d_t^o(y) \qquad (3)$$

Reuna-aluetta päivittäessä jokaisen kiteeseen kuuluvan solun kohdalla korvataan tämän diffuusivinen massa solun itsensä tämän hetkisellä diffuusivisella massalla $d_t^o(x)$. (Gravner – Griffheath 2006.)

2. Jäätyminen

Jokaisellä päivityksellä κ's osa reuna-alueen diffuusivisesta massasta jäätyy (härmistyy) ja lopusta (1-κ) tulee kvasineste massaa.

$$x \in \delta K_t^c$$

$$n'(x) = n_t^o(x) + (1-\kappa) d_t^o(x) \quad (5)$$

$$k'(x) = k_t^o(x) + \kappa d_t^o(x) \quad (6)$$

(Gravner – Griffheath 2006.)

3. Liittyminen

Liittymisvaiheessa jokaisen reuna-alueen solun kohdalla tarkistetaan, liittyykö tämä solu päivityksen kohdalla itse kiteeseen, ts. partikkelidiffuusio. Tässä vaiheessa määräväksi tekijäksi nousee, kuinka monta jäätynyttä naapurisolua kyseisellä solulla on. Mitä enemmän jäätyneitä soluja naapurustossa sijaitsee (eli vapaita liitoskohtia), sitä helpommin solu liittyy itse kiteeseen. (Gravner – Griffheath 2006.)

1 tai 2

Tämä vastaa tilannetta, jossa solu sijaitsee kiteen haarakkeen päässä tai tasaisella reunalla. Tällöin solun liittymistä kiderakenteeseen kontrolloi β arvo.
Jos,

$$n_t^o(x) \geq \beta \rightarrow a_t'(x) = 1 \quad (7)$$

(Gravner – Griffheath 2006.)

> 3

Tilanne jossa solu sijaitsee syvennyksessä, solun liittymistä kiteeseen kontrolloi tällöin α ja θ arvot. Solu liittyy kiteeseen automaattisesti, jos sen reuna-alueen massa > 1.

$$n_t^o(x) \geq 1 \rightarrow a_t'(x) = 1 \quad (8)$$

$$\sum_{y \in Nx} d_t^o(y) < \theta \wedge n_t^o(x) > \alpha \rightarrow a_t'(x) = 1 \quad (9)$$

(Gravner – Griffheath 2006.)

Tilanteessa, jossa solulla on enemmän kuin neljä naapuria, solu liittyy kiteeseen automaattisesti. Liittymiskinetiikka on tässä kohtaa niin suuri, ettei muulla ole väliä. Tämä estää myös yksittäisten solujen jäämisen kiteen ulkopuolelle. (Gravner – Griffheath 2006.)

Kaikissa tapauksissa solun liittyessä kiteeseen senhetkisestä reuna-alueen kvasinestemassasta tulee jäämassaa. Kiteeseen liittyneet solut jätetään kiteeseen eikä niitä enää käsitellä tämän jälkeen.

4. Sulaminen

Viimeinen vaihe kiteen muodostamisessa on sulaminen, joka tapahtuu mallissa reuna-alueilla. Sulamisessa μ's osa reuna-alueen massasta muuntuu takaisin diffuusiviseksi massaksi, sama tapahtuu γ-parametrin määrittelemälle osalle kidemassasta (sublimoituminen).

$$n'_t(x) = (1-\mu)n_t^o(x) \quad (10)$$

$$k'_t(x) = (1-\gamma)k_t^o(x) \quad (11)$$

$$d'_t(x) = d_t^o(x) + \mu n_t^o(x) + \gamma k_t^o(x) \quad (12)$$

(Gravner – Griffheath 2006.)

Satunnaisuus

Vaihtoehtoisesti edelliseen algoritmiin voidaan asettaa viides askel, jolla kuvataan satunnaisuutta (Gravner – Griffheath 2006). Satunnaisuuden saavuttamiseksi muokataan jollain menetelmällä ja halutulla todennäköisyydellä diffuusiivista massaa. Pienet erot diffuusiivisessa massassa riittävät synnyttämään epäsymmetriaa kasvun aikana, johtuen suuresta iteraatioiden määrästä.

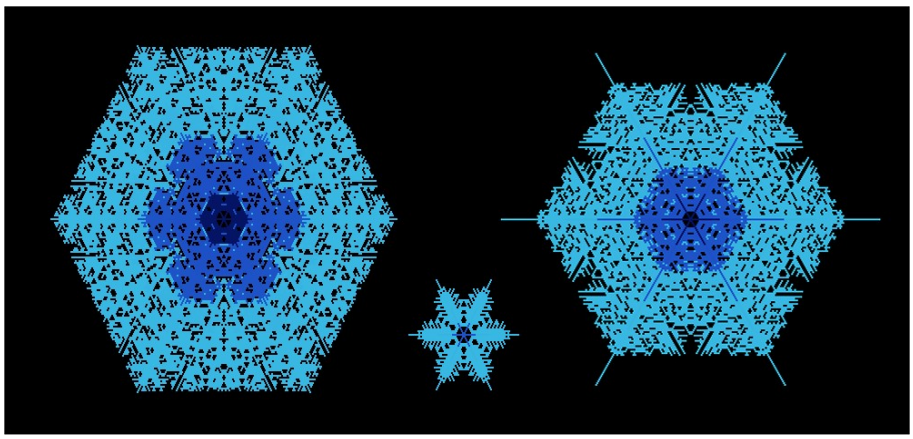
4.3 Lumikiteiden generointi

Reiterin automaatin tapauksessa tuloksien muodostamiseksi ei tarvita laajoja testausajoja, sillä ainoa merkityksellinen parametri on automaatissa käytetty taustakosteuden määrä. Automaatti pystyy ilmaise-

maan lumikidemäisiä muotoja ja on ajettavissa hyvin reaaliaikaisena 512x512 solukolla.

Tulokset eivät kuitenkaan ole kovin hyviä, sillä taustakosteuden lisääntyessä konfiguraatiossa automaatti muodostaa laattamaisia muotoja, päinvastoin kuin reaalimaailmassa. Tätä ongelmaa huomattavampi on se, että vaikka kiteet näyttävät tietyillä ajanhetkillä erilaisilta (kuvio 16), ne itse asiassa kehittyvät aina samalla tavalla noudattaen suurin piirtein samanlaista kaavaa. Tämä ei ole automaatin rakennetta katsottaessa kovinkaan yllättävää.

Taustakosteuden lisäksi muokattavissa ovat lähinnä jäätyneiden kiteiden mukaan muodostettavat painoarvot, mutta näiden muokkaaminen ei juurikaan tuota haluttuja tuloksia, eivätkä muuta sitä perusseikkaa, että automaatti toistaa ajan kuluessa lähinnä samaa kuusikulmaista symmetriaa ja muotoa. Tulokset eivät ole tässä suhteessa kovin tyydyttävät, ja parempaa vaihtoehtoa on etsittävä edelleen.



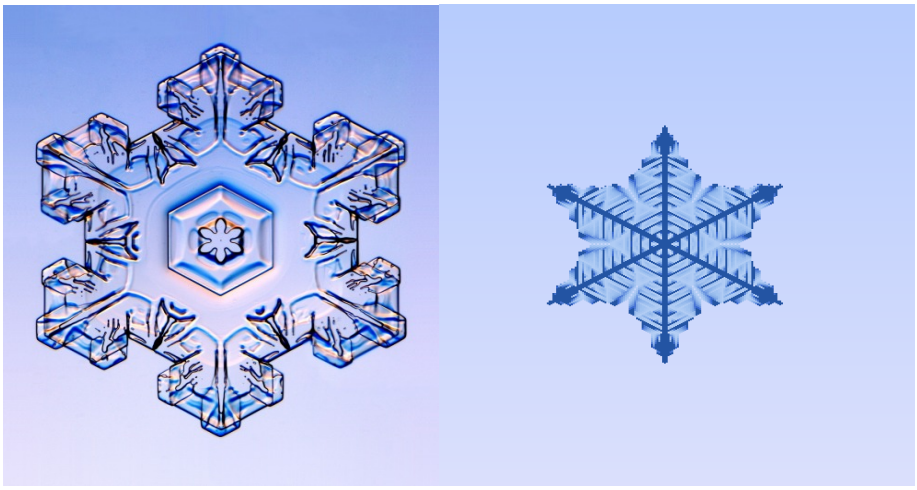
Kuvio 16 Reiterin automaatin tuloksia (värit liioiteltuja selkeyden vuoksi).

Gravner/Griffheath

Koska GG:n automaatin toiminta perustuu huomattavasti useamman parametrin, (7 tai 8) käyttöön muodostettiin useita testisarjoja. Näiden testisarjojen puitteissa automaatilla luotiin kymmenittäin erilaisia kiteitä käyttäen apua GG:n esittämiä arvoja ja toiminnan selostusta sekä valitsemalla satunnaisia arvopareja uusien mahdollisten vaihtoehtojen löytämiseksi. Tällä tavalla nähtiin parempi otos automaatin tuottamista

tuloksista ja siitä, miten ne olisivat hyödynnettävissä simulaation toteutuksessa.

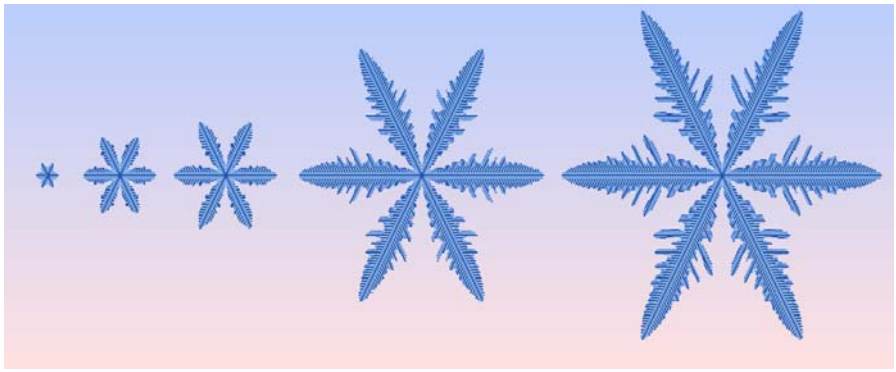
Kuva 1 esittää yhden erinomaisen ehdokkaan, jonka tämä sarja tuotti vertailtuna Libbrechtin kuvaamiin erilaisiin kiteisiin. Kyseessä on luvussa 2 esitellyn luokittelun mukaisesti Sectored Plate. Yleisen muodon lisäksi malli kykenee ottamaan huomioon sisäisen rakenteen muodostumisen luomia yksityiskohtia. Parhaana esimerkkinä on kummassakin ytimen ympärille muodostunut sisäinen kuusikulmio ja sen sisäisen minitähti, vaikkakin simuloidussa kiteessä tähden keskusta puuttuu[2]. Toinen kiinnostava yksityiskohta on kummassakin kiteessä levyjen päästä työntyvät uuden kasvun alut, jotka ilmaisevat että kasvu on muuttumassa enemmän haaroittuvaksi.



Kuva 1 Yksi testisarjan kiteistä vertailtuna aitoon (aito Libbrecht 2007).

GG:n automaatti tuottaa selkeästi parempia tuloksia, kuin edellä mainittu Reiterin automaatti. Monissa tapauksissa kiteiden kasvu vaikutti huomattavasti luonnollisemmalta, kuin itseään toistavassa Reiterin mallissa. Kuva 2 esittää yhden tapauksen kehittymisen ajan myötä. Erityisesti viisikon viimeisessä kiteessä on huomattavissa dendriittien itsesimilaarisuus. Liite 3 antaa paremman kuvan samantyylisestä kiteestä kokonaisuudessaan.

[2] Simuloidussa kiteessä keskusta on aina täysin jäässä (erittäin tumma), eikä voi sulaa. Tätä rajoitusta ei oikeilla kiteillä tietenkään ole.



Kuva 2 Simuloidun kiteen kasvu.

Ongelmalliseksi tilanne kuitenkin muodostuu GG:n kohdalla suorituskyvyssä, sillä automaatti vaatii keskimäärin viisi tuhatta iteraatioita muodostaakseen kokonaisen kiteen. Tämä tarkoittaa, että pystyäkseen n. 10 sekunnin aikana mielenkiintoisen lopputuloksen automaatin on käytävä läpi yli 500 iteraatiota sekunnissa eli n. puoli miljardia soluoperaatiota/sek 512x512 automaatilla.

Huomattava vaihtelu eri kiteiden kasvuajassa esittää oman haasteensa mallin käyttämiseen. Nopeimmat kasvut muodostuva kiteen muutamassa tuhannessa iteraatiossa, kun taas jotkin vaativat lähemmäs 30 000–40 000 iteraatiota. Lisäksi on pystyttävä luomaan jonkinlainen yhteys käyttäjän syötteen ja simulaation saamien arvojen välillä. Tarkoituksena on, että käyttäjä antaa vain yksinkertaisen lämpötila / kosteus informaation. Liite 4 sisältää lisää testisarjojen aikana syntyneitä kiteitä vertailtuina oikeisiin kiteisiin.

5 LUMIKITEIDEN MUODOSTUMISEN SIMULOINTI

5.1 Yleistä ongelmasta

Edellä esitetty GG:n automaatti tuottaa oikeilla arvoilla hyviä tuloksia, on kuitenkin olemassa suhteellisen suuri ongelma näiden parametrien muodostamiseksi lennosta käyttäjän syötteen mukaan. Kuten aiemmin esitettiin käyttäjä antaa kaksi yksinkertaista syötettä: lämpötilan sekä suhteellisen ilman kosteuden. Gravner/Griffheathin automaatin parametreissa ei ole suoraa yhteyttä kahteen näin yleisesti määritettyyn parametriin.

On siis ratkaistava, kuinka käyttäjän syöte voidaan muuntaa sellaisiksi automaatin lähtöarvoiksi, että ne tuottavat kiteitä luvussa 2 esitellyn teorian mukaisesti. Ongelma ei ole aivan yksiselitteinen tai yksinkertainen, sillä kuten kuviota 4 tarkastellessa huomataan, on tiettyjen tyyppien välillä huomattavissa jyrkkiä muutoksia. On siis todettava, etteivät lämpötilan ja kosteuden aiheuttamat muutokset lumikiteiden kasvuun ja sen tyyppiin ole täysin lineaarisia.

Sama voidaan huomata, kun tarkastellaan GG:n automaatin parametreja ja tuloksia. Ainoastaan saturaation muutoksella voidaan nähdä selkeä lineaarinen muutos, johon muut arvot eivät juurikaan vaikuta. GG:n tekijät kuvaavat omaa algoritmiaan ylimuotoilluksi sillä perusteella, että suurin osa kuvauksessa käytettävistä parametreista vaikuttaa pääosin yhteen vaiheeseen kehityksessä (Gravner – Griffeath 2006). Gravner ja Griffheath esittävät julkaisussaan useita arvioita siitä, miten eri parametiarvot vaikuttavat erityyppisten ja muotoisten kiteiden syntyyn. Esityksissä otetaan yksittäinen muuttuja / muuttuja pari tarkasteluun ja tutkitaan sen vaikutusta simuloidun kiteen saavuttamaan muotoon. Tämä antaa pohjan lähteä miettimään sopivaa menetelmää, jolla automaattisesti generoiden voitaisiin suorittaa samanlaista tarkastelua ja yrittää yleistää näitä tuloksia riippuvuussuhteiden luomiseksi.

Kuitenkin kattavan otoksen aikaansaamiseksi olisi simuloitava tuhansia kiteitä ja määritettävä niiden kelvollisuus rakenteen, ulkonäön ja kasvunopeuden suhteen. On selvää, ettei tällaiseen työhön ollut re-

sursseja. On ensiarvoisen tärkeää löytää ratkaisu, jossa ihmisen vaativaa suurien datamäärien seulontaa vältettäisiin. Ulkonäön miellyttävyys on tietenkin ihmismäinen lähtökohta, joten tässä suhteessa automaattisen prosessin käyttö ei ole mahdollista. Kahden muun kriteerin, rakenteen ja kasvunopeuden osalta automaattista seulontaa voidaan kuitenkin hyödyntää.

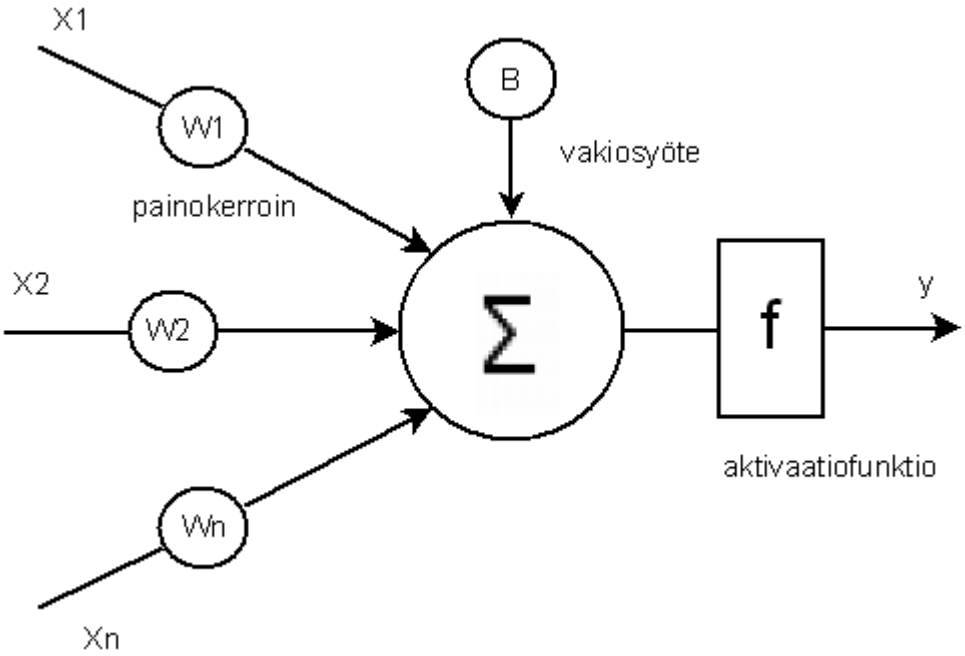
Periaatteessa automaatin tila hetkellä t on riippuvainen 7 parametrisesta funktiosta, jos satunnaisuutta ei oteta huomioon. Testien ja Gravner/Griffheathin mukaan oleellista näyttää olevan tiettyjen parametrien absoluuttinen arvo (erityisesti ρ, kosteus), mutta myös parametrien suhteella on muodostumisen kannalta tärkeä osuus (Gravner – Griffheath 2006). Selkeimmin tämä riippuvuus ilmenee β- ja ρ-arvon välillä, mutta myös α:n ja θ:n välillä.

Funktioiden approksimointiin ja luokitteluongelmiin voidaan soveltaa neuroverkkoja. Seuraavassa on esitelty lyhyesti käytettyjä tekniikoita, joilla ongelmaa lähdettiin ratkaisemaan.

5.2 Neuroverkot

Neuroverkot (engl. neural networks) ovat biologisten hermoverkkojen toiminnan pohjalta muotoiltu malli. Neuraaliverkkoja on tutkittu useita vuosikymmeniä ja niitä on sovellettu monien sellaisten luokittelu ja säätöongelmien ratkaisuun, joihin perinteiset metodit eivät ole hyvin soveltuneet. Neuraaliverkkoja on käytetty esimerkiksi aivokasvainten luokitteluun (Ala-Korpela – Inkinen – Suna 2007, 75,79).

Neuraaliverkkojen rakenne muodostuu kuten biologisessa vastineessaankin neuroneista sekä niihin liittyneistä aksoneista ja dendriiteistä (neuronien väliset liitynnät, synapsit). Jokaiseen neuroniin on liitettynä n-kappaletta synapseja, jotka tuovat tietoa neuroniin. Neuroni itsessään toimii yksinkertaisena summaimena, joka laskee sisääntulevan tiedon perusteella, ylittyykö aktivaatiokynnys. Kynnyksen ylityttyä neuroni virittyy antaa vastesykäyksen siihen liitettyihin synapseihin, jotka on yleensä vastaavasti liitetty seuraavaan neuroniin. (Ala-Korpela ym. 2007 75,76; Buckland 2002, 235, 236; Hämäläinen 1992, 74.)



Kuvio 17 Yksittäisen neuronin kuvaus.

Kuviossa 17 on tyypillinen kuvaus yksittäisestä idealisoidusta neuroverkon neuronista. Jokaiseen neuroniin on liitettynä 1–n syötettä, jotka muodostavat nk. syötevektorin (x_1-x_n), jokaista synapsia ja syötettä kohden on vastaavasti määrätty oma painokerroin (w), joka muokkaa syötettä. Painokerroin voi olla joko positiivinen (exhibitory), vahvistaen signaalia, tai negatiivinen (inhibitory), heikentäen signaalin merkitystä. (Ala-Korpela ym. 2007, 75–77; Buckland 2002, 240, 241; Hämäläinen 1992, 74.)

Kuviossa 17 esitetyn neuronin antama vaste voidaan laskea seuraavan kaavan mukaisesti,

$$y = F\left(\sum_{i=0}^{i=n} w_i x_i + bn\right) \qquad (13)$$

jossa i = syötevektorin alkioiden lukumäärä
x = syötevektorin alkio ja
w = x:ää vastaavan painokerroin
bn = mahdollinen vakiosyöte
F = neuronin aktivaatiofunktio
(Hämäläinen 1992, 74–75)

Aktivaatiofunktio

Aktivaatiofunktio on funktio, joka määrittää neuronin antaman vasteen vahvuuden. Funktioita on useita erilaisia vaihtelevin ominaisuuksin, ja ne valitaan kuhunkin ongelmaan sopivasti sen vaatimuksia silmällä pitäen. Kolme yleisintä käytettyä funktiotyyppiä ovat kynnysfunktio, lineaariset funktiot sekä sigmoidinen funktio. (Buckland 2002, 239–246.)

Kynnysfunktio (step function) on epälineaarinen funktio, jota käytetään, kun neuronin halutaan ilmaisevan ylittyykö jokin tietty kynnysarvo. Vasteeksi muodostuu yksinkertainen boolean informaatio siitä, aktivoituuko neuroni vai ei.

$$y = a > s \qquad (14)$$

jossa s = kynnys
a = summattu aktivaatio
(Buckland 2002, 239.)

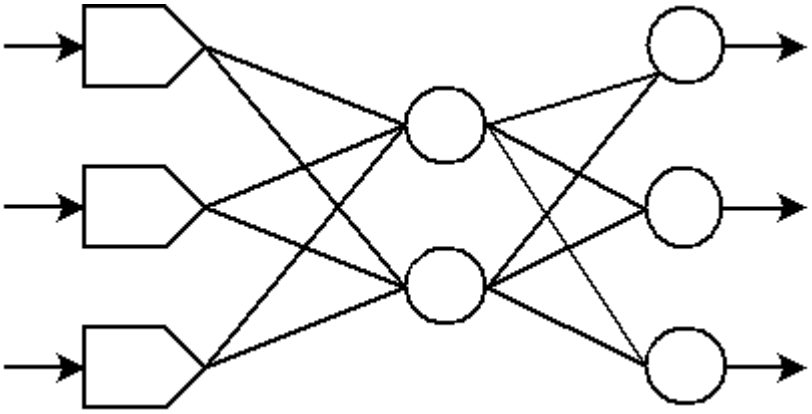
Lineaarisia funktioita käytetään, kun halutaan saavuttaa parempi kvantitatiivinen suhde syötteen ja vasteen välille. Funktioksi käy mikä tahansa lineaarista muutosta kuvaava funktio. (Ala-Korpela ym. 2007, 77.)

Sigmoidinen funktio taas sopii erityisesti epälineaarisiin ongelmiin. Nimensä funktio on saanut S:n muotoisesta kuvaajastaan.

$$y = \frac{1}{1 + e^{-a/p}} \qquad (15)$$

jossa e = neperin luku
a = on summattu aktivaatio
p = on käyrän kontrolliparametri
(Ala-Korpela ym. 2007, 77; Buckland 2002, 239.)

Neuronit järjestetään kerroksittain verkkomaisiksi rakenteiksi (kuvio 18). Ensimmäistä kerrosta, jossa on pelkät liitännät, kutsutaan yleensä syötekerrokseksi (Ala-Korpela ym. 2007, 77). Seuraavaksi kerrostetussa verkossa tulevat laskentakerrokset (hiddenlayer) ja lopulta vastekerros (output layer).



Kuvio 18 MultilayerPerceptron-verkko.

MultiLayerPerceptron-verkko (MLP)

MultiLayerPerceptron on yleisesti käytetty verkon muoto. MLP:ssä tieto kulkee aina yhteen suuntaa (feedforward) eikä takaisinkytkentöjä sallita. Verkko rakentuu useista kerroksista, joissa neuronit ovat aina yhdenmukaisia ja jokainen neuroni on kytketty seuraavan kerroksen jokaiseen neuroniin sisääntulona. Kuvion 18 verkko on tyypillinen MLP-verkko. (Ala-Korpela ym. 2007, 77,78; Buckland 2002, 242.)

Neuroverkoilla ei varsinaisesti ole maksimikokoa, mutta yleisenä nyrkkisääntönä pidetään yksinkertaisuutta. Yksinkertaiset verkot on helpompi kouluttaa ja ne toimivat tehokkaammin. Tavallisimmin käyte-

tään kolmikerrosverkkoja, joissa on yksi laskentakerros syötekerroksen ja vastekerroksen lisäksi (Buckland 2002, 242, Hämäläinen 1992, 74–78).

Verkkojen toiminnan kannalta oleellista on sekä neuronin aktivaatiofunktio että siihen liittyvien tulojen painokertoimet, jotka muokkaavat sisääntulevaa signaalia. Yleisesti ottaen jokaisella neuronilla voisi olla oma erillinen syötevektorinsa, aktivaatiofunktionsa sekä kiinteä syöte, mutta usein verkot rakennetaan yhdenmukaisista neuroneista tehden eroa korkeintaan eri kerrosten välillä. (Ala-Korpela ym. 2007, 75–77.)

Neuroverkkojen oppiminen

Vaatimuksina neuroverkkojen käytölle on, että käytössä on tarpeeksi laaja-aineisto verkon kouluttamiseen tai voidaan määrittää yleisesti käyttäytyminen, jota verkon halutaan noudattavan (Hämäläinen 1992, 74, 78–81). Neuroverkkojen kouluttamisessa on tärkeää säilyttää verkkojen kyky tehdä yleistyksiä (generalisation) annetusta tiedosta. Oppiminen voidaan jakaa karkeasti kahteen eri tyyppiin (Ala-Korpela ym. 2007, 78).

Ohjattu

Ohjatussa oppimisessa annettu syöte ja vaadittu vaste tiedetään tarkkaan. Halutusta syötejoukosta luodaan sopivia opetustapauksia, jotka syötetään verkon läpi ja säädetään verkkoa haluttuun suuntaan. Yleinen tapa opettaa verkko on säätää se erikseen haluttua tulosta kohti määrittelemällä suurin sallittu virhearvo, minkä verkko saa tuottaa. Tämän jälkeen käydään opetustapaukset läpi niin, että jokaisen kohdalla verkon painoarvoja säädetään jonkin menettelyn mukaisesti. Tätä toistetaan, kunnes verkon antama tulos on virhearvon marginaalin sisällä (optimi) kaikissa tapauksissa. Tavallinen tapa ratkaista edellinen on käyttää BackPropagation-menetelmää, jossa virhearvon mukaan säädetään verkon painokertoimia käänteisesti vastekerroksesta syötekerrosta kohti. Ohjattu oppiminen vaatii suuren valmistellun datajoukon, jota voidaan hyödyntää opetustapauksien rakentamiseen. (Ala-Korpela ym. 2007, 78–79; Hämäläinen 1992, 74–81.)

Ohjaamaton

Ohjaamattomassa oppimisessa yritetään hakeutua yleisillä parametreilla kohti ratkaisua, joka antaa toivotun tuloksen. Mahdolliset datan sisältämät luokat ja ryhmittelyt ovat tuntemattomia tai niistä ei haluta esittää arviota, vaan pyritään löytämään uusia alueita. Ohjaamattomasta oppimisesta voidaan myös käyttää termiä itseorganisoituminen. (Ala-Korpela ym. 2007, 78–79; Hämäläinen 1992, 78–81.)

Neuroverkkojen opetuksessa yleisenä ongelmana on verkkojen ylioppiminen. Ylioppimisella tarkoitetaan tapauksia, joissa verkon painokertoimet muotoutuvat niin, että verkko menettää tätä kautta kykynsä yleistää tapauksia. Ylioppimista voidaan testata käyttämällä koulutusaineiston lisäksi erillistä testiaineistoa. Jos tulokset ovat testiaineistolla huonot mutta koulutusaineistolla hyvät, on kyseessä ylioppiminen, eli verkko on oppinut tietyt tapaukset liian hyvin. Eräs tapa toimia ylioppimista vastaan on lisätä satunnaistapauksia verkon opetusaineistoon, jolloin mahdollisuus oppia vain tietyt tapaukset ulkoa vähenee. (Buckland 2002, 319,320 ;Hämäläinen 1992, 74.)

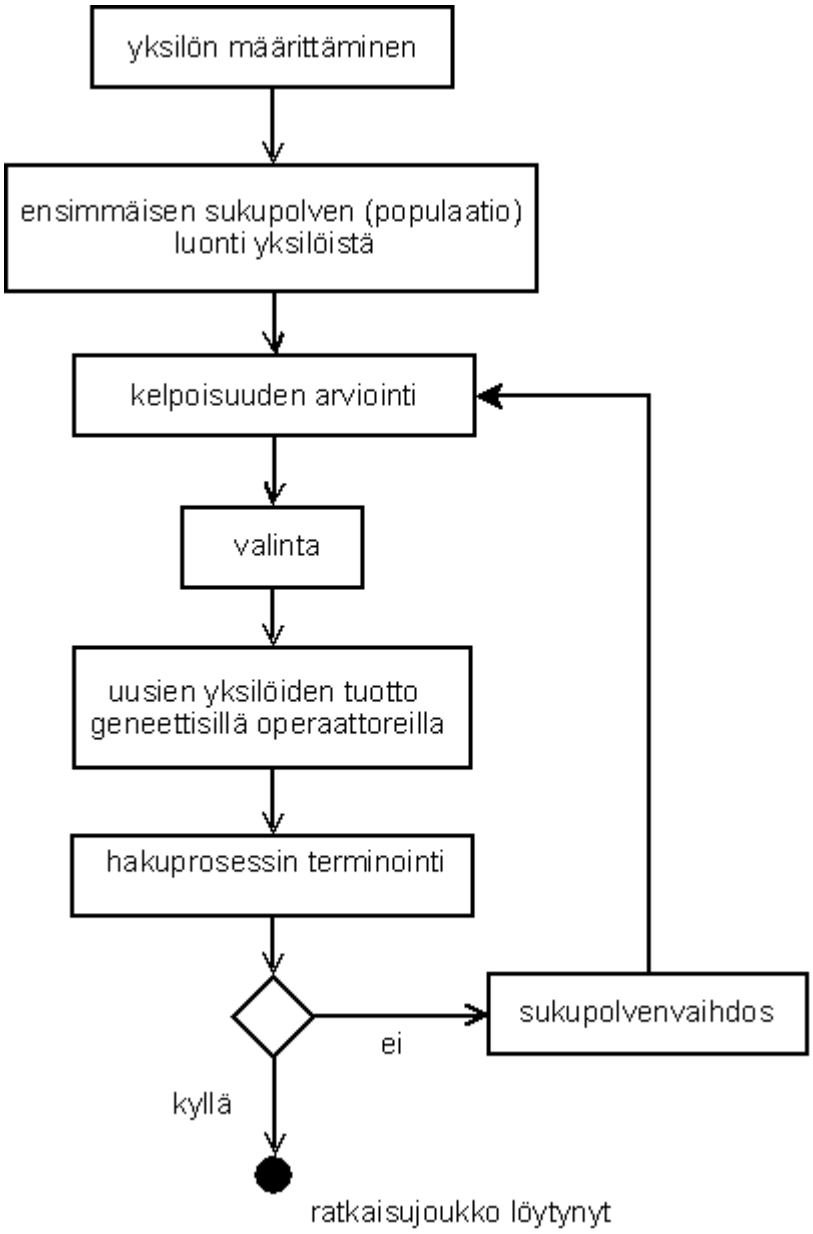
Neuraaliverkkoja voidaan ajatella eräänlaisena blackbox-ratkaisuna, jossa syötetyn tiedon ja vasteen välisiä suhteita ei tunneta mutta tiedetään tai voidaan arvioida, minkälaisen vasteen tietyn syötteen tulee tuottaa. Neuroverkkoja voidaan käyttää paljastamaan piilotettuja lainalaisuuksia ja riippuvuussuhteita datajoukoista (Ala-Korpela ym. 2007, 78–79). Näin voidaan todeta, että neuroverkot toimivat eräänlaisena funktion approksimaattorina.

Jos siis on olemassa funktio, funktioryhmä tai lainalaisuus, joka määrittelee GG:n automaatin arvojen ja yleisten syötearvojen suhteen niin, että ne muodostavat oikean tyyppisiä kiteitä, on toivoa, että neuraaliverkko pystyy tämän ilmaisemaan. Verkon painoarvojen löytämiseksi voidaan käyttää erilaisia menetelmiä. Tässä työssä valittiin käytettäväksi geneettiset algoritmit.

5.3 Geneettiset algoritmit

Geneettiset algoritmit (engl. genetic algorithms) kuuluvat evoluutioalgoritmien joukkoon. Nämä algoritmit perustuvat evoluution ja sen prosessien hyödyntämiseen ratkaisujen etsinnässä. Geneettisiä algorit-

meja on käytetty useissa optimointia vaativissa ongelmissa esimerkiksi biologiassa sekä kuvan prosessoinnissa. (Goldberg 1989, 1, 2, 125–139.)



Kuvio 19 Geneettisen haun prosessi.

Kuvio 19 esittää tyypillisen sukupolvittaisen hakuprosessin. Prosessi alkaa kokoamalla satunnaisesti luoduista yksilöistä populaatio, joka

kehittyy (konvergoituu) toistettavan prosessin aikana kohti haluttua lopputulosta.

Yksilö ja populaatio

Geneettisissä algoritmeissa yksittäistä ratkaisua eli algoritmia kuvaa yksilö. Yksilön geenit ovat tässä yhteydessä algoritmin osia ja genomi tai genotyyppi kokonaisuudessaan on ratkaisumahdollisuus. Genomi koodataan jokaisen ongelman kohdalla sellaiseksi, jotta sitä voidaan käsitellä prosessissa ja muuntaa takaisin käytetteväksi ratkaisuksi (Buckland 2002, 97–99; Goldberg 83–85). Biologian termein tätä ratkaisua kutsuttaisiin fenotyypiksi, yksilön ilmiasuksi. Suosituin tapa esittää ratkaisuja on nk. binäärikoodaus, jossa yksilöt esitetään binäärimerkkijonona (Goldberg 1989, 15–19). Esimerkiksi suunnat pohjoinen, etelä, länsi ja itä voitaisiin ilmaista käyttäen kahta bittiä: ’00’, ’11’, ’10’, ’01’. Tällöin reittiohjeet tiettyyn pisteeseen voitaisiin koodata jonoksi ’11010010’. Yksinkertaisuuden vuoksi käytetään esityksessä vakioetäisyyksiä ja dekoodattuna ohjeet olisivat etelään, itään, pohjoiseen ja länteen palaten siis alkupisteeseen.

Ratkaisujoukkoa kuvaa vastaavasti populaatio, joka on hakuprosessin perusosanen. Hakuprosessin alkuvaiheessa populaatio rakennetaan käyttäen satunnaisesti luotuja yksilöitä, ratkaisuehdotuksia. Populaation oikean koon valinta on kriittinen tekijä geneettisiä algoritmejä hyödynnettäessä. Ongelmaan ja koodaukseen nähden liian pienet populaatiot eivät sisällä tarpeeksi erilaisia yksilöitä ja haut konvergoituvat liian nopeasti, kun taas liian suurten populaatioiden käsittely on hidasta. (Goldberg 1989, 9,101, 111; Buckland 2002.)

Kelpoisuus ja kelpoisuusfunktio

Geneettisien algoritmien kannalta ratkaisevaa on, kuten evolutiivisessa kehityksessäkin, yksilön kelpoisuus (fitness). Kelpoisuus on pystyttävä määrittämään yksiselitteisesti, jotta valintaprosessi toimii ja algoritmi pystyy hakeutumaan kohti optimaalista tulosta (Goldberg 1989, 15–18). Kelpoisuus määritellään yleisesti kelpoisuusfunktion avulla. Tämä muodostetaan kunkin ongelman kohdalla erikseen ottamaan mahdollisimman hyvin halutut ominaisuudet huomioon joita ratkaisun, eli yksilön genomin tulee sisältää (Buckland 2002, 97–99). Kelpoisuu-

den määritys on yksi suurimmista ongelmista geneettisten algoritmien hyödyntämisessä, sillä kelpoisuuden määritys ratkaisee huomattavissa määrin, kuinka tehokas algoritmi on. Kun yksilöiden kelpoisuusarvot on laskettu, voidaan näiden pohjalta suorittaa valintaprosessi.

Valinta

Valinta (selection) on vaihe, jossa sopivaksi katsottuja yksilöitä valitaan uuden populaation muodostamiseen luonnonvalintaa matkien (Buckland 2002). Sopivuuden määrittämiseksi käytetään hyödyksi edellä määriteltyjä kelpoisuusarvoja. Valinnan suorittamiseksi on olemassa useita eri menetelmiä, mutta ne kaikki perustuvat periaatteeseen, että kelpoisuudeltaan hyvillä yksilöillä on suurempi todennäköisyys tulla valituiksi, kun seuraavan sukupolven ”vanhempia” valitaan (Goldberg 1989, 9–11; Buckland 2002, 100). Kannattaa huomioida, että osa kirjallisuudesta lukee valinnan kuuluvaksi geneettisten operaatioiden joukkoon, kun taas osa pitää sitä omana erillisenä vaiheenaan. Tässä noudatetaan jälkimmäistä menettelyä.

Kilpailutusvalinnassa (tournament selection) populaatiosta valitaan n-kappaletta yksilöitä, jotka kilpailutetaan keskenään, ja paras yksilö valitaan. Tätä toistetaan, kunnes populaatio on täydennetty uusilla yksilöillä ja mikään ei estä samaa yksilöä valikoitumasta useaan kertaan uuden populaation muodostuksessa (Buckland 2002, 164, 165). Kilpailutusvalinnan suurin ongelma on liian nopea konvergenssi, jos yhtä yksilöä suositaan liian monta kertaa valinnassa (Buckland 2002).

Geneettiset operaattorit

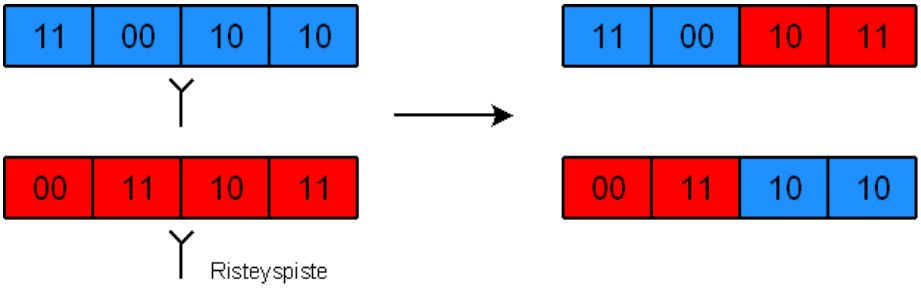
Geneettisten algoritmien toiminta perustuu geneettisten operaattoreiden hyödyntämiseen. Näillä muodostetaan valituista yksilöistä uusia yksilöitä käyttäen hyödyksi biologiasta tuttuja operaatioita, kuten mutaatiota ja risteytystä.

Risteytys, rekombinaatio

Risteytyksessä kahden yksilön geenit (biolog. vanhemmat) risteytetään, jolloin syntyy kaksi uutta vanhempien geenit jakavaa yksilöä. Tällä tavalla luodaan uusia ratkaisuja samalla yrittäen säilyttää edellisten sukupolvien löytämät hyvät tulokset. Risteytykseen soveltuvia operaatioita on useita. Useimmin käytetään risteyspisteisiin perustuvaa, jossa valitaan yksi tai useampia pisteitä joiden kohdalta geenit vaihdetaan. Valinnat on suoritettava varoen, jottei arvokasta informaatiota tuhoudu, sillä muuten risteysoperaattorit voivat käyttäytyä haitallisen mutaation tavoin. (Buckland 2002, 172–173.)

Yksipisteinen risteytys

Edellä esiteltyä reittiohjeiden analogiaa noudattaen esitetään kaksi yksilöä (kuvio 20) ja suoritetaan risteytys toisen geenin kohdalta.



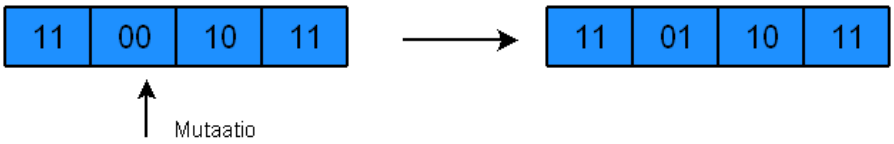
Kuvio 20 Risteytys yhden pisteen kohdalta

Mutaatio

Risteytyksen jälkeen jokaisen yksilön genomissa voi tapahtua pieni satunnaisuuteen perustuva muutos, mutaatio. Mutaatiolla on yleisen näkemyksen mukaan pienempi merkitys, kuin rekombinaatiolla yksilön kehittymisessä kohti optimia. Mutaatio on kuitenkin oleellinen osa geneettisien algoritmien toiminnassa. Se estää liian nopeaa konvergenssiä aiheuttamalla muutoksia yksilöiden genomissa ja seuloo satunnaisesti mahdollisia populaation luonnissa kartoittamatta jääneitä ratkaisun alueita. (Goldberg 1989, 14; Buckland 2002, 223.)

Mutaatio voidaan toteuttaa useilla eri tavoilla kuten rekombinaatiokin, ja mutaatio operaattorit muodostetaan noudatettuun koodaukseen

sopivaksi. Suosituin tapa on nk. pistemutaatio, joka kohdistuu yksittäiseen geeniin (Goldberg 1989 14; Hämäläinen 1992, 78). Vaihtomutaatiossa taas sekoitetaan geenien järjestystä tai siirretään niiden paikkaa. Mutaatioherkkyydessä määritetään pieni todennäköisyys sille, että mutaatio-operaatio suoritetaan tietylle geenille. Mutaationherkkyydeksi valitaan suhteellisen pieni arvo suhteessa risteytyksen todennäköisyyteen (Buckland 2002, 99). Liian aggressiivinen mutaatio tuhoaa mahdollisia ratkaisuvaihtoehtoja ja aiheuttaa haun muuttumisen liian satunnaiseksi. Binäärikoodauksessa suositaan yleisesti pistemutaatiota, jossa valittu bitti saa uudeksi arvokseen negaation itsestään (flip bit, kuvio 21; Goldberg 1989, 16). Tarkastellaan toista edellisessä risteytyksessä syntynyttä ratkaisuvaihtoehtoa, joka käy läpi mutaation jonon toisen geenin viimeisen bitin kohdalla.



Kuvio 21 Pistemutaatio

Edelliset operaattorit muodostavat geneettisten algoritmien toiminnan perustan. Kuitenkin pelkästään näiden operaattoreiden käyttäminen voi johtaa tietyissä tapauksissa suuriin ongelmiin itse algoritmin toimintaa ohjaavien parametrien kuten populaation koon, operaattorien valinnan ja mutaatioherkkyyden valinnassa.

Ongelmaksi voi tulla liian nopea konvergenssi, jolloin jostain lokaalisti optimaalisista yksilöistä tulee liian dominoivia ja niiden jälkeläiset valtaavat koko populaation eikä kunnollista ratkaisua synny. Toisaalta prosessi ei sellaisenaan takaa parhaimman yksilön selviämistä (vain suuremman todennäköisyyden), joten on mahdollisuus menettää hyviä ratkaisutuloksia virheellisten mutaatio tai risteytysvalintojen takia. Ongelmia prosessissa voi vähentää käyttämällä seuraavia menetelmiä. (Golberg 1989 185–190, Buckland 2002, 161, 74,175.)

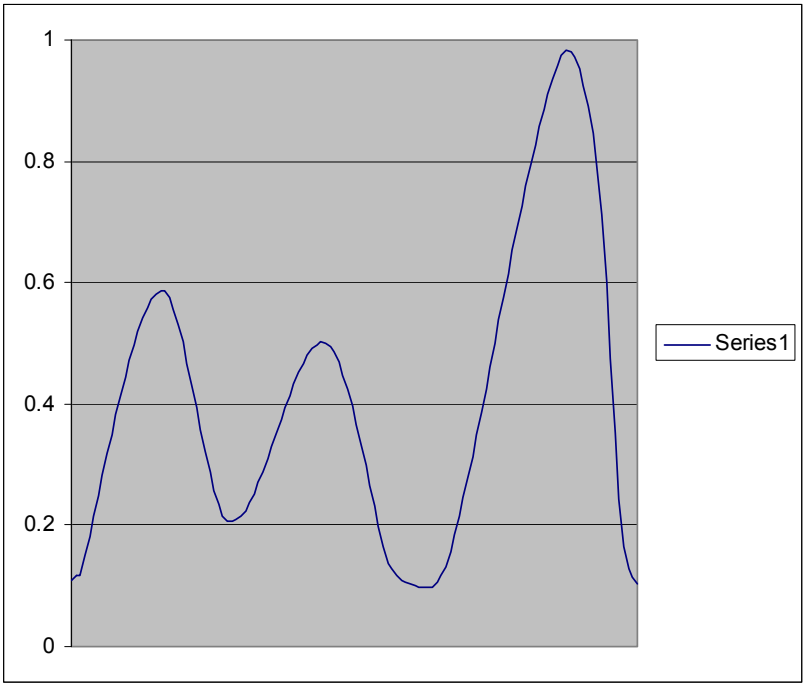
Niching ja elitismi

Niching on tekniikka, jolla estetään liian nopeaa konvergenssia. Geneettisesti liian toisiaan lähekkäisiä yksilöiden kelpoisuutta heikenne-

tään, jotta populaation diversiteetti säilyisi pitempään (Golgberg 1989, 185–190). Tällä estetään vain tiettyjen yksilöiden pääsemistä dominoivaan asemaan populaation kehityksessä. Elitismissä n-kappaletta kelpoisuudeltaan parhaita yksilöitä viedään prosessista läpi muuttumattomina seuraavaan sukupolveen, ts. varmistetaan parhaiden yksilöiden sisältämän informaation säilyminen (Buckland 2002, 137,161). Jos populaatiot korvataan aina kokonaan geneettisten operaatioiden läpi viedyillä yksilöillä, on riskinä sopivien ratkaisujen hukkaaminen.

Geneettisten algoritmien kohdalla voidaan käyttää myös muita biologisesti tärkeitä ja kehittyneempiä operaattoreita, kuten diploidisuus (kaksipuolinen kromosomi) tai dominointi (Golberg 1989, 148–165). Kuitenkin edelliset riittävät perushaun ymmärtämiseen ja sen toteuttamiseksi.

Hakuavaruuden käsite ja geneettisten algoritmien valinnan perustelu



Kuvio 22 Yksiparametrinen hakuavaruus

Kuviossa 22 esitetään erään yksi parametrisen funktion saamat arvot syötteen suhteen. Tavoitteena on maksimoida tai minimoida funktion saama arvo, eli haetaan sitä arvoa (tai väliä), jossa funktio tuottaa maksimaalisen tai minimaalisen vasteen annettuun syötteeseen. Tästä eteenpäin käytetään termiä maksimointi etsittäessä parasta mahdollista ratkaisua. Pisteet, jotka muodostavat käyrän, kuvaavat mahdollista hakuavaruutta. Hakuavaruudessa on useita lokaaleja maksimeja (pienemmät huiput), jotka edustavat suboptimaalisia ratkaisuja. Korkeinta saavuettua huippua voidaan kutsua globaaliksi maksimiksi, joka on ”paras” mahdollinen tulos.

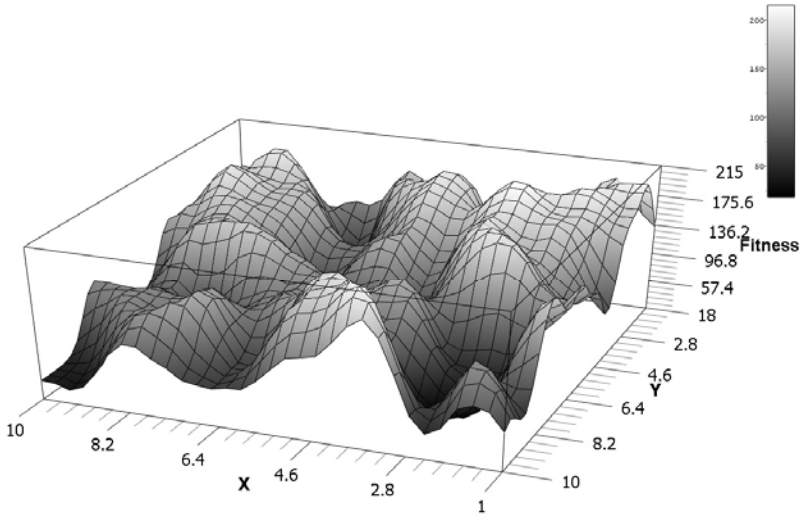
Triviaalinen tapa ratkaista ongelma on läpikäydä jokainen piste (tai riittävän tiheästi) ja tutkia saavutettu tulos. Toinen mahdollisuus on valita satunnaisia pisteitä ja suorittaa sama menettely. Kolmas tyypillinen menettely on käyttää gradienttihakua, joka osoittaa jyrkimmän muutoksen suunnan pisteessä. Tällöin hakua voidaan kutsua myös vuorikiipelyksi (engl. hill climbing). Kaikki edelliset menetelmät sisältävät ongelman: milloin tiedetään, että paras mahdollinen tulos on saavutettu? Koko hakuavaruuden seulonta ei tehokkuuden kannalta katsoen ole järkevää, jos sitä yleensä pystytään edes yrittämään. Satunnaisesti valitsemalla pisteitä voidaan vähentää suoritettavien operaatioiden määrää, mutta ongelma haun päättämisestä säilyy. (Golberg 1989, 2–7.)

Gradienttihaut (gradient search) ovat yleensä laskennallisesti vaativia ja riippuvaisia derivaattojen määrittämisestä. Deriaattoja ei voida määrittää useissa ongelmissa, sillä hakuavaruudet voivat sisältää kohinaa (noise) ja/tai epäjatkuvuuksia. Toinen ongelma ilmenee hyvin tarkasteltaessa kuviota 22: Entä jos haku löytääkin ensimmäisen huipun? Kuinka tällöin siirrytään hakemaan sujuvasti uusia alueita? (Goldberg 1989, 20–23.)

Neuroverkkojen kohdalla ongelma muodostuu siitä, että tavalliset gradienttihaut jäävät helposti jumiin hakuavaruuden lokaaliin maksimiin (Hämäläinen 1992, 74). Tässä haun ongelmana on, että oletusarvoisesti ne lähtevät vain yhdestä kohdasta hakuavaruutta liikkeelle, eli neuroverkon tapauksessa alussa saamat painokertoimet. Myös gradienttihakuihin sopivien funktioiden muodostus, säätäminen ja muok-

kaus osoittautuvat usein hankalaksi tehtäväksi (Goldberg 1989, 3;Hämäläinen 1992, 74).

Geneettinen haku perustuu koodauksessa syntyvien osatulosten (block theory) seulontaan ja yhdistelyyn. Ajatuksena on, että jokaisessa yksilössä on ratkaisun osia, joista muodostetaan kokonaisratkaisu operaattoreiden avulla. GA:t eivät sinänsä ole tehokkaita löytämään eksaktia optimia vaan toimivat tehokkaasti pienentäen hakuavaruuden laajuutta vain löydettyjen optimien alueille. Haut lähtevät useasta pisteestä hakuavaruutta yhtä aikaa ja käyttävät deterministisen siirtymisen sijaan todennäköisyyksin pohjautuvaa. (Goldberg 1989 7, 28–45.)



Kuvio 23 Kaksiparametrinen hakuavaruus, "kelpoisuusmaisema".

Kuviosa 23 esitellään kahden parametrin muodostama hakuavaruus. Kuvion jokaiselle pisteelle on kiinnitetty kelpoisuusarvo, jota esimerkiksi geneettinen algoritmi voi hyödyntää.

Geneettisiä algoritmeja voidaan soveltaa neuroverkoissa itse verkon rakenteen (neuronien määrä, sijoittelu, liitännät) muodostamiseen, verkonpainokertoimien säätämiseen tai itse verkon toimintaa ohjaavien sääntöparametrien muokkaamiseen (Buckland 2002; Hämäläinen 1992, 75). Ongelmina rakenteen muodostamisessa pidetään kilpailevien ratkaisujen (competing conventions) yhdistämistä ja näin ratkaisun kannalta hyödyllisen tiedon häviämistä (Buckland 2002, 347,348).

Verkon painoarvojen säätämisessä voidaan hyödyntää joko pelkästään geneettisiä algoritmeja tai yhdistää se gradienttihakuun. Tässä mallissa ratkaisuvaihtoehtoja lähdetään kartoittamaan laajalta alueelta geneettisillä algoritmeilla ja viimeinen säätö tehdään itse gradienttihakua käyttäen (vaikkapa Backpropagation). Yleisesti ottaen geneettiset algoritmit löytävät optimiratkaisuja nopeammin ja varmemmin, mutta sisältävät usein suuremman virheen, kuin verrattuna Backpropagation. (Hämäläinen 1992, 78.)

Vaikka geneettiset algoritmit eivät välttämättä tuota yhtä hyviä ratkaisuja kuin erikseen ongelmaan kehitetyt ja optimoidut algoritmit tai neuroverkon painoarvojen säätäminen suoraan BackPropagation avulla, on niiden käyttö perusteltua tässä ratkaisussa. Ongelmana Backpropagationin kanssa on suurempi mahdollisuus ylioppimiseen sekä vaadittavien testitapausten suuri määrä (Ala-Korpela ym. 2007, 78,79). Tavoitteena ei ole hakea parasta mahdollista tulosta, ts. pieni virhemarginaali, sillä on oikeastaan hankala määrittää tarkalleen, milloin tämä tavoite on saavutettu. Ei myöskään ole todennäköistä, että olisi olemassa yksittäinen ja selkeä optimiratkaisu. GA:n käyttö luo myös satunnaisuutta, joka estää verkon ylioppimista hakeutuen silti kohti sopivia ratkaisuja (Golberg 1989, 3–15; Hämäläinen 1992 74–76).

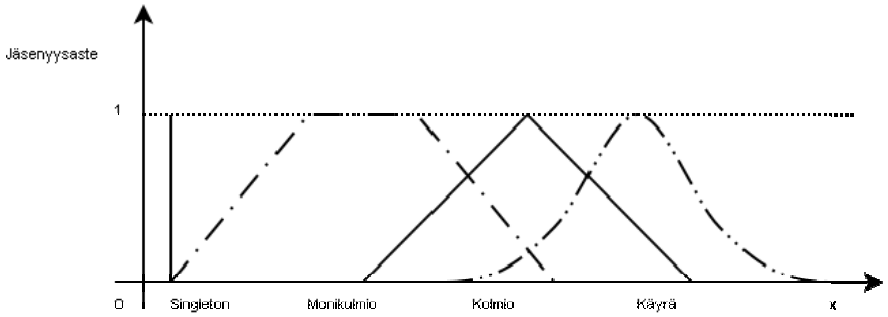
Geneettisten algoritmien hyödyntämiseksi oli työssä pystyttävä määrittelemään eri ratkaisujen kelpoisuus. Kehittyneiden kiteiden ja kelpoisuuden määrittämiseksi hyödynnettiin sumeaa logiikkaa.

5.4 Sumea logiikka

Sumea logiikka (engl. fuzzy logic) on Lofti Zadefin n. 1965 muotoilema tapa lähestyä uudella tavalla sellaisia ongelmia, joiden määrittäminen tavallisen logiikan mukaan on erittäin hankalaa. Ihmiset käsittelevät asioita hyvinkin usein epätäsmällisten tai puutteellisten tietojen perusteella, mikä taas ei tavallisen logiikan puitteissa ole mahdollista. Tästä syystä näiden asioiden mallintaminen tietokoneessa on osoittautunut yhtä suureksi ongelmaksi (Niskanen 2003, 41–42). Sumeaa logiikkaa on sovellettu erityisesti erilaisten säätöjärjestelmien muodostamiseen teollisuudessa ja elektroniikassa (Niskanenen 2003). Sumea logiikka yhdistetään usein muihin tekniikoihin (esim. neuroverkot) parempien tuloksien aikaansaamiseksi (Niskanen 2003, 4, 151–158).

Sumean logiikan perustana voidaan pitää sumeiden joukkojen määrittelemistä. Alkion kuuluvuus joukkoon määritellään jäsenyysasteen perusteella ja kuulumista useaan joukkoon samanaikaisesti tai joukon itsensä kuulumista ei ole poissuljettu (vrt. tavallinen boolean logiikka). Jäsenyysaste määritellään lukuarvolla 0–1 välillä, jossa 1 merkitsee täyttä kuuluvuutta joukkoon (Niskanen 2003, 50–55; Zadeh ym. 1975, 27,28). Totuusarvojen ja vastaavuuden kuvaamiseen käytetään usein kielellisiä termejä kuten melko, noin, hyvin, jossain määrin, jotka antavat määrittelyyn huomattavasti inhimillisemmän lähtökohdan. Vaikka merkintätapa on sama kuin todennäköisyyksissä, sumeudella ei ole mitään tekemistä todennäköisyyksien kanssa.

Sumean logiikan perustyökaluja ovat jäsenyysfunktiot (kuvio 24), joita käytetään määrittämään jäsenyysasteita. Ne myös kuvaavat sumeita joukkoja. Funktiot normalisoidaan usein niin, että ne saavuttavat jossain pisteessä arvon 1 ja funktion kattama pinta-ala on koko sumea joukko. Sumeassa logiikassa käytetään perinteisen logiikan tyyliin erilaisia operaatioita, joiden voidaan katsoa olevan yleistyksiä tavallisen kaksiarvologiikan operaatioista. (Zadeh ym. 1975, 28–32.)



Kuvio 24 Jäsenyysfunktioita.

Sumea päättely

Mallia rakennettaessa ilmiöt ”sumeutetaan” (fuzzification) eli niistä muodostetaan sumeita joukkoja hyödyntäen jäsenyysfunktioita. Tässä voidaan käyttää apuna joko asiantuntemusta (ihmistä) tai jotain automaattista prosessia, joka yrittää etsiä sopivia joukkoja eli funktioita. Näiden avulla muodostetaan käsitteitä kuten *kylmä, viileä, lämmin* ja sääntöjä *jos* huone on *kylmä* niin säädä termostaatti *lämpimälle*. Täs-

sä sumea ehtolause ’laukaisee’ (fires) sääntöön liitetyn johtopäätöksen, joka on myös sumea joukko. Laukaisun voimakkuus voidaan selvittää laskemalla jäsenfunktioiden kattama pinta-ala tietyllä säännöllä. Sumeassa logiikassa laukeavia sääntöjä voi olla samanaikaisesti useita, nämä vain laukeavat erivoimakkuuksilla (firing strenght). (Niskanen 2003, 73-98; Zadeh ym. 199-201.)

Vaikka puhutaan sumeista joukoista ja epämääräisyydestä, ei tämä kuitenkaan tarkoita, että lopputuloksena prosessista olisi jotain epämääräistä tai sumeaa vaan aikaan saadaan yhtä tarkkoja tuloksia kuin matemaattisin menetelminkin. Tätä prosessia kutsutaan vaikkapa selkeytykseksi (defuzzification). Sumean logiikan etuna on pienempien sääntöjoukkojen käyttäminen ja silti hyvän tarkkuuden säilyttäminen. Tavallisimmin näitä ongelmia on tietotekniikassa käsitelty liukuvien arvovälien avulla. Näiden käyttö ei kuitenkaan poista normaalilogiikan aiheuttamia ongelmia, vaan malleihin joudutaan joko rakentamaan erittäin tiheitä välejä tai hyväksymään virhetulkintoja arvojen vaihtuessa yhtäkkiä. Toisena etuna voidaan katsoa olevan se, että sumea logiikka sietää hyvin pieniä virhearvoja tai epämääräisyyksiä lähtöarvoissa. (Niskanen 2003, 96–98; Zadeh ym. 1975, 450, 451.)

5.5 Muodostumisongelman ratkaiseminen

Sopivien parametrien hakuprosessissa hyödynnettiin kaikkia edellä esiteltyjä tekniikoita. Perusperiaatteeltaan prosessi on seuraavanlainen:
Neuroverkkoa käytetään luomaan annetuista kahdesta syötteestä (lämpötila ja kosteus) 7-parametrinen GG:n automaatille sopiva arvojoukko. Arvojoukon saannin jälkeen automaattia suoritetaan tiettyyn pisteeseen. Automaatin luotua kiteen käytetään yksinkertaisia kuviontunnistusmenetelmiä ja sumeaa logiikkaa lumikidetyypin päättelyyn. Mitä paremmin neuroverkko on onnistunut sellaisten parametrien luonnissa, jotka GG:n automaatin kautta tuottavat oikeanlaisia kiteitä, sitä paremmat pisteet se saa geneettisessä algoritmissa, jota käytetään parhaiden vaihtoehtojen seulontaan. Prosessia ajetaan sukupolvittain eteenpäin halutun ajan ja joukosta poimitaan useita (5–10) neuroverkkoja, jotka liitetään simulaatioon tulosten varmistamiseksi.

Syötteiden ja vasteiden valmistelu

Parempien tulosten saavuttamiseksi neuroverkoilla suoritettiin sekä syötteiden että vasteiden kohdalla skaalaus välille 0–1, koska erotukset useiden parametrien kohdalla ovat erittäin suuria (γ tuhannesosia, kun taas β > 1). Skaalauksen ansiosta verkon ei tarvitse muodostaa eroja eri ulostulojen absoluuttisten arvojen osalta, vaan kaikki syötteet ja vasteet ovat samaa kokoluokkaa.

Kelpoisuuden laskeminen

Kelpoisuuden määrittämiseen ei ole yksinkertaisia sääntöjä, vaan kyseessä on aina säätöprosessi, jossa haetaan tasapainoa haluttujen ominaisuuksien välille. Kelpoisuusfunktioksi valittiin jatkuva kelpoisuusfunktio. Eniten pisteitä saa sellainen ratkaisu, joka onnistuu simuloimaan eniten oikean tyyppisiä kiteitä suhteessa syötteisiin ja niiden oletettuihin tyyppeihin. Lisäksi verkon tuottamille parametreille asetetaan raja-arvot, jotta vältyttäisiin simuloimasta kiteitä arvoilla, jotka ovat täysin järjettömiä.

Toinen tähän liittyvä ehto on kasvunopeuden määrittäminen, sillä jotkin raja-arvojen sisälle jäävät parametriryhmät luovat kiteitä, jotka kasvavat erittäin hitaasti tai jopa lopettavat kasvun aikaisessa vaiheessa. Tätä ongelmaa varten määritetään yksinkertainen kasvunopeuden funktio. Jos kiteen osoittama kasvu on tämän funktion mukaan liian hidas, terminoidaan tämän simulointi samantien, jottei aikaa tuhlata tällaisten ratkaisujen kehittämiseen. Bonuksena kasvunopeuden määrittämisellä voidaan suosia (varovasti) suurta kasvunopeutta noudattavia kiteitä ja hakea sellaisia parametreja, jotka tuottavat oikeanlaisia kiteitä mahdollisimman nopeasti. Funktio on luonnollisesti epälineaarinen, koska kasvun nopeus muuttuu ajan myötä huomattavasti.

Sumeaa logiikkaa käytettiin prosessissa kahdessa eri osassa: kiteiden luokitteluun ja arviointiin siitä, minkä tyyppisen kiteen tulisi muodostua tietyllä lämpötila/kosteus syöteparilla.

Sumea logiikka valittiin ratkaisun osaksi, koska ei ole yksiselitteistä tapaa erotella kiteitä toisistaan rajatapauksissa, joita usein syntyy.

Kiteiden luokittelua varten valittiin ensin useita ”ideaalisia” esimerkkitapauksia jokaisesta kideluokasta (fern, dendrites, plates, sectored plates jne.), jotka toimivat prosessissa viitteellisinä esimerkkeinä määritettäessä kiteiden tyyppiä. Kiteitä vertailtiin muutaman yksinkertaisen kuviontunnistus menetelmän avulla ja näiden tulosten perusteella käytettiin sumeaa päättelyä antamaan kuvan siitä mihin kategoriaan kyseinen simuloitu kide kuuluu. Eri menetelmille annettiin erilaiset painoarvot valintaa tehdessä, koska yksittäisen menetelmän käyttö on virhealtista. Eri menetelmiä käytettiin siis tarkentamaan edellisen tuloksia mitä useampi menetelmä toteaa kiteen tyypin olevan tietty, sitä luotettavampia tuloksia saadaan.

Käyttäjän syötearvojen perusteella tehtävässä päättelyssä säännöstöt muodostettiin tekemällä jako luvun 2 kasvuolosuhteita esittävän kaavion perusteella. Sekä lämpötilalle että kosteudelle määritettiin viisi kuvaavaa joukkoa, joiden yhdistelmistä määritetään minkä tyyppinen kide todennäköisesti syntyy. Esimerkiksi, lämpötilan ollessa *keskitasoa* ja kosteuden ollessa *erittäin matala*, muodostuu laattamaisia kuusikulmioita (simple/solid plate). Simuloidun kiteen ja edellisen arvion eroavaisuuden perusteella voidaan määrittää, kuinka hyvin neuroverkko on onnistunut tuottamaan oikean tyyppisen vasteen. Tässä kohtaa ääripäiden poissulkevuus oli tärkeämpää, kuin itse tarkan muodon löytäminen, jonka noudattaminen olisi voinut herkästi johtaa ylioppimiseen.

Neuroverkon koulutus

Neuroverkon tyypiksi valittiin aiemmin luvussa 5.3 esitellyn MLP-tyypin verkko. Verkon painokertoimet koodattiin suoraan reaaliluvuiksi, kuten Hämäläinen (1992) esittää, sillä reaalilukujen muunto binäärimuotoon tuntui tarpeettomalta ylilyönniltä.

Neuroverkon kouluttamiseksi tulee ajaa useita kymmeniä testitapauksia jokaisen verkon kohdalla, jotta saadaan riittävän kattava leikkaus sen kyvystä muodostaa eri kiteitä. Esimerkiksi 40 testisyötteen ja niiden vastinparametrien testaamiseksi on simuloitava vastaavasti 40 kidettä, joiden kasvuajat vaihtelivat n. 5 sekunnista useaan minuuttiin. Yhden populaation sisältäessä 40 mahdollista verkkoa, tämä tarkoittaa että yhden sukupolven testaamiseksi oli käytävä läpi 1600 simuloi-

tua kidettä, joiden simuloiminen vie väistämättä useita kymmeniä tunteja yksittäiseltä koneelta. Verkkojen ja erityisesti geneettisten algoritmien hyödyntäminen vaatii aina jonkinlaista säätöä. Mitä enemmän kokonaisia ajoja pystytään suorittamaan ja tarkistamaan, sitä parempia tuloksia verkoilta voidaan odottaa. Tähän ongelmaan oli pakko löytää ratkaisu, jotta verkkojen käytöstä ja niiden säädöstä tulisi mielekästä.

Koska tapaukset ovat sinänsä toisiinsa liittymättömiä, voidaan niitä käydä läpi eri tietokoneilla samanaikaisesti. Ratkaisuksi riittää tässä yksinkertainen verkotettu laskentajärjestelmä. Palvelin pyörittää geneettisen algoritmin prosessia ja jakaa sukupolviin jaetut tapaukset toisille koneille laskettaviksi. Palvelimena toimiva kone antaa muille koneille niiden pyytäessä testitapauksia. Näiden tapauksien tulokset eli kelpoisuusarvot palautetaan itse palvelimelle. Tällä järjestelyllä verkon koulutusta voidaan nopeuttaa huomattavasti ja viikkojen ajon sijasta tuloksia saadaan muutaman päivän välein.

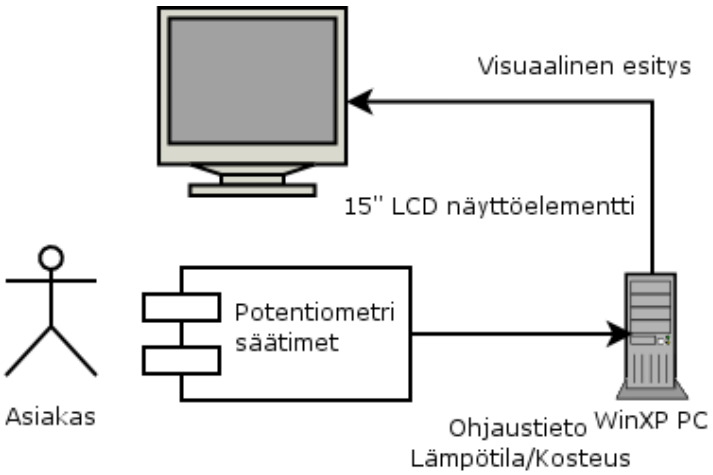
Malli muistuttaa Goldbergin kirjassaan *Genetic Algorithms* esittelemää rinnakkaista semisynkronoitua laskentajärjestelmää (s. 208). Vielä kehittyneempää versiota, jossa olisi käytetty hyödyksi alipopulaatioita ja muuttoa (migration), harkittiin. Tässä kuitenkin voiton vei ensimmäisen vaihtoehdon yksinkertaisuus toteutuksessa ja arvioitu riittävä tehokkuus.

Valinnassa hyödynnettiin ainoastaan kilpailutusvalintaa. Useaa populaation kokoa (25–50) testattiin. Risteytyksessä kokeiltiin useampaa eri vaihtoehtoa, joista useamman pisteen operaattorit osoittautuivat hyödyllisimmiksi. Pisteet valittiin neuronien väliltä, jotta liityntöjen tuhoutumiselta prosessissa vältyttäisiin. Mutaatioksi valittiin pistemutaatio, jossa yhtä painoarvoa muokattiin kerrallaan pienin muutoksin (random perpetuation), sekä vaihtomutaatiota, jossa painot saattoivat vaihtaa paikkaa keskenään. Kokonaan uuden painoarvon muodostavasta mutaatiosta luovuttiin, koska se näytti olevan liian vahingollinen ratkaisun kehittymiselle. Myös Buckland (2002, s. 346–409) ehdottamaa kehittyvää verkkoa kokeiltiin, mutta tulokset eivät näyttäneet olevan oleellisesti parempia. Tässä olisi ehkä vaadittu enemmän testausta ja säätöä, mutta tähän ei aikaa riittänyt.

6 VISUALISOINNIN TOTEUTTAMINEN

6.1 Järjestelmän yleinen kuvaus

Kuviossa 25 esitellään karkealla tasolla järjestelmän fyysinen ympäristö. Asiakas käyttää simulaatiota erikseen rakennettavan käyttöliittymän välityksellä. Tämä käyttöliittymä koostuu kahdesta potentiometrisäätimestä, joiden avulla käyttäjä syöttää haluamansa lämpötilan ja kosteuden simulaatiolle. Syöte tulee järjestelmälle standardimuotoisena hiiren liikeinformaationa, ts. x- ja y-koordinaatteina. Tarkemmat tiedot laitteistosta löytyvät liitteestä 5.



Kuvio 25 Järjestelmän ympäristön fyysinen kuvaus.

Kyseessä on siis eräänlainen informaatiokioskin kaltainen ratkaisu. Järjestelmän käyttäjät voidaan jakaa kahteen ryhmään, ylläpitoon sekä asiakkaisiin (näyttelyssä vierailevat). Käyttötapauksia ei lähdetty erikseen määrittelemään, koska järjestelmä on erittäin yksinkertainen tässä suhteessa. Järjestelmän toiminalle voidaan kuitenkin esittää useita vaatimuksia:

Yksinkertainen käyttöliittymä

Tarkoituksena on saavuttaa mahdollisimman monen käyttäjän mielenkiinto ja halu kokeilla järjestelmää. Luonnollinen, selkeä ja yksinkertainen käyttöliittymä on tähän avainasemassa.

Välitön vaste syötteeseen

Käyttäjän on saatava heti palaute kokeillessaan systeemiä. Tästä syystä pitkät tai vaativat ennakkolaskelmat eivät tule kysymykseen, vaan simulaation on käynnistyttävä heti ja tilanmuutoksen tulee olla selkeä.

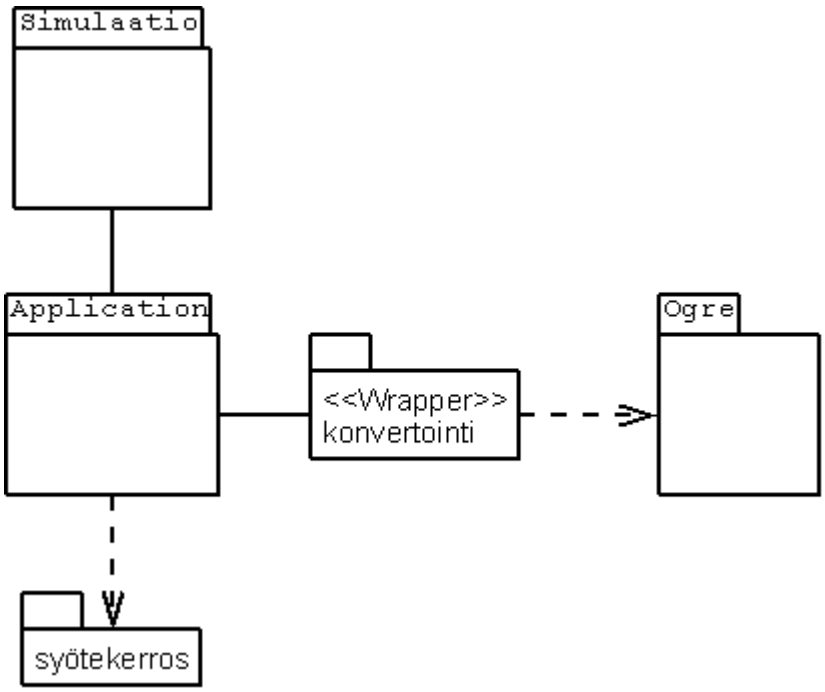
Esitys ei saa kestää myöskään pitkään, vaan tuloksen tulee syntyä lyhyessä ajassa. Tarkoituksena on, että käyttäjä katsoo läpi useamman simulaation tuotoksen eri arvoilla saadakseen kuvan erilaisista kiteistä. Kyseessä on yritys houkutella ihmiset kokeilemaan ja luomaan ”omia” kiteitään.

Järjestelmää käytetään ympäristössä, jossa koneiden käynnistys ja sammutus on ohjattu verkosta. Koneet sammutetaan näyttelypäivinä yön ajaksi ennalta määrätysti, mutta myös satunnaisesti huoltojen ajaksi. Järjestelmän tulee käynnistyä ja asettua käytettäväksi valmiustilaan aina, kun koneet käynnistetään uudelleen. Erillistä syötettä tai ohjausta ei siis voida vaatia järjestelmän käynnistämiseksi.

Järjestelmään mahdollisesti liitettävien ulkopuolisten kirjastojen on oltava lisenssoinniltaan sellaisia, että ne sallivat vapaan käytön kaupalliseen toimintaan eikä järjestelmän lähdekoodia tarvitse antaa julkiseen jakoon.

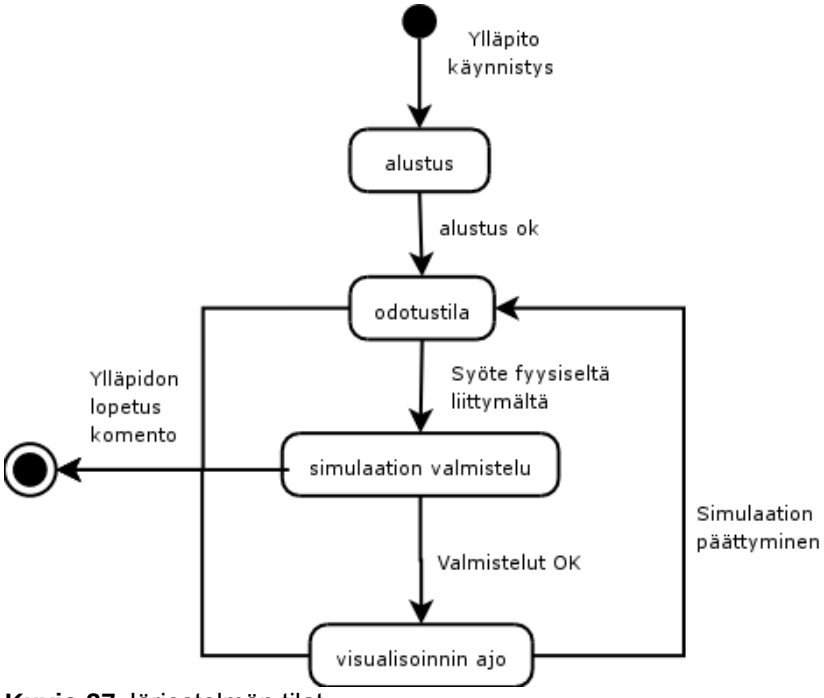
6.2 Visualisoinnin osat

Järjestelmä rakentuu 3 pääkomponentista. Näistä kaksi toteutettiin itse. Graafisen ulkoasun puolesta huolehtii kolmannen osapuolen grafiikkakirjasto Ogre3D, jota hyödynnettiin käyttöliittymän osalta. Kuviossa 26 esitetään tuotetun järjestelmän päärakenne. Koska suoranaista riippuvutta muihin kirjastoihin haluttiin välttää, luotiin ogre3D:n ja itse ohjelmiston välille sovitinluokka. Syötteen vastaanottamisesta, tallettamisesta ja muuntamisesta sopivaan muotoon eli neuroverkoista huolehtii syötekerros.



Kuvio 26 Järjestelmän organisaatio karkeasti.

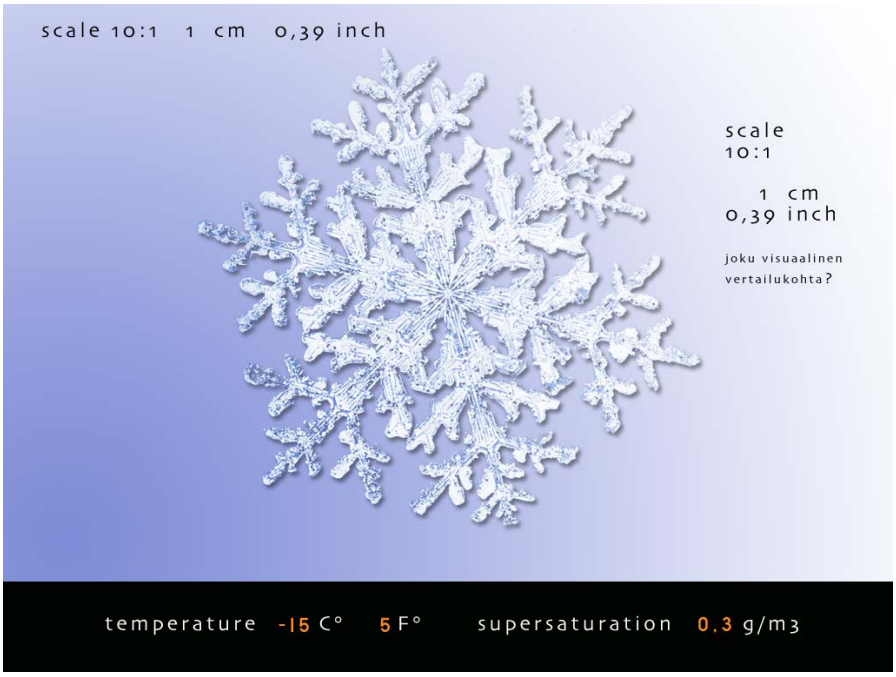
Kuviossa 27 esitetään järjestelmän läpikäymät tilat.



Kuvio 27 Järjestelmän tilat.

Käyttöliittymä

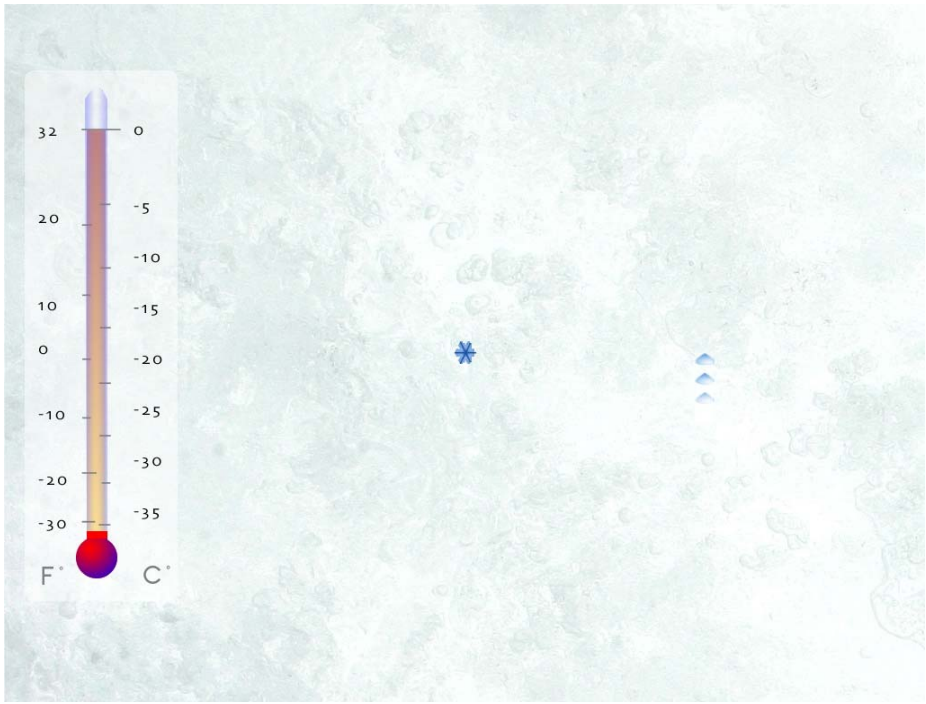
Visualisoinnin käyttöliittymää toteutettaessa oli otettava huomioon näytteilleasettajan vaatimukset sekä selvyys siitä, että lopullista graafista ulkoasua tultaisiin muokkaamaan useita kertoja. Työtä aloitettaessa näyttelyyn ei esimerkiksi vielä ollut sovittu yhteistä väriteemaa, joten tältä osin työ aloitettiin laatimalla yhdessä tilaajan kanssa kuvassa 3 näkyvä näyttömallikonsepti. Tästä konseptista muotoiltiin ajan kuluessa varsinainen käyttöliittymän ulkoasu, grafiikat tähän ulkoasuun tuotti näytteilleasettajan oma graafinen osasto. Muotoilun helpottamiseksi kaikkien komponenttien ulkoasua ja sijaintia pystyy muokkaamaan parametritiedostosta, joka pohjautuu Ogre3D:n overlay-tekniikohin. Tämä mahdollisti sen, että näytteilleasettaja pystyi itse testaamaan erilaisia ulkoasuja ja kokonaisuuksia vapaammin.



Kuva 3 Käyttöliittymänkonsepti (Arktikum).

Kuvassa 4 esitetään lopullinen versio, joka lähetettiin työnantajalle ja joka sisältää kaikkein tärkeimmät elementit. Lopullisessa versiossa päätettiin luopua kokonaan tekstuaalisesta informaatiosta itse esityksessä, koska se koettiin lähinnä häiritseväksi tekijäksi. Ikäjakaumaltaan ja taustaltaan laajan käyttäjäkunnan takia komponentit suunniteltiin mahdollisimman havainnollisiksi, lämpötilan kuvaamiseen käyte-

tään lämpömittaria sekä kosteuden kuvaamiseen pisaroita. Koska käyttöliittymä rakennettiin skaalautuvaksi ja elementit vaihdettaviksi, ei kuvan 4 ulkoasu välttämättä vastaa varsinaista näytteille asetettavaa.



Kuva 4 Varsinainen käyttöliittymä (graafiset elementit Kalle Pohjapelto, Arktikum).

OGRE3D

Ogre3D (Object-Oriented Graphics Rendering Engine) on suosittu avoimen lähdekoodin 3D-grafiikkamoottori. Ogre3D:n kehitys aloitettiin n. 7 vuotta sitten Steven Streetingin huomatessa, että senhetkiset grafiikkamoottoriprojektit eivät olleet riittävän kaukonäköisesti suunniteltuja ja johdettuja (Junker 2006, 1-6).

Streeting kehitti alustavan moottorin arkkitehtuurin, joka seitsemän vuoden kuluessa on kehittynyt yhdeksi suosituimmista avoimen lähdekoodin 3D-moottoriprojekteista. Ogren suosio perustuu hänen mukaansa sen kehitysfilosofiaan, joka on enemmän suunnittelu- ja tehonäkökulmaan perustuva kuin mahdollisimman laajaan ominaisuuskokoelmaan tähtäävä. (Junker 2006 1-6.)

Ogre3D:n lisenssi on nk. LGPL-lisenssi (Lesser GNU Public License). Se on tietyin rajoituksin vapaasti käytettävissä niin kaupallisiin kuin ei-kaupallisiinkin projekteihin, ja näin ollen se täyttää projektin vaatimukset käytöstä kaupalliseen tarkoitukseen. Tarkemmat lisenssitiedot löytyvät www.Ogre3D.org licence -kohdan alta.

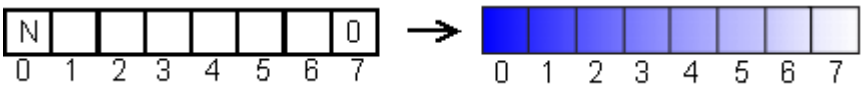
Ogren valinta grafiikkamoottoriksi perustui enemmän haluun tutustua siihen kuin varsinaiseen tarpeeseen. On selvää, että monet Ogren keskeisistä ominaisuuksista jäävät kyseisessä visualisoinnissa käyttämättä. Visualisoinnin ja simuloinnin toteutuskieleksi valittiin C++ tehokkuuden ja kielen tuttuuden vuoksi. Lisäksi huomattavana etuna C++:n valinnassa oli Ogre3D:n alkuperäistoteutuksen saatavuus kyseiselle kielelle.

6.3 Simulaation visualisointi

Luvussa 4.2 esitelty automaatti tuottaa rakenteen, joka sisältää erilaisia tietoja. Nämä tiedot on tulkittava sellaiseen muotoon, jota visualisoinnissa voidaan hyödyntää. Automaatti simuloitiin kaksiulotteisena taulukkona paremman laskentakyvyn saavuttamiseksi, eikä sen konvertointi suoraan tuota haluttuja tuloksia.

Ensimmäinen vaihe halutun tuloksen saavuttamiseksi on valita, mitä tietoja itse automaatista halutaan esittää. Tässä tapauksessa boolean muuttujat sisältävät tarkan kiteen muodon sekä jäätymismassa arvion suhteellisesta jäänpaksuudesta. Nämä ovat kiteen visualisoinnin kannalta oleellisimpia muuttujia.

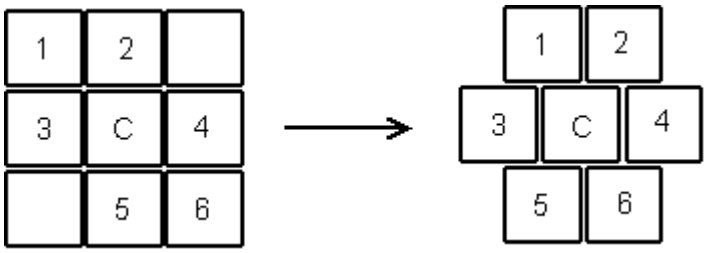
Automaatista visualisointiin valitun osan selvittämisen jälkeen on myös tehtävä päätös siitä, millä tavalla nämä tiedot ilmaistaan. Tässä käytetään hyödyksi ja inspiraatioksi sekä Libbrechtin kirjallisuuden esittelemiä kiteitä että Gravner/Griffheathin esittelemiä tuloksia ja arvioita omasta automaatistaan. Jäänpaksuuden visualisointi tuotetaan kartoittamalla eri automaatin arvot lineaarisesti väriarvoihin (kuvio 28). Mitä paksumpi jää on, sitä tummempana se esitetään.



Kuvio 28 Esimerkki automaatin arvojen kartoituksesta.

Väriarvot muodostuvat 32-bittisestä rgba-esityksestä, jossa jokaiselle komponentille on varattu 8 bittiä. Käyttäjä voi määritellä oman karttansa luomalla 1xb^2 kokoisen bittikartan, joka sisältää halutut värit ja läpinäkyvyyden (b^2= leveys kahden potenssina). Järjestelmä lataa tämän kartan käynnistyksen yhteydessä ja luo siitä oman sisäisen väriskaalansa, jota käytetään automaatin arvojen esittämiseen.

Toinen visuaaliseen ulkoasuun liittyvä seikka on automaatin rakenteen esittäminen niin, että se muodostaa kuusikulmaisen symmetrian edellä mainitun kaksiulotteisen taulukon sijasta. Tämä saavutetaan limittämällä, joka toisella rivillä sijaitsevat pisteet kuvion 29 tavalla. Tämän lisäksi tarvitaan shearing transformaatiotekniikka, jolla sivuttain ”nojaava” kuvio käännetään y-akselin suhteen (kaavat 16 ja 17).



Kuvio 29 Kuusikulmaisuuden saavuttaminen tavallisella näytöllä.

$$(x, y)\begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix} = (x + my, y)$$ horisontaali (16)

$$(x, y)\begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix} = (x, mx + y)$$ vertikaali (17)

jossa x,y = pisteen sijainti
m = transformaatiosuhde

Automaatin informaatiosta luodaan edellä mainittujen tekniikoiden avulla dynaaminen tekstuuri, jolla teksturoidaan yksinkertainen verteksiquadi. Quadin ja tekstuurin sijoittelusta vastaa aiemmin mainittu parametritiedosto ja se käsitellään yhtenä Ogre3D:n overlayna.

Optimointi

Koska kiteiden muodostus testisarjojen perusteella oli liian hidasta miellyttävän esityksen aikaansaamiseksi, on prosessia yritettävä nopeuttaa. Optimoinnissa voidaan hyödyntää lumikiteiden symmetrisyyttä. Kide voidaan jakaa osiin akseleiden suhteen, laskea vain yksi kiteen osa ja peilata tämä osa halutun tuloksen aikaansaamiseksi. Tällä tavalla on riittävää simuloida 1/12 varsinaisesta kiteestä ja itse solukon kooksi riittää ¼ kokonaiseen kiteeseen verrattuna. Tämän hyödyntäminen vaatii kuitenkin erityisesti reuna-ehtojen huomioon ottamista, jotta kiteen kasvu ei häiriinny.

Nopea simulaation profilointi osoitti, että lähes poikkeuksetta yksi algoritminvaihe kulutti eniten prosessointiaikaa. Osia algoritmista kirjoitettiin uudestaan käyttäen vähemmän raskaita operaatioita, puskurointia sekä vähentäen silmukkarakenteissa suoritettavia toimintoja ja yhdistäen operaatioita. Samalla voidaan algoritmin erivaiheissa jättää osa kiteestä huomioimatta. Kokonaisuudessaan optimoinnissa tasapainotettiin simulaation tarkkuutta siten, että visualisoitu kide olisi mahdollisimman tarkka, mutta pitäen samalla huolta, että muodostuminen oli riittävän nopeaa (n. 10 s).

7 JOHTOPÄÄTÖKSET

Lumikiteiden muodostumisen selvittäminen onnistui hyvin. Tässä ensiarvoisen tärkeässä asemassa oli Libbrechtin materiaali lumikiteistä, sillä suomenkielisen materiaalin löytäminen oli hankalaa. Kiteiden luokitteluja oli monia, joten niistä täytyi valita sellainen, mikä toi esille selkeät kidetyypit ja jättää huomioitta monet erikoisemmat muodot ja yhdistelmät.

Varsinaisia tietokonemalleja lumikiteiden muodostukseen löytyi paljon. Monien esittämät tulokset olivat kuitenkin heikkoja, eivätkä ne tuoneet esille yleensä kuin kiteen ulkoisen muodon jättäen monimutkaisen sisäisen rakenteen huomiotta. Oli tavallaan sääli, ettei 3D-kiteiden mallinnusta kannattanut lähteä mallien perusteella edes yrittämään. Olisi ollut erittäin mielenkiintoista nähdä oikean kaltaisen kiteen synty, vaikka sitten ei reaaliaikaisena. Mallin valinta ja simulaation rajaus kaksiulotteisiin oli mielestäni järkevää, paljon laajempaan tutkimukseen näiltä alueilta ei aikaa olisi riittänyt. Käytetyn mallin suhteen pidän soluautomaattien valintaa lähtökohdaksi onnistuneena. Muihin verrattuna erityisesti sisäisten ominaisuuksien osalta GG:n automaatti muodostaa hyviä tuloksia.

Erittäin haastava osuus työssä oli yrittää löytää jonkinlainen menetelmä tai tapa käyttäjän syötteen ja automaatin saamien arvojen välille. Onnistuin tässä mielestäni kohtuullisen hyvin, vaikka useissa kohdin voisikin esittää kysymyksen tehdyistä valinnoista, erityisesti lähtökohdassa neuroverkkojen valinnassa. Olisiko jokin menetelmä luonut paremman tuloksen, jäi vaivaamaan. Oli hyvä, että työllä oli selkeä päivämäärä (18.4.2007), jolloin järjestelmän tuli olla käytettävissä. Prosessissa oli niin monia säädettäviä kohtia, että tässä työssä olisi voinut mennä ikuisuus.

Varsinaisen visualisoinnin toteutus oli kahteen edelliseen työn osaan verrattuna helppoa. Järjestelmä oli yksinkertainen, lähinnä mietintää tuli käyttöliittymän muodostamisessa. Tässä lähtökohtana pidin, että työn tilaaja sai suunnitella haluamansa kaltaisen liittymän. Tästä yksinkertaisuudesta johtuen pidin myös visualisoinnin toteutuksen käsit-

telyn tässä raportissa lyhyenä. Työn aikana tehty järjestelmä on nyt osana Arktikumin jääluolaa ja päänäyttelyä.

Jatkokehitysmahdollisuuksia ja tutkimattomia vaihtoehtoja

Optimointi onnistui melko hyvin automaatin kohdalla, mutta parannettavaa olisi ollut huomattavasti. Parannuksena automaatti ja sen algoritmi olisi ollut mahdollista muuntaa sellaiseksi, jotta se olisi voitu laskea ja käsitellä tietokoneen prosessorin sijasta grafiikkakortin GPU:lla käyttäen pixel shadereita. GPU:t ovat erikoistuneet matriisi- ja vektorilaskentaan, jonka alueilla niiden suorituskyky ylittää selkeästi nykyaikaisilla prosessoreilla vastaavien laskujen suorittamisen. GPU:t ovat ohjelmoitavissa omalla C/assembly- kieltä muistuttavalla kielellään.

Algoritmin askeleet ja operaatiot olisi todennäköisesti voitu kirjoittaa matriisioperaatioiksi. Samalla olisi poistunut tarve kirjoittaa erikseen tulokset tekstuuriksi, joka siirretään grafiikkakortin muistiin. Tämän lisäksi olisi voitu hyödyntää CPU:n ja GPU:n rinnakkaintoimintaa, jolloin ohjelman muut osat olisi voitu jättää CPU:n huoleksi, kun taas raskaampaan laskentaan suunniteltu GPU olisi vastannut simulaation ajosta. Todellisesta suorituskyvyn erosta on vaikea esittää arveluita, mutta ainakaan hitaampi tämä tapa ei olisi ollut. Prosessia nopeuttamalla olisi voitu kasvattaa automaatin tarkkuutta ja parantaa näin visuaalista ulkoasua myös satunnaisuuden parempi huomioon ottaminen olisi ehkä ollut mahdollista.

Nykyään GPU:ta ja sen ominaisuuksia on alettu käyttää laajemmin tieteellisten sovellusten ja simulaatioiden toteutukseen. Yksi grafiikkakorttien valmistajista, Nvidia, on julkaissut oman kehitysalustansa tavallisten grafiikkakorttien paremmaksi hyödyntämiseksi kyseisellä alueella.

Toinen mahdollinen ja täysin todellinenkin parannuksen alue olisi ollut tutkia muita vaihtoehtoja parametrisointiongelman ratkaisuun. Neuraaliverkot sinänsä toimivat, mutta esimerkiksi bayesin verkkojen käyttö olisi saattanut tuottaa parempia tuloksia. Tässä on kuitenkin todettava, että koska toteutusaika oli selkeästi rajallinen ja työlle on asetettava järkevä laajuus, olisi tämän alueen kartoittaminen kasvattanut työmäärää huomattavasti. Yksi mielenkiintoinen vaihtoehto olisi ollut kokeilla

soluautomaatin sääntöjen opettamista suoraan eri verkkoratkaisuille. Eli tietyn solun muodostumisesta olisi päättänyt esimerkiksi koulutettu neuroverkko.

Geneettisten algoritmien puolella kokeiltavia tekniikoita ja ideoita löytyy lähes loputtomasti. Esimerkiksi koevoluutio (coevolution), jossa sekä ratkaisut että testitapaukset pisteytetään, olisi todennäköisesti parantanut tuloksia. Tässä mallissa myös testitapaukset saavat kelpoisuuden sen mukaan, kuinka monta ratkaisuyritelmää ne pystyvät hylkäämään. Sellaiset ratkaisut, jotka onnistuvat näiden selvittämisessä, saavat taas paremmat pisteet kelpoisuusfunktiolta. Toinen kehitysalue olisi ollut testata vielä useampia erilaisia geneettisiä operaattoreita.

Soluautomaatti olisi voitu muodostaa ennemmin komponenteista kuin yksittäisestä kiinteästä luokasta. Tällä olisi saavutettu mahdollisuus käyttää simulaation osuutta erilaisten automaattien luontiin. Itse Järjestelmän koodi ei kuitenkaan sopimusteknisistä syistä ole käytettävissä muuhun kuin esitystarkoituksiin.

Yhteenveto omista kokemuksista

Kokonaisuudessaan työ oli laaja ja monissa kohdin haastava. Oli mielenkiintoista tutustua lumikiteiden syntyyn, josta en alun perin tiennyt juuri mitään. Malleja tutkittaessa oli yllättävää, kuinka paljon kiteiden muodostumista on yritetty erilaisin menetelmin ja kuinka vaihtelevia tulokset ovat olleet. Miellyttävä yllätys oli myös oman intuition kantavuus soluautomaattien mahdollisuuksista ensimmäisestä työnantajan kanssa käydystä keskustelusta lähtien.

Reuna-alueiden käsittelyssä ja reaalilukujen kanssa toimiessa ilmenneet "kaaosefektit" (pienten virheiden kertautuminen), aiheuttivat epäilyjä ettei kiteen simulointi reaaliajassa olisi mahdollista. Kokonaisuudessaan olen tyytyväinen saatuun lopputulokseen ja prosessiin. Opin useista mielenkiintoisista tekniikoista lisää (geneettiset algoritmit, neuroverkot). Tutustuin myös aiemmin paljon kiinnostaneeseen mutta vähälle tutkimiselle jääneeseen aihealueeseen, soluautomaatteihin. Uskon, että tästä mallista tulee tulevaisuudessa entistä tärkeämpi erilaisten ongelmien ratkaisussa.

LÄHTEET

Ala-Korpela, Mika – Inkinen, Sam – Suna Teemu 2007. Kyborgin käsikirja, Havaintoja informaatiosta, ihmisestä ja koneesta, elämästä ja älykkyydestä. Otava, Helsinki.

Buckland, Mat 2002. AI techniques for game programming. Premier Press, Cincinnati.

Emmeche, Claus 1991. Tekoelämä. Tmi Datata, Nummi.

Goldberg, David 1989. Genetic algorithms in Search, Optimization & Machine
Learning. Addison-Wesley Publishing Company, inc. Massachusetts.

Gravner, Janko – Griffeath, David 2006. Modeling snowcrystal growth II: A mesoscopic lattice map with plausiable dynamics. Esipainos
syyskuu 2006.

Hämäläinen, Ari 1992. GA and Neural Networks. Teoksessa Proceedings of the First Finnish Workshop on Genetic Algorithms and their Applications (toim. Jarmo T. Alander), 74–84. Teknillinen korkeakoulu, tietotekniikan osasto, tietojenkäsittelyn laitos, Helsinki.

Junker, Gregory 2006. Pro OGRE 3D Programming. Apress, Newyork.

Kari, Jarkko 1990. Ratkeamattomuus soluautomaateissa. Teoksessa Kaaostutkimus ja tietotekniikka (toim. Ilkka Karanta ja Jouko Seppänen), 13–21. Suomen tekoälyseuran julkaisuja – No 4, Helsinki.

Kim, Theodore – Henson, Michael – Lin, Ming C 2007. A Hybrid Algorithm for Modeling Ice Formation. Osoitteessa http://www.cs.unc.edu/~geom/HYB_ICE/ 12.4.2007.

Libbrecht, Kenneth 2004. The Snowflake Winters Secret Beauty. Colin Baxter Photography Ltd, Grantown-on-Spey.

- - - 2005. The physics of snowcrystals. Osoitteessa http://www.its.caltech.edu/~atomic/publist/rpp5_4_R03.pdf 12.4.2007.

- - - 2007. Snowcrystals.com. A guide to snowcrystals. Osoitteessa http://www.its.caltech.edu/~atomic/snowcrystals/class/class.htm 12.4.2007.

N.A.S.A, 2007. Winter's story resources. Student observation Network: Winter's story. Osoitteessa http://son.nasa.gov/winterstory/snowstorm/index_e.htm 12.4.2007.

Niskanen, Vesa 2003. Sumea Logiikka kirkasta älyä ja mallinnusta. Dark oy, Vantaa.

Oksanen, Tari 1999. Suomen lumipeitteen alueellinen vaihtelu. Helsingin yliopiston geofysiikan laitos pro gradu -tutkielma

Reiter, Clifford A. – Coxe, Angela M. 2007 Fuzzy Hexagonal Automata and Snowflakes. Osoitteessa http://ww2.lafayette.edu/~reiterc/mvp/fz_hx_auto/fz_hx_auto_pp.pdf 7.4.2007.

Wolfram, Stephen 2002. A New Kind of Science. Wolfram Media Inc, Winninpeg.

Zadeh, Lofti A. – Fu, King-Sun – Tanaka, Kokichi – Shimura, Masamichi 1975. Fuzzy sets and their applications to cognitive and decision processes. Academic press inc, New York.

LIITTEET

Lumikide liite 1

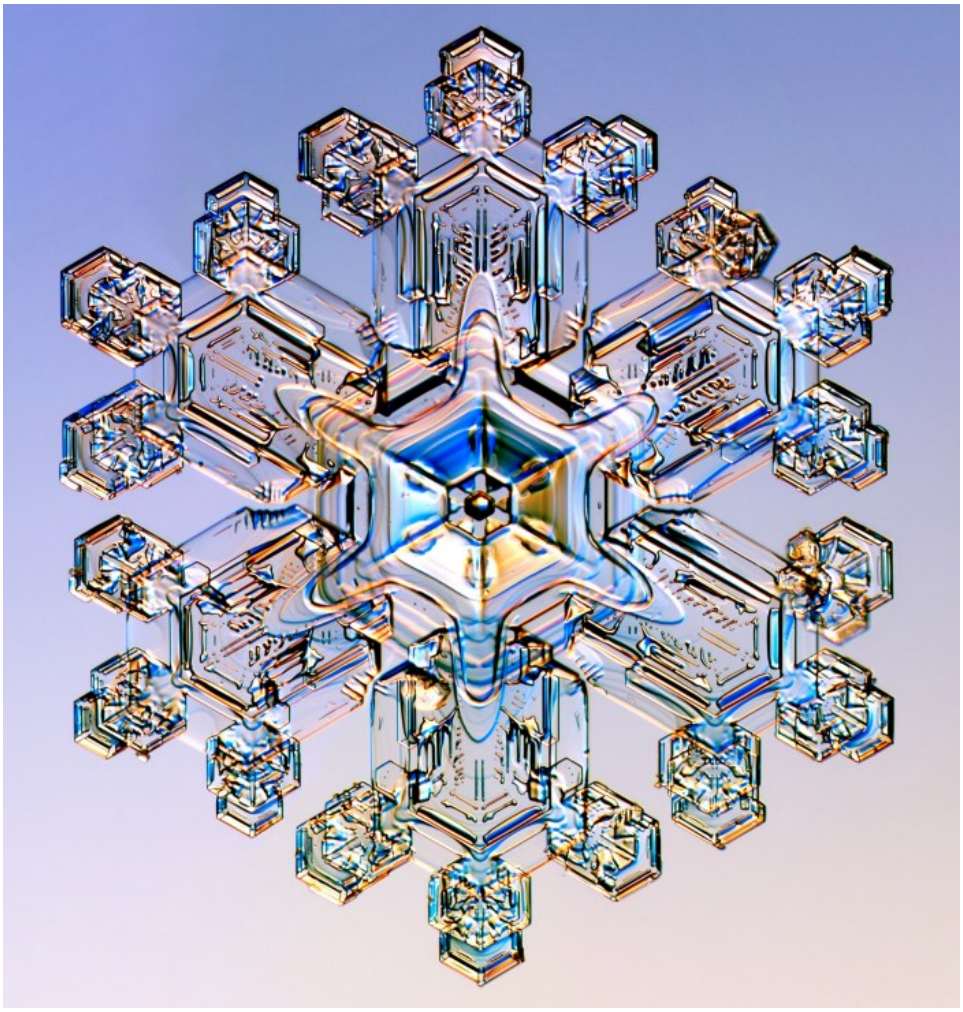
Magono–Lee luokittelu liite 2

Simuloitu lumikide liite 3

Testisarjojen tuloksia liite 4

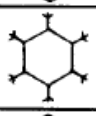
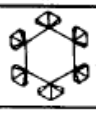
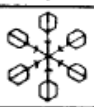
Visualisointilaitteiston kokoonpano liite 5

Liite 1



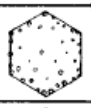
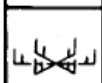
(Libbrecht 2007.)

Liite 2 (1)

	N1a Elementary needle		C1f Hollow column		P2b Stellar crystal with sectorlike ends
	N1b Bundle of elementary needles		C1g Solid thick plate		P2c Dendritic crystal with plates at ends
	N1c Elementary sheath		C1h Thick plate of skeleton form		P2d Dendritic crystal with sectorlike ends
	N1d Bundle of elementary sheaths		C1i Scroll		P2e Plate with simple extensions
	N1e Long solid column		C2a Combination of bullets		P2f Plate with sectorlike extensions
	N2a Combination of needles		C2b Combination of columns		P2g Plate with dendritic extensions
	N2b Combination of sheaths		P1a Hexagonal plate		P3a Two-branched crystal
	N2c Combination of long solid columns		P1b Crystal with Sectorlike branches		P3b Three-branched crystal
	C1a Pyramid		P1c Crystal with broad branches		P3c Four-branched crystal
	C1b Cup		P1d Stellar crystal		P4a Broad branch crystal with 12 branches
	C1c Solid bullet		P1e Ordinary dendritic crystal		P4b Dendritic crystal with 12 branches
	C1d Hollow bullet		P1f Fernlike crystal		P5 Malformed crystal
	C1e Solid column		P2a Stellar crystal with plates at ends		P6a Plate with spatial plates

(N.A.S.A 2007.)

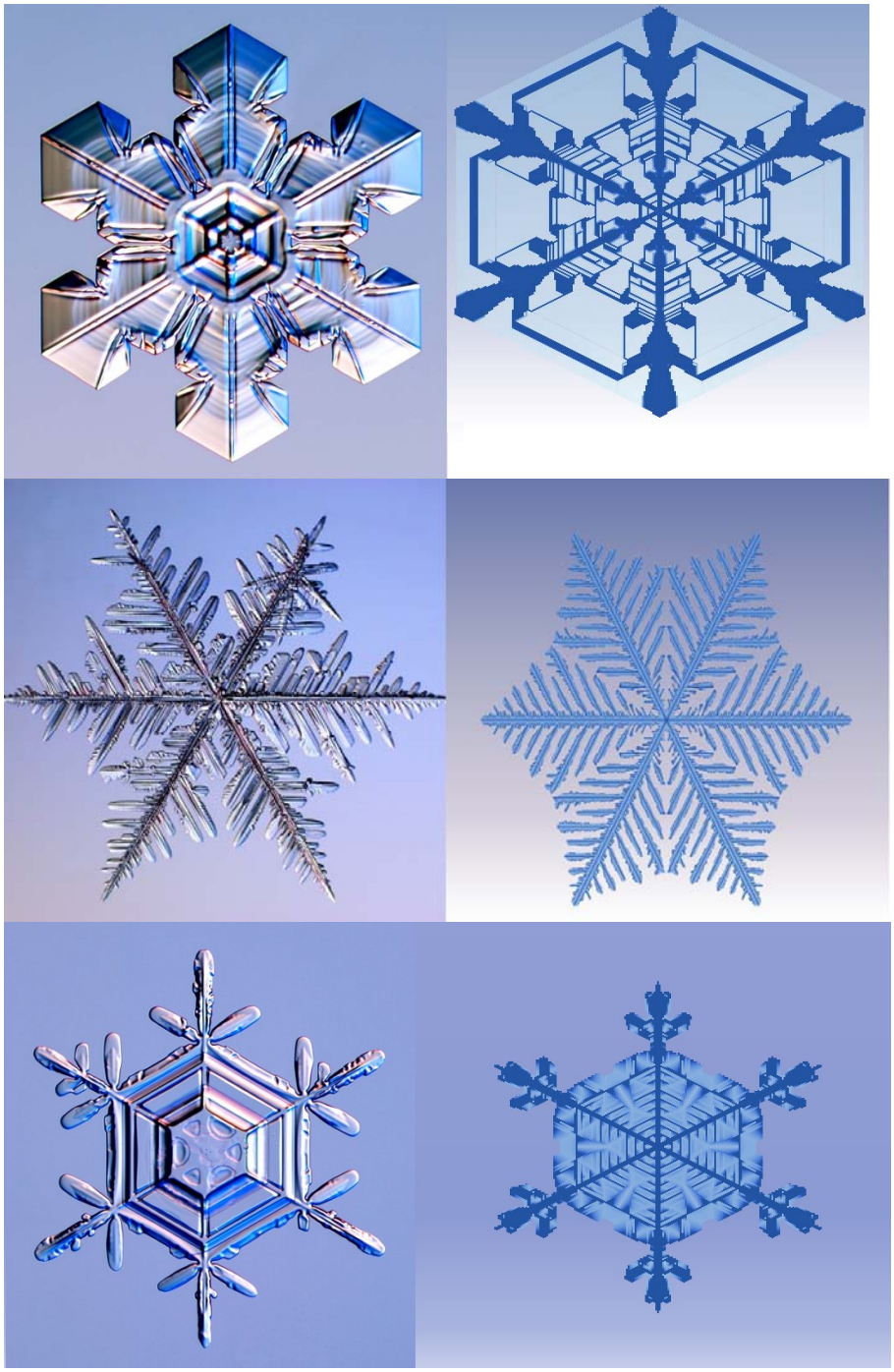
Liite 2 (1)

	P6b Plate with spatial dendrites		CP3d Plate with scrolls at ends		R3c Graupel-like snow with nonrimed extensions
	P6c Stellar crystal with spatial plates		S1 Side planes		R4a Hexagonal graupel
	P6d Stellar crystal with spatial dendrites		S2 Scalelike side planes		R4b Lump graupel
	P7a Radiating assemblage of plates		S3 Combination of side planes, bullets, and columns		R4c Conelike graupel
	P7b Radiating assemblage of dendrites		R1a Rimed needle crystal		I1 Ice particle
	CP1a Column with plates		R1b Rimed columnar crystal		I2 Rimed particle
	CP1b Column with dendrites		R1c Rimed plate or sector		I3a Broken branch
	CP1c Multiple capped column		R1d Rimed stellar crystal		I3b Rimed broken branch
	CP2a Bullet with plates		R2a Densely rimed plate or sector		I4 Miscellaneous
	CP2b Bullet with dendrites		R2b Densely rimed stellar crystal		G1 Minute column
					G2 Germ of skeleton form
	CP3a Stellar crystal with needles		R2c Stellar crystal with rimed spatial branches		G3 Minute hexagonal plate
	CP3b Stellar crystal with columns		R3a Graupel-like snow of hexagonal type		G4 Minute stellar crystal
					G5 Minute assemblage of plates
	CP3c Stellar crystal with scrolls at ends		R3b Graupel-like snow of lump type		G6 Irregular germ

(N.A.S.A 2007.)

Liite 3

Liite 4



(Lumikiteet Libbrecht 2007.)

A3.2 Winter' Art

Tekniikka/ Tech:

Tietokone: prosessori
Optiplex GX620 MT 3,0 GHz, 1,0 GB muistia
(Sound Blaster X-Fi Xtreme)
ATI Radeon X1650 Pro --- tilattu, ei vielä toimitettu

Näyttö:
15" LCD-näyttö
Planar LB1503R

Resoluutio: 1024 x 768
Kuvasuhde: 4:3

2 potentiometriä

Kuvaus/ Description:

Miten lumikiteet muodostuvat

Toteutustapa/Implementation

Idea/Synopsis:

Kaksi potentiometriä kytkettynä tietokoneeseen, joista voidaan valita lämpötilan ja ilmankosteuden suhde. Lopputuloksena näytetään millainen lumikide syntyy vallitsevassa tilassa.

Potentiometrit säätelevät lämpötilaa (min. ja max. lämpötilat!!) ja ilmankosteutta.