Bayes-verkkojen ja geneettisten algoritmien mahdollisuudet

Henry Tirri Petri Myllymäki Jarmo Alander

Esipuhe

Alkusanat

Tiivistelmä

Summary

Sisältö

Johdanto

Erkki Oja

Oppivien ja älykkäiden järjestelmien tutkimus on viime aikoina organisoitunut kokonaisuudeksi, josta käytetään myös nimiä *laskennallinen älykkyys* (computational intelligence) ja *soft computing*. Ajatuksena on, että vaikeissa päättelyongelmissa korvataan eksplisiittisten mallien ja ihmistietämyksen puute laskennallisilla (numeerisilla) malleilla, jotka sovitetaan tutkittavaan kohteeseen suurelta osin mittaustietoja hyväksikäyttäen. Tämä tietysti vain silloin, kun paremman puutteessa on pakko turvautua dataohjattuihin empiirisiin malleihin. Asiantuntijalla on kyllä näkemys kohteensa (esimerkiksi prosessi tai laite) toiminnasta, mutta tarkkojen ja ennustusvoimaisten mallien aikaansaamiseksi voi olla pakko täyttää puuttuvia osia soft computing-malleilla. Lopputulos on silloin monista osista koostuva *hybridimalli.*

Soft computing-alan valtavirrat ovat neuroverkot ja sumeat järjestelmät. Niillä on eniten sovelluksia niin Suomessa kuin muuallakin, alat ovat jo kypsyneet valmiiden ohjelmistopakettien ja hyvätasoisten oppikirjojen tasolle, ja runsaasti käyttökokemusta on olemassa neuroverkkojen tai sumeiden sääntöjen — tai molempien — soveltamisesta laajaan kirjoon käytännön ongelmia. Näitä kahta alaa sovelluksineen on selostettu Tekesin aikaisemmissa raporteissa "Sumean logiikan mahdollisuudet" (1993) ja "Neurolaskennan mahdollisuudet" (1994). Laskennalliseen älykkyyteen kuuluu kuitenkin muitakin metodiikkoja. Tärkeimmät ovat *evoluutiolaskenta*, etenkin geneettiset optimointialgoritmit, sekä *Bayes-verkot*, joita nyt käsillä oleva raportti käsittelee (kuva 1).

Voi tietenkin kysyä, onko yleensä mielekästä puhua tieteenalasta nimeltä laskennallinen älykkyys tai soft computing, joka määritellään tiettynä suljettuna menetelmäjoukkona kuvan 1 mukaisesti. Varmaa ainakin on, että ala on voimakkaassa muutostilassa ja uusia teknologioita ja menetelmiä epäilemättä syntyy lähitulevaisuudessa, joten kuva on koko ajan muutostilassa. Eri menetelmien lähtökohdat ja sovellusmahdollisuudet ovat myös osittain erilaisia.

Mielestäni — ja monen muunkin alan tutkijan mielestä — on järkevää niputtaa nämä uudet alat omaksi metodologiakseen tai tieteenalakseen, koska niillä on kuitenkin joukko yhteisiä piirteitä, jotka erottavat ne perinteisestä komputoinnista. Ensinnäkin niiden välillä on yhteyksiä: esimerkiksi neuroverkkoja käytetään virittämään sumeita järjestelmiä, geneettisillä algoritmeilla voi tehokkaasti optimoida neuroverkkojen rakennetta, ja Bayes-verkot antavat matemaattisesti perustellun ja siksi voimakkaan mallin samoihin päättelyongelmiin, joita ratkotaan neuroverkoilla tai sumeilla järjestelmillä. Toiseksi, ja ehkä vieläkin tärkeämpänä syynä, on menetelmien perusidea ja lähtökohta — enemmän tai vähemmän biologisista tai kognitiivisista prosesseista johdetaan toimintamalleja, jotka yksinkertaistettuina ja muunneltuina sovitetaan oppiviksi ja älykkäiksi ratkaisuiksi käytännön ongelmiin. Tärkeää on nimen omaan laskennallisuuden korostaminen: oppivat ja älykkäät järjestelmät, ainakin realistisissa isoissa ongelmissa, toimivat tehokkaidenkin tietokoneiden äärirajoilla, eikä niitä olisi voinut ajatellakaan käytännön menetelminä vielä kymmenenkään vuotta sitten. Tietokoneiden tehon edelleen kasvaessa tulemme varmasti näkemään uusia laskennallisesti älykkäitä menetelmiä, jotka aina käyttävät kaiken saatavilla olevan laskentatehon yhä parempien ja parempien päättelymallien rakentamiseksi.

Mitä soveltaja tästä alojen niputtamisesta sitten hyötyy? Suoranaisesti ei paljoakaan. Kuitenkin ne valmiit menetelmät, joita hän ongelmiinsa kokeilee tai käyttää, ovat syntyneet tutkimuksen tuloksena, jossa erityyppisten oppivien ja älykkäiden järjestelmien vuorovaikutus on antanut uusia ideoita ja johtanut parempiin tuloksiin.

Bayes-verkot ovat päättelyjärjestelmiä, jotka nojautuvat vanhaan ja koeteltuun todennäköisyyslaskentaan, etenkin ns. bayesiläiseen päättelyyn. Lyhyesti sanottuna tämä tarkoittaa, että kaikille mallituksen elementeille — itse malliarkkitehtuurille, sovitettaville parametreille ja sovitukseen käytettävälle datalle — oletetaan todennäköisyysjakaumat, ja valitsemme sen mallin ja ne parametrit, jotka ovat kaikkein todennäköisimmät ottaen huomioon saamamme mittausdatan. Ajatus on elegantti ja sallii asiantuntijan tietämyksen käyttämisen, koska hän voi kokemuksensa perusteella valita erilaisten mallien ja parametrien etukäteistodennäköisyydet (priorit). Vaikeutena toisaalta on, että todennäköisyysformalismi eleganttisuudessaan voi olla liiankin yleinen — suurta joukkoa tarvittavia todennäköisyysjakaumia on hankala estimoida ja niitä yhdistäviä hyvin moniulotteisia integraaleja on vaikea laskea. Tällöin joudutaan voimakkaisiin yksinkertaistuksiin, jotka murentavat mallien tyylikästä formalismia. Kuitenkin yksinkertaistetutkin mallit ovat monissa käytännön ongelmissa, esimerkiksi luokittelussa, osoittautuneet hyvin tehokkaiksi ja toimiviksi.

Geneettiset algoritmit ovat optimointimenetelmiä. Niissä pyritään löytämään jonkin annetun kohdefunktion optimaalinen minimi tai maksimi. Funktion ei tarvitse olla annettuna matemaattisessa muodossa, vaan pikemminkin menetelmä puree parhaiten ongelmiin joissa kohdefunktion arvot joudutaan laskemaan tai simuloimaan ehkä kalliistikin tietokoneohjelmilla. Kohdefunktio voi kuvata vaikkapa hissin ohjausjärjestelmää tai jonkin suunniteltavan koneenosan muodon hyvyyttä. Erona useimpiin muihin optimointimenetelmiin geneettinen algoritmi käyttää joukkoa — sukupolvea — rinnakkaisia ratkaisuyritteitä, joista tietyillä satunnaisilla menetelmillä yhdistellään vielä parempi sukupolvi uusia ratkaisuyritteitä. Paremmuus mitataan laskemalla kohdefunktion arvot näille yritteille. Näitä sukupolvia lasketaan yhä uusia, kunnes niiden joukosta löytyy jokin riittävän hyvä ratkaisyksilö. Menetelmä on tehokas ja kohtalaisen helppo toteuttaa valmisohjelmistojen avulla. Vaikeutena on ehkä löytää optimointiongelmaa parhaiten kuvaava kohdefunktio ja sopivat muuttujat, joiden varaan ratkaisusukupolvet rakennetaan.

Korostettakoon tässä, että Bayes-verkot ja geneettiset algoritmit ovat käytännössä aivan eri menetelmiä, joiden sovellusalueetkin ovat erilaiset: Bayes-verkoille päättelymallit, diagnostiikka ja luokittelu, ja puolestaan geneettisille algoritmeille yleiset optimointiongelmat. Ne on kuitenkin tässä kirjassa koottu samoihin kansiin käytännön syistä. Selostukset ovat itsenäiset, ja lukija voi tutustua jompaankumpaan menetelmään aivan toisesta riippumatta.

Bayes-verkkojen mahdollisuudet

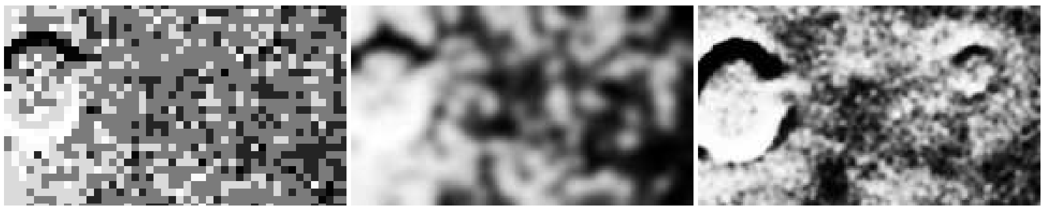
Petri Myllymäki ja Henry Tirri

Luku 1

Bayesiläinen mallinnus

Motivointia. Täydellisen tietokoneperustaisen mallin rakentamisen mahdottomuus. Oppivien ja älykkäiden järjestelmien yleinen rakenne — mallien oppiminen ja niiden soveltaminen. Mallien oppiminen bayesiläisittäin: mallien rakenteen ja parametrien oppimisen erottaminen. Ylioppiminen ja Occamin partaveitsi. Oppimisen ja taustatietämyksen yhdistäminen. Bayesiläisten mallien soveltaminen.

Seuraavassa on lueteltu joukko eri sovellusaloilla (kuvankäsittely, tiedon koodaus ja tiivistys, ohjelmistotekniikka) viime vuosina kehitettyjä merkittäviä teknologioita. Erilaisista sovellusalueistaan huolimatta nämä teknologiat perustuvat samaan taustalla olevaan teoreettisen kehikkoon.



Kuva 1.1: Viking-luotaimen Marsin pinnasta ottamista kuvista muodostettu superresoluutiokuva. Vasemmalla: yksi 24 alkuperäisestä kuvasta. Keskellä: kuva perinteisillä kohinansuodatustekniikoilla käsiteltynä. Oikealla: 24 kuvan aineistosta yhdistetty superresoluutiokuva.

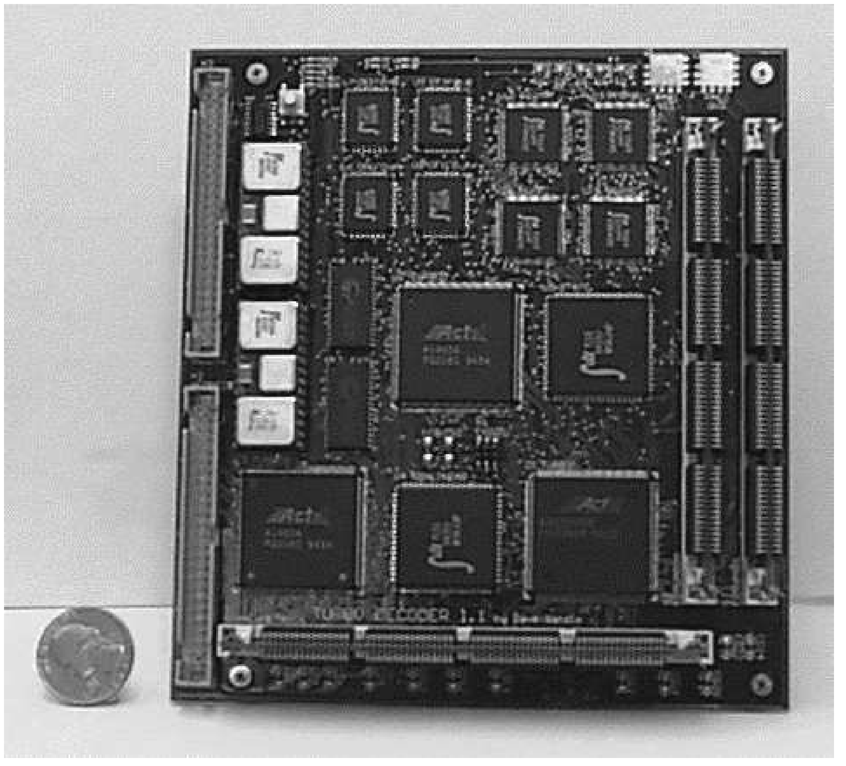
Superresoluutio. Viking-avaruusluotain tuottaa useita kuvia Mars-planeetasta. Kuvat on otettu samankaltaisissa valaistusolosuhteissa ja samoista kohteista. Tällöin instrumenttien pienet orientaatio- ja kohdennuserot tuottavat kuitenkin hieman erilaisia kuvia samasta kohteesta. Tällaisesta kuvajoukosta voidaan yhdistelemällä tuottaa tietokoneella “superresoluutiokuva”:

kuva joka on tarkempi kuin millään kohinansuodatustekniikalla yksittäisestä kuvasta saatu kuva (kuva 1.1). National Aeronautics & Space Administration (NASA) käyttää tätä useaan kuvaan perustuvaa superresoluutiotekniikkaa avaruusluotainten tuottamien kohinaisten kuvien parantamiseen, ja tällaisella kuvankäsittelytekniikalla on merkittäviä sovelluksia mm. lääketieteessä ja biotieteissä.

Turbo-koodaus. Tietoliikenteen määrän jatkuvasti kasvaessa tulee tiedon tehokkaasta koodaamisesta ja tiivistämisestä yhä keskeisempiä ongelmia. Vuonna 1993 kehitettyä turbo-koodausta pidetään koodausteorian merkittävimpänä keksintönä vuosikymmeniin. Turbo-koodaus soveltuu tiedon välittämiseen kohinaista tiedonsiirtokanavaa käyttäen, ja sen tehokkuus tiedon välityksessä on moninverroin parempi aiempiin menetelmiin verrattuna. Turbokoodit ovat erityisen merkittävä edistysaskel avaruusteknologiassa, jossa ne ylittävät mm. Voyagerin käyttämän ns. “katenointikoodin” (concatenated codes) moninkertaisesti. Turbo-koodaus osoittaa myös kuinka nopeasti teoreettinen tutkimus siirtyy käytännön sovellukseksi, erilaisia turbo-koodeihin perustuvia piirejä valmistavat jo useat yhtiöt kuten Efficient Channel Coding Inc. (kuva 1.2).

Älykkäät agentit. Nykyiset tekstin ja kuvien käsittelyyn kehitetyt ohjelmistot ovat laajoja ohjelmakokonaisuuksia joiden monien piirteiden tehokas käyttäminen on vaikeaa, ja käytön oppiminen hidasta. Microsoft Office’97-ohjelmistossa käyttäjän apuna ovat “älykkäät agentit” (Office Assistants), jotka itsenäisinä moduuleina tarkkailevat käyttäjän toimintaa ja tarjoavat apuaan käyttäjälle mm. monimutkaisten muotoilujen toteuttamisessa (kuva 1.3). Tällainen teknologia edellyttää varsin monimutkaista ohjelman toimintojen ja käyttäjän tarpeiden yhteismallinnusta, jotta agentti ei tarjoaisi vain triviaaleja ohjeita. Älykäs agenttiteknologia on erityisesti verkottumisen myötä muodostumassa yhdeksi tärkeimmistä uusista ohjelmistosuuntauksista, ja sen keskeisimpiä sovellusalueita ovat tiedon haussa avustaminen, liikkuva tietojenkäsittely sekä ohjelmistojen ja laitteiden konfiguroinnissa avustaminen.

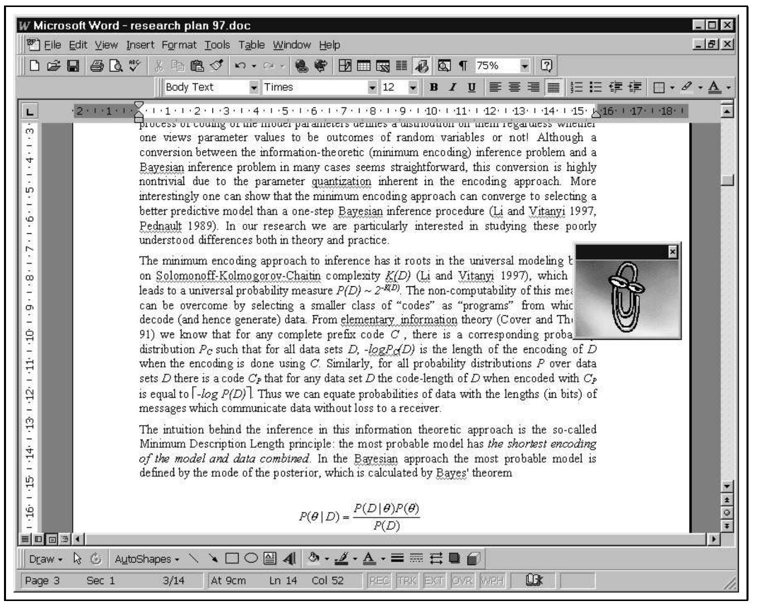
Mikä sitten on yhteistä näille edellä lyhyesti esitetyille tekniikoille? Vastaus liittyy sovellusten taustalla olevaan mallinnukseen joka perustuu ns. bayesiläiseen lähestymistapaan muodostaa malleja ja käyttää niitä päätöksenteossa. Yksinkertaisimmillaan bayesiläisen mallinnuksen voi ymmärtää todennäköisyyslaskentaan perustuvana menetelmänä, joka vastaa kysymykseen, miten käsityksemme jostakin asiasta (uskomuksemme jonkin asian totuudesta) muuttuu kun saamme asiaan liittyvää uutta tietoa. Bayesiläinen mallintaminen on kunnioitettavasta iästään huolimatta — perusteet esitettiin jo 200 vuotta sitten — yksi nopeimmin kehittyviä ja kasvavia mallinnusmenetelmiä, ja sen soveltamisella on kauaskantoisia ja merkittäviä vai-



Kuva 1.2: Turbo-koodauksen toteuttava piiri.

kutuksia mm. lääketieteessä, teollisessa mallintamisessa, ekonometriassa ja yhteiskuntatieteissä.

Raportin tässä osassa käsitellään bayesiläistä mallintamista keskittyen erityisesti ns. Bayes-verkkomalleihin, jotka ovat mm. edellä esitettyjen turbokoodauksen ja Office Assistant-teknologian perustana. Tarkastelemme ensin bayesiläistä lähestymistapaa yleisesti oppivien ja älykkäiden järjestelmien rakentamisen kannalta, ja sitä kuinka lähestymistapaa voidaan soveltaa mallien konstruoimisessa ja epätäydellisen tiedon varassa toimivassa päättelyssä. Luvussa 2 arvioidaan bayesiläisen mallinnuksen suhdetta muihin oppivien ja älykkäiden järjestelmien yhteydessä esiintyviin menetelmiin kuten esimerkkeihin perustuvaan päättelyyn (Case-Based Reasoning, CBR), ja tarkastellaan lyhyesti bayesiläiseen mallintamiseen liittyvää informaatioteoreettista tulkintaa. Luku 3 tarjoaa katsauksen erilaisiin Bayes-verkkomalleihin ja niihin liittyviin käsitteisiin. Luvussa 4 esitellään Bayes-verkkojen sovelluksia ja Bayes-verkkojen sovellusten kehittämiseen suunniteltuja kaupallisia ohjelmistoja. Lopuksi luvun 5 tavoitteena on esitellä alueen keskeisiä tietolähteitä sekä bayesiläiseen mallintamiseen yleensä että erityisesti Bayes-verkkoihin



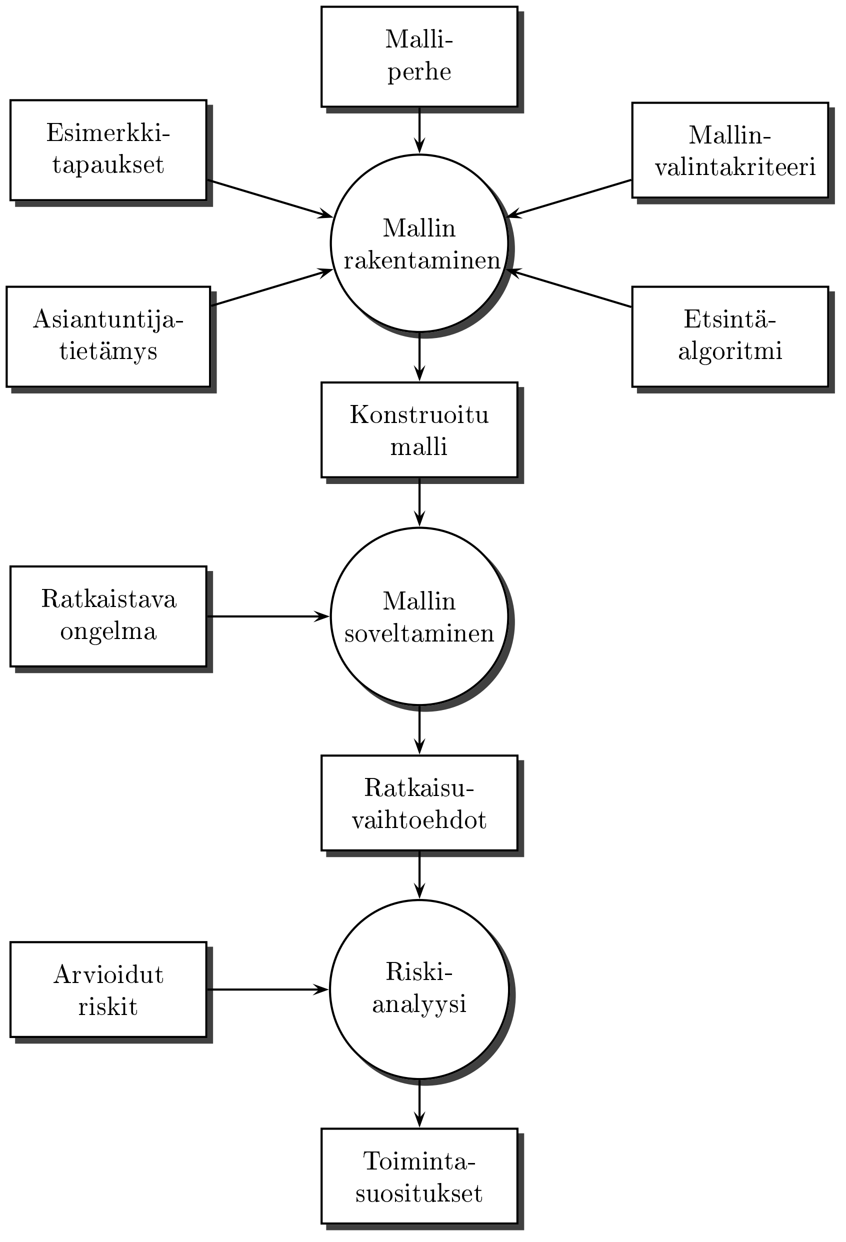
Kuva 1.3: Microsoftin Office Assistant-agentti.

liittyen.

1.1 Bayesiläinen mallintaminen oppivissa ja älykkäissä järjestelmissä

Oppivissa ja älykkäissä järjestelmissä pyritään rakentamaan käsiteltävästä sovellusalueesta yksinkertaistettu matemaattinen malli siten, että mallin toteuttavaa tietokoneohjelmaa voidaan käyttää avuksi erilaisissa ongelmanratkaisutilanteissa. Kuvassa 1.4 on tällaisen järjestelmän rakenne jaettu kolmeen komponenttiin: mallien rakentaminen, mallien soveltaminen, ja riskianalyysi. "Mallien soveltaminen" on tässä yhteydessä intuitiivisesti määritelty käsite, joka kattaa mm. seuraavat ongelmakentät:

Tilastollinen päättely ja ennustaminen:
Rakennettu malli on ongelmakentän matemaattinen kuvaus, ja mallinsovellusmoduulille annettua syötettä voidaan tässä tapauksessa ajatella mallinnetussa maailmassa esiintyvän tilanteen osittaisena kuvauksena. Tilastollisessa päättelyssä ja ennustamisessa on päämääränä selvittää mitä annetuista tiedoista seuraa annetun mallin puitteissa, tai mitkä ovat annettuun tilanteeseen johtaneet syyt. Tähän ryhmään kuuluvat muun muassa luokittelu-, diagnosointi-, ja aikasarjaongelmat.



Kuva 1.4: Oppivien ja älykkäiden järjestelmien yleisrakenne.

Ongelmakentästä kerätyn tiedon analysointi (data mining):
Ongelmakentästä kerätystä tiedosta muodostetaan malli, jonka avulla tutkitaan ongelmakentässä esiintyviä yleisiä säännönmukaisuuksia. Mallille annettu syöte (ratkaistava ongelma) määrittelee tässä tapauksessa esimerkiksi millaisista säännönmukaisuuksista käyttäjä on kiinnostunut.

Teollisten prosessien säätö ja ohjaus:
Rakennettu malli kuvaa jotakin todellisen maailman teollista prosessia. Mallille annettu syöte on tässä tapauksessa joukko malliin vaikuttavia parametreja, ja mallin soveltamisessa on tehtävänä päätellä kuinka määrätä loput parametrit optimaalisella tavalla annetun mallin puitteissa.

Tiedon tiivistäminen ja signaalinkäsittely:
Mallinnettu todellisen maailman prosessi käsittelee tässä tapauksessa signaalien tiivistämistä (compression), suodattamista (filtering), tai korjaamista (correction). Mallin sovellusvaiheessa on tehtävänä optimoida tätä prosessia ohjaavat parametrit.

Mikäli mallien sovellusmoduulin antamien tulosten perusteella tehdään toiminnallisia päätöksiä, on riskianalyysivaiheen tehtävänä arvioida eri vaihtoehdoista seuraavat hyödyt ja haitat.

Bayesiläinen mallintaminen on yksi kiinnostava lähestymistapa oppivien ja älykkäiden järjestelmien rakentamiseksi. Lähestymistapa tarjoaa yhtenäisen, todennäköisyyslaskentaan perustuvan formaalin kehikon, jonka avulla rakennettavan järjestelmän eri vaiheissa esiintyvät osaongelmat voidaan muotoilla ristiriidattomalla, teoreettisesti tyydyttävällä tavalla. Termiä “bayesiläinen” käytetään seuraavassa korostamaan kahta mallintamiselle keskeistä seikkaa. Ensinnäkin verrattuna moniin muihin lähestymistapoihin, bayesiläisen mallintamisen lähtökohtana on konstruoida probabilistinen malli ongelmakentän *yhteistodennäköisyysjakaumalle*, siis malli ongelmakentässä esiintyville tilanteille kokonaisuutena, eikä pelkästään tietylle ongelmakentän osalle (esimerkiksi yhden muuttujan todennäköisyysjakaumalle). Yhteistodennäköisyysjakaumaa voidaan luonnollisesti edelleen soveltaa vain tietyn ongelmakentän osa-alueen jakauman estimointiin. Toiseksi termi viittaa myös siihen, että lähestymistavassa käytetään todennäköisyyksien semanttisessa tulkinnassa modernia, *subjektivistista* näkökantaa, jonka mukaan todennäköisyys on subjektiivinen epävarmuuden mitta, eikä perinteisen, ns. *objektivistisen* lähestymistavan mukainen “toistokoefrekvenssi” (tätä seikkaa käsitellään tarkemmin luvussa 2.2). Todennäköisyyslaskennan käyttämisessä ei tietenkään sinänsä ole mitään subjektiivista, vaan teoria johtaa yhteen tiettyyn ristiriidattomaan l. konsistenttiin tapaan käsitellä epävarmaa tietoa. Subjektivistinen lähestymistapa hyväksyy kuitenkin sen tosiseikan, että mikäli päättelyn lähtökohtana käytetään ihmisten arvioimia todennäköisyyksiä, saattavat eri ihmisten antamat arviot tapahtumien todennäköisyyksistä vaihdella hyvinkin paljon, mikä puolestaan vaikuttaa laskennan lopputulokseen. Yksi käytettävän teoreettisen formalismin suurimmista ansioista on se, että kunkin osaongelman ratkaisemisessa käytettävät perusoletukset on luetteloitava eksplisiittisesti, jolloin eri ratkaisujen järkevyyttä voidaan arvioida paitsi empiirisesti, myös analyyttisesti. Seuraavassa käsitellään kutakin järjestelmän kolmesta komponentista erikseen.

1.2 Mallien rakentaminen

Mallien rakentamista voidaan ajatella yleisesti etsintäprosessina, jossa tavoitteena on löytää ongelmakenttää hyvin kuvaavia numeerisia malleja. Etsintäavaruuden muodostava mallien joukko, *malliperhe*, määritellään yleensä käyttäen ns. parametrisia malliperheitä[1]: yksittäistä mallia voidaan tässä lähestymistavassa ajatella parina (M, θ), missä M määrittelee mallin *struktuurin*, esimerkiksi neuroverkon rakenteen tai sumeiden sääntöjen määrän ja muodon, kun taas θ kiinnittää struktuuriin liittyvät parametrit, esimerkiksi neuroverkon kaarten painot tai sumeiden sääntöjen jäsenyysasteet. *Probabilistisella mallilla* tarkoitetaan seuraavassa mallia joka tuottaa todennäköisyysjakauman mallimaailmassa esiintyville tilanteille.

Käytännössä mallien rakentamisessa joudutaan aina rajoittumaan äärelliseen määrään mallistruktuureja, esimerkiksi joukkoon kolmikerroksisia suunnattuja backpropagation-neuroverkkoarkkitehtuureja, joissa piilosolmujen lukumäärä vaihtelee välillä 1–K. Tässä tapauksessa käytettävissä olevia mallistruktuureja on K kappaletta: $M_1, \ldots, M_K$. Malliperheen voidaan nyt ajatella muodostuvan siitä joukosta malleja (M_k, θ), jotka saadaan aikaan käyttämällä mallistruktuurijoukkoa $\{M_1, \ldots, M_K\}$. Tässä yksittäisen neuroverkon kaarien painoja on merkitty (vektorilla) θ. Mallistruktuuria M_k vastaavaksi *malliluokaksi* kutsutaan sitä malliperheen osajoukkoa, joka sisältää kaikkia muotoa (M_k, θ) olevat mallit. Tyypillisiä esimerkkejä oppivissa ja älykkäissä järjestelmissä käytetyistä mallistruktuureista ovat erilaiset neuroverkkorakenteet, päätöspuut, sumeat sääntöjoukot, ja Bayes-verkot, joita käsitellään tarkemmin luvussa 3.

Bayesiläisessä lähestymistavassa mallien rakentamisprosessissa erotetaan eksplisiittisesti *mallinvalintakriteeri*, jonka perusteella päätetään mitkä mallit ovat sopivia käytettäviksi eri ongelmakentissä, ja *etsintäalgoritmi*, jonka avulla etsintäavaruudesta yritetään löytää mallinvalintakriteerin mielessä hyviä malleja. Mikäli käytetty menetelmä mallien konstruoimiseksi ei tuota toivotun kaltaisia tuloksia, voidaan menetelmän evaluoinnissa tämän erottelun ansiosta keskittyä tarkastelemaan erikseen joko käytettyä valintakriteeriä tai etsintäalgoritmia. Esimerkiksi useissa päätöspuualgoritmeissa ei tämänlaa-

[1] Ei-parametrisia malleja ja esimerkkeihin perustuvaa päättelyä (case-based reasoning) käsitellään luvussa 2.3.

Mallistruktuuri: Rakenne, joka määrää mallien määrittelemiseen tarvittavat osat parametrit). Esimerkiksi neljä syötesolmua, kaksi piilosolmua, ja neljä tulossolmua sisältävä täydellisesti kytketty neuroverkkoarkkitehtuuri muodostaa yhden mallistruktuurin. Yksittäisen mallin määrittelevät parametrit ovat tässä tapauksessa verkon kaarien painot.

Malli: Mallistruktuuri + siihen liittyvien parametrien ilmentymät l. parametrien kiinnitetyt arvot (esimerkiksi 4×2×4–neuroverkkostruktuuri ja siihen liittyvien kaarten painot).

Malliperhe: Joukko mallistruktuureja (esimerkiksi kaikki muotoa $4 \times X \times 4$ olevat neuroverkot, joissa on neljä syötesolmua ja neljä tulossolmua, ja yksi piilokerros). Malliperhettä voidaan ajatella myös kaikkien kiinnitetyllä mallistruktuurijoukolla aikaansaatavien mallien muodostamana joukkona.

Opetusjoukko: Ongelmakentästä saatu otos esimerkkitapauksia.

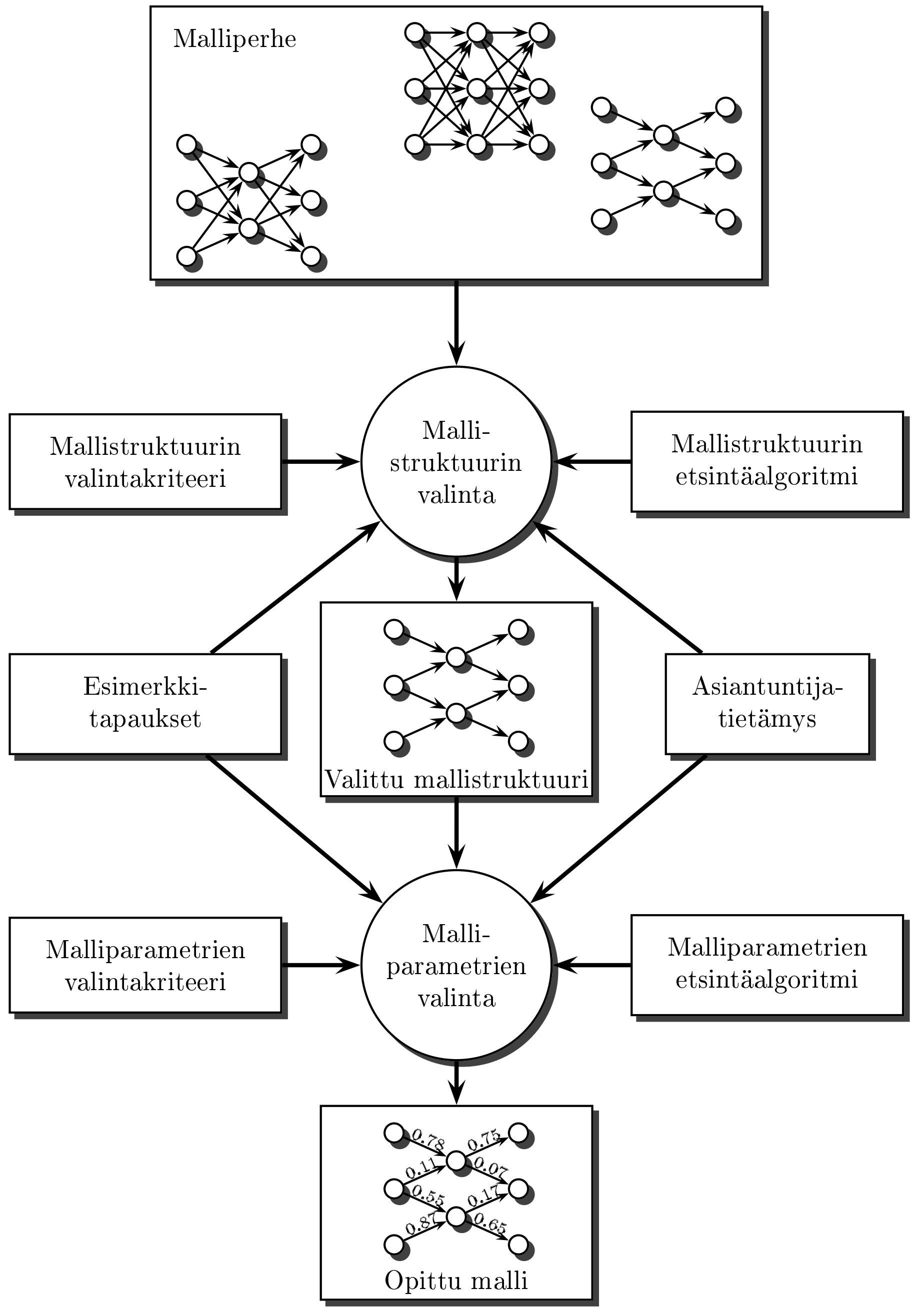
Oppiminen: Mallin (siis mallistruktuurin ja malliparametrien arvojen) valinta kiinnitetystä malliperheestä annettua opetusjoukkoa hyväksikäyttäen.

tuista erottelua ole mahdollista tehdä, vaan etsintä ja valintakriteeri muodostavat yhtenäisen kokonaisuuden, jolloin algoritmien heikkouksien analysointi on vaikeaa.

Kuvassa 1.4 esitetyssä järjestelmässä mallien konstruoiminen tapahtuu käyttäen ongelmankentästä olevaa (asiantuntija)tietämystä, ja/tai saatavilla olevaa ns. *opetusjoukon* (training set) muodostavaa esimerkkiaineistoa. Oletetaan seuraavassa aluksi yksinkertaisuuden vuoksi että käytettävä malli halutaan konstruoida pelkästään opetusjoukon perusteella, jolloin mallien rakennusprosessia kutsutaan mallien *oppimiseksi*, ja käytetään kyseisestä otosjoukosta merkintää $\mathcal{D}$. Asiantuntijatietämyksen yhdistämistä oppimisprosessiin käsitellään luvussa 1.2.4.

Bayesiläisessä mallintamisessa mallien oppiminen voidaan jakaa kahteen erilliseen vaiheeseen: ensimmäisessä vaiheessa määrätään mallistruktuuri M, ja toisessa vaiheessa määrätään kiinnitetyn mallistruktuurin M parametrit θ. Kuvassa 1.5 esitetään tarkennettu kaavio mallien oppimisprosessissta.

Keskeinen ongelma mallien konstruoimisessa on mallien ylisovittamisen välttäminen, jota käsitellään tarkemmin seuraavassa luvussa. Luvuissa 1.2.2



Kuva 1.5: Tarkennettu kuva mallien rakennusvaiheesta.

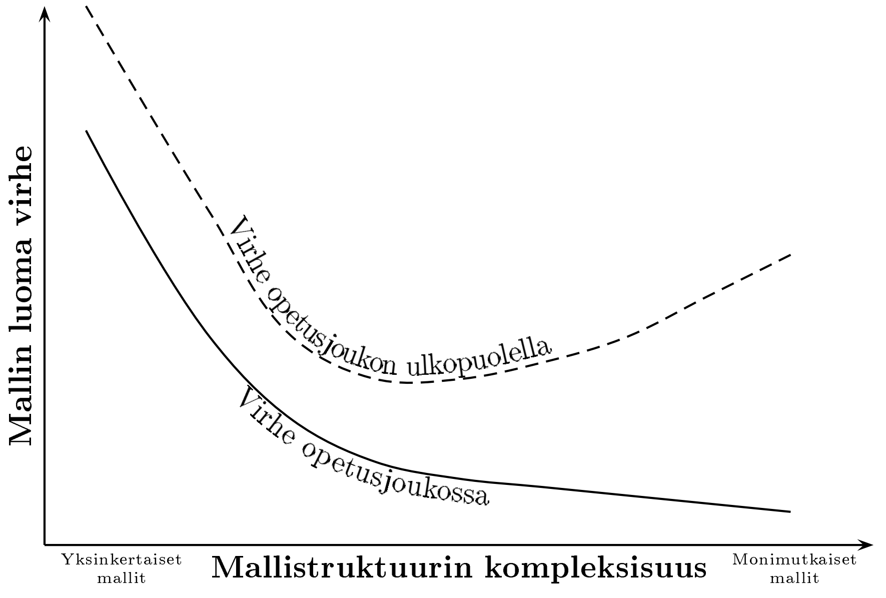
ja 1.2.3 kuvataan bayesiläinen kriteeri mallistruktuurin ja malliparametrien valitsemiseksi. *Tässä yhteydessä on huomattava että bayesiläinen mallintaminen ei ota kantaa oppimisessa käytettävään etsintäalgoritmiin*, joka voi olla mikä tahansa. Yhden vaihtoehdon tällaiseksi algoritmiksi muodostavat *geneettiset algoritmit*, joita käsitellään tämän raportin toisessa osassa.

1.2.1 Mallien ylisovittaminen ja Occamin partaveitsi

Kuten Rissanen "Älykkäiden ja oppivien järjestelmien sovellukset"–teknologiaohjelman evaluointiraportissaan [113] toteaa, valitettavan suuri osa oppiviin ja älykkäisiin järjestelmiin liittyvästä tutkimuksesta on keskittynyt nimenomaan mallien parametrien estimointiongelmaan, vaikka mallin struktuurin valitseminen on käytännön kannalta paljon merkittävämpi ongelma: rakenteellisesti liian monimutkaisissa malleissa on parametrien lukumäärä niin suuri että mallit voidaan *ylisovittaa* (overfit) annettuun opetusjoukkoon. Ylisovittamisen tuloksena syntyvät *ylioppineet* (overtrained) mallit adaptoituvat liian tarkasti opetusjoukkoon, jolloin mallien *yleistyskyky* (generalization capability) eli ennustustarkkuus opetusjoukon ulkopuolisissa tilanteissa on huono. Sanomattakin on selvää, että tällainen järjestelmä, joka toimii hyvin vain tietyissä, ennalta kiinnitetyissä tapauksissa on käytännön sovellusten kannalta hyödytön.

Ylioppiminen: Mallin sovittaminen opetusjoukkoon niin tarkasti, että mallin käyttökelpoisuus on hyvä vain opetusjoukkoon kuuluvissa tapauksissa, ei opetusjoukon ulkopuolella.

Yksi syy ylioppimisilmiöön on se että opetusjoukko on vain suppea ja mahdollisesti virheitä sisältävä otos ongelmakentästä, eikä siten edusta ongelmakentän yhteistodennäköisyysjakaumaa täydellisen tarkasti. Siten opetusjoukkoa vastaavan yhteistodennäköisyysjakauman pikkutarkka estimointi johtaa malliin, joka edustaa huonosti ongelmakentän todellista todennäköisyysjakaumaa. Kuvassa 1.6 on esitetty tyypillinen esimerkki ylioppimisesta: opetusaineistosta malliparametrien sovittamisen tuloksena opitun mallin ennustustarkkuus opetusjoukossa (ennustustarkkuus testattuna käyttäen vain opetusjoukossa annettuja esimerkkitapauksia) on sitä parempi mitä monimutkaisempi käytetty mallistruktuuri on (mitä enemmän mallissa on sovitettavia parametreja), kun taas opitun mallin yleistyskyky eli ennustustarkkuus opetusjoukon ulkopuolisissa tapauksissa alkaa huonontua jossain vaiheessa mallin monimutkaisuuden lisääntyessä. Koska mallien oppimisessa on tavoitteena nimenomaan hyvän yleistyskyvyn saavuttaminen, on ylioppimisen välttäminen koneoppimisen keskeisimpiä ongelmia.



Kuva 1.6: Esimerkki ylioppimisesta.

Ylioppimista voidaan yrittää välttää noudattamalla *Occamin partaveitseksi* kutsuttua periaatetta, jonka mukaan paras mahdollinen malli on sellainen joka sopii ongelmakentästä saatuihin havaintoihin mahdollisimman tarkasti, mutta jonka struktuuri on toisaalta mahdollisimman yksinkertainen. Mallistruktuurin valinnassa on siis löydettävä sopiva tasapaino mallin rakenteen kompleksisuuden ja sen esitysvoimakkuuden välillä: monimutkaiset mallit saadaan sovitettua havaintoihin hyvin tarkasti, mutta niiden struktuurit ovat kompleksisisia, yksinkertaisia malleja ei puolestaan voida useinkaan sovittaa tehtyihin havaintoihin kovin tarkasti. Tilastollisessa oppimisessa tätä tasapainotusongelmaa kutsutaan nimellä *vinouma-varianssi-ongelma (bias-variance dilemma)* (katso esim. [15, 36]).

Vinouma-varianssi-ongelman ratkaisemiseksi on esitetty useita menetelmiä, joista useimmat perustuvat empiirisin argumentein perusteltuihin heuristisiin algoritmeihin (katso esim. [44, 45]). Ongelman ratkaisemiksi on myös esitetty monimutkaisia teoreettisia kehikoita mm. ristiintestauksen (cross-validation) [4] ja VC-ulottuvuuden [137] käsitteisiin perustuen. Esimerkiksi monet neuroverkoissa käytetyt regularisointitekniikat (regularization) [132, 107] pyrkivät ratkaisemaan ylioppisongelman lisäämällä mallinvalintakriteeriin “rankaisutermin”, joka kasvaa neuroverkon rakenteen monimutkaistuessa.

Seuraavassa luvussa näemme kuinka bayesiläinen lähestymistapa sisältää "sisäänrakennetun" *automaattisen Occamin partaveitsi-periaatteen*, minkä ansiosta ylioppimisongelmaa ei tarvitse käsitellä bayesiläisessä lähestymistavassa erikseen, vaan se ratkeaa luonnollisena osana mallien konstruointiprosessia!

1.2.2 Mallistruktuurin valinta

Edellä esitetyssä kaksivaiheisessa oppimisprosessissa tarvitsemme oppimiskriteerin sekä mallistruktuurien että mallien parametrien arvojen evaluoimiseksi. Bayesiläisessä oppimisessa käytettävä kriteeri perustuu todennäköisyyteen: mallistruktuuri M_1 on "parempi" kuin mallistruktuuri M_2 jos se on näistä vaihtoehdoista *todennäköisempi*. Tämä periaate pätee sekä mallistruktuureille että mallien parametreille: mallistruktuurit ja malliparametrit evaluoidaan niiden *posterioritodennäköisyyden* mukaan. Mallistruktuurin tapauksessa sana 'posteriori' viittaa siihen että struktuurin M todennäköisyys $P(M \mid \mathcal{D})$ lasketaan sen jälkeen kun opetusaineisto on nähty, siis *ehdollisena todennäköisyytenä* annettuna opetusjoukko $\mathcal{D}$. Malliparametrien tapauksessa taas käytetään vastaavasti ehdollista jakaumaa $P(\theta \mid \mathcal{D}, M)$.

Todennäköisyyslaskennan perusaksioomia käyttäen on helppo osoittaa että

$$P(M \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid M)P(M)}{P(\mathcal{D})}. \tag{1.1}$$

Tämä kaava tunnetaan *Bayesin teoreemana* jonka esittäjän, Thomas Bayesin (1701–1761), mukaan koko tutkimusalue on saanut nimensä.

Koska todennäköisyyttä $P(\mathcal{D})$ voidaan pitää tässä yhteydessä vakiona, voidaan mallistruktuurien bayesiläisenä oppimiskriteerinä käyttää siis tuloa, jonka tekijöinä ovat oppimisjoukon *kokonaisuskottavuus* eli *evidenssi (marginal likelihood, evidence)* $P(\mathcal{D} \mid M)$, ja mallistruktuurin *prioritodennäköisyys* $P(M)$. Tässä yhteydessä on huomattava että kokonaisuskottavuus eroaa tilastotieteestä tutusta uskottavuustermistä $P_\mathcal{D}$,

$$P_\mathcal{D} = P(\mathcal{D} \mid M, \theta),$$

joka antaa opetusjoukon todennäköisyyden annettuna *täydellinen malli*, siis mallistruktuurin lisäksi myös sitä vastaavien parametrien arvot. Kokonais- eli *marginaaliuskottavuus* saadaan "tavallisesta" uskottavuudesta nimensä mukaisesti *marginalisoimalla* eli *integroimalla* uskottavuustermi yli kaikkien mallistruktuuria M vastaavien mallien, painotettuna parametriasetusten θ prioritodennäköisyydellä:

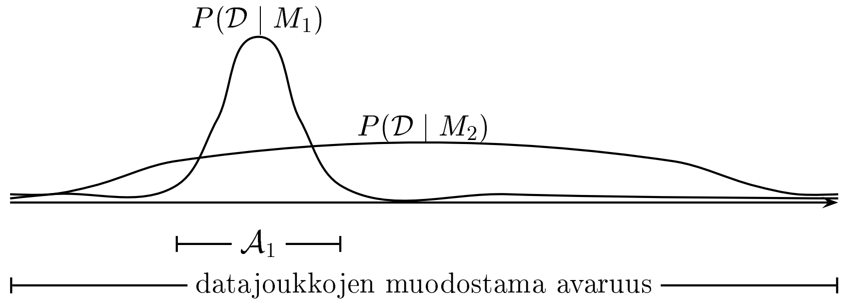
$$P(\mathcal{D} \mid M) = \int P(\mathcal{D} \mid M, \theta) P(\theta \mid M) d\theta. \tag{1.2}$$

Näemme että kokonaisuskottavuutta voidaan pitää normaalin uskottavuustermin odotusarvona niille mallien joukossa joiden struktuuri on M.

Mikäli yllä esiintyvä integraali voidaan laskea, antavat kaavat (1.1) ja (1.2) bayesiläisen kriteerin eri mallistruktuurien vertailemiseksi. Luvussa 3.3 näemme kuinka tämä kriteeri voidaan laskea hyvin tehokkaasti Bayes-verkkojen muodostaman probabilistisen malliperheen tapauksessa. Bayesiläistä lähestymistapaa voidaan soveltaa myös tiettyjen neuroverkkomallien tapauksessa [88, 96, 12, 14, 113], joskaan ei niin suoraviivaisesti kuin probabilististen malliperheiden kanssa.

Yksi tapa soveltaa edellisessä luvussa kuvattua Occamin partaveitsi-periaatetta bayesiläisessä lähestymistavassa saadaan määrittelemällä priorijakauma $P(M)$ siten, että mallistruktuurien prioritodennäköisyys pienenee struktuurin kompleksisuuden kasvaessa. Mikäli mitään muuta syytä — esimerkiksi ongelmakentän asiantuntijan esittämää arvioita mallinnettavan maailman luonteesta — tällaisen priorijakauman käyttämiseen ei kuitenkaan ole, voidaan ei-tasaisen priorijakauman käytöstä luopua, koska *kokonaisuskottavuuskriteeri sisältää jo itsessään automaattisen Occamin partaveitsi-periaatteen!* Tätä aluksi hieman yllättävältä tuntuvaa seikkaa on käsitelty laajemmin mm. teoksissa [110, 88, 96] — asian ymmärtämiseksi intuitiivisella tasolla tarkastelkaamme seuraavaa esimerkkiä.

Olkoon M_1 ja M_2 kaksi mallistruktuuria siten, että M_1 on hyvin yksinkertainen, ja M_2 hyvin monimutkainen mallistruktuuri. Koska siis mallistruktuurin M_2 tapauksessa löytyy aina suuri määrä parametriasetuksia θ, joiden avulla mallistruktuuri voidaan sovittaa tehtyihin havaintoihin $\mathcal{D}$, voimme olettaa että kokonaisuskottavuus $P(\mathcal{D} \mid M_2)$ on kohtalaisen suuri hyvin monelle datajoukolle $\mathcal{D}$. Vastaavasti mallistruktuurin M_1 tapauksessa kokonaisuskottavuus on suuri hyvin rajoitetulle määrälle datajoukkoja; olkoon tämä alue $\mathcal{A}_1$. Koska kokonaisuskottavuus määrää jakauman kaikille mahdollisille datajoukoille $\mathcal{D}$, on selvää että $P(\mathcal{D} \mid M_2)$ ei voi olla suurempi kuin $P(\mathcal{D} \mid M_1)$ kaikilla $\mathcal{D}$. Kuva 1.7 havainnollistaa tätä seikkaa tarkemmin: monimutkaisen mallistruktuurin M_2 tapauksessa on kokonaisuskottavuus levittynyt tasaisesti laajalle, monien datajoukkojen $\mathcal{D}$ muodostamalle alueelle, kun taas mallistruktuurin M_1 tapauksessa kokonaisuskottavuuden merkitsevät arvot saadaan alueella $\mathcal{A}_1$, ja sen ulkopuolella kokonaisuskottavuus on lähellä nollaa. Yksinkertaisempaa mallistruktuuria M_1 vastaava kokonaisuskottavuus on siis suurempi, mikäli käytettävä datajoukko on alueelta $\mathcal{A}_1$!



Kuva 1.7: Intuitiivinen perustelu sille miksi kokonaisuskottavuus noudattaa automaattisesti Occamin partaveitsi-periaatetta: molempien käyrien rajaaman alueen pinta-alan on oltava yhtä suuri (yksi), jolloin yksinkertainen mallistruktuuri M_1 saa jollakin alueella (tässä $\mathcal{A}_1$) suuremman kokonaistodennäköisyyden arvon kuin monimutkaisempi mallistruktuuri M_2.

Occamin partaveitsi-periaatteen toteutumisen näemme myös soveltamalla kokonaisuskottavuuden laskemisessa ns. BIC-approksimaatiota [120, 109, 65], jonka mukaan

$$P(\mathcal{D} \mid M) \approx \frac{P(\mathcal{D} \mid \tilde{\theta}(\mathcal{D}))}{N^{\frac{d(M)}{2}}},$$

missä merkintä $\tilde{\theta}(\mathcal{D})$ tarkoittaa uskottavuuden $P(\mathcal{D}|\theta(\mathcal{D}))$ maksimoivia *suurimman uskottavuuden parametreja (maximum likelihood parameters)*, $\mathrm{d}(M)$ mallistruktuuria M vastaavien malliparametrien lukumäärää, ja N datajoukon $\mathcal{D}$ kokoa. Näemme siis että kokonaisuskottavuus tasapainoilee Occamin partaveitsi-periaatteen mukaisesti mallin esityskyvyn ja kompleksisuuden välillä: mallin struktuurin kompleksisuuden kasvaessa osoittajassa esiintyvä uskottavuustermi kasvaa, mutta samanaikaisesti kasvaa myös nimittäjässä esiintyvä mallin kompleksisuudesta rankaiseva termi.

Jos esitämme BIC-approksimaation hieman toisessa muodossa (ottamalla logaritmit kyseisistä termeistä), saamme

$$-\log P(\mathcal{D} \mid M) \approx -\log P(\mathcal{D} \mid \tilde{\theta}(\mathcal{D})) + \frac{d(M)}{2} \log N,$$

mistä näemme että approksimaatio on sama kuin Rissasen teoksessa [109] esittämä *Minimum Description Length (MDL)*-periaatteeseen perustuva kaksiosainen kriteeri mallistruktuurin valitsemiseksi. Bayesiläisen ja informaatioteoreettisten lähestymistapojen yhteyksiä käsitellään myöhemmin tarkemmin luvussa 2.4.

1.2.3 Malliparametrien valinta

Kun kriteerin (1.1) mielessä optimaalinen mallistruktuuri M on löydetty, on seuraava tehtävä oppimisprosessissa määrätä mallistruktuuriin liittyvät parametrit θ. Monissa ei-probabilistisissa malliperheissä, kuten esimerkiksi neuroverkkomalleissa, käytetty mallinvalintakriteeri on tässä tapauksessa yleensä estimointivirhe opetusjoukossa. Kuten edellä nähtiin, tämä johtaa helposti ylioppimiseen mikäli mallistruktuuria ei ole valittu oikein. Käytetyt etsintäalgoritmit (esimerkiksi backpropagation) ovat näissä tapauksissa yleensä yksinkertaisia iteratiivisia gradienttimenetelmän variaatioita joiden konvergenssi saattaa olla hyvin hidasta. Lisäksi gradienttimenetelmät ovat lokaaleja menetelmiä jotka parhaassakin tapauksessa konvergoivat vain lokaaliin optimiin, eivätkä välttämättä löydä niitä parametriarvoja jotka maksimoivat käytetyn parametrien evaluointikriteerin.

Bayesiläisessä mallintamisessa sitä vastoin parametrien arvojen valinnassa käytettävä oppimiskriteeri on sama kuin mallistruktuurien valinnassa: parametriarvot θ evaluoidaan niiden posterioritodennäköisyyden $P(\theta \mid M, \mathcal{D})$ mukaan. Todennäköisyyslaskennan perusaksioomista saamme että

$$P(\theta \mid M, \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta, M) P(\theta \mid M)}{P(\mathcal{D} \mid M)}, \tag{1.3}$$

Yksittäisen mallin θ posterioritodennäköisyys saadaan siis kertomalla opetusjoukon uskottavuus $P(\mathcal{D} \mid \theta, M)$ mallin prioritodennäköisyydellä $P(\theta \mid M)$, ja normalisoimalla syntyvä tulo kokonaisuskottavuudella (1.2), jonka arvo on jo määrätty mallistruktuurin valinnassa. Tietyissä probabilistissa malliperheissä posteriorijakauman maksimikohta voidaan laskea suljetussa muodossa yksinkertaisella matemaattisella kaavalla, jolloin *mallin parametrien säätämiseen ei tarvita lainkaan iteratiivista aikaavievää prosessia*! Esimerkin tällaisesta malliperheestä muodostavat tässä raportissa käsiteltävät Bayes-verkot, joita tarkastellaan tarkemmin luvussa 3. Luvussa 3.3.2 esitetään yksinkertainen laskukaava, jonka avulla annetun Bayes-verkon parametrit voidaan määrätä suoraan ilman iteratiivista oppmisprosessia.

1.2.4 Oppimisen ja asiantuntijatietämyksen yhdistäminen

Edellä oletettiin mallistruktuurin valinnassa käytettävän pelkästään tilastollista havaintoaineistoa $\mathcal{D}$. Usein saatavilla on kuitenkin esimerkkitapauksien lisäksi myös runsaasti "ihmistietämystä": ongelmakentän asiantuntijoiden kokemusperäistä tietoa, tai "maalaisjärkeen" (common sense) perustuvaa tietoa ongelmakentän luonteesta. Monien malliperheiden, kuten esimerkiksi neuroverkkojen, tapauksessa merkittäväksi ongelmaksi on muodostunut se, kuinka nämä kaksi tietolähdettä saadaan yhdistettyä. Bayesiläisessä mallintamisessa yhdistäminen on suoraviivaista: kaavoissa (1.1) ja (1.2) esiintyy kaksi prioritodennäköisyysjakaumaa, $P(M)$ ja $P(\theta \mid M)$. Sana 'priori' viittaa tässä yhteydessä siihen että nämä jakaumat on määrättävä "ennen" opetusjoukon $\mathcal{D}$ tarkastelua siinä mielessä että ne eivät saa riippua opetusjoukon muodostavasta tilastollisesta aineistosta. Bayesiläinen formalismi antaa nyt mahdollisuuden koodata ongelmakentästä saatavilla olevaa yleistä tietämystä prioritodennäköisyyksien avulla.

Luvussa 3.3 näemme että Bayes-verkkojen tapauksessa mallistruktuurien prioritodennäköisyyksien arvioiminen on erityisen luontevaa, koska kukin Bayes-verkkostruktuuri edustaa joukkoa muuttujien välisiä riippuvuuksia, ja tällaisten riippuvuuksien arviointi on ongelmakentän asiantuntijoille monessa tapauksessa sangen helppoa. Lisäksi priorijakauma $P(\theta \mid M)$ voidaan formalisoida Bayes-verkkojen tapauksessa tavalla, joka antaa asiantuntijoille mahdollisuuden paitsi koodata prioritietämyksensä, myös arvioida prioritiedon merkityksellisyyttä suhteessa saatavilla olevaan tilastolliseen aineistoon. Toisaalta on myös korostettava, että koska Bayes-verkkojen tapauksessa on sekä mallistruktuurilla että siihen liittyvillä parametreilla selvä semanttinen merkitys, voidaan Bayes-verkkoja konstruoida aivan hyvin myös ilman tilastollista aineistoa, käyttäen puhtaasti asiantuntijoilta saatua tietämystä.

1.3 Mallien soveltaminen

Kuten edellä näimme, bayesiläisessä oppimisessa muodostetaan ongelmakentästä malli (M, θ) käyttäen apuna opetusesimerkkejä $\mathcal{D}$, ja/tai ongelmakentän asiantuntijoiden prioritietämystä. Oletetaan jatkossa että ongelmakentän kuvauksessa käytetään n attribuuttia (satunnaismuuttujaa) $X_1, \ldots, X_n$. Merkinnällä $X_i = x_i$ tarkoitetaan sitä että attribuutin X_i arvo on x_i, missä x_i voi olla joko reaalilukuku, jolloin kyseessä on *jatkuva-arvoinen* muuttuja, tai sitten x_i kuuluu johonkin äärelliseen joukkoon mahdollisia arvoja, missä tapauksessa attribuuttia X_i kutsutaan *diskreetiksi*. Jatkossa keskitymme yksinkertaisuuden vuoksi hyvin paljon diskreetteihin muuttujiin — jatkuvien muuttujien käsittelystä puhutaan enemmän luvussa 3.4.

Seuraavassa oletamme että käytetty malliperhe on probabilistinen, missä tapauksessa käytettävä malli (M, θ) määrittelee yhteistodennäköisyysjakauman $P(X_1 = x_1, \ldots, X_n = x_n \mid M, \theta)$ muuttujien $X_1, \ldots, X_n$ eri arvokombinaatioille. Tätä yhteistodennäköisyysjakaumaa käyttäen on mahdollista rat-

Muuttujajoukko $\mathbf{S}_1$:
: Ne muuttujat joiden arvo ei ole tiedossa, mutta joiden arvojen tuntemisen katsotaan olevan edellytyksenä käsillä olevan ongelmanratkaisutilanteen ratkaisemiksi.

Muuttujajoukko $\mathbf{S}_2$:
: Muuttujat joiden arvo on tunnettu.

Muuttujajoukko $\mathbf{S}_3$:
: Muuttujat joiden arvo ei ole tiedossa, eikä arvojen tuntemisen katsota olevan välttämätön käsillä olevan ongelmanratkaisutilanteen kannalta.

Probabilistinen päättely:
: Todennäköisyysjakauman $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ estimointi.

kaista erilaisia mallien soveltamisessa esiintyviä estimointitehtäviä. Tätä havainnollistaaksemme olettakaamme että attribuutit on jaettu kolmeen erilliseen osajoukkoon siten että ensimmäisen osajoukon $\mathbf{S}_1$ muodostavat ne attribuutit joiden arvoa halutaan estimoida, toisen osajoukon $\mathbf{S}_2$ ne attribuutit joiden arvo on annettu, ja kolmannen osajoukon $\mathbf{S}_3$ muut attribuutit. *Probabilistisessa päättelyssä* tavoitteena on laskea ehdollinen todennäköisyysjakauma kiinnostuksen kohteena oleville attribuuteille, annettuna tunnettujen attribuuttien arvot. Tuloksena on siis jakauma $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$, missä merkintä $\mathbf{S}_2 = \mathbf{s}_2$ tarkoittaa että kukin attribuutti joukossa $\mathbf{S}_2$ on asetettu tunnettuun arvoonsa.

Tässä yhteydessä on korostettava että attribuuttien jako yllämainittuihin kolmeen osajoukkoon ei ole millään tavalla kiinteä, vaan voi vaihdella tilanteen mukaan. Niinpä probabilistisessa päättelyssä ei tarvitse etukäteen päättää mitä halutaan estimoida tai mitä tietoa estimointitehtävässä on saatavilla. Tämä poikkeaa monista vaihtoehtoisista lähestymistavoista, kuten esimerkiksi päätöspuista ja monissa neuroverkkomalleista, joissa estimointitehtävä (esimerkiksi luokittelutehtävä) on päätettävä etukäteen jo ennen mallin rakentamista. Probabilistisessa päättelyssä sitä vastoin riittää, jos yhteistodennäköisyydet $P(X_1, \ldots, X_n \mid M, \theta)$ on annettu: todennäköisyyslaskennan perusaksioomista seuraa että ehdolliset todennäköisyydet voidaan esittää muodossa

$$
\begin{aligned}
P(\mathbf{S}_1 = \mathbf{s}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta) &= \frac{P(\mathbf{S}_1 = \mathbf{s}_1, \mathbf{S}_2 = \mathbf{s}_2 \mid M, \theta)}{P(\mathbf{S}_2 = \mathbf{s}_2 \mid M, \theta)} \\
&= \frac{\sum_{\mathbf{S}_3} P(\mathbf{S}_1 = \mathbf{s}_1, \mathbf{S}_2 = \mathbf{s}_2, \mathbf{S}_3 \mid M, \theta)}{\sum_{\mathbf{S}_1, \mathbf{S}_3} P(\mathbf{S}_1, \mathbf{S}_2 = \mathbf{s}_2, \mathbf{S}_3 \mid M, \theta)}, \qquad (1.4)
\end{aligned}
$$

missä merkintä $\sum_{\mathbf{S}_i}$ tarkoittaa että summassa käydään läpi kaikki joukossa $\mathbf{S}_i$ esiintyvien muuttujien arvokombinaatiot. Mikä tahansa ehdollinen todennäköisyys saadaan siis laskettua summaamalla eli *marginalisoimalla* yli tietyn yhteistodennäköisyyksien joukon. Jotta tätä lähestymistapaa olisi mahdollista soveltaa käytännössä, on ratkastava seuraavat kaksi ongelmaa:

1. Kuinka tallettaa yhteistodennäköisyydet $P(X_1, \ldots, X_n \mid M, \theta)$ niin että niiden talletus ei vaadi eksponentiaalista muistitilaa, ja että ne ovat tehokkaasti käytettävissä?

2. Kuinka laskea kaavassa (1.4) esiintyviä marginaalisummia tehokkaasti?

Luvussa 3 osoitamme kuinka yllä esitetyt ongelmat voidaan ratkaista kun yhteistodennäköisyysjakaumat määritellään käyttäen Bayes-verkkojen malliperhettä: luvussa 3.1 käsitellään yhteistodennäköisyyksien tallettamisen ongelmaa, ja luvussa 3.2 marginalisointiongelmaa Bayes-verkkojen tapauksessa.

Probabilistisessa päättelyssä estimoidaan siis ehdollista jakaumaa $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$. Seuraavassa luvussa käsitellään sitä miten tätä jakaumaa voidaan käyttää hyväksi päätöksenteossa arvioimalla eri toimenpiteistä seuraavien hyötyjen ja haittojen odotusarvoja. Ennen sitä kuitenkin esittelemme joitakin tyypillisiä probabilistisen päättelyn muotoja.

Yksinkertaisimmassa tapauksessa joukko $\mathbf{S}_1$ käsittää vain yhden muuttujan X, ja tehtävänä on määrätä muuttujan X arvojen jakauma lähtötietojen $\mathbf{S}_2 = \mathbf{s}_2$ pohjalta. Tyypillisen esimerkin tällaisesta ongelmasta muodostavat *luokitteluongelmat*, joissa tehtävänä on määrittää diskreetin luokkamuuttujan X arvojen todennäköisyydet (katso esimerkki 1).

Esimerkissä 1 käytettyjen diskreettien muuttujien jakauma voidaan määrätä laskemalla todennäköisyys kullekin muuttujan mahdolliselle arvolle. Mikäli X on jatkuva-arvoinen, jakauma voidaan tietyin jakaumaoletuksin kuvata antamalla jakauman tunnusluvut, esimerkiksi odotusarvo ja varianssi (katso esimerkki 2).

Esimerkissä 1 esitetyissä diskreeteissä luokitteluongelmissa probabilistista päättelyä käytettiin diagnostiseen päättelyyn, eli syiden (esim. sairaus) todennäköisyyksien arviointiin, annettuna havaitut seuraukset (oireet). Toisaalta, kuten edellä mainittiin, muuttujien jakoa joukkohin $\mathbf{S}_1$, $\mathbf{S}_2$ ja $\mathbf{S}_3$ voi-

Esimerkki 1 Yhden diskreetin muuttujan jakauman estimointi (luokitteluongelma).

Ongelma1:	Sairastaako potilas tautia X havaittujen oireiden ja tehtyjen kokeiden ($\mathbf{s}_2$) perusteella?
Ongelma2:	Mikä on asiakkaan luottokelpoisuusluokka (X) havaitun osittaisen asiakasprofiilin ($\mathbf{s}_2$) perusteella?
Ratkaisu1:	Laske $P(X = 1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$.
Ratkaisu2:	Laske $P(X = x \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ kaikille luottokelpoisuusluokille x.

Esimerkki 2 Jatkuvan muuttujan jakauman estimointi.

Ongelma1:	Mikä on tuotteen valmistuskustannusten (X) oletettava arvo, jos täytämme asiakkaan tuotteelle asettamat vaatimukset ($\mathbf{s}_2$)?
Ongelma2:	Mikä on ulostuloventtiilin X virtauksen oletettava voimakkuus jos säätimet $\mathbf{S}_2$ asetetaan arvoihin $\mathbf{s}_2$?
Ratkaisu:	Laske jakauman $P(X \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ odotusarvo.

daan muuttaa dynaamisesti mielivaltaisella tavalla. Niinpä siirtämällä muuttujia joukosta toiseen voimme soveltaa probabilistista päättelyä myös päinvastaiseen suuntaan, seurausten ennustamiseen syiden pohjalta. Tämänkaltaista päättelyä voidaan käyttää esimerkiksi ongelmakentän luonteen analysointiin (data mining) esimerkeissä 3 esitetyillä tavoilla.

Esimerkki 3 Ongelmakentän analysointi.

Ongelma1:	Kuinka suuri osa tautia X sairastavista potilaista kärsii äkillisistä tajuttomuuskohtauksista ($Y = 1$)?
Ongelma2:	Mikä on parhaaseen luottokelpoisuusluokkaan (AAA) kuuluvien asiakkaiden yleisin tilimuoto (Y)?
Ratkaisu1:	Laske $P(Y = 1 \mid X = 1, M, \theta)$.
Ratkaisu2:	Laske $P(Y = y \mid X = AAA, M, \theta)$ kaikille tilimuodoille y.

Esitettyjä kahta päättelyn muotoa (seurauksien arviointi syistä, tai syiden päättely seurauksista) voidaan tietenkin edelleen sekoittaa mielivaltaisella tavalla ratkaisemaan mm. esimerkissä 4 esitettyjen tapausten kaltaisia ongelmia.

Mikäli joukko $\mathbf{S}_1$ sisältää useita muuttujia, tulee jakauman $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ täydellisestä määräämisestä vaikeaa, koska se vaatii diskreetissäkin tapauksessa kaikkien joukossa $\mathbf{S}_1$ olevien muuttujien arvokombinaatioiden lä-

Esimerkki 4 Eri päättelytapojen yhdistäminen.

Ongelma1:	Onko oletettavaa että potilas voi kärsiä tajuttomuuskohtauksista ($Y = 1$), jos hänellä on tauti X, ja hänen ruumiinlämpötilansa on z?
Ongelma2:	Onko tehtyjen havaintojen ($\mathbf{s}_2$) perusteella oletettavaa että potilas sairastaa tautia X, ja että hän voi saada vakavia tajuttomuuskohtauksia?
Ratkaisu1:	Laske $P(Y = 1 \mid X = 1, Z = z, M, \theta)$.
Ratkaisu2:	Laske $P(X = 1, Y = 1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$.

pikäynnin, ja näitä arvokombinaatioita on tietenkin eksponentiaalinen määrä. Niinpä tällaisessa tilanteessa tyydytäänkin usein etsimään se muuttujien $\mathbf{S}_1$ arvokombinaatio $\mathbf{s}_1$ (tai n tässä mielessä parasta arvokombinaatiota) jolla on korkein todennäköisyys, ts. arvokombinaation joka maksimoi todennäköisyyden $P(\mathbf{S}_1 = \mathbf{s}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$. Koska

$$\max_{\mathbf{s}_1} P(\mathbf{S}_1 = \mathbf{s}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta) = \max_{\mathbf{s}_1} P(\mathbf{S}_1 = \mathbf{s}_1, \mathbf{S}_2 = \mathbf{s}_2 \mid M, \theta), \qquad (1.5)$$

näemme että tässä mielessä optimaalinen arvokombinaatio täydentää vapaille muuttujille $\mathbf{S}_1$ arvot siten että syntyvän kokonaisarvokombinaation todennäköisyys on maksimaalinen. Tyypillisiä esimerkkejä tämänkaltaisen probabilistisen päättelyn sovellusalueista ovat konfiguraatio-ongelmat, joissa täydennetään osakonfiguraatio siten että täydennetyn konfiguraation todennäköisyys on mahdollisimman suuri annetun mallin muodostamassa todennäköisyysjakaumassa. Jos malli (M, θ) muodostetaan esimerkiksi jotakin tuotetta valmistavan yrityksen valmistettujen tuotteiden tietokannasta, voidaan tällä tavalla täydentää asiakkaan tuotteelle esittämät toivomukset siten että uusi tuote on mahdollisimman samankaltainen jo valmistettujen tuotteiden kanssa, ja siten otaksuttavasti edullinen valmistaa (katso esimerkki 5).

Esimerkki 5 Optimaalisen konfiguraation etsiminen.

Ongelma1:	Täydennä valmistettavan tuotteen kuvaus ($\mathbf{S}_1$) siten että tuote täyttää asiakkaan sille esittämät vaatimukset ($\mathbf{s}_2$), ja sopii yrityksen tuoteprofiiliin (eli siitä tehtyyn malliin) mahdollisimman hyvin.
Ongelma2:	Konstruoi asiakkaan täydellinen asiakasprofiili ($\mathbf{S}_1$) annettujen tietojen $\mathbf{s}_2$ pohjalta.
Ratkaisu:	Etsi $\mathbf{s}_1$ siten että todennäköisyyden $P(\mathbf{S}_1 = \mathbf{s}_1, \mathbf{S}_2 = \mathbf{s}_2 \mid M, \theta)$ arvo on maksimaalinen.

Ehdollisia todennäköisyyksiä voidaan käyttää myös etäisyysmittana, jonka avulla voidaan etsiä esimerkiksi valmistettujen tuotteiden joukosta se tapaus joka muistuttaa eniten asiakkaan antamaa osittaista tuotteen kuvausta. Tällaista probabilistista etäisyysmetriikkaa käsitellään lähteessä [70].

Konfiguraatio-ongelmia läheisesti muistuttavan sovellusalueen muodostavat asiakasprofilointitehtävät (katso esimerkki 5). Tavoitteena on täydentää asiakkaan profiili asiakkaasta kerättyjen tietojen ja kaikkien asiakkaiden muodostamasta tietokannasta muodostetun mallin perusteella. Syntyvää täydennettyä asiakasprofiilia voidaan käyttää hyväksi mm. suoramarkkinoinnin suuntaamisessa: probabilistista päättelyä voidaan käyttää esimerkiksi ennustamaan olisiko asiakas kiinnostunut tietyntyyppisestä luottokortista annettuna hänestä kerätyt tiedot.

Koska probabilistista päättelyä käyttäen voidaan muodostaa täydellinen mallimaailman tilanteen kuvaus, voidaan yhteistodennäköisyyden maksimoivia konfiguraatioita käyttää myös probabilistisen päättelyn selitysmekanismina: muodostetusta täydennetystä konfiguraatiosta voidaan esimerkiksi päätellä mitä muita tauteja tai oireita jo diagnosoidulla potilaalla mahdollisesti on (katso esimerkki 6). Ennustettujen lisäoireiden havaitseminen lisää luonnollisesti jo tehdyn diagnoosin luotettavuutta.

Esimerkki 6 Päättelyä tukevat selitykset.

Ongelma:	Oletetaan että potilas sairastaa tautia X ja että hänellä on oireet $\mathbf{S}_2 = \mathbf{s}_2$. Mitä muita oireita potilaalla luultavasti on? Sairastaako potilas mahdollisesti myös jotain muuta tautia?
Ratkaisu:	Etsi $\mathbf{s}_3$ siten että todennäköisyyden $P(X = 1, \mathbf{S}_2 = \mathbf{s}_2, \mathbf{S}_3 = \mathbf{s}_3 \mid M, \theta)$ arvo on maksimaalinen.

Kuten edellä esitetyistä esimerkeistä huomaamme, puuttuva tieto käsitellään probabilistisessa päättelyssä erityisen elegantilla tavalla: jos esimerkiksi alkuperäinen suunnitelmamme oli estimoida jakaumaa $P(X_1 \mid X_2 = x_2, \ldots, X_5 = x_5)$, mutta muuttujan X_3 arvoa ei olekaan saatavilla, käytämme päättelyssä yksinkertaisesti jakaumaa $P(X_1 \mid X_2 = x_2, X_4 = x_4, X_5 = x_5)$ — kullakin hetkellä käytetään siis jakaumaa joka saadaan ottamalla huomioon kaikki annettu tieto. Monien muiden malliperheiden tapauksessa on puuttuvan tiedon käsittely huomattavan hankalaa: esimerkiksi päätöspuiden toimivuuden perusedellytyksenä on se, että kussakin päätössolmussa käytettävän muuttujan arvo on aina mahdollista saada selville.

Probabilistista päättelyä voidaan käyttää myös puuttuvan tiedon estimointiin: sen sijaan että estimoisimme jakaumaa $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$, asetamme estimoitavat muuttujat johonkin arvokombinaatioon $\mathbf{s}_1$, ja kysymme kuinka muuttujat $\mathbf{S}_3$ on asetettava kun halutaan maksimoida halutun arvokombinaation todennäköisyys $P(\mathbf{S}_1 = \mathbf{s}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, \mathbf{S}_3, M, \theta)$. Esimerkissä 7 esitetyt tapaukset kuvaavat tämäntyyppisiä sovellusalueita.

Esimerkki 7 Puuttuvan tiedon estimointi.

Ongelma1:	Mitä lääkettä ($\mathbf{S}_3$) potilaalle on annettava jotta toipumisen todennäköisyys olisi maksimaalinen?
Ongelma2:	Kuinka asettaa säätimet $\mathbf{S}_3$ siten että läpivirtaus putkessa X olisi optimaalinen tilanteessa jossa säätimet $\mathbf{S}_2$ halutaan pitää nykyisissä asetuksissa $\mathbf{s}_2$?
Ongelma3:	Muodosta valmistettavan tuotteen kuvaus ($\mathbf{S}_3$) siten että se on konsistentti asiakkaan vaatimusten ($\mathbf{s}_2$) kanssa, ja voitto on maksimaalinen.
Ratkaisut1&2:	Etsi $\mathbf{s}_3$ siten että todennäköisyyden $P(X = 1 \mid \mathbf{S}_2 = \mathbf{s}_2, \mathbf{S}_3 = \mathbf{s}_3, M, \theta)$ arvo on maksimaalinen.
Ratkaisu3:	Etsi $\mathbf{s}_3$ siten että jakauman $P(X \mid \mathbf{S}_2 = \mathbf{s}_2, \mathbf{S}_3 = \mathbf{s}_3 \mid M, \theta)$ odotusarvo on maksimaalinen.

Tässä yhteydessä on huomautettava että esimerkissä 7 esiintyvä päättelyn muoto, puuttuvan tiedon estimointi, on monissa tilanteissa laskennallisesti raskaampi tehtävä kuin jakauman $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ estimointi, eli päättely puuttuvan tiedon vallitessa.

Paitsi probabilistisessa päättelyssä esiintyvissä syötearvoissa, puuttuvaa tietoa voi esiintyä myös opetusaineistossa $\mathcal{D}$. Lähteessä [72] tarkastellaan bayesiläisen lähestymistavan käyttämistä tämän ongelman ratkaisemisessa.

1.4 Riskianalyysi

Edellisessä luvussa kuvattiin kuinka probabilistisessa päättelyssä on tavoitteena muodostaa ehdollinen todennäköisyysjakauma estimoitaville muuttujille, annettuna tunnettujen muuttujien arvot. Oletetaan seuraavassa yksinkertaisuuden vuoksi että estimoimme jakaumaa $P(X \mid \mathbf{S}_2 = \mathbf{s}_2, \theta, M)$, missä X on diskreetti muuttuja jonka arvo meidän on kiinnitettävä annettujen lähtötietojen $\mathbf{s}_2$ pohjalta. Intuitiivisesti luonnollisimmalta tuntuvin vaihtoehto on tietenkin valita se muuttujan X arvo x joka maksimoi todennäköisyyden $P(X = x \mid \mathbf{S}_2 = \mathbf{s}_2, \theta, M)$. Käytännön sovelluksissa saatetaan kuitenkin tehdä muuttujan arvon valinnan perusteella päätöksiä, joiden seuraukset voivat olla hyvin kauskantoisia, ja siksi päätöksenteon huolellinen analyysi on paikallaan. Todennäköisyyslaskenta tarjoaa *päätösteoriana (decision theory)* tunnetun teoreettisen kehikon eri päätöstilanteissa esiintyvien ongelmien ratkaisemiseksi. Päätösteorian mukaan meidän on valittava se vaihtoehto, joka minimoi odotettavissa olevan haitan (maksimoi odotettavissa olevan hyödyn). Esimerkki 8 havainnollistaa tätä periaatetta.

Esimerkki 8 Päätösteoreettinen riskianalyysi.

Työpaikkalääkäri saa eteensä työntekijän, jolle tehdyt alustavat testit osoittavat henkilön potevan erästä tautia 60% todennäköisyydellä. Tauti on sinänsä vaaraton, mutta jos se pääse hoitamattomana taudin myöhempään vaiheeseen, joutuu henkilö kymmenen päivän sairaslomalle. Tauti voidaan diagnosoida 100% luotettavuudella erään nopean, mutta melko kalliin lisätestin avulla. Testin hinta vastaa työntekijän yhden päivän työpanosta. Jos henkilöllä tämän testin avulla havaitaan kyseinen tauti ennen kuin se kehittyy myöhempään vaiheeseensa, voidaan tarvittavien sairaspäivien määrä vähentää puoleen.

Lääkäri pohtii tulisiko hänen määrätä työntekijälle kallis lisätesti, vai luokitella henkilö terveeksi ja lähettää hänet takaisin töihin. Eri vaihtoehdoissa kustannukset muodostuvat siis seuraavasti, kun yksikkönä käytetään menetettyjä työpäiviä:

Aiheutuvat kustannukset:	Potilas on sairas	Potilas on terve
Tehdään lisätesti	6	1
Ei tehdä lisätestiä	10	0

Jos potilaalle tehdään lisätesti, on aiheutuvien kustannusten odotusarvo $0.6 \cdot 6 + 0.4 \cdot 1 = 4$, kun taas toisessa tapauksessa aiheutuvien kustannusten odotusarvo on $0.6 \cdot 10 + 0.4 \cdot 0 = 6$. Päätösteorian mukaan eri toimintavaihtoehdoista on valittava se, josta seuraava odotettavissa oleva kustannus on minimaalinen (tai vastaavasti odotettavissa oleva hyöty maksimaalinen), joten lääkärin tulisi määrätä potilas lisätesteihin. Jos lääkärille tulee 1000 samanlaista tapausta vuodessa, toimimalla päätösteorian mukaisesti hän säästää yhtiölleen vuosittain (keskimäärin) 2000 työpäivää vastaavan määrän rahaa.

Jos ehdolliset todennäköisyydet $P(X = x \mid \mathbf{S}_2 = \mathbf{s}_2, \theta, M)$ on arvioitu oikein, bayesiläisen päätösteorian mukaan toimivan agentin toiminta on optimaalista, tulkittuna siten, että tietyllä ajanjaksolla saavutettavan hyödyn odotusarvo on tätä politiikkaa käyttäen maksimaalinen. Vaikka päätöksissä käytettävä todennäköisyysjakauma oliskin arvioitu väärin, bayesiläiseen päätösteoriaan perustuvan toiminnan voidaan osoittaa silti olevan ristiriidattomasti rationaalista siinä mielessä, että ei ole mahdollista konstruoida sellaista päätöksentekotilanteiden jonoa, jonka tuloksena bayesiläinen päätöksentekijä häviäisi varmasti (ns. "Dutch book"-argumentti, katso esim. [10, 9, 32]). Lisäksi voidaan osoittaa että bayesiläinen päätösteoria on *ainoa* tässä mielessä rationaalinen päätöksentekopolitiikka [102].

Päätösteoriaa on kritisoitu siitä että eri tilanteista seuraavien haittojen arviointi on usein kohtuuttoman vaikeaa tai jopa mahdotonta. Tämä on usein totta: esimerkissä 8 haittojen arvottaminen oli yksinkertaista, koska niitä arvioitiin pelkästään yhtiölle muodostuvien kustannusten kautta, mutta jos mukaan olisi otettu työntekijälle koituvien kärsimysten arviointi, tilanne olisi ollut paljon monimutkaisempi. Päätösteoriassa toteutuu kuitenkin bayesiläisen mallintamisen pääperiaate, jonka mukaan meidän on kirjattava täsmällisesti kaikki ongelmanratkaisussa käyttämämme oletukset. Jos käyttämämme järjestelmä ei toimi toivotulla tavalla, tiedämme että syy on joko siinä, että olemme arvioineet haittavaikutukset väärin, tai siinä että emme ole pystyneet estimoimaan käytettyjä ehdollisia todennäköisyyksiä riittävän tarkasti. Päätösteoria antaa meille kuitenkin teoreettisesti optimaalisen päämäärän, jota voimme approksimoida parhaan kykymme mukaan, ja jos järjestelmämme ei toimi halutulla tavalla, voimme yrittää analysoida mikä tekemistämme approksimaatioista ei pidä paikkaansa.

Tässä yhteydessä on korostettava että ihmisen rooli bayesiläisessä mallintamisessa on ensiarvoisen tärkeä: valittavissa olevista vaihtoehdoista seuraavien haittojen (tai hyötyjen) arviointi on tehtävä joka jää viime kädessä ihmisen tehtäväksi. Päätösteoria voi antaa toimintasuosituksen annettujen haittavaikutusten kannalta, mutta ihmisen tehtäväksi jää arvioida tehtyjen oletusten paikkansapitävyys ja siten toimintasuositusten järkevyys.Tästä syystä epätäsmällistä päättelyä suorittavia järjestelmiä onkin ryhdytty kutsumaan yhä useammin *päätöksentukijärjestelmiksi (decision support systems)* entisen termin *asiantuntijajärjestelmä (expert system)* sijaan.

Luku 2

Bayesiläisen lähestymistavan arviointia

Bayesiläisen mallintamisen edut: ristiriidaton päättely, ylioppimisen välttäminen, tilastollisen aineiston ja asiantuntijatietämyksen yhdistäminen, puuttuvan tiedon luonteva käsittely, teoreettinen kehikko hybridimalleille. Vastauksia bayesiläisen lähestymistavan kritiikkiin. Vertailua esimerkkeihin perustuvaan päättelyyn ja ei-parametrisiin malleihin. Yhteys informaatioteoreettisiin lähestymistapoihin: MDL, MML ja stokastinen kompleksisuus.

Luvussa 1 esitettyä bayesiläistä mallinnuskehikkoa kohtaan voidaan esittää kritiikkiä kohdistamalla se joko parametriseen mallintamislähestymistapaan sinänsä, tai nimenomaan bayesiläiseen mallintamiseen. Luvussa 2.3 käsittelemme lyhyesti esimerkkeihin perustuvaa päättelyä ja ei-parametrisia malliperheitä, jotka eivät kuitenkaan osoittaudu kovin erilaisiksi lähestymistavoiksi parametriseen mallintamiseen verrattuna. Sitä ennen kertaamme luvussa 2.1 tärkeimmät bayesiläisen lähestymistavan tarjoamat edut, ja arvioimme baysiläisyyttä kohtaan esitettyä kritiikkiä luvussa 2.2. Luvussa 2.4 käsitellään informaatioteoreettista lähestymistapaa mallintamiseen, ja osoitetaan sen läheinen yhteys bayesiläisyyteen.

2.1 Bayesiläinen mallintamisen etuja

Bayesiläinen mallinnus tarjoaa teoreettisesti elegantin, yhtenäisen lähestymistavan kaikkiin oppivien ja älykkäiden järjestelmien rakentamisessa esiintyviin ongelmiin. Mikäli käytettävä malliperhe on lisäksi probabilistinen, saavutetaan lähestymistavalla monia käytännön sovellusten kannalta merkittäviä etuja:

Ristiriidaton kalkyyli pätäsmällisen tiedon käsittelemiseksi.
Todennäköisyyslaskenta tarjoaa epätäsmällisen tiedon käsittelemiseksi matemaattisen kalkyylin, missä uskomusastetta tiettyyn väitteeseen kuvataan reaaliluvulla väliltä [0,1]. Syntyvä kalkyyli on *ristiriidaton l. konsistentti*, mikä tarkoittaa sitä että arvioidessamme jonkin suureen todenperäisyyden astetta, on todennäköisyyslaskennan tarjoama vastaus aina yksikäsitteinen. Mielivaltaisen lukuja väliltä [0,1] manipuloivan numeerisen järjestelmän ristiriidattomuus ei suinkaan ole itsestäänselvyys. Itse asiassa voidaan osoittaa että tiettyjen oletusten vallitessa todennäköisyyslaskenta on ainoa olemassaoleva konsistentti eätäsmällistä tietoa käsittelevä kalkyyli. Vaikka todistuksen perustana olevia oletuksia ei hyväksyttäisikään, osoittaa tämä teoreettinen tulos että konsistentin kalkyylin luominen on erittäin vaikeaa (enemmän aiheesta löytyy lähteestä [133]).

Todennäköisyyslaskennan käyttäminen poistaa siten yhden epätäsmällistä päättelyä suorittavien järjestelmien ongelmista: jos esimerkiksi rakennamme sumean sääntökannan ja käytämme jotakin mielivaltaisesti valittua kalkyyliä (esimerkiksi jotakin min-max-yhdistelysääntöjen variaatiota), ja järjestelmä ei toimi toivotulla tavalla, emme voi tietää onko meidän syytä vaihtaa käytettyä kalkyylia, vai järjestelmän parametreja (jäsenyysasteita). Bayesiläisessä lähestymistavassa kalkyyli on yksikäsitteinen, joten järjestelmän puutteiden korjaamiseksi riittää tarkastella käytetyn mallin parametreja.

Ylioppimista välttävä mallinvalintakriteeri.
Kuten luvussa 1.2.1 todettiin, bayesiläinen mallinvalintakriteeri sisältää "automaattisen Occamin partaveitsen" minkä ansiosta mallin struktuurin kompleksisuus voidaan valita siten että mallin tarkkuus myös opetusjoukon ulkopuolella (mallin yleistyskyky) on mahdollisimman hyvä. Bayesiläinen oppiminen pyrkii siis automaattisesti tasapainoon mallien kompleksisuuden ja niiden kuvausvoiman välillä. Vaikka bayesiläistä mallinvalintakriteeriä on luonnollisinta käyttää aidosti probabilististen malliperheiden, kuten esimerkiksi Bayes-verkkojen tapauksessa (katso luku 3.3), voidaan kriteeriä soveltaa myös muiden malliperheiden, esimerkiksi neuroverkkojen [88, 96, 12, 14, 112] ja päätöspuiden [110, 100] oppimisessa.

Tilastollisen aineiston ja asiantuntijatietämyksen yhdistäminen luonnollisella, teoreettisesti tyydyttävällä tavalla.
Luvussa 1.2.4 hahmoteltiin kuinka asiantuntijatietämys voidaan yhdistää tilastolliseen aineistoon priorijakaumia käyttäen. Toisaalta monissa probabilistisissa malliperheissä (kuten esimerkiksi luvussa 3 esitettyjen Bayes-verkkojen tapauksessa) voidaan sekä mallistruktuureille että malliparametreille antaa

selkeä semanttinen tulkinta, jolloin malleja voidaan konstruoida myös suoraan asiantuntijatietämystä käyttäen, ilman tilastollista oppimista (toisin kuin neuroverkkojen ja muiden ns. "black box"-mallien tapauksessa). Luvussa 3.3 sovellamme näitä periaatteita käytännössä, ja esitämme kuinka Bayes-verkkomalleja voidaan konstruoida käyttäen pelkästään tilastollista aineistoa, pelkästään asiantuntijatietämystä, tai yhdistämällä nämä kaksi tietolähdettä.

Parametrien oppimisen nopeus.
Kuten luvussa 1.3 todettiin, bayesiläisessä mallintamisessa voidaan optimaaliset malliparametrit löytää monessa tapauksessa suoraan ilman aikaavievää iteratiivista oppimisprosessia (vrt. neuroverkkojen parametrien oppiminen backpropagation-algoritmilla). Luvussa 3.3.2 annamme yksinkertaisen laskukaavan, jonka avulla Bayes-verkkomallin parametrit voidaan laskea suoraan annetusta opetusjoukosta $\mathcal{D}$.

Tarvittavan tilastollisen aineiston vähyys.
Bayesiläinen mallintaminen mahdollistaa asiantuntijatietämyksen hyväksikäytön mallien oppimisprosessissa. Jos asiantuntijatietämystä on saatavilla, on tarvittavan tilastollisen aineiston määrä hyvin pieni — itse asiassa mallit voidaan konstruoida suoraan asiantuntijatietämyksen perusteella, ilman tilastollista oppimista! Toisaalta, vaikka asiantuntijatietämystä ei olisikaan saatavilla, todellisilla aineistoilla suoritetut empiiriset kokeet [74] ovat osoittaneet että bayesiläisessä oppimisessa jo hyvin pieni esimerkkiotos (parhaissa tapauksissa vain muutamista esimerkeistä muodostuva joukko) riittää hyvän tuloksen saavuttamiseksi parametrien oppimisessa.

Monipuoliset sovellusmahdollisuudet yhteisjakauman mallintamisen ansiosta.
Kuten luvussa 1.3 näimme, bayesiläinen mallintaminen mahdollistaa hyvin monenlaisten ongelmien kuvaamisen yhdessä ristiriidattomassa teoreettisessa kehikossa. Koska probabilistiset mallit kuvaavat mallinnetussa maailmassa esiintyvien tilanteiden yhteisjakaumaa, voidaan samaa probabilistista mallia käyttää joustavasti hyvin erityyppisissä sovelluksissa.

Mahdollisuus sekoittaa jatkuvia ja diskreettejä muuttujia.
Probabilistisissa malleissa voidaan käyttää joko moniarvoisia diskreettejä muuttujia, jatkuva-arvoisia muuttujia, tai molempia yhtäaikaa. Tähän kysymykseen palataan luvussa 3.4.

Puuttuvan tiedon käsittelymekanismit.
Kuten luvussa 1.3 näimme, probabilistinen lähestymistapa tarjoaa selkeän teoreettisen kehikon puuttuvan tiedon käsittelemiseksi: jos puuttuva tieto ei sinänsä ole kiinnostuksen kohteena, probabilistinen päättely voidaan suorittaa olemassaólevan tiedon perusteella marginalisoimalla (integroimalla) puuttuvan tiedon vaikutus pois. Puuttuva tieto voidaan toisaalta myös täydentää optimaalisella tavalla vertailemalla eri täydennysten todennäköisyyksiä.

Toimintavaihtoehtojen päätösteoreettinen analyysi.
Vaihtoehtoisista lähestymistavoista poiketen bayesiläinen päätösteoria tarjoaa mahdollisuuden arvioida eri toimintavaihtoehtoihin liittyviä odotettavia riskejä ja hyötyjä. Päätösteoriaa voidaan käyttää odotusarvoisesti pienimmän riskin (tai suurimman hyödyn) tuottavan vaihtoehdon etsimiseksi luvussa 1.4 esitetyllä tavalla, tai jo tehdyn päätöksen seurauksien analysoimiseksi. Päätösteoriaa voidaan luonnollisesti soveltaa myös ei-probabilististen mallien antamien tulosten käsittelyssä, jos mallien tulokset esitetään todennäköisyysjakaumana. On kuitenkin huomattava että pelkkä numeroiden suora normeeraus siten että niiden summa on yksi ei välttämättä tuota todennäköisyysjakaumaa joka vastaisi ongelmakentän todellista jakaumaa hyvin, eikä päätösteorian soveltaminen tällaiselle jakaumalle tuota hyviä tuloksia.

Teoreettinen kehikko hybridimallien konstruoimiseksi.
Luvussa 1.2 esitetyssä lähestymistavassa valittiin annetun opetusdatan $\mathcal{D}$ perusteella yksi mallistruktuuri M, ja yksi parametriarvoilmentymä $\hat{\theta}$, joiden avulla konstruoitiin ongelmakenttää mallintava yhteistodennäköisyysjakauma $P(X_1 = x_1, \ldots, X_n = x_n \mid M, \hat{\theta}, \mathcal{D})$. On kuitenkin huomattava, että vaikka esitetty bayesiläinen kriteeri valitsee kaikkein todennäköisimmän parametrivektorin $\hat{\theta}$, saattaa olla mahdolllista että on muita parametriarvovektoreita, jotka ovat lähes yhtä todennäköisiä kuin $\hat{\theta}$. Jos päämääränä on tehdä tarkkoja ennustuksia saatavilla olevan aineiston $\mathcal{D}$ perustella, on helppo nähdä että ennustusten tarkkuutta voidaan parantaa käyttämällä yhden mallin sijasta joukkoa todennäköisiä parametrivektoreita, painottaen niiden antamia tuloksia parametrivektorien todennäköisyydellä. Itse asiassa bayesiläinen lähestymistapa ei pysähdy edes vielä tähän, vaan tarkasti ottaen bayesiläisessä mallintamisessa edellytetään, että optimaalisen ennustustarkkuuden saavuttamiseksi on käytettävä jakaumaa ,joka saadaan laskemalla integraali

("painotettu summa") yli *kaikkien* (äärettömän monen) parametrivektorin θ:

$$P(X_1 = x_1, \ldots, X_n = x_n \mid M, \mathcal{D}) \\ = \int P(X_1 = x_1, \ldots, X_n = x_n \mid M, \theta) P(\theta \mid M, \mathcal{D}) d\theta.$$

Tämä lähestymistapa johtaa odotusarvoparametrien käyttöön, joista puhutaan luvussa 3.3.2. Toisaalta vastaavaa ajatuskulkua voidaan soveltaa myös mallistruktuurin valinnassa, jolloin päädytään käyttämään jakaumaa

$$P(X_1 = x_1, \ldots, X_n = x_n \mid \mathcal{D}) \\ = \sum_M (X_1 = x_1, \ldots, X_n = x_n \mid M, \mathcal{D}) P(M \mid \mathcal{D}).$$

Bayesiläisen filosofian mukaan opetusdatan perusteella saadaan tarkin ongelmakentän jakauman esitys muodostamalla painotettu summa saatavilla olevista mallistruktuureista, kun painokertoimina käytetään mallistruktuurien posterioritodennäköisyyksiä $P(M \mid \mathcal{D})$. Näin voidaan yhdistää esimerkiksi monen eri Bayes-verkon antamat tulokset, tai muodostaa vaikkapa Bayes-verkon, neuroverkon, ja päätöspuun muodostama hybridimalli. Jälkimmäisessä tapauksessa ongelmaksi saattaa kuitenkin muodostua se, kuinka konstruoida jakaumamalli, joka antaa eri malliperheistä tuleville mallistrktuureille yhteismitalliset todennäköisyydet. Lisäksi on huomattava että hybridimallien tulkittavuus ei ole kovin hyvä, joten lähestymistapaa ei voida käyttää ongelmakentän analysointiin tähtäävissä sovelluksissa (data mining).

2.2 Bayesiläinen lähestymistavan kritiikkiä

Edellisessä luvussa esitetyistä bayesiläisen mallintamisen eduista huolimatta on bayesiläinen mallinnus toistaiseksi jäänyt ehkä vähemmälle huomiolle kuin vaihtoehtoiset lähestysmistavat. Yksi syy on se, että bayesiläiseen mallinnukseen liittyy sen formaalin luonteen vuoksi joitakin teknisiä ongelmia, mutta erityisesti luvussa 3 kuvatun Bayes-verkkomalliperheen syntyminen on poistanut monet näistä ongelmista. Saavutetut tulokset ovat kuitenkin verrattain uusia (Bayes-verkkoteoriaa on tutkittu laajemmin vasta noin 10 vuoden ajan), eivätkä siksi vielä kovin laajalti tunnettuja. Maailmalla liikkuukin bayesiläisestä mallintamisesta monia harhakäsityksiä, jotka eivät pidä paikkaansa.

Seuraavassa kokoelma yleisimpiä bayesiläiseen lähestymistapaan kohdistuvia kriittisiä kommentteja.

Bayesiläinen mallinnus edellyttää epärealistisen suuren parametrijoukon arvojen määräämistä.
Bayesiläinen mallinnuksen myönnetään usein olevan *teoriassa* hieno lähestymistapa, mutta probabilististen mallien uskotaan edellyttävän niin suuren parametrijoukon arvojen tarkkaa määrittämistä, että lähestymistapa ei ole käytännössä mahdollinen. Luvussa 3 esitetyt Bayes-verkkomallit ovat vasta muutamia vuosia sitten kehitetty elegantti ratkaisu tähän ongelmaan: Bayes-verkkoja käyttäen tarvittavien parametrien määrä saadaan vähennettyä käytännön sovellukset mahdollistavalle tasolle. Toisaalta, kuten edellä jo todettiin, bayesiläinen mallintaminen tarjoaa mahdollisuuden oppia malleja suoraan tilastollisesta aineistosta, suoraan asiantuntijatietämystä soveltaen, tai yhdistämällä molemmat tietolähteet. Tämä mahdollisuus siirtää parametrien määräämistehtävän ihmiseltä koneelle suoritettavaksi. Lisäksi on vielä todettava että bayesiläisten mallien on empiirisesti havaittu kestävän hyvin parametrien arvojen epätarkkuutta, jolloin parametrien tarkka määrittäminen ei ole tärkeää—käytännön sovelluksia ajatellen on usein riittävää kunhan ne ovat oikeaa suuruusluokkaa [57].

Priorijakaumien määrääminen on vaikeaa.
Kuten edellä näimme, bayesiläinen mallintaminen edellyttää että järjestelmälle annetaan kaksi priorijakaumaa: mallistruktuurien priorijakauma $P(M)$ ja malliluokan parametrien priorijakauma $P(\theta \mid M)$. Tämä seikka mainitaan usein bayesiläisen mallintamisen heikkoutena, vaikka tosiasiassa se on yksi lähestymistavan suurimpia etuja — mahdollistaahan priorijakaumien käyttö nimenomaan tilastollisen tiedon ja asiantuntijatietämyksen yhdistämisen luonnollisella tavalla. Mikäli prioritietämystä ei ole saatavilla, hyvä vaihtoehto käytännössä on käyttää ns. ei-informatiivista, tasaista priorijakaumaa. Toinen käyttökelpoinen vaihtoehto on käyttää informaatioteoreettisia lähestymistapoja joita tarkastellaan luvussa 2.4.

Bayesiläinen päättely on epäintuitiivista.
Bayesiläistä lähestymistapaa on kritisoitu siitä, että sen tarjoama epätäsmällisen päättely olisi jollakin tavoin “epäintuitiivista”, erilaista kuin ihmisten suorittama epätäsmällinen päättely. Suurin syy tähän uskomukseen on vanha objektivistinen todennäköisyyskäsitteen tulkinta, jonka mukaan todennäköisyydet määritellään toistokokeissa esiintyvinä frekvensseinä. Uudemman, *subjektivistisen* todennäköisyystulkinnan mukaan todennäköisyydet määritellään uskomusastetta kuvaavina lukuina väliltä $[0, 1]$. Tämän ansiosta voidaan subjektivistisessa lähestymistavassa puhua esimerkiksi todennäköisyydestä sille että maailmanloppu tulee vuonna 2000, kun taas vanhassa fre-

kventistisessä mallissa kyseisen todennäköisyyden käyttäminen oli ongelmallista. Luvussa 3.2 annamme probabilistisesta päättelystä esimerkkejä, jotka tukevat intuitiivisesti hyvin luonnolliselta tuntuvia päättelyn muotoja (esimerkiksi ns. syiden poisselittämisilmiötä).

Myös lukuisat päätösteoriaan liittyvät paradoksit aiheuttavat sekaannusta. yypillisessä esimerkissä koehenkilölle annetaan kaksi vaihtoehtoa: vaihtoehdossa A hän saa kymmenen miljoona markkaa todennäköisyydellä 1 (siis varmasti), kun taas vaihtoehdossa B hän saa kymmenen miljoona markkaa todennäköisyydellä 0.25, neljäkymmentä miljoonaa markkaa todennäköisyydellä 0.25, ja ei mitään todennäköisyydellä 0.5. Vaihtoehto A on varmaankin itse kunkin mielestä paljon houkuttelevampi, vaikka päätösteoria suosittelee vaihtoehdon B valitsemista, koska voiton odotusarvo on yli kymmenen miljoona markkaa! Esimerkki paljastaa kuinka tärkeää päätösteoriassa on odotettavissa olevan hyödyn mittaaminen oikein: siinä oletettavassa lähtötilanteessa, jossa koehenkilö ei ole miljonääri, ei neljänkymmenen miljoonan markan voittaminen ole hänelle neljä kertaa arvokkaampaa kuin kymmenen miljoonan voittaminen: jo kymmenen miljoonaa riittää siihen että henkilö voi elää loppuelämänsä mukavissa olosuhteissa ilman tarvetta palkkatyöhön, joten voittosumman kasvattaminen nelinkertaiseksi ei tuo merkittävää lisäystä voiton aikaansaamaan elämänmuutokseen. Jos eri markkamääriä vastaavat todelliset arvostukset otetaan huomioon, päätösteoria toimii inhimillisesti katsoen “järkevällä“ tavalla.

Toisaalta on vielä todettava, että vaikka bayesiläinen päättely olisikin jossakin mielessä epäintuitiivista, ehkäpä se olisi vain hyvä asia: ihmiset ovat monesti osoittautuneet huonoiksi päätöksentekijöiksi monimutkaisia suureita käsittelevissä tilanteissa. Kuten edellä kuvattu “paradoksi” osoittaa, ihmiselle jää oppivien ja älykkäiden järjestelmien käyttämisessä tärkeä rooli: on vaikeaa kuvitella tietokoneohjelmaa joka pystyisi tyydyttävästi määräämään ongelmakentästä tärkeimmät olemassaolevat toimintavaihtoehdot, ja arvioimaan niihin liihin liittyviä hyötyjä ja haittoja.

Bayesiläinen mallintaminen käsittelee vain tiedon epävarmuutta, ei tiedon epätäsmällisyyttä.

Todennäköisyyslaskentaa on kritisoitu siitä, että se käsittelee perinteisiä, täsmällisiä objekteja, eikä epätäsmällisyyttä kuvaavia ns. sumeita käsitteitä (katso esim. [58]). Koska todennäköisyysjakaumia voidaan määritellä mielivaltaisen moniarvoisille tai jatkuva-arvoisille muuttujille, voidaan bayesiläisessä mallintamisessa kuitenkin käsitellä samoja kysymyksiä kuin sumeassa logiikassakin [21, 20].

Bayesiläisen mallintamisen soveltaminen on vaikeaa.
Bayesiläisen mallintamisen edellyttämä kaikkien oletusten eksplisiittisen formalisoinnin täydellinen ymmärtäminen edellyttää tiettyjä matemaattisia valmiuksia. Tällaisen mallintamisen soveltaminen käytännössä ei kuitenkaan edellytä näiden teknisten yksityiskohtien syvempää ymmärrystä, vaan bayesiläisen mallintamisen teoriaa voidaan käyttää tuottamaan yksinkertaisia laskenta-algoritmeja, kuten lukuisat saatavilla olevat ohjelmistot todistavat (ks. luku 4.2). Toisaalta monet heuristisesti kehitetyt "ad hoc"-menetelmät voidaan tulkita todennäköisyyslaskennan tarjoamassa kehikossa. Tämän avulla on pystytty esimerkiksi selittämään ne epärealistiset riippumattomuusoletukset, jotka joudutaan (implisiittisesti) tekemään käytettäessä ns. varmuuskertoimiin perustuvia sääntökantoja [46].

2.3 Esimerkkeihin perustuva päättely ja ei-parametriset mallit

Ei-parametrisissa (non-parametric) lähestymistavoissa luovutaan mallien eksplisiittisestä, mallistruktuurin kiinnittämästä parametroidusta esitysmuodosta, ja yritetään ratkaista ongelmakentässä esiintyvät tehtävät suoraan esimerkkitapauksia soveltaen. *Esimerkkeihin perustuvassa päättelyssä (case-based reasoning, CBR)*[1] päättelyongelmat ratkaistaan kaksivaiheisella prosessilla. Ensimmäisessä, *esimerkkien vertailuvaiheessa (case matching)*, käsillä olevaa tilannetta verrataan muistissa oleviin aikaisempiin esimerkkitapauksiin (cases), ja tapauksille annetaan yhteensopivuusaste (matching score), joka on sitä suurempi mitä enemmän muistissa oleva tapaus muistuttaa nykyistä tilannetta. Päättelyn toisessa, *esimerkkien sovellusvaiheessa (case adaptation)* muodostetaan ratkaisuehdotus muokkaamalla suurimman yhteensopivuusasteen saaneiden vanhojen tapausten ratkaisemisessa käytetyistä menetelmistä ehdotus nykyisen tilanteen ratkaisemiseksi.

Yksinkertaisin esimerkki CBR-tyyppisestä lähestymistavasta on ns. *lähimmän naapurin menetelmä (nearest neighbor method)*, jossa esimerkkien yhteensopivuusasteena käytetään niiden (euklidista) etäisyyttä, ja tarvittavat tiedot kopioidaan suoraan korkeimman yhteensopivuusasteen saaneesta esimerkkitapauksesta. Monimutkaisemmissa malleissa tarvittavat tiedot muodostetaan käyttämällä opetusjoukon tapauksien painotettua summaa, käyttäen painokertoimina tapausten yhteensopivuusasteita. Esimerkkeihin

[1] Lisätietoa esimerkkeihin perustuvasta päättelystä löytyy esimerkiksi lähteistä [68, 139]. Lähestymistavasta käytetään myös termejä *memory-based reasoning* [131],*instance-based learning* [2], *lazy learning* [1] ja *transductive inference* [137].

perustuvan päättelyn toimintaperiaate on hyvin samanlainen kuin tilastotieteessa *ydinestimaattoreiksi (kernel estimators)* kutsutuilla menetelmillä (katso esim. [121]). Ydinestimaattoreissa muodostetaan annetun opetusjoukon kunkin tapauksen ("ytimen") ympärille jakauma jonka keskikohta on ytimessä, ja kokonaisjakauma saadaan tällaisten ydinjakaumien summana. Neuroverkkotutkimuksen alueella vastaavia menetelmiä kutsutaan *ydinkantafunktioiksi (radial basis functions)* [92, 107] tai *probabilistisiksi neuroverkoiksi (probabilistic neural networks)* [128].

On tärkeää huomata että nimestään huolimatta ei-parametriset mallit eivät suinkaan ole parametrittomia: esimerkiksi vaikka ydinestimaattoreissa on ydinjakaumien määrä, ja niiden keskikohta kiinnitetty, on jakaumien tarkka muoto määrättävä joukolla parametreja (esim. ydinjakaumakohtaiset varianssit), jotka on arvioitava esimerkkiaineistosta. Esimerkkeihin perustuvassa päättelyssä puolestaan on tapausten yhteensopivuusasteen laskemisessa käytettävä metriikka käyttäjän vapaasti määrättävissä, ja eri metriikat voidaan ajatella jonkin (mahdollisesti hyvin monimutkaisen) malliperheen parametrisoituina malleina. Se, halutaanko puhua eri metriikoista ja mallien soveltamisessa käytettävissä laskentamenetelmistä, vai parametrisista malleista, näyttää olevan pitkälti makuasia.

Bayesiläisiin parametrisiin malleihin perustuvilla lähestymistavoilla on se etu, että menetelmä pakottaa määrittelemään täsmällisesti kaikki oletukset, joiden varassa rakennettavan järjestelmän toiminta on. Ilman tällaista lähestymistapaa päädytään ad hoc-tyyppisiin menetelmiin, joiden toimivuudesta ei ole teoreettisia takeita. Voidaan esimerkiksi kysyä mihin perustuu useimmissa CBR-menetelmissä implisiittisesti tehtävä oletus, jonka mukaan euklidisen etäisyysmitan mielessä toisiaan muistuttavien tapausten ratkaisemisessa voidaan soveltaa samoja menetelmiä? Tämä oletus saattaa tietenkin pitää paikkansa ongelmasta, ja sen tapausten koodaamisessa käytetystä menetelmästä riippuen, mutta menetelmien käyttäminen ilman tehtyjen implisiittisten oletusten tiedostamista saattaa johtaa vakaviin virhepäätelmiin.

Ei-parametristen lähestymistapojen ero parametriseen mallintamiseen näyttää olevan siis lähinnä siinä, että ei-parametrisissa malleissa parametrien lukumäärä ei ole kiinteä, vaan se riippuu käytettävissä olevan opetusjoukon koosta. Kuten edellä kuitenkin näimme, baysiläisessä mallintamisessa on mallistruktuurin valinta kiinteä osa oppimisprosessia, ja valittava mallistruktuuri (ja sitä kautta malliparametrien lukumäärä) riippuu käytettävästä opetusjoukosta[2]! Parametristen ja erilaisten ei-parametristen lähestymistapojen ero ei siis näytä olevan periaattellinen, vaan lähinnä terminologinen. Itse asiassa

[2]Tästä syystä tähän periaatteeseen perustuvia malleja kutsutaan joskus *semiparametrisiksi (semi-parametric)* malleiksi.

sekä esimerkkeihin perustuva päättely että ydinestimaatorit (ja muut vastaavat mallit) voidaan ajatella erikoistapauksena *äärellisistä sekajakaumista (finite mixture models)* [35, 136], joita käsitellään enemmän luvussa 3.4. Äärellisten sekajakaumamallien ja esimerkkeihin perustuvan päättelyn samankaltaisuus osoitetaan lähteissä [135, 134, 95]. Yleisempää todennäköisyyslaskentaan perustuvaa formalismia esimerkkeihin perustuvalle päättelylle on ehdotettu lähteessä [69]. On huomattava, että ehdotetut bayesiläiset formalismit eivät välttämättä noudata esimerkkeihin perustuvan päättelyn normaalia kaksivaiheista päättelyprosessia, vaan päättely voi tapahtua suoraan ilman esimerkkien yhteensopivuusasteen laskemista. Bayesiläinen menetelmä tapausten yhteensopivuusasteen laskemiseksi esitetään lähteessä [70].

2.4 Informaatioteoreettiset lähestymistavat: MDL ja MML

2.4.1 MDL-periaate ja stokastinen kompleksisuus

Mallistruktuurin valinnassa voidaan käyttää myös informaatioteoreettista lähestymistapaa, joka on hyvin läheisessä suhteessa bayesiläiseen mallintamiseen. Rissasen *Minimum Description Length (MDL)*-periaatteen [110] mukaan mallistruktuureista valitaan se, jonka avulla opetusjoukko $\mathcal{D}$ voidaan koodata siten että syntyvä koodi on mahdollisimman lyhyt. Rissanen kutsuu lyhintä tällaista koodinpituutta opetusjoukon $\mathcal{D}$ *stokastiseksi kompleksisuudeksi (stochastic complexity)*, annettuna mallistruktuuri M. Tällä koodausteoreettiselle lähestymistavalle voidaan antaa intuitiivisesti selkeä tulkinta: lyhimmän koodin tuottava mallistruktuuri on ongelmakentän paras kuvaus siksi että lyhyen koodin aikaansaamiseksi on mallistruktuuriin koodattava kaikki mahdolliset aineistossa $\mathcal{D}$ esiintyvät säännönmukaisuudet.

Kuinka stokastinen kompleksisuus sitten määritellään? Ensiksikin on korostettava, että on mahdotonta konstruoida määritelmää joka tuottaisi lyhimmän koodin kaikille mahdollisille opetusjoukoille $\mathcal{D}$: koodausteorista tiedämme että jos $L(\mathcal{D} \mid M)$ kuvaa opetusjoukkoa $\mathcal{D}$ vastaavan koodin pituutta mallistruktuurin M tapauksessa, määrittelee

$$P(\mathcal{D} \mid M) = 2^{-\log L(\mathcal{D}|M)}$$

todennäköisyysjakauman yli kaikkien mahdollisten joukkojen $\mathcal{D}$. Koska

$$\sum_{\mathcal{D}} P(\mathcal{D}|M) = 1,$$

ei ole mahdollista että olisi todennäköisyysjakauma P^* jolle pätisi: $P^*(\mathcal{D}|M) > P(\mathcal{D}|M)\ \forall P, \mathcal{D}$. Siispä ei myöskään voi olla sellaista koodia L, joka tuottaisi lyhimmän mahdollisen kuvauksen kaikille joukoille $\mathcal{D}$. Stokastinen kompleksisuus täytyykin määritellä siten että se tuottaa lyhyimmän kuvauksen 'tyypillisimmille' joukoille $\mathcal{D}$. Se kuinka 'tyypillisyys' määritellään, ratkaisee viime kädessä eri stokastisten määritelmien hyvyyden.

Varhaisessa tuotannossaan [109] Rissanen määritteli stokastisen kompleksisuuden seuraavasti:

$$L(\mathcal{D} \mid M) = -\log P(\mathcal{D} \mid M, \tilde{\theta}) + \frac{d(M)}{2} \log N, \tag{2.1}$$

missä $(M, \tilde{\theta})$ on malliluokan M *suurimman uskottavuuden (maximum likelihood)*-malli, ts. malli, joka maksimoi uskottavuustermin $P(\mathcal{D} \mid M, \theta)$, $d(M)$ on mallin parametrien lukumäärä, ja N opetusjoukon $\mathcal{D}$ koko. Näemme että tämä informaatioteoreettinen stokastisen kompleksisuuden määritelmä antaa saman tuloksen kuin luvussa 1.2.2 esitetty kokonaisuskottavuuden BIC-approksimaatio.

Myöhemmin(katso esim. [110]) Rissanen osoitti, että määritelmä

$$L(\mathcal{D} \mid M) = -\log P(\mathcal{D} \mid M) = -\log \int P(\mathcal{D} \mid M, \theta) P(\theta \mid M) d\theta \tag{2.2}$$

tuottaa koodeja, joiden voidaan osoittaa olevan monessa suhteessa parempia kuin määritelmän (2.1) tuottamat koodit. Määritelmästä (1.1) näemme että tämä kriteeri vastaa bayesiläistä lähestymistapaa, jossa mallistruktuurien prioriksi on oletettu tasainen jakauma, ts. $P(M_1) = P(M_2) = \ldots = P(M_K)$, ja mallistruktuurin valintakriteerinä käytetään siis yksinomaan kokonaisuskottavuutta $P(\mathcal{D} \mid M)$. Äskettäin [111] Rissanen esitteli uuden, tietyssä mielessä vielä tehokkaamman koodin, jossa stokastinen kompleksisuus määritellään seuraavasti:

$$L(\mathcal{D} \mid M) = -\log \frac{P(\mathcal{D} \mid \tilde{\theta}(\mathcal{D}), M)}{\sum_{\mathcal{D}'} P(\mathcal{D}' \mid \tilde{\theta}(\mathcal{D}'), M)}, \tag{2.3}$$

missä $\tilde{\theta}(\mathcal{D})$ on opetusjoukkoa $\mathcal{D}$ vastaava suurimman uskottavuuden malli, ja summa nimittäjässä käy yli kaikkien mahdollisten opetusjoukkojen. Voidaan osoittaa että tämä uusi stokastisen kompleksisuuden määritelmä antaa asymptoottisesti saman tuloksen kuin määritelmä (2.2), jos käytämme integraalissa (2.2) parametrien priorijakaumana ns. Jeffreyn prioria. Tätä kysymystä käsitellään tarkemmin lähteissä [75, 76].

2.4.2 MML-periaate

Wallacen kehittämä *MML (minimum message length)*-periaate on läheistä sukua Rissasen MDL-periaatteelle, mutta MML-lähestymistavassa keskitytään malliparametrien valitsemisen ongelmaan, kun taas MDL-periaatetta sovelletaan yleensä mallistruktuurin valintaongelmaan. MML-lähestymistavassa hylätään edellä esitetty mallien kaksivaiheinen oppiminen, ja sulautetaan kaikki malliluokat M yhdeksi suureksi kaikkien mallien muodostamaksi malliperheeksi. Koska malliluokan käsitettä ei tässä lähestymistavassa käytetä, malliperhe muodostuu erikokoisista malleista eli eripituisista parametrivektoreista θ.

MML-periaatteen mukaan *paras malli $\bar{\theta}$ on se joka tuottaa lyhimmän koodin kun kun koodataan käytetty malli $\bar{\theta}$, ja opetusaineisto $\mathcal{D}$ käyttäen mallia $\bar{\theta}$*:

$$\bar{\theta} = \arg\min_{\theta} L(\theta) + L(\mathcal{D} \mid \theta).$$

Jos mallien parametrit oletetaan reaaliluvuiksi, ei koodia kuitenkaan voida konstruoida, koska puhtaiden reaalilukujen esittäminen edellyttää kykyä esittää lukuja äärettömän tarkasti, mikä taas vaatii äärettömän pitkiä koodeja. Siksi MML-lähestymistavassa oletetaan että mallien parametrit pystytään esittämään vain äärellisellä tarkkuudella (mikä tietysti on realistinen oletus käytännön sovelluksia ajatellen), ja mallit θ diskretisoidaan äärellisen koodin saamiseksi. Diskretisointi on tehtävä niin, että syntyvän koodin pituus on mahdollisimman lyhyt.

Baysiläisessä oppimisessa toisaalta yksittäisen mallin θ soveltuvuus mitattiin sen posterioritodennäköisyydellä

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta)P(\theta)}{P(\mathcal{D})}. \tag{2.4}$$

Jos unohdamme nimittäjässä olevan vakion $P(\mathcal{D})$, saamme

$$-\log P(\theta \mid \mathcal{D}) \propto -\log P(\mathcal{D} \mid \theta) - \log P(\theta).$$

Kun merkitsemme $L(\theta) = -\log P(\theta)$ ja $L(\mathcal{D} \mid \theta) = -\log P(\mathcal{D} \mid \theta)$, näemme että MML-periaate on bayesiläinen siinä mielessä että se valitsee mallin joka maksimoi posterioritodennäköisyyden (2.4), kun mallit on diskretisoitu optimaalisella tavalla. MML-periaatteesta löytyy lisämateriaalia lähteistä [99, 98, 6], ja Monash-yliopiston tutkimusryhmän ylläpitämältä WWW-sivulta[3].

[3] URL: http://www.cs.monash.edu.au/~lloyd/tildeMML/'

Luku 3

Bayes-verkot

Mitä ovat Bayes-verkot? Kausaalinen mallintaminen Bayes-verkoilla. Päättely epätäydellisillä tiedoilla Bayes-verkkoja käyttäen. Bayes-verkkojen muodostaminen automaattisesti: verkon rakenteen oppiminen ja parametrien oppiminen. Bayes-verkkojen muunnelmia: päätösverkot, jatkuva-arvoiset Bayes-verkot, Naiivi Bayes-malli, kvalitatiiviset Bayes-verkot.

Kuten kuvasta 1.4 näemme, ensimmäinen askel bayesiläisessä mallinnusprosessissa on malliperheen valinta. Vaikka bayesiläistä mallinnusta voi periaatteessa soveltaa myös ei-probabilistisiin malliperheisiin, kuten esimerkiksi neuroverkkoihin (ks. [88, 96, 12, 14, 113]), saavutetaan bayesiläisen lähestymistavan edut helpoimmin käyttäen probabilistisia malleja. *Bayes-verkot* (Bayesian networks, Bayesian belief networks) muodostavat tällaisen malliperheen, joka on saavuttanut viime aikoina suurta huomiota. Pääsyy Bayes-verkkoihin kohdistuvassa kiinnostuksessa johtuu 80-luvulla kehitetystä teoreettisesta kehikosta, jonka avulla probabilististen mallien luomiseen tarvittavien parametrien määrää voidaan huomattavasti pienentää. Luvussa 3.1 selvitetään lyhyesti Bayes-verkkojen pääperiaatteet, puuttumatta tässä yhteydessä tarkemmin teknisiin yksityiskohtiin, jotka selviävät esimerkiksi lähteistä [105, 97, 60, 18]. Luvussa 3.2 näemme kuinka luvussa 1.3 esitettyjä probabilistisen päättelyn eri muotoja on mahdollista toteuttaa Bayes-verkkomallien avulla. Bayes-verkkojen konstruoimista käsitellään luvussa 3.3.

3.1 Bayes-verkkomalliperhe

Esitämme seuraavassa Bayes-verkkomalliperheen perusominaisuudet yksinkertaistettujen esimerkkien avulla. Tässä yhteydessä on korostettava, että vaikka käytetyissä esimerkeissä esiintyy yleensä vain binäärisiä diskreettejä muuttujia, voidaan lähestymistavassa käyttää myös sekä moniarvoisia diskreettejä muuttujia että jatkuvia muuttujia. Tähän kysymykseen palaamme luvussa 3.4.

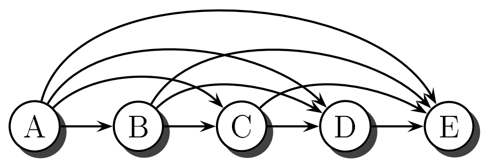
Oletetaan seuraavassa esimerkissä että ongelmäkentän kuvauksessa käytetään viittä binääristä attribuuttia (satunnaismuuttujaa) A, B, C, D, E, ja käytetään binäärimuuttujien mahdollisista arvoista merkintöjä 0 ja 1. Tässä matemaattisessa mallissa maailman kaikki mahdolliset tilat koostuvat muuttujien A, B, C, D, E arvojen muodostamista binäärivektoreista, joita on tässä yksinkertaisessa viiden binäärimuuttujan tapauksessa kaikkiaan $2^5 = 32$ kappaletta. Bayesiläisessä mallinnuksessa on päämääränä muodostaa matemaattinen malli, jonka avulla voidaan määrätä todennäköisyys mille tahansa mahdolliselle maailman tilalle (5-komponenttiselle binäärivektorille). Tällaista *yhteistodennäköisyysjakaumaa* hyväksikäyttäen voidaan suorittaa luvussa 1.3 esitettyjä probabilistisen päättelyn eri muotoja, kuten luvussa 3.2 tulemme näkemään.

Bayesiläisen mallintamisen päämääränä oleva yhteistodennäköisyysjakauma voidaan luonnollisesti määrätä luettelemalla kaikki mahdolliset tilat ja niitä vastaavat todennäköisyydet, mutta koska tilojen lukumäärä on eksponentiaalinen suhteessa muuttujien lukumäärään, ei tämä lähestymistapa ole käytännössä esiintyvissä tilanteissa mahdollinen. Toisaalta todennäköisyyslaskennan perusaksioomia käyttäen on helppo osoittaa että

$$\begin{aligned} &P(A, B, C, D, E) = \\ &\qquad P(A)P(B|A)P(C|A, B)P(D|A, B, C)P(E|A, B, C, D). \end{aligned} \tag{3.1}$$

Muuttujien A, B, C, D, E yhteistodennäköisyysjakauman määräämiseksi siis riittää, että tunnemme yhtälössä (3.1) oikealla puolella esiintyvät ehdolliset todennäköisyydet. Tässä yhteydessä on huomattava, että se mitä ehdollisia todennäköisyyksiä tarvitaan riippuu käytetystä muuttujien järjestyksestä. Jos yhtälöä (3.1) sovelletaan esimerkiksi muuttujien järjestykseen (E, D, C, B, A), saadaan yhtälön oikealle puolelle ehdolliset todennäköisyydet $P(E)$, $P(D \mid E)$, $P(C \mid E, D)$, $P(B \mid E, D, C)$, ja $P(A \mid E, D, C, B)$. Käytettyä permutaatiota (ja siis sitä vastaavaa ehdollisten todennäköisyyksien joukkoa) voidaan havainnollistaa graafisesti: kuvassa 3.1 on esitetty syklitön suunnattu verkko, jossa muuttujasta Y on muuttujaan X kaari ainoastaan silloin kun jokin hajotelmassa (3.1) käytettävistä ehdollisista todennäköisyyksistä on muotoa $P(X \mid *, Y, *)$.

Kuvassa 3.1 esitetty suunnattu verkko määrää siis minkä tyyppisiä ehdollisia todennäköisyyksiä on käytettävä, jos halutaan määrätä muuttujien yhteisjakauma $P(A, B, C, D, E)$ käyttäen ketjusääntöä (3.1). Kiinnittämäl-



Kuva 3.1: Hajotelman (3.1) esitys suunnattuna verkkona.

lä verkkoesitystä vastaavien ehdollisten todennäköisyyksien arvot on yhteisjakauma yksikäsitteisesti määrätty. Toisaalta, kuten luvussa 1.2 esitettiin, bayesiläisessä mallintamisessa malli määritellään parina (M, θ), missä M eli mallin struktuuri määrää mitä parametrejä mallin identifioimiseksi on määrättävä, ja θ on joukko parametriarvoja jotka kiinnittävät yhden mallin. Kuvassa 3.1 esitettyä graafista verkkoesitystä voidaan siis ajatella mallin struktuurina, ja graafia vastaavia ehdollisia todennäköisyyksiä voidaan pitää mallin parametreina.

Ottaen huomioon että $P(X = 1 \mid *) = 1 - P(X = 0 \mid *)$ voimma laskea että käyttäen kuvassa 3.1 esitettyä mallistruktuuria tarvitsemme $1 + 2 + 4 + 8 + 16 = 31$ parametria (ehdollisen todennäköisyyden arvoa) yhteistodennäköisyysjakauman määräämiseksi. Koko avaruuden tilojen lukumäärään (32) nähden emme ole siis säästäneet paljoa. Keskeinen tekijä Bayes-verkkoteoriassa onkin ehdollisen riippumattomuuden käsite, jota käyttäen tarvittaen parametrien määrää voidaan radikaalisti pienentää:

Määritelmä 1 (Ehdollinen riippumattomuus)
Olkoot $\mathbf{X}$*,* $\mathbf{Y}$*, ja* $\mathbf{Z}$ *erillisiä muuttujajoukkoja. Joukon* $\mathbf{X}$ *sanotaan olevan* ehdollisesti riippumaton *muuttujajoukosta* $\mathbf{Y}$*, annettuna joukko* $\mathbf{Z}$*, jos*

$$P(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}) = P(\mathbf{X} \mid \mathbf{Z}).$$

Selventääksemme ehdollisen riippumattomuuden käsitettä olettakaamme esimerksi että ongelmakentäämme liittyy lääketieteelliseen diagnosointiin, ja esimerkkimuuttujillamme A, B, C, D, E on taulukossa 3.1 esitetyt semanttiset tulkinnat[1].

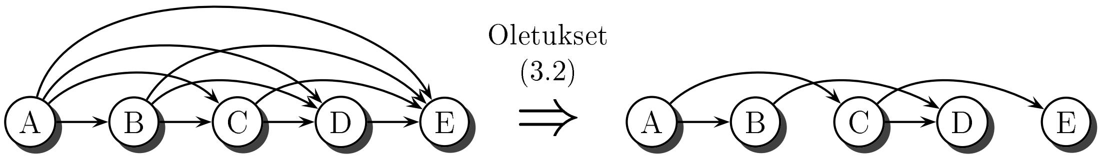
Ongelmakentän hyvin tunteva asiantuntija saattaisi kertoa muuttujien suhteesta esimerkiksi seuraavaa:

> Vakavien päänsärkykohtausten todennäköisyys ($P(E = 1)$) riippuu ainoastaan siitä, onko henkilöllä aivokasvain vain ei (C). Toisaalta jos tiedetään henkilön veren kalsiumpitoisuus (B) ja se, onko henkilöllä kasvain vai ei (C), voidaan määrätä tajuttomuuskohtauksien todennäköisyys ($P(D = 1)$), joka ei tällöin riipu

[1] Esimerkki on teoksesta [105, sivu 196].

Merkintä	*Tulkinta*
$A = 1$:	Tutkittavalla henkilöllä on aivosyöpä.
$B = 1$:	Tutkittavan henkilön veren kalsiumpitoisuus on noussut epänormaalille tasolle.
$C = 1$:	Tutkittavalla henkilöllä on aivokasvain.
$D = 1$:	Tutkittava henkilö saa tajuttomuuskohtauksia.
$E = 1$:	Tutkittava henkilö kärsii vakavista pääkivuista.

Taulukko 3.1: Lääketieteellisessä esimerkissä käytetyt merkinnät.



Kuva 3.2: Ehdollisen riippumattomuuden vaikutus verkkoesitykseen.

> päänsäryn esiintymisestä (E), tai (suoraan) siitä, onko henkilöllä aivosyöpä (A) vai ei. Aivokasvaimen (C) esiintymisen todennäköisyys riippuu vastaavasti suoraan ainoastaan siitä, onko henkilöllä aivosyöpä (A), ei muista tekijöistä.

Yllä mainitut tiedot voidaan toisaalta myös päätellä tilastollisesta aineistosta ilman asiantuntijan apua, mikäli saatavilla on riittävä määrä esimerkkitapauksia. Ehdollisen riippumattomuuden käsitettä soveltaen saatu tieto voidaan nyt koodata seuraavasti:

$$\begin{aligned} &P(E \mid A, B, C, D) = P(E \mid C) \\ &P(D \mid A, B, C) = P(D \mid B, C) \\ &P(C \mid A, B) = P(C \mid A) \end{aligned} \tag{3.2}$$

Kun otamme nämä ehdolliset riippumattomuudet huomioon, kuvassa 3.1 esitetty verkko muuttuu kuvassa 3.2 esitettyyn muotoon. Kun edelleen piirrämme verkon hieman eri tavalla, ja merkitsemme kuvaan selvyyden vuoksi kutakin solmua vastaavan muuttujan semanttisen tulkinnan, saamme kuvassa 3.3 esitetyn Bayes-verkkostruktuurin.

Bayes-verkot määritellään formaalisti seuraavasti:

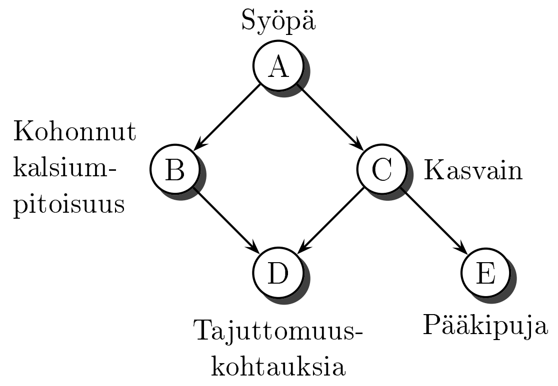
Määritelmä 2 (Bayes-verkkomalli)
Bayes-verkkomalli on pari (M, θ), missä M on syklitön suunnattu verkko, jonka solmut vastaavat ongelmakentän muuttujia, ja jonka topologia täyttää

seuraavan ehdon: kukin muuttuja on ehdollisesti riippumaton kaikista muuttujista, jotka eivät ole muuttujan jälkeläisiä verkossa, annettuna muuttujan edeltäjät, ja muuttujan edeltäjien muodostama joukko on suppein muuttujajoukko joka täyttää tämän ehdon. Parametrijoukon θ muodostavat ehdolliset todennäköisyydet muotoa $P(X \mid F(X))$, missä $F(X)$ on muuttujan X edeltäjien joukko verkossa M.

Tässä yhteydessä on korostettava, että vaikka sekä neuroverkko että Bayes-verkko ovat molemmat verkkoesityksiä, esitysten abstraktiotasossa on huomattava ero. Neuroverkothan ovat lähinnä *funktionaalisen* tason kuvauksia siinä mielessä että neuroverkkoalgoritmien voidaan ajatella toimivan neuroverkkostruktuurin kaltaisessa arkkitehtuurissa, jossa verkon solmut ovat yksinkertaisia laskentayksiköitä jotka lähettävät toisilleen yksittäisistä reaaliluvuista muodostuvia sanomia. Bayes-verkko puolestaan on *käsitteellisen (symbolisen)* tason esitys: Bayes-verkon solmut vastaavat ongelmakentän kuvaamiseen käytettyjä attribuutteja, ja kaaret kuvaavat attribuuttien välisiä riippuvuuksia. Bayes-verkon rakenne ei siis välttämättä suoraan kuvaa Bayes-verkkoalgoritmien toimintaa samalla tavoin kuin neuroverkon rakenne kuvaa neuroverkkoalgoritmien toimintaa (tähän kysymykseen palaamme luvussa 3.2). Erityisesti on huomattava että toisin kuin neuroverkoissa, joissa kuhunkin kaareen liittyy yksi parametri (kaaren paino), Bayes-verkoissa parametrit eivät liity suoranaisesti verkon kaariin, vaan kutakin verkon *solmua* kohden tarvitaan joukko parametreja, joiden lukumäärä on eksponentiaalinen solmun edeltäjien lukumäärän suhteen. Tarvittavien parametrien lukumäärän pienentämiseksi voidaan tätä parametrijoukkoa yrittää approksimoida lineaarisella määrällä parittaisia, muotoa $P(X = x \mid Y = y)$ parametreja, joiden voidaan ajatella liittyvän verkon kaariin: yhtä tällaista approksimaatioita (ns. Noisy-Or-mallia) käsitellään tarkemmin luvussa 3.4.

Kuvassa 3.3 on annettu esimerkki Bayes-verkkomallista kiinnittämällä verkon parametrit lähteessä [105, sivu 197] annettuihin arvoihin (on muistettava että kyseessä on yksinkertaistettu, rajoitettu esimerkki, joten annettuihin parametriarvoihin ei pidä suhtautua lääketieteelliseltä kannalta liian vakavasti). Kaikkien mahdollisten Bayes-verkkomallien (kaikki mahdolliset Bayes-verkkostruktuurit + kaikki mahdolliset struktuuria vastaavien parametrien asetukset) muodostamaa joukkoa kutsutaan seuraavassa *Bayes-verkkomalliperheeksi*. Bayes-verkkojen oppimisella tarkoitetaan yhden Bayes-verkkomallin etsimistä tästä joukosta. Luvussa 3.3 paneudumme tähän kysymykseen tarkemmin.

Bayes-verkkojen määritelmästä ja ehdollisen riippumattomuuden käsitteestä seuraa tärkeä Bayes-verkkomallien avulla koodattujen todennäköisyysjakaumien ominaisuus: jos ongelmakentän kuvauksessa käytetyt muuttujat



$P(A = 1)$	$=$	0.20
$P(B = 1 \mid A = 1)$	$=$	0.80
$P(B = 1 \mid A = 0)$	$=$	0.20
$P(C = 1 \mid A = 1)$	$=$	0.20
$P(C = 1 \mid A = 0)$	$=$	0.05
$P(D = 1 \mid B = 1, C = 1)$	$=$	0.80
$P(D = 1 \mid B = 0, C = 1)$	$=$	0.80
$P(D = 1 \mid B = 1, C = 0)$	$=$	0.80
$P(D = 1 \mid B = 0, C = 0)$	$=$	0.05
$P(E = 1 \mid C = 1)$	$=$	0.80
$P(E = 1 \mid C = 0)$	$=$	0.60

Kuva 3.3: Esimerkki Bayes-verkkomallista.

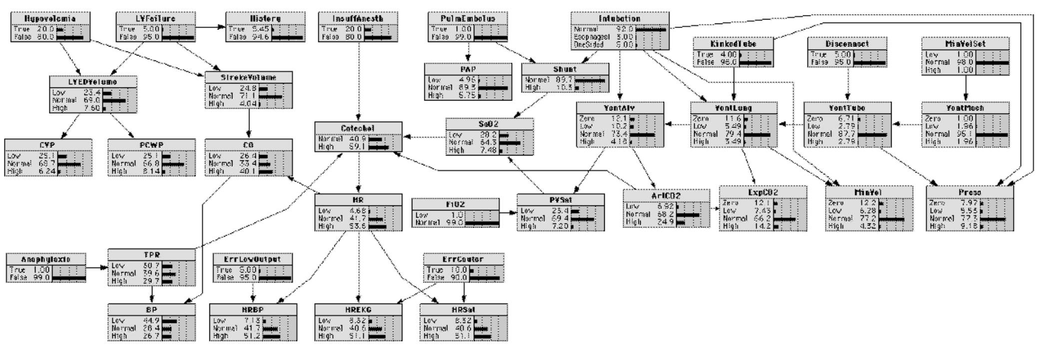
ovat $X_1, \ldots, X_n$, niin minkä tahansa muuttujien arvokombinaation yhteistodennäköisyys saadaan laskettua kaavalla

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid F(X_i) = f(X_i)), \qquad (3.3)$$

missä kuten edellä, $F(X_i)$ on muuttujan X_i edeltäjät käytetyssä Bayes-verkossa, ja $f(X_i)$ joukkoa $F(X_i)$ vastaava arvovektori. Näemme, että Bayes-verkkomalli (M, θ) on probabilistinen sillä annetun verkkostruktuurin parametrit kiinnittävät yhteistodennäköisyysjakauman muuttujille $X_1, \ldots, X_n$. Luvussa 3.2 esitamme kuinka hajotelmaa (3.3) voidaan käyttää hyväksi toteutettaessa luvussa 1.3 esitettyjä probabilistisen päättelyn muotoja.

Kuvassa 3.3 esitetyssä Bayes-verkkomallissa on tarvittavien parametrien lukumäärä 11, mikä on edelleen yli 30% koko avaruuden koosta. On kuitenkin huomattava, että tarvittavat parametrit, kuten esimerkiksi $P(B|A)$, ovat muodoltaan sangen yksinkertaisia, ja siten myös helposti asiantuntijoiden määrättävissä, toisin kuin käsitteellisesti hankalat yhteistodennäköisyydet $P(A, B, C, D, E)$. Lisäksi on korostettava että tarvittavien parametrien suhde avaruuden kokoon pienenee dramaattisesti muuttujien lukumäärän kasvaessa: kuvassa 3.4 on sairaalapotilaiden monitoroinnissa käytetty 37-solmuinen ALARM-verkkona tunnettu Bayes-verkkostruktuuri (katso [8]), jossa tarvittavien parametrien (ei näytetty kuvassa) lukumäärä on 509, ainoastaan $2,94 \times 10^{-12}$% koko avaruuden koosta ($2^{13} \times 3^{17} \times 4^{7} = 1,73 \times 10^{16}$). Kuvassa esitetyt todennäköisyydet ovat eri muuttujien arvojen prioritodennäköisyyksiä, jotka saadaan laskettua ALARM-verkon parametreista. Kuva on tuotettu Norsys-yhtiön Netica-ohjelmistolla, joka on Bayes-verkkojen konstruoimiseen ja niiden käyttämiseen tarkoitettu sovelluskehitin (katso luku 4.2).

Käytännön sovellusten kannalta on kuvassa 3.4 esitetty Bayes-verkkokin



Kuva 3.4: Sairaalapotilaiden valvonnassa käytettävä ALARM-verkko.

edelleen melko pieni — Bayes-verkko-ohjelmistoja valmistava HUGIN-yhtiö ilmoittaa teollisessa käytössä olevan yli tuhat solmua sisältäviä Bayes-verkkoja. Tässä yhteydessä on muistettava, että toisin kuin neuroverkoissa, ei Bayes-verkoissa solmujen lukumäärä ole vapaa parametri, vaan se määräytyy ongelmakentän kuvaamisessa käytettävien attribuuttien lukumäärästä: jo sadallakin solmulla (attribuutilla) voidaan kuvata sangen monimutkaisia ongelmakenttiä. Toisaalta on korostettava, että vaikka solmujen lukumäärä on kiinnitetty, ei Bayes-verkkojen konstruoiminen ole kuitenkaan välttämättä yksinkertaista, koska mahdollisten verkkotopologioiden lukumäärä kasvaa erittäin nopeasti solmujen lukumäärän suhteen (katso luku 3.3).

3.1.1 Bayes-verkot ja kausaalisuus

Kuvassa 3.3 esitetyssä esimerkissä voidaan Bayes-verkkostruktuurin kaarille antaa selkeä kausaalinen tulkinta: esimerkiksi muuttujan A, joka on muuttujien B ja C edeltäjä verkossa, voidaan ajatella olevan muuttujien B ja C kausaalinen *syy*. Mielenkiintoinen kysymys onkin kuinka Bayes-verkkostruktuurin kaaret suhtautuvat kausaalisen syy-seuraussuhteen käsitteeseen — onko kaikkien Bayes-verkkojen kaarilla sama kausaalinen tulkinta?

Yleisesti ottaen vastaus tähän kysymykseen on kielteinen: Bayes-verkon kaarien ei välttämättä tarvitse kuvastaa kausaalisia syy-seuraussuhteita. Tätä seikkaa havainnollistaa seuraava yksinkertainen esimerkki: olkoon ongelmakentän kuvauksessa käytettävät muuttujat A ja B, joilla on seuraavat semanttiset tulkinnat:

A: Tutkittavalla henkilöllä on flunssa (A=1) tai ei (A=0).

B: Tutkittavan henkilön nenä vuotaa (B=1) tai ei (B=0).

Vaihtoehtoisia Bayes-verkkostruktuureja on kaksi: $A \rightarrow B$ ja $A \leftarrow B$. Tarkastellaan ensiksi ensimmäistä vaihtoehtoa, ja oletetaan että verkkostruktuuria vastaavat parametrit ovat

$$\begin{aligned} P(B=1 \mid A=1) &= 0.8, \\ P(B=1 \mid A=0) &= 0.01, \text{ ja} \\ P(A=1) &= 0.001. \end{aligned}$$

Nämä määrittelevät seuraavan yhteisjakauman:

$$\begin{aligned} &P(A=0,B=0) = P(A=0)P(B=0|A=0) = 999/1000 * 99/100 = 0.98901, \\ &P(A=0,B=1) = P(A=0)P(B=1|A=0) = 999/1000 * 1/100 = 0.00999, \\ &P(A=1,B=0) = P(A=1)P(B=0|A=1) = 1/1000 * 2/10 = 0.0002, \text{ja} \\ &P(A=1,B=1) = P(A=1)P(B=1|A=1) = 1/1000 * 8/10 = 0.0008. \end{aligned}$$

Toisaalta, jos otamme verkkostruktuurivaihtoehdon $A \leftarrow B$, ja asetamme struktuuria vastaavat parametrit seuraavasti:

$$\begin{aligned} P(B=1) &= 0.98921, \\ P(A=1 \mid B=1) &= 0.074143, \\ P(A=1 \mid B=0) &= 0.000202, \end{aligned}$$

voidaan yksinkertaisella laskutoimituksella osoittaa että syntyvä yhteistodennäköisyysjakauma on täsmälleen sama kuin ensimmäisessä tapauksessa. Esitettyjen verkkostruktuurien sanotaan olevan *ekvivalentteja*, mikä tarkoittaa sitä että niillä voidaan esittää täsmälleen sama joukko todennäköisyysjakaumia.

Kumpi kahdesta ekvivalentista verkkostruktuurista on sitten "oikea", kumpaa tulisi käyttää käytännön sovelluksissa? Puhtaasti pragmaattiselta kannalta katsottuna tällä kysymyksella ei ole merkitystä: koska syntyvät mallit esittävät täsmälleen samoja todennäköisyysjakaumia, ne käyttäytyvät täsmälleen samalla tavalla. Bayes-verkkojen soveltamisen kannalta on siis tässä mielessä yhdentekevää, käytetäänkö esimerkissämme verkkostruktuuria $A \rightarrow B$, joka vastannee useimpien käsitystä muuttujien kausaalisesta syy-seuraus-riippuvuussuhteesta, vai verkkostruktuuria $A \leftarrow B$. Bayes-verkkojen kaaret eivät siis välttämättä vastaa kausaalisia syy-seuraussuhteita.

Toisaalta on kuitenkin todettava, että vaikka kaarien kausaalisella tulkinnalla ei ole merkitystä Bayes-verkkojen soveltamisen kannalta, saattaa kausaalisuuden käsitteellä olla suuri käytännön merkitys Bayes-verkkojen rakentamisprosessissa: on helppo nähdä että muotoa $P(\text{seuraus} \mid \text{syy})$ olevien todennäköisyyksien arviointi on huomattavasti helpompaa kuin muotoa $P(\text{syy} \mid \text{seuraus})$ olevien todennäköisyyksien. Esimerkkitapauksessamme on varmaankin huomattavasti helpompi arvioida mikä on nenän valumisen todennäköisyys, annettuna että henkilöllä on flunssa, kuin arvioida todennäköisyyttä sille että henkilöllä on flunssa, annettuna että hänen nenänsä valuu. Samoin flunssan esiintymisen prioritodennäköisyyden arvioiminen on helpompaa kuin nenän valumisen prioritodennäköisyyden valuminen, koska jälkimmäisessä tapauksessa on arvioitava kaikki mahdolliset tilanteet joissa nenän valumista voi esiintyä. Jos malli siis konstruoidaan asiantuntijatietämystä käyttäen, on käytännössä järkevää pyrkiä rakentamaan verkko siten että solmujen väliset kaaret vastaavat muuttujien kausaalisia riippuvuussuhteita.

Vaikka kysymyksellä Bayes-verkkojen kaarien kausaalisesta tulkinnasta ei ole käytännön merkitystä jos malleja sovelletaan probabilistiseen päättelyyn, on kysymys sitäkin tärkeämpi ongelmakentän ominaisuuksien tilastollisen analyysin (data mining) kannalta. Bayes-verkkoja, joissa kaikki verkon kaaret kuvaavat kausaalisia riippuvuussuhteita, kutsutaan *kausaaliverkoiksi (causal networks)*. Bayes-verkkotutkimus onkin avannut täysin uusia lähtökohtia kausaalisuuden tutkimukselle sinänsä (katso esim. [106, 130, 47, 52]).

3.2 Päättely Bayes-verkoissa

Kuten luvussa 1.3 näimme, on probabilistisessa päättelyssä tavoitteena estimoida muotoa $P(\mathbf{S}_1 \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ olevia todennäköisyyksiä joko niin, että tuloksena on yhteisjakauma joukon $\mathbf{S}_1$ muodostaville muuttujille, tai niin että tuloksena saadaan n todennäköisintä arvokombinaatiota $\mathbf{s}_1$. Esittelemme seuraavassa kuinka tämä toteutetaan Bayes-verkkojen tapauksessa. Yksinkertaisuuden vuoksi rajoitumme aluksi esimerkissä 9 kuvattuun tapaukseen, jossa joukon $\mathbf{S}_1$ koko on yksi, eli tarkastelemme muotoa $P(X = x \mid \mathbf{S}_2 = \mathbf{s}_2, M, \theta)$ olevia todennäköisyyksiä. Yleistys monimutkaisempiin tapauksiin on suoraviivainen.

Esimerkissä 9 kuvattua ongelmakenttää voidaan mallintaa kuvassa 3.5 esitetyllä Bayes-verkolla, kun formalisoimme esimerkin kuvauksessa annetut oletukset seuraavasti: nenän vuotamiseen (muuttuja C) vaikuttavat muuttujat ovat A (onko flunssa vai ei) ja B (onko talossa kissa vai ei). Huonekaluista löytyvien raapimisjälkiin (muuttuja D) vaikuttaa se, onko talossa kissa, eivät muut mallissa käytetyt attribuutit.

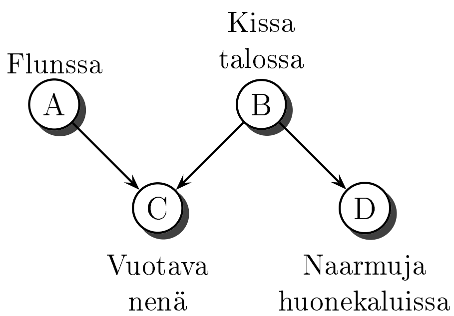
Mallimaailmassa on kaikkiaan 16 mahdollista tilaa, joita vastaavat yhteistodennäköisyydet saadaan laskettua esitetyn Bayes-verkkomallin avulla soveltamalla kaavaa (3.3), josta näemme että

$$P(A, B, C, D) = P(A)P(B)P(C \mid A, B)P(D \mid B).$$

Esimerkki 9 Kuvassa 3.5 esitettyyn Bayes-verkkoon liittyvän ongelmakentän kuvaus.

Olet matkalle vierailulle kun muistat äkkiä että et muistanut ottaa nenäliinoja mukaan vaikka kaupungissa on paha influenssaepidemia (arvioit tartunnan todennäköisyyden suuruudeksi 0.05). Olet myös allerginen kissoille, etkä muista varmasti onko isäntäperheellä kissa vai ei. Arvioit todennäköisyydeksi sille että talossa on kissa 0.05. Olet huolestunut koska sekä flunssa että kissa-allergia saa nenäsi yleensä vuotamaan, jolloin nenäliinoille olisi ollut tarvetta. Kokemuksesta tiedät että jos sinulla on joko flunssa, tai olet kosketuksissa kissojen kanssa, nenäsi alkaa vuotaa todennäköisyydellä 0.95, ja arvelet että jos molemmat pitäisivät yhtäaikaa paikkansa, nenän vuotamisen todennäköisyys nousisi arvoon 0.99. Toisaalta, ilman edellä mainittuja syitä on äärimmäisen epätodennäköistä (todennäköisyysarviosi on 0.001) että nenäsi vuotaisi.

Päätät perille tultuasi ensitöikseksi tarkkailla näkyykö huonekaluissa kissan tekemiä naarmuja, joita kokemuksesi syntyy todennäköisyydellä 0.99, mikäli talossa asuu kissa. On tietysti myös mahdollista että huonekaluista löytyy kissan tekemiä naarmuja, vaikka talossa ei kissaa olisikaan (huonekaluthan voivat olla vaikkapa ostettu käytettyinä), mutta arvioit tämän mahdollisuuden todennäköisyydeksi ainoastaan 0.1.



$$
\begin{aligned}
&P(A=1)=0.05\\
&P(B=1)=0.05\\
&P(C=1 \mid A=0, B=0)=0.001\\
&P(C=1 \mid A=1, B=0)=0.95\\
&P(C=1 \mid A=0, B=1)=0.95\\
&P(C=1 \mid A=1, B=1)=0.99\\
&P(D=1 \mid B=1)=0.99\\
&P(D=1 \mid B=0)=0.1
\end{aligned}
$$

Kuva 3.5: Esimerkin 9 Bayes-verkkokuvaus.

Vastaavat 16 yhteistodennäköisyyttä on esitetty taulukossa 3.2.

Taulukossa 3.2 esitettyjä yhteistodennäköisyyksiä hyväksikäyttäen voidaan laskea (priori)todennäköisyys sille että nenä alkaa vuotaa esimerkissä 9

A	B	C	D	P(A)	P(B)	P(C|A,B)	P(D|B)	P(A,B,C,D)
1	1	1	1	0.05	0.05	0.99	0.99	2.450E-03
1	1	1	0	0.05	0.05	0.99	0.01	2.475E-05
1	1	0	1	0.05	0.05	0.01	0.99	2.475E-05
1	1	0	0	0.05	0.05	0.01	0.01	2.500E-07
1	0	1	1	0.05	0.95	0.95	0.1	4.513E-03
1	0	1	0	0.05	0.95	0.95	0.9	4.061E-02
1	0	0	1	0.05	0.95	0.05	0.1	2.375E-04
1	0	0	0	0.05	0.95	0.05	0.9	2.138E-03
0	1	1	1	0.95	0.05	0.95	0.99	4.467E-02
0	1	1	0	0.95	0.05	0.95	0.01	4.513E-04
0	1	0	1	0.95	0.05	0.05	0.99	2.351E-03
0	1	0	0	0.95	0.05	0.05	0.01	2.375E-05
0	0	1	1	0.95	0.95	0.001	0.1	9.025E-05
0	0	1	0	0.95	0.95	0.001	0.9	8.123E-04
0	0	0	1	0.95	0.95	0.999	0.1	9.016E-02
0	0	0	0	0.95	0.95	0.999	0.9	8.114E-01
								1.000E+00

Taulukko 3.2: Kuvassa 3.5 esitetyn Bayes-verkon määrittelemä yhteistodennäköisyysjakauma.

esitetyssä mallissa:

$$\begin{aligned}
P(C=1) &= \sum_{a\in\{0,1\}}\sum_{b\in\{0,1\}}\sum_{d\in\{0,1\}} P(C=1, A=a, B=b, D=d) \\
&= 2.450\text{E-}03 + 2.475\text{E-}05 + 4.513\text{E-}03 + 4.061\text{E-}02 \\
&\quad +4.467\text{E-}02 + 4.513\text{E-}04 + 9.025\text{E-}05 + 8.123\text{E-}04 = 9.36\%.
\end{aligned}$$

Vastaavasti saamme naarmujen esiintymisen (priori)todennäköisyyden $P(D=1)$ arvoksi 14.45%. Flunssan (priori)todennäköisyys $P(A=1)$, samoin kuin kissan esiintymisen (priori)todennäköisyys $P(B=1)$ on 5%, mitkä nähdään suoraan käytetystä Bayes-verkkomallista.

Sana "priori" viittaa edellä todennäköisyyteen, joka saadaan Bayes-verkkomallista joko suoraan tai yksinkertaisen laskutoimituksen kautta, ilman että tehdyt havainnot vaikuttavat tilanteeseen. Probabilististen mallien käyttökelpoisuus perustuu suoraviivaiseen kalkyyliin, jonka avulla eri satunnaismuuttujien todennäköisyydet muuttuvat dynaamisesti sitä mukaa kun saamme ongelmakentästä uusia havaintoja. Kun havaitset esimerkiksi että nenäsi alkaa vuotaa ($C=1$), voit *päivittää* uskomuksesi siihen että sinulla on flunssa laskemalla flunssan ehdollisen todennäköisyyden *annettuna tehdyt havainnot*:

$$\begin{aligned}
P(A=1 \mid C=1) &= \frac{P(A=1,C=1)}{P(C=1)} \\
&= \frac{\sum_{b\in\{0,1\}}\sum_{d\in\{0,1\}} P(A=1,B=b,C=1,D=d)}{P(C=1)} \\
&= \frac{2.450\text{E-}03 + 2.475\text{E-}05 + 4.513\text{E-}03 + 4.061\text{E-}02}{9.36\text{E-}02} \\
&= 50.84\%.
\end{aligned}$$

Flunssan todennäköisyys siis yli kymmenkertaistui tehdyn havainnon seurauksena. Vastaavasti saamme todennäköisyydeksi sille että talossa on kissa, annettuna se havainto että nenäsi vuotaa,

$$P(B=1 \mid C=1) = 50.84\%,$$

ja todennäköisyydeksi sille että huonekaluissa on naarmuja

$$P(D=1 \mid C=1) = 55.25\%.$$

Edellä näimme että annettuna havainto, "nenä vuotaa", saamme sekä flunssan että kissan esiintymisen todennäköisyydeksi 0.5084 ja naarmujen esiintymisen todennäköisyydeksi 0.5525. Jos kysymme nyt mikä on todennäköisin mallimaailman tila, annettuna havainto $C = 1$, intuitiivisesti houkuttelevalta vaihtoehdolta tuntuisi asettaa kukin binäärimuuttuja itsenäisesti siihen arvoon jonka todennäköisyys on suurempi, jolloin vastaus olisi

$$P(A=1,B=1,C=1,D=1) = 2.450\text{E-}03.$$

Taulukosta 3.2 on kuitenkin helppo nähdä että muuttujien konfiguraatioavaruudesta löytyy kolme seikan $C = 1$ kanssa konsistenttia tilaa, joiden todennäköisyys on suurempi:

$$\begin{aligned}
P(A=0,B=1,C=1,D=1) &= 4.467\text{E-}02, \\
P(A=1,B=0,C=1,D=0) &= 4.061\text{E-}02, \text{ ja} \\
P(A=1,B=0,C=1,D=1) &= 4.513\text{E-}03.
\end{aligned}$$

Kokonaistodennäköisuuden maksimoivan tilan etsiminen on siis aivan eri ongelma kuin kunkin yksittäisen satunnaismuuttujan ehdollisen todennäköisyysjakauman laskeminen.

Kuten edellä esitetystä esimerkistä näimme, mikä tahansa Bayes-verkkoesitystä vastaavan yhteistodennäköisyysjakauman arvo saadaan laskettua

Selitysten poissulkeminen (explaining away)
Annettuna nenän vuotamisesta tehty havainto saamme sekä flunssan että kissan esiintymisen todennäköisyydeksi 50.84%, siis vain hieman yli 0.5. Ensi näkemältä voi tuntua siltä, että tehty havainto ei ole merkittävästi lisännyt tietoamme ongelmakentästä, mutta täytyy muistaa että sekä flunssan että kissan esiintymisen prioritodennäköisyys oli mallissamme hyvin pieni, vain 5%, joten näiden muuttujien todennäköisyys on yli kymmenkertaistunut. Toisaalta näemme että tehty havainto ei tässä vaiheessa auta meitä päättämään, kumpi seikka on nenän valumisen syy, flunssa vai kissa-allergia. Jos havaitsemme edelleen että talon huonekaluissa on kissan tekemiä naarmuja, satunnaismuuttujien posterioritodennäköisyydet päivittyvät jälleen:

$$\begin{aligned} P(B=1 \mid C=1, D=1) &\approx 91\% \\ P(A=1 \mid C=1, D=1) &\approx 13\% \end{aligned}$$

Tässä uudessa tilanteessa voimme olettaa että nenän vuotamisen aiheuttaja on melko varmasti kissa-allergia. Tämä esimerkki havainnollistaa todennäköisyysmalleissa esiintyvää havaintojen selityksiin liittyvää *"poisselittämis" (explaining away)*-ilmiötä: yhden vaihtoehtoisen selityksen todennäköisyyden kasvaminen vähentää automaattisesti vaihtoehtoisten selitysten uskottavuutta, vaikka selitykset eivät sinänsä olisikaan poissulkevia. Tämän intuitiivisesti luontevalta tuntuvan periaatteen on havaittu esiintyvän myös inhimillisessä päättelyssä [55, 104, 87].

kertomalla asiaankuuluvat Bayes-verkon parametrit keskenään. Näin ollen probabilistisen päättelyn suorittamiseksi ei tarvitse tallettaa taulukon 3.2 kaltaista todennäköisyystaulukkoa, vaan riittää tallettaa Bayes-verkon parametrit, joita on yleensä pieni määrä koko malliavaruuden kokoon verrattuna, kuten edellisessä luvussa näimme. Toisalta probabilistisessa päättelyssä käytettävien ehdollisten todennäköisyyksien laskeminen edellyttää yllä esitetyssä muodossaan monia tekijöitä sisältävien marginalisointisummien laskemista, mikä saattaa suurten Bayes-verkkojen tapauksessa olla käytännössä mahdotonta. Bayes-verkkojen määritelmään liittyvät ehdolliset riippumattomuusoletukset tarjoavat kuitenkin ratkaisun myös tähän ongelmaan, kuten seuraava esimerkki osoittaa.

Olettakaamme että tehtävänä on laskea kuvassa 3.5 esitettyyn Bayes-verkkomalliin liittyen ehdollinen todennäköisyys $P(B = 1 \mid C = 1, D = 1)$.

Todennäköisyyslaskennan perusaksioomia käyttäen näemme helposti että

$$P(B=1 \mid C=1, D=1) = \frac{P(d^+ \mid b^+, c^+)P(c^+ \mid b^+)P(b^+)}{P(c^+, d^+)},$$

missä arvoasetuksesta $X=1$ on käytetty lyhennettä x^+ (asetuksesta $X=0$ käytetään jatkossa lyhennettä x^-). Bayes-verkkoihin liittyvän ehdollisen riippumattomuuden määritelmän perustella muuttuja D on ehdollisesti riippumaton muuttujasta C, annettuna muuttujan B arvo, joten saamme

$$P(b^+ \mid c^+, d^+) = \frac{P(d^+ \mid b^+)P(c^+ \mid b^+)P(b^+)}{P(c^+, d^+)}.$$

Toisaalta, koska

$$P(b^+ \mid c^+, d^+) = \frac{P(b^+ \mid c^+, d^+)}{P(b^+ \mid c^+, d^+) + P(b^- \mid c^+, d^+)},$$

saamme että

$$\begin{aligned} P(b^+ \mid c^+, d^+) &= \frac{\frac{P(d^+|b^+)P(c^+|b^+)P(b^+)}{P(c^+,d^+)}}{\frac{P(d^+|b^+)P(c^+|b^+)P(b^+)}{P(c^+,d^+)} + \frac{P(d^+|b^-)P(c^+|b^-)P(b^-)}{P(c^+,d^+)}} \\ &= \frac{P(d^+ \mid b^+)P(c^+ \mid b^+)P(b^+)}{P(d^+ \mid b^+)P(c^+ \mid b^+)P(b^+) + P(d^+ \mid b^-)P(c^+ \mid b^-)P(b^-)}. \end{aligned}$$

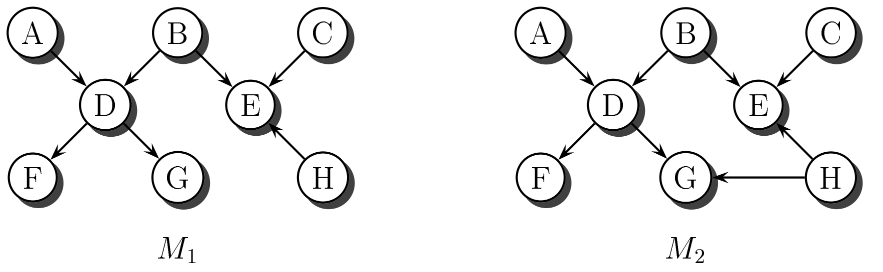
Näemme että koska nimittäjissä esiintyvä termi $P(c^+, d^+)$ ei riipu muuttujan B arvosta (eli on B:n ehdollisen todennäköisyysjakauman suhteen vakio), sen arvoa ei tarvitse laskea, vaan se kumoutuu pois! Haluamamme ehdollisen todennäköisyyden saamiseksi riittää siis laskea kaksi tuloa,

$$\begin{gathered} P(d^+ \mid b^+)P(c^+ \mid b^+)P(b^+) \text{ ja} \\ P(d^+ \mid b^-)P(c^+ \mid b^-)P(b^-), \end{gathered}$$

jotka molemmat sisältävät vain Bayes-verkkomalliin talletettuja parametreja.

Bayes-verkoissa tapahtuvan probabilistisen päättelyn havainnollistamiseksi ajatellaan kuvan 3.5 Bayes-verkkoa tietokoneohjelma-arkkitehtuurina, jossa kukin muuttuja X vastaa moduulia, jonka tehtävänä on laskea ehdollinen todennäköisyysjakauma $P(X \mid \mathbf{S}_2 = \mathbf{s}_2)$, ja oletetaan että Bayes-verkon parametrit on talletettu siten, että kukin moduuli X näkee muotoa $P(X \mid F(X))$ olevat parametrit. Kuten edellä näimme, yllä esitetyssä tapauksessa moduuli B tarvitsee seuraavat luvut voidakseen määrätä ehdollisen todennäköisyysjakauman $P(b^+ \mid c^+, d^+)$:

$$P(d^+ \mid b^+), P(c^+ \mid b^+), P(b^+), P(d^+ \mid b^-), P(c^+ \mid b^-), \text{ ja} P(b^-).$$



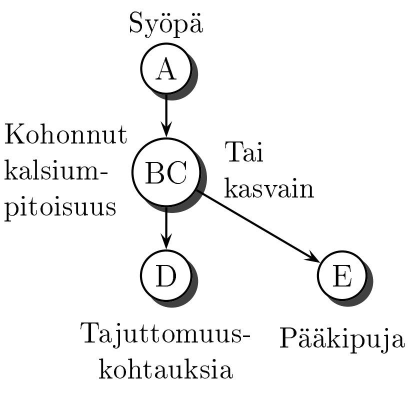
Kuva 3.6: Yksipolkuinen (M_1) ja monipolkuinen (M_2) Bayes-verkkostruktuuri.

Luvut $P(b^+)$ ja $P(b^-)$ on talletettu paikallisesti moduuliin B Bayes-verkkomallin parametreina. Koska $P(c^+ \mid b^+) = P(c^+ \mid a^-, b^+) + P(c^+ \mid a^+, b^+)$, moduli C voi laskea tämän tarvittavan luvun, samoin kuin luvun $P(c^+ \mid b^-)$, nopeasti ja tehokkaasti käyttäen vain paikallisesti talletettua informaatiota (moduuliin C talletettuja Bayes-verkon parametreja). Moduuli D voi vastaavasti laskea arvot $P(d^+ \mid b^+)$ ja $P(d^+ \mid b^-)$. Laskeakseen todennäköisyyden $P(b^+ \mid c^+, d^+)$ modulin B riittää siis lähettää tarvittavien lukujen laskupyyntö moduleille C ja D, ja kertoa saadut vastaukset edellä esitetyllä tavalla halutun ehdollisen todennäköisyysjakauman tuottamiseksi.

Edellä kuvattua viestinvälitysarkkitehtuuriin perustuva menetelmää ehdollisten todennäköisyyksien laskemiseksi voidaan soveltaa kaikissa *yksipolkuisissa (singly connected)* Bayes-verkoissa: yksipolkuisissa verkoissa ei minkään kahden solmun välillä ole kuin yksi suuntaamaton polku, missä suuntaamaton polku tarkoittaa verkon kaaria seuraavaa reittiä, joka voi kulkea suunnattuja kaaria kumpaan suuntaan tahansa. Kuvassa 3.6 yksipolkuinen Bayes-verkkostruktuuri M_1 muutetaan *monipolkuiseksi (multi-connected)* yhdistämällä solmut H ja G, jolloin syntyy kaksi (suuntaamatonta) polkua esimerkiksi solmujen B ja G välille.

Yksipolkuisissa Bayes-verkoissa voidaan ehdollisten todennäköisyysjakaumien laskeminen suorittaa erittäin tehokkaasti, lineaarisessa ajassa Bayes-verkon parametrien lukumäärään nähden. Algoritmin yksityiskohtainen kuvaus löyttyy esimerkiksi alan perusoppikirjoista [105, 97]. Maksimitodennäköisyyskonfiguraation etsiminen voidaan niinikään suorittaa samankaltaisella algoritmilla.

Edellä kuvatunlaiseen viestinvälitystekniikkaan perustuvat tehokkaat probabilistiset päättelyalgoritmit perustuvat siihen tosiseikkaan, että yksipolkuisissa verkoissa kukin solmu osittaa verkon kahteen erilliseen osaan, jotka voidaan käsitellä erikseen, ja sama pätee rekursiivisesti jokaiselle verkon solmulle. Monipolkuisissa verkoissa menetelmä ei valitetavasti toimi. Itse asiassa voidaan teoreettisesti osoittaa, että muotoa $P(X = 1 \mid \mathbf{S}_2 = \mathbf{s}_2)$ olevien



$P(A = 1)$	$=$	0.20
$P(BC = 00 \mid A = 1)$	$=$	$0.20 \times 0.80 = 0.16$
$P(BC = 01 \mid A = 1)$	$=$	$0.20 \times 0.20 = 0.04$
$P(BC = 10 \mid A = 1)$	$=$	$0.80 \times 0.80 = 0.64$
$P(BC = 11 \mid A = 1)$	$=$	$0.80 \times 0.20 = 0.16$
$P(BC = 00 \mid A = 0)$	$=$	$0.80 \times 0.95 = 0.76$
$P(BC = 01 \mid A = 0)$	$=$	$0.80 \times 0.05 = 0.40$
$P(BC = 10 \mid A = 0)$	$=$	$0.20 \times 0.95 = 0.19$
$P(BC = 11 \mid A = 0)$	$=$	$0.20 \times 0.05 = 0.01$
$P(D = 1 \mid BC = 11)$	$=$	0.80
$P(D = 1 \mid BC = 01)$	$=$	0.80
$P(D = 1 \mid BC = 10)$	$=$	0.80
$P(D = 1 \mid BC = 00)$	$=$	0.05
$P(E = 1 \mid BC = 01)$	$=$	0.80
$P(E = 1 \mid BC = 11)$	$=$	0.80
$P(E = 1 \mid BC = 00)$	$=$	0.60
$P(E = 1 \mid BC = 10)$	$=$	0.60

Kuva 3.7: Kuvan 3.3 Bayes-verkko muutettuna yksipolkuiseksi verkoksi.

ehdollisten todennäköisyyksien laskeminen [28] tai edes niiden approksimointi [29, 115], ja myös maksimitodennäköisyystilojen etsiminen [124], on erittäin vaikea ongelma monipolkuisten verkkojen tapauksessa: jos näille ongelmille olisi aina polynomisessa ajassa toimiva ratkaisumenetelmä, menetelmää voitaisiin käyttää ratkaisemaan kaikki ns. NP-täydelliset päätösongelmat polynomisessa ajassa, minkä ei yleisesti uskota olevan mahdollista. On kuitenkin syytä muistaa näiden teoreettisten tulosten käsittelevän pahimman tapauksen (worst-case) skenaariota: yhdenkin eksponentiaalisen ajan vaativan tapauksen löytäminen riittää, vaikka kaikki muut tapaukset voitaisiin käsitellä polynomisessa ajassa.

Yleisin lähestymistapa probabilistisen päättelyn toteuttamiseksi monipolkuisissa Bayes-verkoissa on yrittää muuttaa monipolkuinen verkko yksipolkuiseksi, jolloin päättely on tehokasta kuten edellä näimme. Yleisimmin käytetty transformointitekniikka on muuttujien *ryhmittely (clustering)*. Tarkastellaan esimerkiksi kuvassa 3.3 esitettyä Bayes-verkkoa. Ryhmittelemällä muuttujat uudelleen siten että ne muodostavat neljä ryhmää $\{A\}$, $\{B, C\}$, $\{D\}$, ja $\{E\}$ näemme, että jos sulautamme muuttujat B ja C uudeksi muuttujaksi BC, syntyvä Bayes-verkko on yksipolkuinen (katso kuva 3.7). Koska muuttujilla B ja C oli kummallakin 2 mahdollista arvoa, on yhdistetyllä muuttujalla BC 4 mahdollista arvoa; merkitään näitä arvoja symboleilla 00,01,10 ja 11. Syntyneeseen uuteen verkkoon liittyvät todennäköisyydet on helppo laskea alkuperäisen verkon parametriarvoista: muotoa $P(D \mid BC)$ olevat todennäköisyydet vastaavat suoraan muotoa $P(D \mid B, C)$ olevia todennäköisyyksiä, ja ehdollisen riippumattomuuden määritelmän mukaan näemme että $P(E \mid BC) = P(E \mid C)$ ja $P(BC \mid A) = P(B \mid A)P(C \mid A)$.

Edellä kuvatussa yksinkertaisessa tapauksessa oli helppo nähdä kuinka annettu monipolkuinen verkko voitiin muuttaa yksipolkuiseksi yhdistämällä kaksi muuttujaa, mutta monimutkaisemmissa verkoissa tehtävä on huomattavasti vaikeampi eivätkä ratkaisut ole aina yhtä yksinkertaisia. Käytännön kannalta tärkeä kysymys on voidaanko tämä transformaatioprosessi automatisoida — onko mahdollista kehittää algoritmi joka tuottaa mistä tahansa annetusta monipolkuisesta Bayes-verkosta yksipolkuisen Bayes-verkon polynomisessa ajassa? Vastaus on hieman yllättäen kyllä: tällaisia algoritmeja on kehitetty itse asiassa lukuisia (katso esim. [105, 97, 60, 18]). Useimmat esitetyistä menetelmistä perustuvat kaksivaiheiseen algoritmiin, jossa ensimmäisessä vaiheessa annettu Bayes-verkko *moralisoidaan*[2] yhdistämällä solmut joilla on yhteinen seuraaja, ja poistamalla kaarien suunnat. Menetelmän toisessa vaiheessa syntynyt suuntaamaton verkko *kolmioidaan (triangulation)* lisäämällä verkkoon suuntaamattomia kaaria kunnes kaikissa verkossa sijaitsevien yli kolmen kaaren pituisissa silmukoissa on joitakin kahta silmukan solmua yhdistävä kaari. Jos kolmioidussa verkossa esiintyvien maksimaalisten *klikkien* (suurimpien sellaisten verkosta löytyvien solmujoukkojen joissa kaikki solmut on kytketty toisiinsa) solmut yhdistetään yhdeksi muuttujaksi, niin syntyvien yhdistettyjen muuttujien avulla on mahdollista luoda alkuperäisen monipolkuisen Bayes-verkon kanssa ekvivalentti yksipolkuinen verkko. Syntyvää yksipolkuista verkkoa kutsutaan *liittymäpuuksi (junction tree)*. Nimitys johtuu siitä, että liittymäpuun solmusta voi olla kaari toiseen solmuun vain silloin jos solmuja vastaavien klikkien leikkaus ei ole tyhjä. Liittymäpuiden konstruoimisessa käytettävä algoritmi on melko monimutkainen, eikä sen esittämän tämän raportin yhteydessä ole mielekästä. Algoritmin ksityiskohdat selviävät mm. lähteistä [105, 97, 60, 18].

Liittymäpuihin perustuvat menetelmät antavat siis algoritmin probabilistisen päättelyn suorittamiseksi monipolkuisissa Bayes-verkoissa. Koska syntyvä liittymäpuu on puu, siis yksipolkuinen Bayes-verkko, voidaan probabilistista päättelyä suorittaa syntyneessä verkossa lineaarisessa ajassa verkon parametrien lukumäärään nähden. Tässä yhteydessä on kuitenkin huomattava että syntyvän liittymäpuun parametrien lukumäärä ei ole sama kuin alkuperäisen monipolkuisen Bayes-verkon parametrien lukumäärä, vaan se voi olla paljon suurempi, pahimmassa tapauksessa eksponentiaalisen suuri alkuperäiseen lukumäärään verrattuna. Voidaan itse asiassa osoittaa [142] että sellaisen kolmioinnin löytäminen joka tuottaa minimaalisen määrän liittymäpuuparametreja on NP-täydellinen ongelma. Alkuperäisen ongelman vaikeutta ei siis voi paeta tiedon esitystapaa muuttamalla (yhdistämällä muuttujia), mikä on tietenkin muutenkin itsestään selvää. Becker ja Geiger [7] ovat esittä-

[2] Moralisointi-termi (moralization) johtuu siitä että prosessissa yhdistetään solmut (vanhemmat) joilla on yhteinen seuraaja (lapsi).

neet "hajota-ja-hallitse"-periaatteeseen perustuvan algoritmin, jonka tuottama kolmiointi on aina vähintään tietyn vakioetäisyyden päässä optimaalisesta. Kjærulff [66] on tutkinut stokastisten optimointmenetelmien soveltamista tämän ongelman ratkaisemisessa.

Käytännön sovelluksista saadut kokemukset osoittavat että liittymäpuihin perustuvat algoritmit toimivat yleensä varsin hyvin — useimmat luvussa 4.2 kuvatuista ohjelmistoista perustuvat tähän lähestymistapaan. Pearlin *ehdollistamismenetelmä (conditioning)* [104] on läheistä sukua solmujen yhdistämiseen perustuville tekniikoille: ehdollistamisessa monipolkuista verkkoa ei kuitenkaan eksplisiittisesti muuteta yksipolkuiseksi verkoksi, vaan ongelmia aiheuttavat polut poistetaan implisiittisesti kiinnittämällä (ehdollistamalla) polulla sijaitsevia muuttujia johonkin arvoon. Probabilistisen päättelyn toteuttamiseksi voidaan myös luopua laskennan deterministisyydestä, ja käyttää stokastisia päättelyalgoritmeja. Teoksessa [93] kuvataan kuinka kuinka tämän ongelman ratkaisemisessa voidaan käyttää Boltzmannin koneena tunnettua stokastista neuroverkkoarkkitehtuuria. Muita vaihtoehtoisia lähestymistapoja probabilistisen päättelyn toteuttamiseksi monipolkuisissa Bayes-verkoissa on kuvattu lähteissä [18, 56, 31, 116, 85, 143].

3.3 Bayes-verkkojen rakentaminen

Kuten luvussa 3.1 näimme, Bayes-verkkomallien komponenteille voidaan antaa intuitiivisesti selkeä semanttinen tulkinta: Bayes-verkkostruktuuri kuvaa ongelmakentän kuvaamisessa käytettyjen attribuuttien välisiä riippuvuuksia, Bayes-verkon parametreina käytetyt ehdolliset todennäköisyydet taas riippuvuuksien voimakkuuksia. Tämäntyyppinen tieto on usein helposti saatavilla ongelmakentän asiantuntijoilta, jolloin Bayes-verkkojen konstruointi on helppoa. Bayes-verkkokehittimiä (katso luku 4.2) käyttäen Bayes-verkkojen soveltaminen ei edellytä syvempää Bayes-verkkoteorian ymmärtämistä, jolloin asiantuntijat voivat myös konstruoida Bayes-verkkomalleja suoraan itse, ja siirtyä samalla testaamaan kehitettyjen mallien toimivuutta. Eric Horvitzin mukaan[3] United Airlinesin insinöörit testasivat ensimmäisiä itse kehittämiään Bayes-verkkomalleja jo muutaman tunnin kuluttua siitä kun Bayes-verkkomallien käsite ja niiden konstruoimiseen soveltuvat kehittimet oli heille esitelty.

Toisaalta, kuten luvussa 1.2 todettiin, bayesiläisen mallinnuksen yksi suurimpia vahvuuksia on se, kuinka lähestymistapa mahdollistaa asiantuntijatietämyksen yhdistämisen tilastolliseen oppimiseen esimerkkiaineistosta. Mallien oppiminen jaettiin edellä kahteen vaiheeseen: mallistruktuurin oppiminen

[3]Katso URL: http://www.auai.org/BN-Testimonial.html

ja malliparametrien oppiminen. Kuvaamme seuraavassa kuinka laskea oppimiskriteeri eri Bayes-verkkostruktuureille, ja kuinka kiinnittää Bayes-verkon parametrit, annettuna struktuuri. Samassa yhteydessä selvitämme kuinka asiantuntijatietämys voidaan priorijakaumia käyttäen ottaa huomioon oppimisprosessissa.

3.3.1 Verkkostruktuurin oppiminen

Luvussa 1.2.2 näimme että bayesiläisessä oppimisessa mallistruktuurit evaluoidaan vertailemalla mallien posterioritodennäköisyyksiä

$$P(M \mid \mathcal{D}) \propto P(\mathcal{D} \mid M)P(M),$$

ja luvussa 1.2.4 näimme kuinka asiantuntijatietämys voidaan integroida oppimisprosessiin priorijakaumien kautta. Mallistruktuurin oppimisen päämääränä on siis iteratiivisen etsintäalgoritmin kautta etsiä mallistruktuuri M joka maksimoi tämän todennäköisyyden. Bayes-verkkojen tapauksessa etsintäavaruus on valtava: erilaisten syklittömien suuntamattomien verkkojen lukumäärä saadaan lähteessä [114] esitetystä rekursioyhtälöstä, josta näemme että jo kymmenellä solmulla saadaan aikaan noin 4.2×10^{18} erilaista Bayes-verkkostruktuuria! Tässä valossa ei olekaan lainkaan yllättävää että Bayes-verkkojen oppimisongelman on osoitettu olevan NP-kova ongelma [25]. On kuitenkin jälleen kerran korostettava, että kyseessä on pahimman tapauksen analyysi, joka ei kerro mitään ongelman keskimääräisestä vaikeudesta. Bayes-verkkojen oppimisessa onkin sovellettu hyvällä menestyksellä tunnettuja heuristisia etsintäalgoritmeja — viitteitä käytettyihin algoritmeihin löytyy esimerkiksi teoksista [48, 122].

Jos ongelmakentästä ei ole saatavilla priori-informaatiota, on järkevää olettaa eri struktuureille tasainen priorijakauma, jolloin todennäköisyyttä $P(M)$ voidaan pitää vakiona, ja käyttää oppimiskriteerinä yksinomaan kokonaisuskottavuutta $P(\mathcal{D} \mid M)$. Mikäli priori-informaatiota on saatavilla, voidaan tätä tietämystä käyttää hyväksi priorijakauman $P(M)$ määräämisessä. Koska mahdollisten Bayes-verkkojen määrä on kuitenkin tavattoman suuri, ei ole käytännössä mahdollista määrätä prioritodennäköisyyttä $P(M)$ erikseen kullekin Bayes-verkkostruktuurille. Priori-informaatiota voidaan kuitenkin käyttää hyväksi yhden (tai muutaman) ns. prioriverkon konstruoimisessa. Prioriverkoilla tarkoitetaan verkkostruktuureja joiden prioritodennäköisyyden arvellaan olevan hyvin suuri (verrattuna muihin verkkostruktuureihin). Prioriverkkoja voidaan käyttää hyödyksi mallistruktuurin etsimisessä esimerkiksi heurististen etsintäalgoritmien aloituspisteinä, tai varsinaisen priorijakauman estimoinnissa lähteessä [50] esitetyllä tavalla.

Bayes-verkkojen kokonaisuskottavuuden $P(\mathcal{D} \mid M)$ laskemiseksi on tehtävä joitakin teknisiä oletuksia, joista tärkein koskee parametrien priorijakaumaa $P(\theta \mid M)$. Kuten luvussa 3.1 näimme, Bayes-verkkostruktuuriin liitettävät parametrit θ ovat muotoa $P(X \mid F(X))$ olevia ehdollisia todennäköisyyksiä, missä $F(X)$ on muuttujan X edeltäjien joukko. Numeroidaan joukon $F(X)$ kaikki mahdolliset arvokombinaatiot, ja oletetaan että niitä on q_i kappaletta. Merkinnällä θ_{ijk} tarkoitamme seuraavassa parametria $P(X_i = k \mid F(X_i) = j)$. Jos oletamme nyt että priorijakauma $P(\theta_{ij} \mid M)$ on ns. *Dirichlet-jakauma* (ks. esim. [39]) *hyperparametrein*[4] $N'_{ij1}, \ldots, N'_{ijr_i}$, niin voidaan osoittaa [27, 50] että kokonaisuskottavuus voidaan laskea kaavalla (3.4).

Kaavassa (3.4) esiintyvät tilastolliset tunnusluvut N_{ijk} on helppo laskea opetusjoukosta $\mathcal{D}$, ja kokonaisuskottavuus voidaan laskea ajassa $O(Nn^2r)$, missä N on joukon $\mathcal{D}$ koko, n muuttujien lukumäärä, ja r suurin mahdollinen muuttujien arvojen lukumäärä. Käyttäjän tehtäväksi jää määrätä hyperparametrit N'_{ijk}. Mikäli prioritietämystä ei ole, voidaan käyttää tasaista priorijakauma joka saadaan asettamalla $N'_{ijk} = 1$ kaikilla i:n j:n, ja k:n arvoilla. Jos prioritietämystä on saatavilla, antaa kaava (3.4) vihjeen siitä kuinka asiantuntijatietämys voidaan koodata hyperparametreiksi N'_{ijk}: hyperparametrit ja tilastolliset tunnusluvut N_{ijk} ovat käytettävässä kaavassa täysin identtisessä asemassa, joten on helppo päätellä, että jos käyttäjä asettaa hyperparametrin N'_{ijk} arvoksi esimerkiksi viisi, on tämän asetuksen vaikutus kokonaisuskottavuuteen sama kuin jos käyttäjä olisi lisännyt opetusjoukkoon $\mathcal{D}$ viisi esimerkkiä, joissa kussakin muuttujan X_i arvo olisi k, ja muuttujan edeltäjät $F(X_i)$ olisivat arvokombinaatiossa j. Käyttäjä voi siis muotoilla henkilökohtaisen näkemyksensä muuttujan X_i arvojen jakaumasta muuttamalla hyperparametreja N'_{ijk}, ja hyperparametrien määräämisessä käytetyn arvoalueen skaalaa muuttamalla saadaan vaihdeltua prioritietämyksen vaikutuksen suhdetta opetusjoukkoon vaikutukseen verrattuna. Tätä lähestymistapaa kutsutaan *ekvivalentin otoskoon (equivalent sample size)* menetelmäksi: asiantuntija voi painottaa tietämyksensä merkitystä ajattelemalla sitä "priorioptusjoukkona", joukkona opetusjoukon ulkopuolisia tapauksia joista asiantuntija on oman kokemusperäisen tietämyksensä muodostanut.

Kaavassa (3.4) esitetty menetelmä kokonaisuskottavuuden laskemiseksi pätee diskreeteille muuttujille, kun muuttujien arvot otosjoukossa $\mathcal{D}$ oletetaan jakautuneiksi multinomijakauman mukaan. Vastaavanlainen tulos voidaan kuitenkin itse asiassa johtaa hyvin laajalle joukolle erilaisia jakaumia

[4] Nimitys "hyperparametri" johtuu siitä että kyseessä on "toisen kertaluvun" parametri, yhteistodennäköisyysjakauman määrittelemisessä käytettävän parametrin jakauman määrittelemisessä käytetty parametri.

Kokonaisuskottavuuden laskeminen:

$$P(\mathcal{D} \mid M) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}, \tag{3.4}$$

missä

- Γ on ns. Gamma-funktio, tavallisen kertomafunktion yleistys myös muille kuin kokonaisluvuille (minkä ansiosta hyperparametrien ei tarvitse välttämättä olla kokonaislukuarvoisia),
- n on muuttujien lukumäärä,
- q_i on Solmun X_i edeltäjien mahdollisten arvokombinaatioiden lukumäärä.
- r_i on muuttujan X_i arvojen lukumäärä,
- N_{ijk} on niiden tapausten lukumäärä joukossa $\mathcal{D}$, missä muuttujan X_i arvo on k, ja muuttujan X_i edeltäjät $F(X_i)$ ovat arvokombinaatiossa j,
- N'_{ijk} on parametria θ_{ijk} vastaavan priorijakauman määräämisessä käytetty hyperparametri,
- $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, ja
- $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$.

(ns. eksponentiaaliselle perheelle), myös jatkuville jakaumille, ja jatkuvien ja diskreettien muuttujien muodostamille sekajakaumille [10]. Esimerkki Bayes-verkon kokonaisuskottavuuden laskemisesta jatkuvien jakaumien tapauksessa löytyy lähteestä [40].

Yllä kuvattu lähestymistapa perustuu Bayes-verkkostruktuurien vertailuun niiden posterioritodennäköisyyden (tai käytännössä yleensä kokonaisuskottavuuden) perusteella. Luvussa 3.1 esitetty Bayes-verkkojen määritelmä tarjoaa myös vaihtoehtoisen lähestymistavan tähän ongelmaan: koska kukin Bayes-verkkostruktuuri vastaa tiettyä joukkoa riippumattomuusoletuksia, voidaan opetusjoukosta ensin tilastollisesti päätellä ongelmakentässä vallitsevat riippumattomuudet, ja konstruoida sitten pääteltyjä riippumattomuuksia vastaava Bayes-verkkostruktuuri. Tähän lähestymistapaan perustuvia algoritmeja on esitetty esimerkiksi lähteessä [130]. Täydellisten riippumattomuuksien sijaan voidaan yrittää tilastollisesti päätellä muuttujien osittaisia järjestyksiä siten, että syntyvässä Bayes-verkossa muuttujasta X voi olla kaari muuttujaan Y vain jos X on Y:n edeltäjä kiinnitetyssä järjestyksessä. Tilastollisesti pääteltyjä osittaisia järjestyksiä voidaan myös yhdistää yllä esitettyyn MAP-metriikkaan perustuviin oppimisalgoritmeihin [125]. Osittaisen järjestyksen määräämiseksi voidaan tilastollisten testien sijaan käyttää tietenkin myös priori-informaatiota. Itse asiassa monet MAP-metriikkaa soveltavat oppimisalgoritmit, kuten esimerkiksi Cooperin ja Herskovitsin K2-algoritmi [27], edellyttävät tällaisen osittaisen järjestyksen määräämistä ennen oppimisprosessin alkua.

3.3.2 Bayes-verkon parametrien oppiminen

Suurimman todennäköisyyden parametrit Kun Bayes-verkkostruktuuri M on kiinnitetty, on seuraava vaihe mallin oppimisessa mallin parametrien θ valinta. Kuten luvussa 1.2.3 näimme, bayesiläisessä lähestymistavassa parametrit kiinnitetään MAP-arvoonsa, siis arvoon $\hat{\theta}$ joka maksimoi posterioritodennäköisyyden $P(\theta \mid \mathcal{D}, M)$:

$$\hat{\theta} = \arg\max_{\theta} P(\theta \mid \mathcal{D}, M).$$

Samojen teknisten oletusten vallitessa, joita käytettiin edellisessä luvussa esitetyn kokonaisuskottavuuden kaavan (3.4) johtamisessa, voidaan osoittaa että malliparametrien posterioritodennäköisyys maksimoituu asettamalla

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + N'_{ijk} - 1}{N_{ij} + N'_{ij} - r_i}, \tag{3.5}$$

missä merkinnät ovat kuten edellä kaavassa (3.4). MAP-parametriarvojen laskemisessa ei siis tarvita iteratiivista oppimisalgoritmia, vaan ne voidaan määrätä suoraan käyttäen kaavaa (3.5)!

Suurimman uskottavuuden parametrit On tärkeää huomata, että yllä annettu laskukaava MAP-parametrien määräämiseksi poikkeaa klassisesta tilastotieteestä tutusta *suurimman uskottavuuden (maximum likelihood, ML)* parametriasetuksesta $\tilde{\theta}$,

$$\tilde{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}} = \frac{N_{ijk}}{\sum_{k=1}^{r_i} N_{ijk}}, \tag{3.6}$$

joka maksimoi datan uskottavuuden $P(\mathcal{D} \mid \theta, M)$. Koska

$$P(\theta \mid \mathcal{D}, M) \propto P(\theta \mid M) P(\mathcal{D} \mid \theta, M),$$

näemme että ML- ja MAP-parametrit ovat samat siinä tapauksessa, että priorijakauma $P(\theta \mid M)$ on tasainen, ja siis vakio. Suurimman uskottavuuden parametrit saadaan siis erikoistapauksena kaavasta (3.5), jos käytämme tasaisia priorijakaumia, eli asetamme kaikkien hyperparametrien N'_{ijk} arvoksi yksi.

Odotusarvoparametrit Parametrien θ asettaminen MAP-arvoihinsa on bayesiläisen perusidelogian mukainen vastaus kysymykseen kuinka kiinnittää parametrit annettuna mallistruktuuri. Mikäli syntyvää mallia (M, θ) kuitenkin käytetään probabilistisessa päättelyssä (eikä siis ainoastaan ongelmakentän analysoinnissa), näemme hiukan yllättäen että MAP-parametreja käyttävälle mallille löytyy vaihtoehto, joka tuottaa tarkempia ennustuksia probabilistisessä päättelyssä. Olkoon nimittäin $P(X_1, \ldots, Xn \mid \theta, M)$ mallin (M, θ) tuottama jakauma. Koska emme tiedä varmasti mikä mallin θ tuottama jakauma vastaa ongelmakentän todellista todennäköisyysjakaumaa, arvioimme eri mallien hyvyyttä niiden posterioritodennäköisyyden $P(\theta \mid \mathcal{D}, M)$ mukaan käyttämällä apuna ongelmakentästä saatua otosta $\mathcal{D}$. Yllä esitetyssä MAP-lähestymistavassa valittiin siis parametrit, jotka maksimoivat tämän todennäköisyyden. Tarkempi jakauma saataisiin kuitenkin aikaan ottamalla esimerkiksi kaksi tässä mielessä parasta mallia θ_1 ja θ_2, ja muodostamalla niiden painotettu keskiarvo:

$$P(X_1, \ldots, Xn \mid \theta_1, M) P(\theta_1 \mid \mathcal{D}, M) + P(X_1, \ldots, Xn \mid \theta_2, M) P(\theta_2 \mid \mathcal{D}, M).$$

Seuraamalla tätä päättelyketjua eteenpäin, näemme että optimaalisessa tapauksessa meidän pitäisi käyttää *kaikkia (äärettömän monia) mahdollisia parametriasetuksia*, jolloin saamme jakauman

$$\int P(X_1, \ldots, Xn \mid \theta, M) P(\theta \mid \mathcal{D}, M) d\theta = P(X_1, \ldots, Xn \mid M).$$

Voidaan kuitenkin osoittaa että syntynyt jakauma on identtinen jakauman kanssa, joka saadaan käyttäen yhtä mallia jossa kukin parametriarvo on kiinnitetty odotusarvoonsa $E(\theta_{ijk} \mid \mathcal{D}, M)$. Odotusarvomallin käyttäminen vastaa siis, paradoksaalista kyllä, tilannetta jossa emme kiinnitä parametriarvoa lainkaan vaan käytämme kaikkia äärettömän monia parametriasetuksia! Odotusarvot saadaan laskettua kaavasta

$$\hat{\theta}_{ijk} = E(\theta_{ijk} \mid \mathcal{D}, M) = \frac{N_{ijk} + N'_{ijk}}{N_{ij} + N'_{ij}}. \tag{3.7}$$

Odotusarvoihin perustuvien mallien on empiirisesti havaittu tuottavan paljon tarkempia estimaatteja ongelmakentän todennäköisyysjakaumasta, etenkin kun käytettävissä olevan opetusjoukon $\mathcal{D}$ koko on pieni [74, 75, 133].

Kolme laskukaavaa malliparametrien määräämiseksi:

- Suurimman todennäköisyyden parametrit (3.5):

$$\theta_{ijk} = \frac{N_{ijk} + N'_{ijk} - 1}{N_{ij} + N'_{ij} - r_i}.$$

- Suurimman uskottavuuden parametrit (3.6):

$$\theta_{ijk} = \frac{N_{ijk}}{N_{ij}}.$$

- Odotusarvoparametrit (3.7):

$$\theta_{ijk} = \frac{N_{ijk} + N'_{ijk}}{N_{ij} + N'_{ij}}.$$

3.4 Bayes-verkkojen variaatioita

3.4.1 Päätösverkot (influence diagrams)

Luvussa 1.4 näimme kuinka probabilistisessa päätelyssä tuotettuja ehdollisia todennäköisyysjakaumia voidaan käyttää hyväksi päätöksentekoprosessissa päätösteoriana tunnetun formalismin kautta. *Päätösverkko (influence diagrams)* on Bayes-verkko, jossa päätöksentekoon liittyvät vaihtoehdot koodataan erityisinä päätössolmuina, ja päätössolmujen eri arvoihin liittyvä odotettavissa olevan hyödyn määrä lasketaan suoraan probabilistisen päättelyn yhteydessä (katso esim. [123, 101]). Päätössolmujen käsittelymahdollisuus on liitetty mukaan useisiin Bayes-verkkoja manipuloiviin ohjelmistoihin (katso luku 4.2).

3.4.2 “Noisy-or”-malli

Kuten luvussa 3.2 näimme, probabilistisen päättelyn eri muodot voidaan toteuttaa yksipolkuisilla (joko aidosti yksipolkuisilla, tai monipolkuista verkkoa vastaavalla liittymäpuulla) Bayes-verkoilla ajassa joka on suoraan verrannollinen verkkoa vastaavien parametrien määrään. Luvussa 3.1 näimme että nämä parametrit ovat muotoa $P(X \mid F(X))$, missä $F(X)$ on solmun X edeltäjien joukko annetussa verkossa. Tätä muotoa olevia parametreja on eksponentiaalinen määrä solmun X edeltäjien arvojen lukumäärän suhteen, joten jos solmulla on hyvin suuri joukko edeltäjiä, saattaa tarvittavien parametrien lukumäärä olla liian suuri käytännön tarpeita ajatellen[5]. Ns. "noisy-or"-mallissa muotoa $P(X \mid F(X))$ olevia parametreja approksimoidaan kaavan

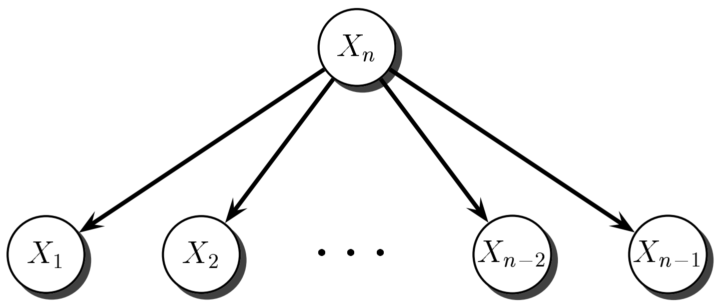
$$P(X \mid Y_1, \ldots, Y_n) = 1 - \prod_{i=1}^{n}(1 - P(X \mid Y_i))$$

mukaan. "Noisy-or"-mallia käyttäen voidaan tarvittavien parametrien lukumäärää pienentää siis huomattavasti: kutakin solmua X kohden riittää tallettaa $|F(X)|$ parametria, missä $|F(X)|$ on solmun edeltäjien lukumäärä. "Noisy-or"-approksimaation on empiirisesti havaittu toimivan useissa käytännön tilanteissa hyvin, vaikka toisinaan sen sisältämät implisiittiset riippumattomuusoletukset eivät pidä paikkaansa jolloin approksimaatio ei ole riittävän tarkka käytännön sovelluksia ajatellen [53, 97].

3.4.3 Jatkuva-arvoiset muuttujat

Tässä raportissa on havainnollisuuden vuoksi keskitytty tarkastelemaan diskreettejä Bayes-verkkoja, joissa erilaisia mahdollisia muuttujien arvoja on vain äärellinen määrä. Koska Bayes-verkko kuitenkin viime kädessä on yhteistodennäköisyysjakauman hajotelman graafinen esitys, joka kertoo kuinka muuttujajoukon yhteistodennäköisyys saadaan laskettua ehdollisten todennäköisyyksien tulona (katso kaava 3.3), ei mikään estä käyttämästä tässä hajotelmassa myös jatkuvia ehdollisia tiheysfunktioita: sekä luvussa 3.2 hahmoteltu probabilistisen päättelyn toteuttamiseen käytetty viestinvälitysalgoritmi [83, 3], että luvussa esitetty Bayes-verkkojen rakentamisessa käytetty mallinvalintakriteeri [40] voidaan määritellä myös jatkuvien muuttujien tapauksessa, ja myös sekatapauksessa jossa käytetään sekä jatkuvia että diskreettejä muuttujia. Malliperheen laajennus tällä tavalla saattaa joissakin tapauksissa aiheuttaa teknisiä ongelmia — yksinkertaisinta jatkuvien muuttujien käsittely on ns. äärellisissä sekajakaumamalleissa, joita käsittelemme luvussa 3.4.5. On myös huomattava että käytännön sovelluksissa reaaliarvoisten muuttujienkaan arvot eivät todellisuudessa ole äärettömän tarkkuuden reaalilukuja, vaan jollakin äärellisellä tarkkuudella esitettyjä numeroita. Niinpä monissa sovelluksissa jatkuvia arvoja voidaankin yhtä hyvin käsitellä suoraan diskreetteinä muuttujina, tai ne voidaan ensin diskretisoida. Bayesiläistä lähestymistapaa diskretisointiin on käsitelty lähteissä [37, 71].

[5] Useimmat käytännön sovelluksissa käytettävät Bayes-verkot ovat suhteellsien harvoja, jolloin tätä ongelmaa ei esiinny.



Kuva 3.8: Naiivi Bayes-mallin Bayes-verkkoesitys. Juurimuuttuja X_n on luokkamuuttuja, jonka suhteen muiden muuttujien riippumattomuusoletus tehdään.

3.4.4 Naiivi Bayes-malli

Luokitteluongelmissa tehtävänä on laskea diskreetin luokkamuuttujan jakauma annettuna muiden muuttujien (mahdollisesti osittainen) arvokombinaatio $\mathbf{S}_2 = \mathbf{s}_2$. Niin sanotussa *Naiivi Bayes*-mallissa muut muuttujat oletetaan riippumattomiksi, jos luokkamuuttujan arvo on tunnettu. Tästä oletuksesta seuraa että muuttujien yhteistodennäköisyysjakauma voidaan esittää muodossa

$$P(X_1, \dots, X_n) = P(X_1, \dots, X_{n-1} \mid X_n)P(X_n) = P(X_n)\prod_{i=1}^{n-1} P(X_i \mid X_n),$$

missä muuttujat on oletettu järjestetyiksi siten että luokkamuuttujan indeksi on n. Tämän hajotelman Bayes-verkkoesitys on kuvassa 3.8.

Naiivi Bayes-mallia on sovellettu monissa erilaisissa luokitteluongelmissa varsin hyvällä menestyksellä. Yksi syy mallin suosioon on probabilistisen päättelyn tehokkuus: koska Bayes-verkko on tässä tapauksessa kaksitasoinen puu, on laskenta äärimmäisen tehokasta. Itse asiassa Naiivi Bayes-mallissa tapahtuva probabilistinen päättely voidaan toteuttaa feedforward-neuroverkon tapaisena arkkitehtuurina [95, 94]. Mallin toinen merkittävä etu on se että koska mallin arkkitehtuuri on kiinnitetty, ei mallistruktuurin oppimisongelmaa ole, ja mallin parametrit voidaan asettaa suoraan luvussa 3.3.2 esitettyjä kaavoja käyttäen.

Vaikka Naiivi Bayes-malliin johtava riippumattomuusoletus saattaa tuntua melko epärealistiselta, toimii Naiivi Bayes-malli käytännössä usein yllättävän hyvin [133, 82]. Itse asiassa monet Bayes-verkkojen kaupallisista sovelluksista käyttävät tätä yksinkertaista mallia (esimerkiksi Microsoftin Answer Wizard ja Office Assistant, ks. luku 4.1). Hyvä esimerkki Naiivi Bayes-mallin sovellettavuudesta saatiin “Knowledge Discovery and Data Mining”-konferenssin yhteydessä kesällä 1997 julkistetussa maailmanlaajuisessa ennustuskilpailussa, jossa kaksi kolmesta parhaasta ratkaisusta käytti juuri Naiivi Bayes-mallia [103]. Tämän yksinkertaisen mallin menestykselle voidaan antaa seuraava intuitiivinen selitys: monissa luokitteluongelmissa (esimerkiksi lääketieteellisissä ongelmissa) on muuttujien arvojen määrittäminen kallista, koska se edellyttää esimerkiksi erityisten laboratoriokokeiden suorittamista. Niinpä kahden toisistaan riippuvan muuttujan mukaanottaminen on resurssien tuhlausta, koska jo toinenkin näistä muuttujista sisältää tarvittavan informaation. Vaikuttaa siltä että monet reaalimaailman datajoukot on kerätty resursseja säästäen, minkä seurauksena Naiivi Bayes-mallin riippumattomuusoletus saattaa olla hyvinkin järkevä.

Naiivi Bayes-mallin on toisaalta havaittu luokittelevan hyvin myös sellaisissa (keinotekoisissa) tilanteissa, joissa käytetty datajoukko on konstruoitu sellaiseksi, että Naiivi Bayes-mallin riippumattomuusoletus rikkoutuu varmasti [5]. Mallin luokittelutarkkuuden on näissäkin tilanteissa ollut suuri — malli pystyy siis diskriminoimaan suurella tarkkuudella annetut esimerkit vastaaviin luokkiin — mutta jos tarkastellaan mallin antamaa jakaumaa luokkamuuttujalle, nähdään että mallin estimoima jakauma ei ole kovinkaan tarkka. Naiivi Bayes-mallin riippumattomuusoletuksesta seuraa siis ilmeisen konsistetti vinouma (bias) estimoituun luokkamuuttujan jakaumaan. Vinouma estää luokkamuuttujan jakauman tarkan estimoinnin, mutta koska oletuksista seurannut vinouma on kaikissa tilanteissa johdonmukainen, saattaa menetelmän erottelukyky olla edelleen hyvä. Tämä tarkoittaa siis sitä että Naiivi Bayes-malli saattaa olla hyvä vaihtoehto silloin jos luokittelussa riittää erotella annetut tapaukset eri luokkiin, mutta naiivi Bayes-mallin tarjoaman luokkamuuttujan jakauman käyttöön riskianalyysissä kannattaa suhtautua varauksella.

3.4.5 Latentit muuttujat ja äärelliset sekajakaumat

Standardit Bayes-verkkomallit perustuvat siihen oletukseen että mallintamisessa käytettävät muuttujat $X_1, \ldots, X_n$ on valittu siten että ongelmakentälle voidaan konstruoida toimiva malli valittuja muuttujia käyttäen. Mikäli näin ei ole, voidaan tietysti kyseenalaistaa koko tehtävänasettelu, ja valita käytettävät muuttujat uudelleen. Jos muuttujajoukon uudelleenmäärittely ei kuitenkaan tällaisessa tilanteessa ole mahdollista, saattaa Bayes-verkkojen (tai minkä tahansa vain annettuja muuttujia käyttävän malliperheen) soveltaminen tuottaa epätyydyttäviä tuloksia. Seuraava yksinkertainen esimerkki havainnollistaa tätä seikkaa.

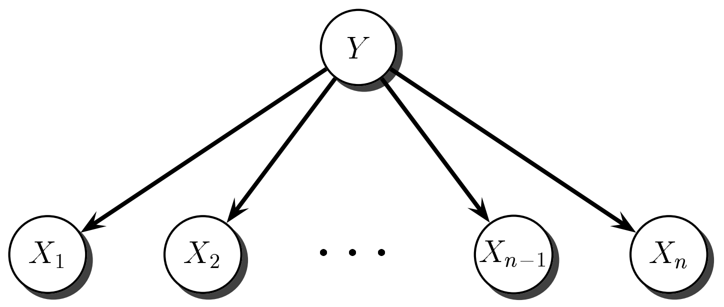
Oletetaan että käytettävissä olevia muuttujia on kaksi, X_1, ja X_2, ja muuttujan X_1 arvo kertoo hukkumiskuoleminen viikottaisen lukumäärän Suomessa, ja muuttujan X_2 arvo Suomessa viikottain myydyn jäätelön määrän. Tilastojen perusteella on helppo nähdä että muuttujilla on selvästi jonkinlainen riippuvuussuhde, joten Bayes-verkkostruktuuri jossa muuttujat eivät ole yhdistettyjä toisiinsa, ei mallinna ongelmakenttää tyydyttävällä tavalla. Jäljelle jää vain kaksi mahdollista Bayes-verkkostruktuuria: $X_1 \rightarrow X_2$ ja $X_1 \leftarrow X_2$. Intuitiivisesti ottaen nämä tapaukset näyttäisivät kuitenkin indikoivan että jompi kumpi muuttuja olisi toisen muuttujan syy, mikä ei varmaankaan pidä paikkaansa todellisuudessa. Jos meidän nyt sallitaan lisätä malliimme uusi muuttuja Y, niin näemme että Bayes-verkkorakenne $X_1 \leftarrow Y \rightarrow X_2$ tuottaa muuttujien X_1 ja X_2 välille *epäsuoran riippuvuussuhteen*, joka mallintaa tyydyttävästi intuitiivista kuvaamme ongelmakentästä.

Edellisessä esimerkissä käytettyä muuttujaa Y kutsutaan *latentiksi muuttujaksi* tai *piilomuuttujaksi*, koska muuttuja ei ollut mukana alkuperäisessä muuttujajoukossa, ja sen arvoja ei siten ole annettu opetusjoukossa $\mathcal{D}$. Yksinkertaisessa esimerkissämme on käytetylle latentille muuttujalle helppo keksiä semanttinen tulkinta (vuodenaika), mutta aina näin ei ole asianlaita, eikä se bayesiläisen mallintamisen kannalta ole välttämättä tarpeenkaan: latenttien muuttujien arvoja opetusjoukossa voidaan pitää puuttuvana datana, joka marginalisoidaan pois käytetyistä laskukaavoista.

Mikäli latentteja muuttujia on vain yksi, ja latentista muuttujasta on kaari kaikkiin muihin muuttujiin, ja muiden muuttujien välillä ei ole kaaria, kutsutaan syntyvää Bayes-verkkomallia *äärelliseksi sekajakaumamalliksi (finite mixture model)* [35, 136]. Syntyvä yhteistodennäköisyysjakauma on tässä tapauksessa

$$\begin{aligned} P(X_1, \ldots, X_n) &= \sum_y P(X_1, \ldots, X_n, Y = y) \\ &= \sum_y P(X_1, \ldots, X_n \mid Y = y) P(Y = y) \\ &= \sum_y P(Y = y) \prod_{i=1}^{n} P(X_i \mid Y = y), \end{aligned}$$

missä summa käy yli kaikkien mahdollisten piilomuuttujan Y arvojen. Vastaava Bayes-verkkostruktuurin graafinen esitys on kuvassa 3.9. On tärkeää huomata että vaikka syntyvä mallistruktuuri on muodollisesti identtinen Naiivi Bayes-mallin (kuva 3.8) kanssa, on juurimuuttuja sekajaumamallien tapauksessa latentti muuttuja (ja mahdollinen luokkamuuttuja on siis puun lehti siinä missä muutkin muuttujat), joten malli ei tee samoja riippumattomuusoletuksia kuin Naiivi Bayes-malli. Itse asiassa on helppo nähdä että



Kuva 3.9: Äärellisen sekajakaumamallin Bayes-verkkoesitys. Juurimuuttuja Y on latentti piilomuuttuja, jonka suhteen muiden muuttujien riippumattomuusoletus tehdään.

mikä tahansa Bayes-verkkomalli voidaan muuttaa äärelliseksi sekajaumamalliksi, eli sekajaumamallit muodostavat universaalin malliperheen.

Bayes-verkkoihin verrattuna sekajakaumamalleilla on se etu, että koska syntyvä Bayes-verkkostruktuuri on yksinkertainen puu, on sekajakaumamalleilla suoritettava päättely laskennallisesti aina erittäin tehokasta. Lisäksi mahdollisia verkkostruktuureja on vain yksi, joten mallistruktuuriavaruus ei ole kovin suuri: ainoa vapaa parametri on latentin muuttujan arvojen lukumäärä, jonka voidaan olettaa olevan vähemmän kuin joukon $\mathcal{D}$ koko. Juurimuuttujan latenttiudesta seuraa kuitenkin että kokonaisuskottavuus $P(\mathcal{D} \mid M)$, joka Bayes-verkkojen tapauksessa lasketaan helposti kaavaa (3.4) käyttäen, ei ole sekajaumien tapauksessa laskettavissa käytännössä tarkasti: kokonaisuskottavuus saadaan marginalisoimalla yli kaikkien mahdollisten piilomuuttujan arvoista muodostuvien vektorien Z yli,

$$P(\mathcal{D} \mid M) = \sum_Z P(\mathcal{D}, Z \mid M),$$

joita on eksponentiaalinen määrä. Sekajakaumamallien kokonaisuskottavuuden laskemiseksi on kuitenkin kehitetty monia tehokkaita approksimaatiomenetelmiä (katso [73, 81, 77, 72, 26]).

Kokonaisuskottavuuden laskemisen lisäksi piilomuuttujaoletuksesta implisiittisesti seuraava puuttuva informaatio tekee myös sekajakumamallien malliparametrien laskemisen vaikeammaksi kuin normaalien Bayes-verkkojen tapauksessa. Tähänkin tehtävään on kehitetty tehokkaita approksimaatiomenetelmiä, joista yleisin tunnetaan nimellä *Expectation-Maximization (EM)* [33, 91, 62, 13, 78]. Koska latenttia piilomuuttujaa Y voidaan ajatella eräänlaisena ”klusterimuuttujana”, joka ryhmittelee annetut esimerkkitapaukset, voidaan EM-algoritmia pitää probabilistisena ryhmittelyalgoritmina. Algoritmin tuottamia tapausryhmiä voidaan soveltaa ongelmakentän probabilistisessa analysoinnissa ja visualisoinnissa (data mining) samalla tavoin kuin esimerkiksi SOM-neuroverkkomallia [67]. Algoritmi tarjoaa siis bayesiläisen lähestymistavan *numeerisena taksonomiana* tunnetulle tutkimusalueelle [43].

Kuten luvussa 2.3 todettiin, sekajakaumamalliperhe tarjoaa yhtenäisen teoreettisen kehikon monille näennäisesti erilaisille lähestymistavoille, kuten esimerkkeihin perustuva päättely [135, 134, 95, 69, 70], ydinestimaattorit, ydinkantafunktiot (radial basis functions), sekä probabilistiset neuroverkot [12], itseorganisoivat kartat [14], sekä piilo-Markov-mallit (hidden Markov models) [126, 30, 108]. Lähteissä [133, 135] raportoidut laajat empiiriset testitulokset osoittavat että mallistruktuurin näennäisestä yksinkertaisuudesta huolimatta sekajakaumallit (ja myös jopa Naiivi Bayes-mallit) tuottavat konsistentisti erittäin hyviä tuloksia vaihtoehtoisilla menetelmillä (kuten esimerkiksi neuroverkot ja päätöspuut) saavutettuihin tuloksiin verrattuna. Erityisen kiinnostavaa on että bayesiläiset mallit näyttävät suoriutuvan hyvin myös tilanteissa, joissa opetusdataa on hyvin vähän: saatavilla olevan datan määrän vähentäminen jopa 90 prosentilla ei näytä juurikaan huonontavan mallien antamia tuloksia [74, 75].

3.4.6 Kvalitatiiviset Bayes-verkot

Kvalitatiivisissa Bayes-verkoissa (qualitative Bayesian networks) [140, 34] luovutaan verkon parametrien (ehdollisten todennäköisyyksien) esittämisestä kvantitatiivisina reaalilukuina, ja siirrytään äärelliseen joukkoon mahdollisia kvalitatiivisia arvoja, esimerkiksi joukkoon $\{-, 0, +\}$. Verkon kaaret kuvaavat tässä tilanteessa kuinka tietyn solmun edeltäjät vaikuttavat vastaavaan muuttujaan X: vähentääkö ('-') vai lisääkö ('+') edeltäjien arvokombinaatio muuttujan X todennäköisyyttä, vai onko arvokombinaatio X:n suhteen irrelevantti ('0'). Näyttää siltä että kvalitatiivisten Bayes-verkkojen tapauksessa on mahdollista suorittaa määrättyjä probabilistisen päättelyn muotoja hyvin tehokkaasti myös monipolkuisten verkkojen tapauksessa, mutta teoreettisia tuloksia asiasta ei vielä juuri ole. Suomenkielinen katsaus kvalitatiivisia verkkoja käsittelevän tutkimuksen nykytilaan löytyy lähteestä [127].

3.4.7 Bayes-verkkojen ja neuroverkkojen yhteyksiä

Kuten edellä on jo mainittu, bayesiläisiä tekniikoita voidaan periaatteessa soveltaa myös muiden malliperheiden kuin Bayes-verkkojen tapauksessa. Mackay [88] ja Neal [96] kuvaavat väitöskirjoissaan bayesiläisten tekniikoiden soveltamista suunnattujen neuroverkkojen oppimisessa. Mackay voitti kehittämällään ohjelmistolla vuonna 1993 järjestetyn laajan aikasarjaennustamiskilpailun [89]. Sekä Mackayn[6] että Nealin[7] kehittämät ohjelmistot ovat vapaasti saatavilla kehittäjiensä kotisivuilta.

Paitsi neuroverkkojen oppimisalgoritmien kehittämisessä, Bayes-verkkojen ja neuroverkkojen välille on luotu myös lukuisia muita yhteyksiä. Michael Jordanin tutkimusryhmä MIT:ssa on tutkinut Bayes-verkkojen ja suunnattujen neuroverkkojen välimuotoa, jossa Bayes-verkkojen muuttujien välisten riippuvuuksien voimakkuuksia kuvataan neuroverkoista tutulla sigmoid-funktiolla. Malliperheestä käytetään nimitystä *sigmoid belief networks* [117]. Bishop kuvaa kirjassaan [13] ydinkantafunktioiden (radial basis functions), probabilististen neuroverkkojen, ydinestimaattoreiden ja sekajakaumallien välisiä yhteyksiä. Myllymäki kuvaa väitöskirjassaan [93] kuinka Bayes-verkoissa suoritettava probabilistinen päättely voidaan suorittaa massiivisesti rinnakkaisella Boltzmannin kone-neuroverkkoarkkitehtuurilla. Baysiläinen kehikko itseorganisoiville kartoille [67] on kuvattu artikkelissa [14].

[6] URL: http://wol.ra.phy.cam.ac.uk/mackay/homepage.html
[7] URL: http://www.cs.utoronto.ca/~radford/

Luku 4

Bayes-verkkojen sovelluksia

Bayes-verkkojen sovellusesimerkkejä: lääketieteellinen diagnosointi, prosessikontrolli, vikadiagnostiikka, tiedon analysointi. Millä ohjelmistoilla Bayes-verkkomalleja voidaan rakentaa (Bayes-verkkokehittimet): kaupalliset ohjelmistot, tutkimusprototyypit.

Bays-verkkosovellukset voidaan jakaa kahteen ryhmään: yksittäiset Bayes-verkkotekniikkoihin perustuvat sovellukset, ja yleiset Bayes-verkkokehittimet, jotka ovat geneerisiä työkaluja erillisten sovellusten tuottamista varten. Bayes-verkkotutkimukseen on koko sen vasta noin kymmenvuotisen historian aikana liittynyt selkeä pyrkimys kehitettyjen menetelmien kaupalliseen soveltamiseen, ja sovellusten määrä onkin viime vuosina ollut voimakkaassa kasvussa. Lisäkiinnostusta Bayes-verkkojen hyödyntämismahdollisuuksia kohtaan herätti Bill Gatesin lokakuussa 1996 antama lausunto, jonka mukaan Microsoftin kilpailuetu tulevaisuudessa perustuu heidän eritysosaamiseensa Bayes-verkkojen alueella [54]. Microsoftilla onkin yksi maailman johtavia Bayes-verkkojen tutkimusyksiköitä (katso luku 5.2.1).

Luvussa 4.1 esitellään joitakin yksittäisiä Bayes-verkkosovelluksia. Luvussa ei pyritä antamaan tyhjentävää luetteloa kaikista olemassaolevista Bayes-verkkosovelluksista, vaan tyypillisiä esimerkkejä Bayes-verkkotekniikoita käyttävistä sovelluksista erilaisilta sovellusalueilta. Jotkut mainituista sovelluksista on luotu käyttäen kaupallisia Bayes-verkkokehittimiä, joita käsitellään luvussa 4.2, toiset taas ovat käsityönä luotuja räätälöityjä ohjelmistoja. Lisätietoa Bayes-verkkosovelluksista löytyy mm. David Heckermin laatimasta listasta[1], johon on koottu yli sata viitettä Bayes-verkkojen käytännön sovelluksia raportoiviin artikkeleihin vuosilta 1985–1995. Heckermanin lista on melko laaja, mutta jo hieman vanhentunut. AUAI-järjestö (Association for

[1] URL: ftp://ftp.research.microsoft.com/pub/dtg/david/BN-APPS.TXT

Uncertainty in Artificial Intelligence) on koonnut suppeahkon listan joistakin merkittävimmistä viimeaikaisista sovelluksista. Lista on saatavilla järjeston kotisivulta[2], mistä löytyy myös kokoelma käytännön soveltajilta kerättyjä kommentteja siitä, miksi Bayes-verkot ovat osoittautuneet onnistuneeksi ratkaisuksi heidän sovellusympäristössään.

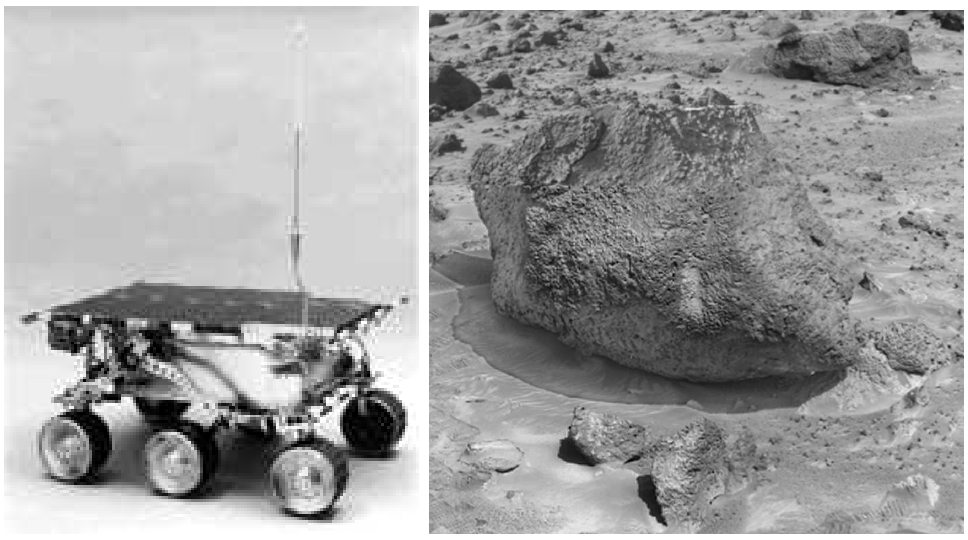
4.1 Sovellusesimerkkejä

Luvussa 3.2 esitettiin kuinka probabilistisen päättelyn eri muotoja voidaan toteuttaa käytännössä Bayes-verkkomalleja hyväksikäyttäen. Laajat julkisesti saatavilla olevilla luokitteluaineistoilla tehdyt vertailut osoittavat, että bayesiläiset mallit tuottavat konsistentisti erittäin hyviä tuloksia vaihtoehtoisilla menetelmillä (kuten esimerkiksi neuroverkot ja päätöspuut) saavutettuihin tuloksiin verrattuna [133, 135, 74, 79, 80]. Kuten luvussa 1.3 näimme, probabilistista päättelyä voidaan soveltaa luokitteluongelmien lisäksi ratkaisemaan hyvin monentyyppisiä ongelmatilanteita, joten Bayes-verkkojen sovellusalueitakin on lukuisia. Seuraavassa joitakin tyypillisimpiä sovellusesimerkkejä sovellusalueittain luokiteltuna.

Lääketieteellinen diagnosointi Lääketieteellinen diagnosointi on perinteisesti ollut yksi probabilististen menetelmien suosituimmista sovellusalueista. Yksi syy tähän varmaankin on se, että probabilististen menetelmien kanssa on tulosten luotettavuutta mahdollista arvioida tekemällä riskianalyysi luvussa 1.4 esitetyllä tavalla. Yksi ensimmäisiä tällaisia ohjelmistoja oli rintasyöpäpotilaiden diagnosointiin suunniteltu PathFinder [51], joka on myöhemmin kaupallistettu BiopSys Medical-yhtiön[3] IntelliPath-nimisenä tuotteena. IntelliPath-ohjelmisto on tällä hetkellä käytössä sadoissa eri sairaaloissa ympäri maailmaa. Muita vastaavia eri sairaaloissa ja tutkimuslaitoksissa aktiivisessa käytössä olleita järjestelmiä ovat mm. ACORN, DXPLAIN ja Iliad[4]. Yksi suurisuuntaisimpia tämän ongelma-alueen hankkeita on Microsoftin OnParenting-ohjelmisto[5], joka on lasten sairauksien diagnosointiin suunniteltu laaja järjestelmä. Järjestelmä käyttää sairauksien diagnosointiin Knowledge Industries-yhtiön kehittämää Bayes-verkkomallia.

Tiedon analysointi (data mining) Koska Bayes-verkkoja käyttäen voidaan tilastollisesta esimerkkiaineistosta konstruoida ongelmakentästä semant-

[2] URL: http://www.auai.org/
[3] URL: http://www.biopsys.cim
[4] URL: http://www-uk.hpl.hp.com/people/ewc/list-main.html
[5] URL: http://www.onparenting.msn.com



Kuva 4.1: NASA:n Pathfinder-projektin Marsin pinnalle lähettämä Sojounerrobotti, ja sen lähettämistä kuvista superresoluutiotekniikalla tuotettu kuva noin metrin kokoisesta kivenlohkareesta ("Yogi") Marsin pinnalla. Kuva löytyy osoitteesta http://mpfwww.jpl.nasa.gov/mpf/high-res.html.

tisesti tulkittavissa oleva malli, voidaan syntyviä malleja käyttää ongelmakentän ominaisuuksien analysoimisessa ja visualisoimisessa. NASAn Amestutkimuskeskuksen AutoClass-projektissa[6] on sovellettu bayesiläistä mallintamista menestyksellisesti mm. IRAS tähtikartaston kuvien, Yhdysvaltojen eri lentokenttien, ja eri DNA-sekvenssien analysoimisessa [24].

Kuvankäsittely NASAn tutkimusryhmät ovat kehittäneet menetelmän ns. *superresoluutiokuvien (super-resolution images)* tuottamiseksi. Menetelmässä on kyse useiden samasta kohteesta otettujen huonolaatuisten kuvien yhdistämisestä yhdeksi hyvälaatuiseksi ("superresoluutio"-) kuvaksi. Kuvassa 4.1 on tällainen Sojourner-robotin tuottamista kuvista bayesiläisellä tekniikalla muodostettu superresoluutiokuva kivenlohkareesta Marsin pinnalla.

Superresoluutiokuvia tuottava tekniikka ei perustu suoraan Bayes-verkkoteoriaan, mutta se seuraa puhtaasti edellä esitettyjä bayesiläisen mallintamisen periaatteita (katso [23]). Paitsi Pathfinder-projektin aineistosta, NASAn tutkijat ovat tuottaneet superresoluutiokuvia myös Galileo-luotaimen aineistosta, sekä Viking-luotaimen Marsin pinnasta ottamista kuvista[7].

[6]URL: http:/ic-www.arc.nasa.gov/ic/projects/bayes-group/autoclass/

[7]URL: http://ic.arc.nasa.gov/ic/projects/super-res/

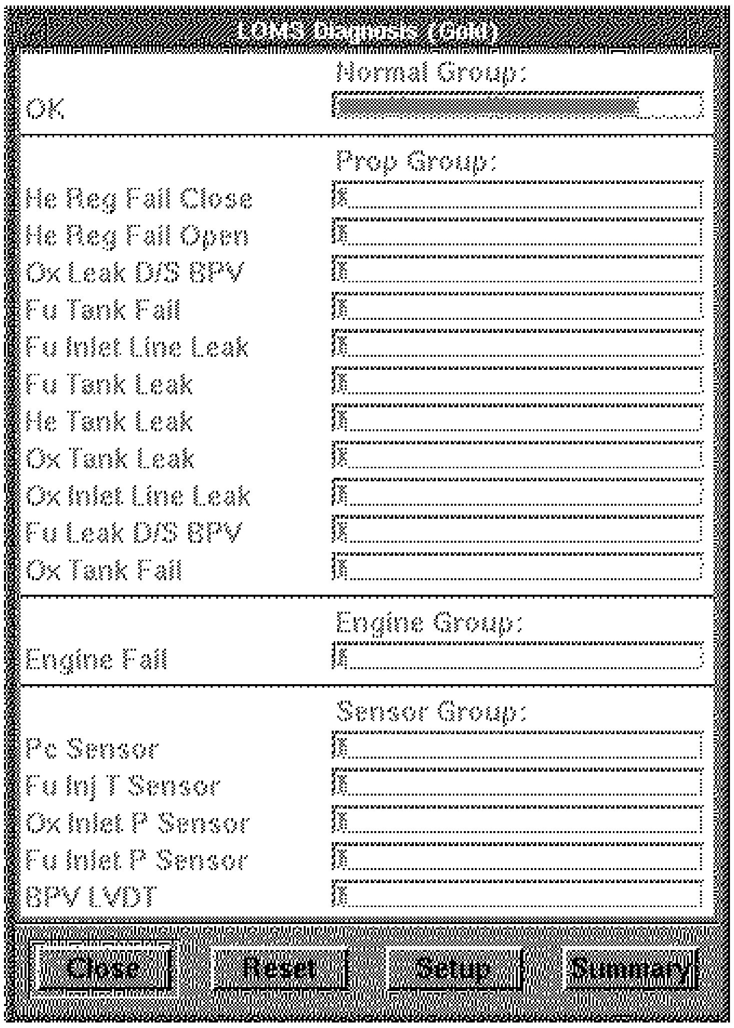
Hahmontunnistus Bayesiläistä mallinnusta on sovellettu hahmontunnistuksessa menestyksellisesti mm. NASA:n tutkimuskeskuksissa LandSat-satelliittien ottamien kuvien alueiden automaattisessa tunnistamisessa [64]. Käytetty Bayes-verkkomalli oli tässä tapauksessa luvussa 3.4 esitetty äärellinen sekajakaumamalli. Tavanomaisten Bayes-verkkojen käyttöä kuvantunnistuksessa käsitellään mm. lähteissä [61, 42]. Bayes-verkoilla on myös läheinen yhteys ns. *piilo-Markov-malleihin (hidden Markov models)* (katso [126, 30, 108]), joita käytetään yleisesti esimerkiksi puheentunnistukseen liittyvien hahmontunnistusongelmien ratkaisemisessa.

Vikadiagnostiikka Microsoftin tutkimusosasto on kehittänyt useita kymmeniä Bayes-verkkoihin perustuvia vikadiagnostiikkajärjestelmiä (troubleshooting wizards), jotka auttavat tietokoneen käyttäjää erilaisissa ongelmatilanteissa[8]. Intel käyttää Bayes-verkkoja prosessoripiirien valmistusprosessin laaduntarkkailussa ja American Airlines ohjelmistovirheiden etsimisessä[9].

Prosessikontrolli Yksi esimerkki prosessikontrolliin liittyvistä sovelluksista on kuvassa 3.4 esitetty ALARM-verkko jota käytetään sairaalapotilaiden tilaa tarkastelevan mittausprosessin ohjaamisessa [8]. NASA on kehittänyt laajan Bayes-verkkoihin perustuvan järjestelmän avaruussukkuloiden lentojen kontrollomiseksi. Järjestelmä paitsi ilmoittaa mahdollisista vikatilanteista, myös ehdottaa päätösteorian mukaisesti parhaat mahdolliset toimenpiteet kriisitilanteissa. Lisäksi ohjelmisto muokkaa aktiivisesti lentojen valvonnassa käytettäviä graafisia näyttöjä siten, että kullakin hetkellä näytetään vain tärkeimmät sukkuloiden tilaa kuvaavat mittaustulokset lentojen kulkua seuraavalle henkilökunnalle, mikä helpottaa teknikkojen työtä, ja nopeuttaa päätöksentekoa ongelmatilanteissa (katso kuva 4.2). VISTA-järjestelmä on ollut käytössä NASA:n Houstonin avaruussukkuloiden lennonjohtokeskuksessa jo useita vuosia. Enemmän tietoa VISTA-projektista löytyy projektin WWW-sivulta[10].

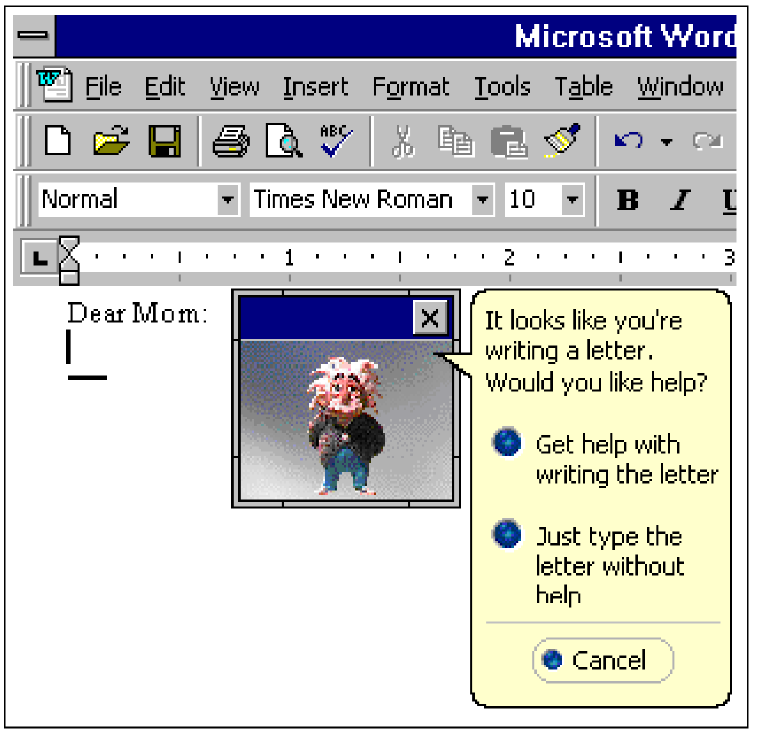
Bayes-verkkoja on sovellettu menestyksellisesti myös mm. lentokoneiden suihkumoottorien suunnittelussa (General Electric[11]), sekä erilaisten asejärjestelmien (Mitre/US Navy[12]) ja miehittämättömien vedenalaisten ajoneuvojen (Lockheed/DARPA[13]) ohjausprosesseissa.

[8] URL: http://support.microsoft.com/support/
[9] URL: http://www.auai.org/BN-Routine.html
[10] URL: http://www.research.microsoft.com/research/dtg/horvitz/vista.htm
[11] URL: http://www3.us.com/kic/kic/
[12] URL: http://www.afit.af.mil/Schools/EN/ENG/LABS/AI/BayesianNetworks/fieldedSystems.html#imsi
[13] URL: http://hugin.dk/lockheed-9604.html



Kuva 4.2: Esimerkki VISTA-järjestelmän muokkaamasta adaptiivisesta graafisesta näytöstä, jonka perusteella avaruussukkuloiden valvontakeskus seuraa lentojen kulkua.

Älykkäät agentit Älykkäät agentit ovat perussovellusohjelmiin liitettäviä itsenäisiä moduleita, jotka seuraavat järjestelmän toimintaa, ja helpottavat sovellusten käyttöä tarjoamalla aktiivisesti apua silloin kun katsovat sen käyttäjän kannalta tarpelliseksi. Microsoft on soveltanut Bayes-verkkotekniikkaa älykkäiden agenttien toteuttamisessa, ja tekniikka on jo käytössä Office'97-ohjelmistossa nimellä *Office Assistant*-nimisenä tuotteena (katso kuva 4.3). Microsoftin pitkän tähtäyksen suunnitelmana on toteuttaa Bayes-verkkotekniikoihin perustuva *adaptiivinen käyttöjärjestelmä*, koodinimeltään *Lumiere*, joka seu-



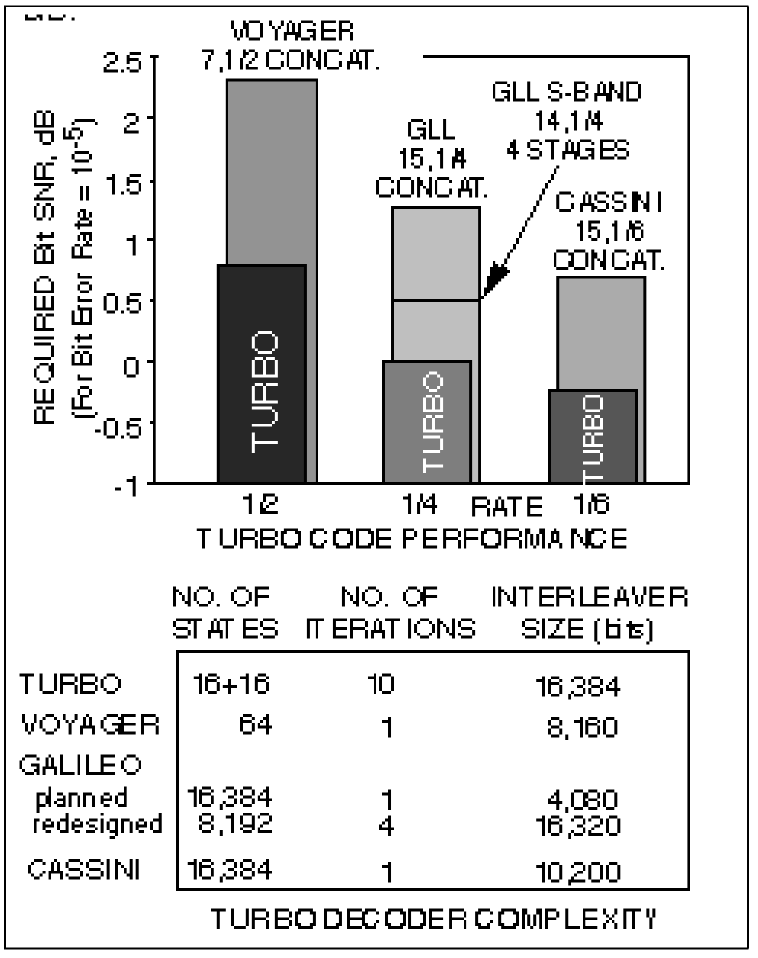
Kuva 4.3: Microsoftin Office Assistant-agentti.

raa tietokoneen käyttäjän toimintaa, ja adaptoituu toimimaan siten että tietokoneen käyttö on mahdollisimman tehokasta. Lisätietoja Lumiere-projektista löytyy projektin WWW-sivulta[14].

Tiedon kompressointi Kuten luvussa 2.4 näimme, bayesiläiselle lähestymistavalle voidaan antaa myös informaatioteoreettinen tulkinta todennäköisyysjakaumien ja koodinpituuksien läheisen yhteyden ansiosta. Niinpä ei olekaan yllättävää että Bayes-verkkoteoriaa voidaan soveltaa myös erilaisten koodausmenetelmien kehittämisessä, ja kehitettyjä koodeja voidaan soveltaa esimerkiksi tiedon kompressoinnissa. Vuonna 1993 esiteltiin uusi vallankumouksellinen koodausmenetelmä, *turbo-koodaus* [11], jonka on empiirisesti havaittu olevan huomattavasti tehokkaampi koodausmenetelmä kuin aikaisemmin käytetyt koodit (katso kuva 4.4[15]).

Teoksessa [90] osoitetaan kuinka turbo-koodaaja voidaan esittää Bayes-verkkomallina. Turbo-koodausta pidetään yhtenä tärkeimmistä koodausteo-

[14] URL: http://www.research.microsoft.com/research/dtg/horvitz/lum.htm
[15] Kuva löytyy WWW-sivulta URL: http://www331.jpl.nasa.gov/public/JPLtcodes.html



Kuva 4.4: Turbo-koodin vertailu avaruusluotaimissa nykyisin käytössä oleviin koodausmenetelmiin. Huomaa että histogrammikuvan asteikko on logaritminen, joten erot eri menetelmien välillä ovat huomattavia.

rian tuottamista tuloksista käytännön sovellusten kannalta. Menetelmä on erityisen kiinnostava avaruustutkimuksen kannalta, koska avaruusluotaimien lähetyskapasiteetti on melko pieni, jolloin tarvitaan tehokkaita tiedon pakkausalgoritmeja.

4.2 Bayes-verkkokehittimet

Koska uusia Bayes-verkko-ohjelmistoja syntyy kiihtyvässä tahdissa, oheinen lista saatavilla olevista Bayes-verkkokehittimistä vanhenee nopeasti. Elektronisesti saatavilla oleva Russell Almondin ylläpitämä lista löytyy Almondin kotisivulta[16], ja toinen vastaavanlainen lista CoSCo-ryhmän kotisivulta[17].

Ohjelmistot on seuraavassa jaoteltu kaupallisiin tuotteisiin ja tutkimusprototyyppeihin, mutta on huomattava että joistakin kaupallisista ohjelmistoista on saatavilla ilmainen versio tutkimus- tai opetuskäyttöön. Lisäksi useimmat kaupalliset yritykset tarjoavat tuotteistaan vapaasti kokeiltavia demo-versioita, jotka on saatavilla alla annettujen WWW-linkkien kautta. On myös todettava, että jotkut ilmaiseksi jaettavat, ja siksi tutkimusprototyypeiksi luokitellut ohjelmistot ovat ohjelmistoteknisesti kaupallisten ohjelmistojen tasolla (esim. Microsoftin MSBN).

Alla olevat ohjelmistot on jaoteltu sen mukaan, voidaanko ohjelmistoja käyttää Bayes-verkkojen oppimiseen datasta, vai tukevatko ohjelmat ainoastaan verkkojen manuaalista konstruomista. Oppimista tukevien ohjelmien lukumäärä ei ole toistaiseksi kovin suuri, mutta tilanne tullee muuttumaan lähitulevaisuudessa radikaalisti kun tuoreet aiheeseen liittyvät tutkimustulokset saadaan toteutettua sovellusasteelle.

4.2.1 Kaupalliset ohjelmistot

Oppimista tukevat ohjelmistot

- Ergo/Cogito
  Lisätietoja: Noetic Systems,
  URL: http://www.noeticsystems.com/ergo.shtml
- MIM
  Lisätietoja: David Edvards,
  URL: http://www.math.auc.dk/~jhb/CoCo/mim.html
- TETRAD
  Lisätietoja: Carnegie-Mellon University,
  URL: http://hss.cmu.edu/HTML/departments/philosophy/TETRAD/ tetrad.html

Manuaaliseen mallin konstruoimiseen perustuvat ohjelmistot

- Baron
  Lisätietoja: KC Associates,
  URL: mailto: kchang@gmu.edu
- Analytica
  Lisätietoja: Lumina Decision Systems,
  URL: http://www.lumina.com/software/index.html

[16] URL: http://bayes.stat.washington.edu/almond/belief.html
[17] URL: http://www.cs.Helsinki.FI/research/cosco/

- Decision Tools Suite
  Lisätietoja: Palisade,
  URL: http://www.palisade.com/

- DPL: Decision Programming Language
  Lisätietoja: Applied Decision Analysis,
  URL: http://www.dpl.adainc.com/

- DX Solution Series
  Lisätietoja: Knowledge Industries,
  URL: http://www.kic.com/

- GRAPHICAL-BELIEF
  Lisätietoja: Russell Almond,
  URL: http://bayes.stat.washington.edu/almond/gb/graphical-belief.html

- HUGIN
  Lisätietoja: Hugin Expert,
  URL: http://www.hugin.dk/

- NETICA
  Lisätietoja: Norsys Software,
  URL: http://www.norsys.com/netica.html

- STRATEGIST
  Lisätietoja: Prevision,
  URL: http://www.prevision.com/strategist.html

- DATA
  Lisätietoja: TreeAge Software,
  URL: http://www.treeage.com/

4.2.2 Tutkimusprototyypit

Oppimista tukevat ohjelmistot

- Belief Network Power Constructor
  Lisätietoja: Jie Cheng,
  URL: http://193.61.148.131/jcheng/bnpc.htm

- BIFROST
  Lisätietoja: Bo Thiesson,
  URL: http://www.sp.dk/~sorenh/software.html

- BKD: Bayesian Knowledge Discoverer
  Lisätietoja: Marco Ramoni,
  URL: http://kmi.open.ac.uk/~marco/projects/bkd/software/
- BNG
  Lisätietoja: Peter Haddawy,
  URL: http://www.cs.uwm.edu/public/dsail/research/bng.html
- BUGS
  Lisätietoja: Bugs-projekti,
  URL: http://www.mrc-bsu.cam.ac.uk/bugs/
- COCO
  Lisätietoja: Jens Henrik Badsberg,
  URL: http://www.math.auc.dk/~jhb/CoCo/cocoinfo.html

Manuaaliseen mallin konstruoimiseen perustuvat ohjelmistot

- BAYES
  Lisätietoja: Carnegie-Mellon University,
  URL: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasonng/probabl/bayes/0.html
- BAYONNET
  Lisätietoja: Yoichi Motomura,
  URL: http://www.etl.go.jp/etl/suri/motomura/BN/bn-java.html
- JAVABAYES
  Lisätietoja: Fabio Gozman,
  URL: http://www.cs.cmu.edu/~javabayes/Home/
- Bayesian Network Editor
  Lisätietoja: Sreekanth Nagarajan,
  URL: http://eel.engr.orst.edu/~nagarasr/bayesnet/index.html
- BELIEF
  Lisätietoja: Carnegie-Mellon University,
  URL: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasonng/probabl/belief/0.html
- IDEAL
  Lisätietoja: Rockwell International Science Center,
  URL: http://rpal.rockwell.com/ideal.html
  URL: mailto: ideal-request@rpal.rockwell.com

- MacEvidence
  Lisätietoja: Prakash Shenoy,
  URL: http://stat1.cc.ukans.edu:80/~pshenoy/
- Microsoft Bayes Networks (MSBN)
  Lisätietoja: Microsoft Research,
  URL: http://www.research.microsoft.com/research/dtg/msbn/
- Pulcinella
  Lisätietoja: IRIDIA,
  URL: http://iridia.ulb.ac.be/pulcinella/Welcome.html
- S-ElimBel
  Lisätietoja: Nicholas Thiéry,
  URL: http://www.spaces.uci.edu/thiery/elimbel/
- SPI
  Lisätietoja: Bruce D'Ambrosio,
  URL: http://www.cs.orst.edu/~dambrosi/
- TresBel
  Lisätietoja: Hong Xu,
  URL: http://iridia.ulb.ac.be/Projects/imple.html
- XBaies
  Lisätietoja: Robert Cowell,
  URL: mailto:xbaies-list@stats.ucl.ac.uk
- X-pert
  Lisätietoja: AI Group, Cantabria University,
  URL: http://ccaix3.unican.es/~AIGroup/
- XPress
  Lisätietoja: Mark Stitson,
  URL: mailto:M.Stitson@rhbnc.ac.uk

Yllä olevaan listaan on otettu mukaan vain standardeja Bayes-verkkoja käyttävät sovellukset. Latentteja muuttujia käyttäviä sekajakaumamalleja käyttäviä ohjelmisto on lukuisia — tunnetuimpia näistä ohjelmistoista ovat:

- AutoClass
  Lisätietoja: NASAn AutoClass-projekti,
  URL: http://ic-www.arc.nasa.gov:80/ic/projects/bayes-group/group/autoclass/

- Snob
  Lisätietoja: Snob-kotisivu,
  URL: http://www.cs.monash.edu.au/~dld/Snob.html

Sekä AutoClass että Snob ovat tutkimusprototyyppejä, joiden toiminta perustuu mallien bayesiläiseen oppimiseen datasta. Linkkeja sekajakaumamallien kaupallisiin sovelluksiin, ja muihin tutkimusprototyyppeihin löytyy David Dowen ylläpitämältä WWW-sivulta[18].

Tilastollisista sovelluksistaan tunnettu SPSS-yhtiö on myös ryhtynyt soveltamaan bayesiläisiä tekniikoita kaupallisissa ohjelmistoissaan. Käytettävä malliperhe ei tosin ole Bayes-verkot, vaan neuroverkkomallit. Yhtiön mainoksen mukaan bayesiläisen lähestymistavan avulla voidaan välttää ylioppiminen, ja mallien oppiminen on myös myös yleensä nopeampaa. Lisätietoja bayesiläisiä neuroverkkoja soveltavasta "Neural Connection 2"-tuotteesta löytyy WWW-sivulta URL: http://www.spss.com/software/Neuro/. Vastaavanlainen vapaasti saatavilla olevilla ohjelmisto on raportoitu Radford Nealin väitöskirjassa [96], ja on saatavilla Nealin kotisivulta[19]. Nealin ohjelmistoa on laajennettu hiljakkoin käsittämään neuroverkkojen lisäksi myös äärellisiä sekajakaumamalleja.

[18] URL: http://www.cs.monash.edu.au/~dld/mixture.modelling.page.html
[19] URL: http://www.cs.utoronto.ca/~radford/

Luku 5

Tietolähteitä

Bayes-verkkojen aktiiviset kaupalliset soveltajat ja niitä markkinoivat yritykset. Bayes-verkkoja koskevaa kirjallisuutta. Tutkimusalan tutkimusryhmät ja järjestöt.

5.1 Kaupallinen sektori

Bayes-verkkojen kaupallistamisen pioneereja on tanskalainen HUGIN-yhtiö, joka soveltaa Aalborgin yliopistossa toimivan, professori Steffen Lauritzenin johtaman tutkimusryhmän tuloksia kaupallisen Hugin-ohjelmiston kehittämisessä. Kuten Bill Gatesin antamasta lausunnosta [54] saattaa arvata, merkittävin kaupallinen tekijä Bayes-verkkojen sovellusalueella on tällä hetkellä eittämättä Microsoft, jonka Bayes-verkkotutkimusryhmä on tutkimusalan suurimpia.

HUGIN-yhtiö on keskittynyt kehittämään ja myymään yhtä tuotetta, Hugin-kehitintä, kun taas Microsoft tuottaa sisäisiä sovelluksia, jotka liitetään yhtiön kaupallisiin tuotteisiin. Monet Bayes-verkkosovelluksia tekevistä yhtiöistä toimivat HUGIN-yhtiön tapaan tuottaen geneerisiä Bayes-verkkokehittimiä, mutta yhä useammat yhtiöt laajentavat toimintaansa myös konsultointialueelle, ja auttavat asiakkaitaan rakentamaan laajoihin järjestelmiin integroitavia räätälöityjä sovelluksia, samalla tavoin kuin Microsoft liittää oman tutkimuskeskuksensa tulokset tuotteisiinsa. Esimerkkejä tällaisista räätälöidyistä sovelluksista löytyy AUAI-järjestön kotivulta[1], samoin kuin lista syistä jotka ovat johtaneet Bayes-verkkojen käyttöönottoon[2].

Bayes-verkkotekniikoiden aktiivisiin kaupallisiin soveltajiin kuuluvat tällä hetkellä seuraavat yritykset:

[1] URL: http://www.auai.org/BN-Routine.html
[2] URL: http://www.auai.org/BN-Testimonial.html

- Applied Decision Analysis, URL: http://www.dpl.adainc.com/
- Decisioneering, URL: http://www.decisioneering.com/
- Hugin Expert, URL: http://www.hugin.dk/
- Knowledge Industries, URL: http://www.kic.com/
- Lumina Decision Systems, URL: http://www.lumina.com/
- Microsoft, URL: http://www.research.microsoft.com/research/dtg/
- Noetic Systems, URL: http://www.noeticsystems.com/
- Norsys Software, URL: http://www.norsys.com/home.html
- Palisade, URL: http://www.palisade.com/
- Prevision, URL: http://www.prevision.com/
- ThinkBank, URL: http://www.Thinkbank.COM/
- TreeAge Software, URL: http://www.treeage.com/
- Ultimode Systems, URL: http://www.ultimode.COM/index.html

5.2 Bayes-verkkotutkimus

5.2.1 Tutkimusryhmiä

Yksi merkittävimmistä Bayes-verkkotutkimusryhmistä on perinteisesti ollut professori Steffen Lauritzenin johtama tutkimusryhmä Aalborgin yliopistossa Tanskassa. Ryhmä on osallistunut aktiivisesti myös HUGIN-ohjelmistotuotteen (ks. luku 4.2) rakentamiseen. Muita merkittäviä tutkimusryhmiä Euroopassa löytyy mm. Suomesta, Belgiasta, ja Espanjasta (ks. allaoleva lista).

Maailmanlaajuisesti suurin tutkimusryhmä tällä hetkellä on luultavasti Microsoftilla, joka on haalinut koko 90-luvun ajan lupaavia tutkijoita tutkimuskeskukseensa. Microsoftin asema lienee vain vahvistumassa tulevaisuudessa, koska tutkimusryhmä on tuottanut jo useita menestyksellisiä Bayes-verkkosovelluksia (ks. luku 4.1), ja yhtiö on lisäksi lisäämässä tutkimusrahoitustaan moninkertaiseksi nykyiseen verrattuna. Yrityksen pääjohtajan Bill Gatesin lausunnot [54] kuvaavat myös vahvaa luottamusta Bayes-verkkotekniikkaan. Toinen merkittävä tekijä tutkimusalueella Yhdysvalloissa on NASA, jolla on ollut lukuisia menestyksellisiä Bayes-verkkoprojekteja.

Monash-yliopiston tutkimusryhmä Australiassa on tehnyt pioneerityötä sekajakaumamallien tutkimuksen alueella, samoin kuin CoSCo-ryhmä Helsingin yliopistossa.

Alla oleva lista sisältää joitakin WWW-linkkejä Bayes-verkkotutkimusta tekeviin ryhmiin Suomessa, Euroopassa, ja muualla maailmassa. Lista sisältää luonnollisesti vain muutaman esimerkin Bayes-verkkotutkimusta tekevistä tutkimusryhmistä; lisää hyödyllisiä WWW-linkkejä (mm. yli 100 nimeä käsittävä luettelo bayesiläistä mallintamista tutkivista henkilöistä[3]) löytyy AUAI-järjestön kotisivulta (ks. luku 5.2.4).

- Helsingin yliopisto
  URL: http://www.cs.Helsinki.FI/research/cosco/
- Aalborgin yliopisto,
  URL: http://www.cs.auc.dk/general/ES/
- Cantabrian yliopisto/Cornell University
  URL: http://ccaix3.unican.es/~AIGroup/
- IRIDIA,
  URL: http://iridia.ulb.ac.be/Projects/imple.html
- NASA
  URL: http://fi-www.arc.nasa.gov/fia/projects/bayes-group/
- Microsoft
  URL: http://www.research.microsoft.com/research/dtg/
- Monash University
  URL: http://www.cs.monash.edu.au/~dld/

5.2.2 Kirjallisuutta

Bayes-verkkokirjallisuuden klassikkona pidetään Pearlin vuonna 1988 julkaisemaa kirjaa [105], joka vielä hieman vanhentuneenakin sopii hyväksi johdatukseksi Bayes-verkkoteoriaan. Pearlin lähestymistapa perustuu pitkälti verkkoteoreettisiin käsitteisiin, kun taas lähes samaan aikaan julkaistu Neapolitanin oppikirja [97] tarjoaa matemaattisesti hieman pragmaattisemman, asiantuntijajärjestelmien rakentamiseen tähtäävän yleisesityksen. Charniakin artikkeli [19] antaa selkeän, ei-matemaattisen johdatuksen Bayes-verkkojen peruskäsitteisiin. Teknisempi lyhyt katsaus aiheeseen löytyy artikkelista [129].

[3] URL: http://bayes.stat.washington.edu/bayes_people.html

Edellä mainitut artikkelit, samoin kuin Pearlin ja Neapolitanin kirjat, ovat hieman vanhentuneita mm. siinä mielessä, että niissä ei juurikaan käsitellä Bayes-verkkojen oppimista. Uudempia johdatuksia Bayes-verkkoteoriaan tarjoavat Spirtesin et al. [130], Jensenin [60], Schaferin [118], ja Castillon et al. [18] kirjat, sekä Heckermanin artikkeli [49]. Spirtesin kirja käsittelee kausaalisia Bayes-verkkoja, ja niiden ei-bayesiläistä oppimista. Jensenin ja Schaferin kirjat keskittyvät kuvaamaan probabilistisen päättelyn toteuttamista Bayes-verkoissa, eivätkä käsittele juuri lainkaan Bayes-verkkojen oppimista. Castillon kirja on selkeää luettavaa, mutta Bayes-verkkojen oppimista käsittelevä luku sisältää joitakin epätarkkuuksia. Heckermanin artikkeli on hyvä yleiskatsaus Bayes-verkkotutkimuksen tuoreimpiin tuloksiin. Lauritzenin [84] ja Whittakerin [141] kirjat käsittelevät yleisemmin graafisiin verkkoesityksiin perustuvan mallintamisen teoriaa.

Bayes-verkkojen oppimiseen keskittyvää materiaalia löytyy lähteistä [50, 48, 16, 17]. Suomenkielisia katsauksia tutkimusalueeseen löytyy tutkielmista [122, 5]. Yleisemmin bayesiläistä mallinmuodostusta käsitellään lähteissä [22, 88, 96]. Näistä Mackay [88] ja Neal [96] keskittyvät baysiläisen lähestymistavan soveltamiseen neuroverkkojen oppimisessa. Muita neuroverkkojen ja bayesiläisen mallintamisen välisiä yhteyksiä käsitteleviä teoksia ovat [13, 14, 93, 117].

Teoreettisempia katsauksia todennäköisyyslaskennan soveltamistekniikoihin löytyy teoksista [41, 10, 59]. Hyviä johdatuksia päätösteoriaan löytyy lähteistä [32, 9, 38, 86]. Epätäsmällisyyden käsitettä yleisemmin käsitellään teoksissa [119, 102, 138]. Todennäköisyyslaskennan ja sumean logiikan suhdetta tarkastellaan artikkeleissa [21, 20].

Lisää viitteitä Bayes-verkkoihin liittyviin teoksiin löytyy seuraavilta WWW-sivuilta:

- URL: http://www.cs.pitt.edu/~tsamard/bnpointers.html
- URL: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/ldc/WWW/bayes-net-research/bayes-net-research.html
- URL: http://www.Ultimode.com/~wray/graphbib/
- URL: http://www.lis.pitt.edu/~dsl/da-books.html

Elektronisesti luettavia (HTML-formaatissa olevia) johdatuksia Bayes-verkkoteoriaan löytyy mm. seuraavilta WWW-sivuilta:

- URL: http://www.afit.af.mil/Schools/EN/ENG/LABS/AI/BayesianNetworks/

- URL: http://www.cm.cf.ac.uk/Dave/AI2/AI_notes.html
- URL: http://www.research.att.com/~dmac/notes/bayesian.html
- URL: http://yoda.cis.temple.edu:8080/UGAIWWW/lectures97/uncertainty/bindex.html

Lisäksi useiden Bayes-verkkotutoriaalien/kurssien kalvot ovat myös saatavilla elektronisessa muodossa. Näitä löytyy mm. seuraavilta WWW-sivuilta:

- URL: http://www.cs.Helsinki.FI/~myllymak
- URL: http://www.cs.Helsinki.FI/~tirri
- URL: http://www.research.microsoft.com/research/dtg/heckerma/heckerma.html
- URL: http://www.ultimode.COM/~wray/
- URL: http://www.auai.org/auai-tutes.html

5.2.3 Konferensseja ja lehtiä

Yksinomaan Bayes-verkkotutkimukseen keskittynyttä jokavuotista tieteellistä konferenssia ei toistaiseksi ole perustettu, joskin *Uncertainty in Artificial Intelligence (UAI)*-konferenssit ovat käytännössä muodostuneet sellaiseksi — lähes kaikki konferenssissa nykyään julkaistavat artikkelit käsittelevät Bayes-verkkoja. Bayesiläiseen mallinnukseen yleisesti liittyviä asioita käsitteleviä konferensseja ovat mm. *International Workshops on Artificial Intelligence and Statistics*, *Valencia International Meetings on Bayesian Statistics*, *International FLAIRS Conferences (Special Track on Uncertain Reasoning)*, ja *International Workshops on Maximum Entropy and Bayesian Methods*. Koska bayesiläinen mallinnus liittyy läheisesti koneoppimiseen ja muihin tekoälyn osa-alueisiim, julkaistaan alueeseen liittyviä artikkeleita paljon mm. seuraavissa konferensseissa: *Knowledge Discovery and Data mining (KDD)*, *International Conferences on Machine Learning (ICML)*, *International Joint Conferences on Artificial Intelligence (IJCAI)*, ja *National Conferences on Artificial Intelligence (AAAI)*. Neuroverkkokonferensseista ovat jokavuotiset *Neural Information Processing Systems (NIPS)*-konferenssit julkaisseet yhä kasvavassa määrin bayesiläiseen mallinnukseen liittyviä artikkeleita. Eurooppalaiset epätäsmällisen tiedon käsittelyyn liittyvät konferenssit, kuten esimerkiksi *International Joint Conferences on Qualitative and Quantitative Practical Reasoning (ECSQARU)*, *European Symposiums on Intelligent*

Techniques, ja *The European Congresses on Intelligent Techniques and Soft Computing (EUFIT)*, ovat keskittyneet pääasiassa ei-bayesiläisten lähestymistapojen käsittelemiseen.

Bayes-verkkotutkijat eivät toistaiseksi ole perustaneet omaa, Bayes-verkkotutkimukselle omistautunutta tieteellistä julkaisua, vaan alan tutkimustulokset ilmestyvät monissa eri tekoälyn osa-alueita käsittelevissä lehdissä, joita ovat mm. *Artificial Intelligence*, *Journal of Artificial Intelligence Research*, *Journal of Approximate Reasoning*, ja *Machine Learning*.

5.2.4 Järjestöjä

Koska edellä mainittu UAI-konferenssi on muodostunut Bayes-verkkotutkimusalueen tärkeimmäksi vuosittaiseksi tapaamiseksi, on konferenssin järjestävästä *Association for Uncertainty in Artificial Intelligence (AUAI)*-organisaatiosta luonnollisesti muovautunut eräänlainen Bayes-verkkotutkijoiden kattojärjestö. Muita merkittäviä alan tutkimukseen liittyviä järjestöjä ovat *International Society for Bayesian Analysis (ISBA)*, joka sponsoroi *Valencia International Meeting on Bayesian Statistics*- ja *International Workshop on Maximum Entropy and Bayesian Methods*-kokouksia, sekä *Society for Artificial Intelligence and Statistics*, joka järjestää joka toinen vuosi pidettävän *International Workshop on Artificial Intelligence and Statistics*-kokouksen. Muita tutkimusalueeseen liittyviä järjestöjä on lueteltu alla. Eurooppalaisista organisaatioista on huomautettava, että kuten eurooppalaiset alan konferenssit, myös tieteelliset organisaatiot (kuten esim. ERUDIT, NeuroCOLT, ja MLnet) ovat keskittyneet pääasiassa ei-bayesiläisiin epätäsmällisyyden käsittelytapoihin.

- Association for Uncertainty in Artificial Intelligence (AUAI),
  URL: http://www.auai.org/
- Society for Artificial Intelligence and Statistics,
  URL: http://www.vuse.vanderbilt.edu/~dfisher/ai-stats/society.html
- International Society for Bayesian Analysis (ISBA)
  URL: http://omega.albany.edu:8008/isba/
- ASA Section on Bayesian Statistical Sciences
  URL: http://www.isds.duke.edu/sbss/sbss.html
- European Association for Decision Making
  URL: http://huizen.dds.nl/~eadm/

- Decision Analysis Society
  URL: http://www.fuqua.duke.edu/faculty/daweb/
- Society for Risk Analysis (SRA)
  URL: http://www.sra.org/
- Classification Society of North America (CSNA)
  URL: http://www.pitt.edu/~csna/
- Network of Excellence for Uncertainty Modeling and Fuzzy Technology in the European Union (ERUDIT)
  URL: http://ss-m3.mitgmbh.de/erudit/
- ESPRIT Working Group in Neural and Computational Learning (NeuroCOLT)
  URL: http://www.cs.rhbnc.ac.uk/research/compint/neurocolt/
- ESPRIT Network of Excellence in Machine Learning (MLnet)
  URL: http://omega.gmd.de/ml-archive/MLnet/MLnet.html

Kirjallisuutta

[1] D. Aha. Editorial for a special issue on lazy learning. *Artificial Intelligence Review (to appear)*, 1997.

[2] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] S. Alag and A. Agogino. Inference using message propagation and topology transformation in vector gaussian continuous networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, pages 20–27, Portland, Oregon, 1996.

[4] S. Amari, N. Murata, K.-R. Müller, M. Finker, and H. Hua Yang. Asymptotic statistical theory of overtraining and cross-validation. *IEEE Transactions on Neural Networks*, 8(5):985–996, February 1997.

[5] T. Antti-Poika. Bayes-verkkojen käyttö luokittelussa. Master's thesis, Deparment of Computer Science, University of Helsinki, 1997.

[6] R.A. Baxter and J.O. Oliver. MDL and MML: Similarities and differences. Technical Report 207, Department of Computer Science, Monash University, 1994.

[7] Ann Becker and Dan Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, pages 81–89, Portland, Oregon, 1996.

[8] I. Beinlich, H. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conf. on Artificial Intelligence in Medicine*, pages 247–256, London, UK, August 1989.

[9] J.O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 1985.

[10] J.M. Bernardo and A.F.M Smith. *Bayesian theory.* John Wiley, 1994.

[11] G. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding: Turbo codes. In *Proceedings of the 1993 International Conference in Communications*, pages 1064–1070, geneve, May 1993.

[12] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[13] C.M. Bishop, M. Svensén, and C.K.I. Williams. EM optimization of latent-variable density models. In D.S. Touretzky, M.C.Mozer, and M.E.Hasselmo, editors, *Advances in Neural Information Processing Systems 8.* MIT Press, 1996.

[14] C.M. Bishop, M. Svensén, and C.K.I. Williams. GTM: A principled alternative to the self-organizing map. In *Advances in Neural Information Processing Systems 9 (to appear).* MIT Press, 1997.

[15] L. Breiman. Bias, variance, and arcing classifiers. Technical Report TR 460, University of California, Statistics Department, April 1996.

[16] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[17] W. Buntine. A guide to the literature on learning graphical models. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210, 1996.

[18] E. Castillo, J. Gutiérrez, and A. Hadi. *Expert Systems and Probabilistic Network Models.* Monographs in Computer Science. Springer-Verlag, New York, NY, 1997.

[19] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[20] P. Cheeseman. Probabilistic versus fuzzy reasoning. In Kanal and Lemmer [63], pages 85–102.

[21] P. Cheeseman. In defense of probability. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, volume 2, pages 1002–1009, 1995.

[22] P. Cheeseman. On Bayesian model selection. In D. Wolpert, editor, *The Mathematics of Generalization*, volume XX of *SFI Studies in the Sciences of Complexity*, pages 315–330. Addison-Wesley, 1995.

[23] P. Cheeseman, B. Kanefsky, R. Hanson, and J. Stutz. Super-resolved surface reconstruction from multiple images. Technical Report FIA-94-12, NASA Ames Research Center, Artificial Intelligence Branch, October 1994.

[24] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 6. AAAI Press, Menlo Park, 1996.

[25] D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, 1994.

[26] D.M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of incomplete data given a Bayesian network. In E. Horvitz and F. Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 158–168, Portland, Oregon, August 1996. Morgan Kaufmann Publishers.

[27] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[28] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, March 1990.

[29] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

[30] T. Dean and M. Wellman. *Planning and Control*. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[31] R. Dechter and I. Rish. A scheme for approximating probabilistic inference. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–97)*, pages 132–141, San Francisco, CA, 1997. Morgan Kaufmann Publishers.

[32] M.H. DeGroot. *Optimal statistical decisions*. McGraw-Hill, 1970.

[33] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[34] M. Druzdzel and M. Henrion. Efficient reasoning in qualitative probabilistic networks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 548–553, Washington, DC, July 1993. AAAI Press/MIT, Menlo Park, CA.

[35] B.S. Everitt and D.J. Hand. *Finite Mixture Distributions.* Chapman and Hall, London, 1981.

[36] J.H. Friedman. On bias, variance, 0/1–loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–78, 1997.

[37] N. Friedman and M. Goldszmidt. Discretizing continuous attributes while learning Bayesian networks. In L. Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 157–165. Morgan Kaufmann Publishers, 1996.

[38] P. Gärdenfors and N.-E. Sahlin, editors. *Decision, Probability, and Utility.* Cambridge University Press, New York, 1988.

[39] D. Geiger and D. Heckerman. A characterization of the Dirichlet distribution through global and local independence. Technical Report MSR-TR-94-16, Microsoft Research, November (revised February 1995) 1994.

[40] D. Geiger and D. Heckerman. Learning bayesian networks. Technical Report MSR-TR-95-02, Microsoft Research, December 1994.

[41] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis.* Chapman & Hall, 1995.

[42] Shaogang Gong and Hilary Buxton. Bayesian nets for mapping contextual knowledge to computational constraints in motion segmentation and tracking. In *British machine vision conference*, Guildford, Surrey, UK, September 1993.

[43] M. Gyllenberg, T. Koski, and M. Verlaan. Classification of binary vectors by stochastic complexity. Technical Report A5, University of Turku, Institute for Applied Mathematics, November 1994.

[44] M. Hassoun. *Fundamentals of Artificial Neural Networks.* MIT Press, Cambridge, Massachusetts, 1995.

[45] R. Hecht-Nielsen. *Neurocomputing.* Addison-Wesley Publishing Company, Reading, MA, 1990.

[46] D. Heckerman. Probabilistic interpretation for MYCIN's certainty factors. In Kanal and Lemmer [63], pages 167–196.

[47] D. Heckerman. A Bayesian approach to learning causal network. In P. Besnard and S. Hanks, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*, pages 285–295, Montreal, Canada, 1995.

[48] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, One Microsoft Way, Redmond, WA 98052, 1996.

[49] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79–119, 1997.

[50] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995.

[51] D. Heckerman and B. Nathwani. An evaluation of the diagnostic accuracy of Pathfinder. *Computers and Biomedical Research*, 25:56–74, 1992.

[52] D. Heckerman and R. Schachter. Decision-theoretic foundations for causal reasoning. *Journal of Artificial Intelligence Research*, 3:405–430, 1995.

[53] D. Heckerman and M. Shwe. Diagnosis of multiple faults: A sensitivity analysis. In D. Heckerman and A. Mamdani, editors, *Uncertainty in Artificial Intelligence 9*, pages 80–87. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[54] L. Helm. The future of software may lie in the obscure theories of an 18th century cleric called Thomas Bayes. *Los Angeles Times*, Monday, October 28 1996.

[55] M. Henrion. Uncertainty in artificial intelligence: Is probability epistemologically and heuristically adequate? In J. Mumpower, L. Philipps, O. Renn, and Uppuluri V., editors, *Expert Judgements and Expert Systems*, NATO ASI Series F: Computer and Systems Science, pages 106–129. Springer-Verlar, Berlin, 1987.

[56] M. Henrion. An introduction to algorithms for inference in belief nets. In M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 129–138. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1990.

[57] M. Henrion, M. Pradhan, B. Del Favero, K. Huang, G. Provan, and P. O'Rorke. Why is diagnosis using belief networks insensitive to imprecision in probabilities? In E. Horvits and F. Jensen, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Twelfth Conference*, pages 307–314, Portland, Oregon, August 1996. Morgan Kaufmann Publishers, San Francisco, CA.

[58] P. Isomursu, V. Niskanen, C. Carlsson, and P. Eklund. Sumean logiikan mahdollisuudet. Technical Report 34/93, Teknologian kehittämiskeskus (TEKES), 1993.

[59] E. Jaynes. Probability theory: The logic of science. Under construction. A draft copy can be dowloaded from 'ftp://bayes.wustl.edu/Jaynes.book'., 1996.

[60] F. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

[61] F. Jensen, H. Christensen, and J. Nielsen. Bayesian methods for interpretation and control in multi-agent vision systems. *Applications of Artificial Intelligence X: Machine Vision and Robotics, SPIE Proceedings Series*, 1708, 1992.

[62] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[63] L.N. Kanal and J.F. Lemmer, editors. *Uncertainty in Artificial Intelligence 1*. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1986.

[64] B. Kanefsky, J. Stutz, P. Cheeseman, and W. Taylor. An improved automatic classification of a Landsat/TM image from Kansas (FIFE). Technical Report FIA-94-01, NASA Ames Research Center, Artificial Intelligence Branch, May 1994.

[65] R.E. Kass and A.E. Raftery. Bayes factors. Technical Report 254, Department of Statistics, University of Washington, 1994.

[66] U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2:7–17, 1992.

[67] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995.

[68] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, 1993.

[69] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. Bayes optimal lazy learning. Technical Report NC-TR-97-031, ESPRIT Working Group 8556: Neural and Computational Learning (NeuroCOLT), 1997.

[70] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. A Bayesian approach for retrieving relevant cases. In P. Smith, editor, *Artificial Intelligence Applications (Proceedings of the EXPERSYS-97 Conference)*, pages 67–72, Sunderland, UK, October 1997. IITT International.

[71] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. A Bayesian approach to discretization. In *Proceedings of the European Symposium on Intelligent Techniques*, pages 265–268, Bari, Italy, March 1997.

[72] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. Comparing stochastic complexity minimization algorithms in estimating missing data. In *Proceedings of WUPES'97, the 4th Workshop on Uncertainty Processing*, pages 81–90, Prague, Czech Republic, January 1997.

[73] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. On the accuracy of stochastic complexity approximations. In *Proceedings of the Causal Models and Statistical Learning Seminar*, pages 103–117, London, UK, March 1997.

[74] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald. Comparing predictive inference methods for discrete domains. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 311–318, Ft. Lauderdale, Florida, January 1997. Also: NeuroCOLT Technical Report NC-TR-97-004.

[75] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald. On predictive distributions and Bayesian networks. In W. Daelemans, P. Flach, and A. van den Bosch, editors, *Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning (BeNeLearn'97)*, pages 59–68, Tilburg, the Netherlands, October 1997.

[76] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald. On predictive distributions and Bayesian networks. Technical Report NC-TR-97-032, ESPRIT Working Group 8556: Neural and Computational Learning (NeuroCOLT), 1997.

[77] P. Kontkanen, P. Myllymäki, and H. Tirri. Comparing Bayesian model class selection criteria by discrete finite mixtures. In D. Dowe, K. Korb, and J. Oliver, editors, *Information, Statistics and Induction in Science*, pages 364–374, Proceedings of the ISIS'96 Conference, Melbourne, Australia, August 1996. World Scientific, Singapore.

[78] P. Kontkanen, P. Myllymäki, and H. Tirri. Constructing Bayesian finite mixture models by the EM algorithm. Technical Report NC-TR-97-003, ESPRIT Working Group 8556: Neural and Computational Learning (NeuroCOLT), 1996.

[79] P. Kontkanen, P. Myllymäki, and H. Tirri. Predictive data mining with finite mixtures. In E. Simoudis, J. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 176–182, Portland, Oregon, August 1996.

[80] P. Kontkanen, P. Myllymäki, and H. Tirri. Some experimental results with finite mixture models. In *Book of Abstracts, First European Conference on Highly Structured Stochastic Systems*, pages 112–115, Rebild, Denmark, May 1996.

[81] P. Kontkanen, P. Myllymäki, and H. Tirri. Experimenting with the Cheeseman-Stutz evidence approximation for predictive modeling and data mining. In D. Dankel, editor, *Proceedings of the Tenth International FLAIRS Conference*, pages 204–211, Daytona Beach, Florida, May 1997.

[82] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 223–228, San Jose, CA, July 1992. MIT Press.

[83] S. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1992.

[84] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[85] Z Li and B. D'Ambrosio. Efficient inference in bayes nets as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1):55–81, 1994.

[86] D.V. Lindley. *Making Decisions*. John Wiley & Sons, London, second edition, 1992.

[87] Waldmann M. and K. Holyoak. Predictive and diagnostic learning within causal models: Asymmetries of cue competition. *Journal of Experimental Psychology: General*, 121:222–236, 1992.

[88] D. Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992.

[89] D. J. C. MacKay. Bayesian non-linear modelling for the prediction competition. In *ASHRAE Transactions, V.100, Pt.2*, pages 1053–1062, Atlanta Georgia, 1994. ASHRAE.

[90] R. McEliece, D. Mackay, and J.-F. Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *International Journal of Selected areas in Communication*, 1997. (to appear).

[91] G. McLachlan and K. Thriyambakam, editors. *The EM Algorithm and Extensions*. John Wiley & Sons, New York, 1997.

[92] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[93] P. Myllymäki. *Mapping Bayesian Networks to Stochastic Neural Networks: A Foundation for Hybrid Bayesian-Neural Systems*. PhD thesis, Report A-1995-1, Department of Computer Science, University of Helsinki, December 1995.

[94] P. Myllymäki and H. Tirri. Bayesian case-based reasoning with neural networks. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 422–427, San Francisco, March 1993. IEEE, Piscataway, NJ.

[95] P. Myllymäki and H. Tirri. Massively parallel case-based reasoning with probabilistic similarity metrics. In S. Wess, K.-D. Althoff, and M Richter, editors, *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Artificial Intelligence*, pages 144–154. Springer-Verlag, 1994.

[96] R. Neal. *Bayesian Learning for Neural Networks.* Number 118 in Lecture Notes in Statistics. Springer-Verlag, New York, NY, 1996.

[97] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley & Sons, New York, NY, 1990.

[98] J.O. Oliver and R.A. Baxter. MML and Bayesianism: Similarities and differences. Technical Report 206, Department of Computer Science, Monash University, December 1994.

[99] J.O. Oliver and D.J. Hand. Introduction to minimum encoding inference. Technical Report 205, Department of Computer Science, Monash University, July 1994.

[100] J.O. Oliver and D.J. Hand. On pruning and averaging decision trees. In A. Prieditis, editor, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 430–437. Morgan Kaufmann Publishers, 1995.

[101] R. Oliver and J. Smith. *Influence diagrams, belief nets and decision analysis.* John Wiley & Sons, 1990.

[102] J. Paris. *The Uncertain Reasoner's Companion: A Mathematical Prespective*, volume 39 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, Cambridge, 1994.

[103] I. Parsa. KDD-97 knowledge discovery and data mining tools competition. *Knowledge Discovery Nuggets*, 97(19), 1997. An electronic journal, available at `http://www.kdnuggets.com/`.

[104] J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[105] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[106] J. Pearl and T. Verma. A theory of inferred causation. In A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[107] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[108] W. Poland. *Decision Analysis with Continuous and Discrete Variables: A Mixture Distribution Approach.* PhD thesis, Department of Engineering Economic Systems, Stanford University, 1994.

[109] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society*, 49(3):223–239 and 252–265, 1987.

[110] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, New Jersey, 1989.

[111] J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[112] J. Rissanen. Information theory and neural nets. In P. Smolensky, M. Mozer, and D. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*, chapter 16. Lawrence Erlbaum Associates, 1996.

[113] J. Rissanen. Oppivien ja älykkäiden järjestelmien sovellukset 1994–1998, tutkimusprojektien arviointi. Technical Report Teknologiaohjelmaraportti 2/97, Teknologian kehittämiskeskus (TEKES), 1997.

[114] R. Robinson. Counting unlabeled asyclic graphs. In C. Little, editor, *Combinatorial Mathematics*, number 622 in Lecture Notes in Mathematics. Springer-Verlag, 1977.

[115] D. Roth. On the hardness of approximate reasoning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 1, pages 613–618, 1993.

[116] E. Santos and S. Shimony. Belief updating by enumerating high-probability independence-based arguments. In R. Lopez de Mantaras and D. Poole, editors, *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 506–513, Seattle, Washington, July 1994. Morgan Kaufmann Publishers, San Francisco, CA.

[117] L. Saul, T. Jaakkola, and M. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

[118] G. Schafer, editor. *Probabilistic Expert Systems.* Society for Industrial and Applied mathematics (SIAM), 1996.

[119] G. Schafer and J. Pearl, editors. *Readings in Uncertain Reasoning.* Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[120] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[121] D.W. Scott. *Multivariate Density Estimation. Theory, Practice, and Visualization*. John Wiley & Sons, New York, 1992.

[122] M. Seppänen. Bayes-verkon muodostaminen datasta. Master's thesis, Department of Computer Science, University of Helsinki, 1996.

[123] R.D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36(4):589–604, July-August 1988.

[124] S.E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68:399–410, 1994.

[125] M. Singh and M. Valtorta. An algorithm for construction of Bayesian network structures from data. In E. Horvits and F. Jensen, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Ninth Conference*, pages 259–265. Morgan Kaufmann Publishers, San Francisco, CA, 1993.

[126] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independece networks for hidden Markov probability models. Technical Report A.I. Memo 1565, C.B.C.L Memo 132, M.I.T, February 1996.

[127] M. Sorsa. Kvalitatiiviset bayesverkot. Master's thesis, Department of Computer Science, University of Helsinki, 1996.

[128] D. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[129] D. Spiegelhalter, P. Dawid, S. Lauritzen, and R. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8(3):219–283, 1993.

[130] P. Spirtes, C. Glymour, and R. Scheines, editors. *Causation, Prediction and Search*. Springer-Verlag, 1993. Out of print. Can be downloaded from 'URL: http://hss.cmu.edu/html/departments/philosophy/TETRAD.BOOK/book.html'.

[131] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[132] A.N. Tikhonov. On solving incorrectly posed problems and method of regularization. *Doklady Akademii Nauk USSR*, 151:501–504, 1963.

[133] H. Tirri. *Plausible Prediction by Bayesian Inference.* PhD thesis, Report A-1995-1, Department of Computer Science, University of Helsinki, June 1997.

[134] H. Tirri, P. Kontkanen, and P. Myllymäki. A Bayesian framework for case-based reasoning. In I. Smith and B. Faltings, editors, *Advances in Case-Based Reasoning*, volume 1168 of *Lecture Notes in Artificial Intelligence*, pages 413–427. Springer-Verlag, Berlin Heidelberg, November 1996.

[135] H. Tirri, P. Kontkanen, and P. Myllymäki. Probabilistic instance-based learning. In L. Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 507–515. Morgan Kaufmann Publishers, 1996.

[136] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley & Sons, New York, 1985.

[137] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[138] P. Walley. Measures of uncertainty in expert systems. *Artificial Intelligence*, 83:1–58, 1996.

[139] I. Watson and F. Marir. Case-based reasoning: A review. *The Knowledge Engineering Review*, 9(4):327–354, 1994.

[140] M.P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44(3):257–304, August 1990.

[141] J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* John Wiley & Sons, 1990.

[142] M. Yannakis. Computing the minimal fill-in is NP-complete. *SIAM Journal of Algebraic Discrete Methods*, 2:77–79, 1981.

[143] L. Zhang and D. Poole. Sidestepping the triangulation problem in Bayesian net computations. In *Proceedings of the Eight Annual Conference on Uncertainty in Artificial Intelligence*, pages 360–367, Palo Alto, CA, July 1992. Morgan Kaufmann Publishers.

Geneettisten algoritmien mahdollisuudet

Jarmo Alander

Luku 6

Evoluutio ja optimointi

> "Vaikuttaako valinnan laki, jonka olemme nähneet olevan niin voimakkaan aseen ihmisen kädessä, myöskin luonnontilassa?"
>
> Charles Darwin, *Lajien synty [55]*

Teollisuuden ja talouden toiminta perustuu paljolti, tai ainakin sen pitäisi perustua optimointiin: Kustannuksia pyritään karsimaan samalla kun tuotteiden laatua ja suorituskykyä maksimoidaan. Matemaattisesti tarkasteltuna optimointiongelmat ovat yleensä kovin vaikeita, usein jopa ratkeamattomia. Voidaan matemaattisesti todistaa, että monet käytännön ongelmat, jotka eivät päällisin puolin katsoen kovin kummoisilta näytä, ovat sellaisia, että niitä ei voida tunnetuilla matemaattisilla tai laskennallisilla menetelmillä tehokkaasti ratkaista. Tämä tieto on osaltaan johtanut siihen, että käytännössä vierastetaan matemaattista optimointitapaa ja ongelmat ratkotaan ammattitaitoon tukeutuen intuitiivisesti ja peukalosääntöjen avulla. On nähtävissä selvä juopa käytännön (teollisuuden suunnittelijat) ja teorian (korkeakoulujen ja tutkimuslaitosten tutkijoiden) välillä (kts. optimointikyselyn tulokset sivulla 158). Ihminen onkin ylivertaisen joustava ratkaisemaan käytännön ongelmia tietokoneen mekaaniseen laskentaan nähden. Hän ei kuitenkaan ole erehtymätön eikä pysty käsittelemään tehokkaasti ja virheettömästi suurempia tietomääriä. Lisäksi yksilöiden kyvyt tässäkin suhteessa vaihtevat niin kokemuksen kuin muidenkin henkilökohtaisten ominaisuuksien mukaan.

Tietotekniikka tarjoaa välineitä suurten tietomäärien käsittelyyn, tarkkojen laskelmien tekoon ja ennen muuta samojen rutiinien uupumattomaan toistamiseen. Minkä tietokone älyssä kiistatta häviää kompensoituu usein sillä, että se pystyy nopeasti käymään läpi lukemattomia vaihtoehtoja ja valitsemaan niistä parhaat. Yhdistämällä ihmisen ammattitaito tietokoneen muistiin ja laskentakapasiteettiin saadaan työpari, joka pärjää kovassakin kilpailussa.

Seuraavassa esiteltävä evolutiivinen tai geneettinen algoritmi on menetelmä, joka on helppo yhdistää ihmisen intuitiiviseen työskentelyyn käytännön optimointitehtäviä ratkottaessa. Paraskaan suunnitelma ei ole yleensä optimaalinen, jos ei sitä ole vartavasten jotakin tavoitetta varten optimoitu — eikä yleensä silloinkaan, kuten tullaan huomaamaan.

Esiteltävä menetelmä perustuu vaihtoehtojen tuottamiseen rekombinoimalla hyvin samaan tapaan kuin luonnossa tapahtuu perintötekijöiden uudelleenjärjestäytyminen. Tämä yhdistettynä huonojen ratkaisujen karsintaan, johtaa käytännössä siihen, että populaation keskimääräinen hyvyys hyvin todennäköisesti kasvaa vähitellen sukupolvien mukana. Lopputuloksena on usein annetut ehdot täyttävä riittävän hyvä ratkaisu käytännön optimointiongelmaan.

6.1 Evoluutio

Charles Darwin esitti 1800-luvun puolessa välissä teorian lajien vähittäisestä kehityksestä, joka selitti mm. lajien väliset ilmeiset sukulaisuussuhteet, jotka puolestaan Carl Linné oli esittänyt lajien hierarkisessa sukulaisuusluokittelujärjestelmässään jo edellisellä vuosisadalla.

Teoria herätti ilmestyttyään kiihkeää keskustelua puolesta ja vastaan, eikä tämä keskustelu ole vieläkään täysin laantunut, vaikka biologian, varsinkin molekyylibiologian, uudet löydöt ovat olleet hyvin sopusoinnussa Darwinin perusideoiden kanssa.

Darwin pystyi näkemään eläinjalostuksen, biologian ja paleontologian tarjoamassa aineistossa viitteitä lajien hitaasta, mutta selvästä kehityksestä. Luonnossa kehitys tapahtuu niin hitaasti, että sen suora havainnointi on hyvin vaikeaa. Varsin yleinen käsitys on, että luonto edustaa jotakin hyvin pysyvää ihmisen kulttuurin katoavaisten saavutusten rinnalla.

Tällä hetkellä keskustellaan joidenkin havaintojen mahdollisesti osoittavan eräiden eliöiden kehittyneen niin “nopeasti”, että kehitys olisi havaittavissa jo muutaman sukupolven aikana [216]. Tulokset eivät kuitenkaan ole kovin selviä.

Kasvin- ja eläinjalostuksessa on kuitenkin jo jopa vuosituhansia käytetty menestyksellisesti hyväksi eliöiden perimän muokkaamista mahdollisimman nopeasti valinnan avulla haluttujen ominaisuuksien kehittämiseksi silminnähtävin ja hämmästyttävin tuloksin. Hyvän esimerkin tarjoaa koirarotujen laaja kirjo. Suhteellisen lyhyessä ajassa on luotu mitä mielikuvituksellisempia rotuja, joita ei hevin samaksi lajiksi uskoisi, ainakaan ulkomuodon perusteella. Vastaavia esimerkkejä löytyy muitakin. Darwin itse harrasti lähinnä kyyhkysten jalostusta.

Kirjallisuutta [55, 57, 58, 59, 60, 91, 92, 93, 165, 120, 215]

6.2 Optimointi

Evoluutioteoria ja genetiikka tarjoavat menetelmiä hyödyntää yksinkertaisesti tietokoneiden suurta laskentakapasiteettia vaikeiden optimointiongelmien ratkaisemisessa. Harvalla tietotekniikan menetelmällä on miljardien vuosien kenttätestit takanaan niin kuin geneettisellä algoritmilla! Vielä hieman pitempään ja massiivisempaan testaukseen yltää vain ns. termodynaaminen menetelmä (engl. simulated annealing), jonka esikuva on kiinteän aineen jähmettymisprosessi. Ikivanhaa perua ovat myös hermoverkot noin 600 miljoonan vuoden iällään. Ei siis ihme, että moiset tietojenkäsittelymenetelmät herättävät jonkin verran kiinnostusta tietotekniikan tutkijoissa ja soveltajissa.

Käytännön optimointitehtävät ovat usein niin monimutkaisia, että tehokkaita matemaattisia menetelmiä niiden ratkaisemiseksi ei ole olemassa useinkaan edes teoriassa. Esimerkiksi analyyttiset menetelmät asettavat varsin tiukat vaatimukset optimoitavan funktion ominaisuuksille. Tehtävä voi olla sellainen, että sitä ei voida, ainakaan helposti, palauttaa esimerkiksi lineaariseksi, jolloin se olisi suhteellisen helposti ja tehokkaasti ratkaistavissa lineaarisen ohjelmoinnin menetelmin. Voi myös olla niin, että optimoitava on olemassa tietokoneohjelmana, laitteena, koejärjestelynä tms muodossa, jossa sitä on käytännössä mahdotonta matemaattisesti analysoida.

Tietokoneohjelman tapauksessa on kuitenkin aina olemassa mahdollisuus lähteä etsimään ratkaisua käyttäen tietokoneen suurta laskentakapasiteettia. Vain muutaman vaihtoehdon kyseessä ollen tilanne on helppo: lasketaan kaikki vaihtoehdot ja valitaan paras tulos. Käytännössä kaikkien vaihtoehtojen läpikäynti ei onnistu nopeimmallakaan tietokoneella vähänkään monimutkaisemmassa tapauksessa. Tästä pitää huolen ns. kombinatorinen räjähdys, joka tulee lähes aina yllättävän nopeasti vastaan.

Esimerkki 1 (Kombinatorinen räjähdys) *Tarkastellaan tehtävää, jossa pitäisi ratkaista optimaalinen järjestys n:lle työlle (tai vaikka järjestää kirjahyllyn n kirjaa). Yksinkertaisin tapa löytää optimi olisi käydä läpi kaikki mahdolliset n työn permutaatiot. Näitä on tunnetusti olemassa kaikkiaan $n! = n(n-1)(n-2)\cdot 1$ kappaletta. Oheisessa taulukossa on laskettu permutaatioiden lukumäärät eli kertomafunktion arvot, kun n vaihtelee välillä $[1, 20]$. Kuten huomataan, permutaatioiden lukumäärä kasvaa erittäin nopeasti ja* *on käytännössä saavuttamaton nopeimmillekin tietokoneille viimeistään, kun $n \approx 20$. Taulukossa on vertailun vuoksi esitetty funktio 3^n, joka myös kasvaa nopeasti, mutta jää kuitenkin nopeasti jälkeen kertomafunktion räjähdysmäisestä kasvusta.*

n	$n!$	$T_{n!}$	3^n	T_{3^n}
1	1	1μs	3	3μs
2	2	2μs	9	9μs
3	6	6μs	27	27μs
4	24	24μs	81	81μs
5	120	120μs	243	243μs
6	720	720μs	729	729μs
7	5040	5ms	2187	2,1ms
8	40320	40ms	6561	6,5ms
9	362880	363ms	19683	19,6ms
10	3628800	3,6s	59049	59ms
11	39916800	40s	177147	177ms
12	479001600	8min	531441	531ms
13	6227020800	1h 44min	1594323	1,6s
14	87178291200	1vrk	4782969	4,8s
15	1307674368000	15vrk	14348907	14,3s
16	20922789888000	242vrk	43046721	43s
17	355687428096000	11,3a	129140163	2min 9s
18	6402373705728000	203a	387420489	6min 20s
19	121645100408832000	3.857a	1162261467	19min 22s
20	2432902008176640000	77.140a	3486784401	58min 7s

Taulukko 6.1: Kombinatorinen räjähdys eli kertomafunktion $n!$ arvoja, kun $1 \leq n \leq 20$. Kolmannessa sarakkeessa on esitetty laskenta-aika T sillä oletuksella, että yhden permutaation laskentaan kuluu (epärealistisen lyhyt aika) yksi mikrosekunti ($= 10^{-6}$s). Neljännessä ja viidennessä sarakkeessa on vastaavasti funktio 3^n.

6.3 Monte Carlon malliin

Yksinkertaisin keino yrittää välttää käymästä kaikkia vaihtoehtoja läpi on hakea ratkaisua "vähän sieltä sun täältä" niin, että saadaan mahdollisimman kattava kuva hakuavaruuden eri osista. Kun näytepisteet arvotaan satunnaisesti esim. tasaista jakaumaa käyttäen, on kyseessä ns. Monte Carlo -menetelmä, jonka etuna on sen yksinkertaisuus: tarvitaan tuskin muuta kuin satunnaislukugeneraattori ja optimoitavan funktion laskennan suoritta-va rutiini. Etsintä voidaan lopettaa, kun riittävän hyvä ratkaisu on löytynyt tai etsintään varattu aika on käytetty loppuun, jolloin siihen asti lasketuista vaihtoehdoista paras saa kelvata. Menetelmän huonona puolena on se, että se ei mitenkään hyödynnä optimoitavan funktion ominaisuuksia, joten etsintä on yleensä varsin tehotonta. Kuitenkin jo muutaman vaihtoehdon läpikäynti voi tuottaa huomattavan parannuksen lähtötilanteeseen nähden ja olla siten kannattava. Ja joskushan sitä voi onnekkaan kohdalle sattua täysosuma...

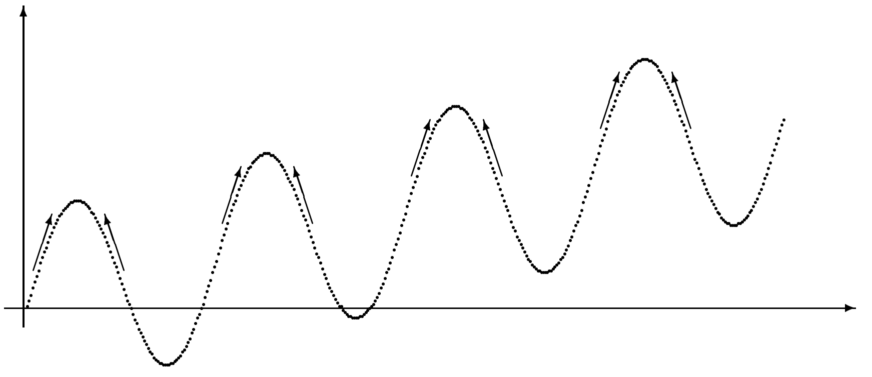
6.4 Kiipeilyä rinteessä

> Mikä nousee ja laskee mutta silti pysyy koko ajan paikallaan?
>
> – v: mäki,

Funktion arvoja tutkimalla voidaan usein ennakoida alueita, joihin etsintä ilmeisesti kannattaisi ensisijassa kohdistaa. Vaikka funktion tarkkaa käyttäytymistä ei tunnettaisikaan, niin yleensä sen arvot ainakin paikallisesti korreloivat keskenään, eli hyvän arvon lähellä on hyvin todennäköisesti vieläkin parempia eli osa etsintätyöstä kannattaa keskittää hyvien ratkaisujen ympäristöön. Suotuisassa tapauksessa voidaan paikallisuutta käyttää hyväksi turvautuen ns. hill-climbing -menetelmään eli pyritään etenemään suuntaan, jossa funktion arvo näyttäisi kasvavan (tai pienenevän) (kuva 6.1). Toistettaessa etenemistä joudutaan yleensä jo muutaman askeleen jälkeen paikalliseen optimiarvoon; ikään kuin oltaisiin kiivetty ylös kukkulan rinnettä huippua kohti.

Tähän periaatteeseen perustuvat useat perinteiset numeeriset optimointimenetelmät. Jos funktion derivaatat voidaan laskea, niin tällainen menetelmä on erityisen nopea, koska etenemisen suunta ja suuruus voidaan arvioida varsin tarkasti. Menetelmällä pyritään itse asiassa derivaatan nollakohtiin, joissa jatkuvasti derivoituvien funktioiden ääriarvot sijaitsevat, milloin eivät satu etsintäalueen rajalle.

Menetelmän kiusallinen ongelma on, että käytännössä optimoitava on usein monihuippuinen. Tällöin rinnettä ylös kapuamalla löydetäänkin useimmiten vain paikallinen optimi. Todellinen globaali optimi voi olla lähes missä vain. Aloittamalla kiipeily toistuvasti uudelleen eripuolilta hakuavaruutta, lopputulos on huomattavasti parempi. Tällöin on kyse ns. uudelleenaloitus (engl. restart) menetelmästä.



Kuva 6.1: Ns. hill-climbing -menetelmä.

Kirjallisuutta [97, 110, 211]

Luku 7

Geneettinen algoritmi

Jos paikalliset ääriarvot ovat riittävän säännönmukaisesti sijoittuneet, voidaan niistä yrittää ennustaa lupaavia uusia etsintäalueita. Jos säännönmukaisuus on tuntematon, kuten useimmiten monimutkaisen tehtävän tapauksessa valitettavasti on, ennustusmenetelmävaihtoehtoja on vähän. Yksi niistä on ns. geneettinen algoritmi (GA), joka perustuu siihen olettamukseen, että yhdistelemällä hyvien ratkaisujen osia satunnaisesti saadaan huomattavalla todennäköisyydellä vielä parempia ratkaisuja eli hyvä ratkaisu koostuu hyvistä osaratkaisuista.

Luonnon evoluutio näyttäisi perustuvan tähän ns. **rakennuspalikkahypoteesi**in (engl. building block hypothesis). Vanhemmilta perityt ominaisuudet voivat jälkeläisissä yhdistyä niin, että jälkeläiset pystyvät vanhempiaan paremmin kilpailemaan hyvin rajallisista resursseista. Luonnon evoluution hämmästyttävä kyky luoda hyvin monimutkainen ekosysteemi ja sen lukuisat eliöt on herättänyt tutkijoiden kiinnostuksen geneettisen algoritmin mahdollisuuksiin vaikeiden optimointiongelmien ratkaisijana monilla tieteen ja tekniikan aloilla.

Tekniikassa kannattaa muistaa, että rakennuspalikkahypoteesi ei saa tarkoittaa suoraviivaisesti sitä, että kaikki osat optimoidaan pelkästään jonkun teknisen ominaisuuden suhteen. Tällöin ehkä päädytään teknisesti hyvään tulokseen, kun kaikki komponentit ovat "viimeisen päälle", mutta taloudellisesti ratkaisu voi olla katastrofaalinen.

Toistaiseksi GA:n matemaattisessa teoriassa on edistytty vain vähän. Sen sijaan erilaisia menestyksellisiä sovelluksia on syntynyt kiihtyvällä vauhdilla. Myös suomalaistutkijat ovat olleet hyvin aktiivisia: Suomen GA tutkimuksen volyymi on julkaistujen artikkeleiden määrässä mitattuna moninkertainen muiden Pohjoismaiden yhteenlaskettuun tutkimukseen nähden ja samaa suuruusluokka kuin suurissa Euroopan Unionin teollisuusmaissa (taulukko 10.6 sivulla 167). Tutkimuskohteena ovat olleet sekä GA:n ominaisuudet että soveltaminen mm. ohjaukseen ja säätöön, koneen osien muodon optimointiin, hermoverkkojen optimointiin, signaalien käsittelyyn sekä laitetoteutukset. (kappale 10.9 sivulla 167).

7.1 Toimintaperiaate

GA:n ydin on helppo toteuttaa millä ohjelmointikielellä tahansa muutamalla kymmenellä tai sadalla ohjelmarivillä, kun mukaan ei lasketa optimoitavaa funktiota, joka sitten voi olla hyvinkin laaja ohjelma, esimerkiksi jonkin laitteen rakenteen ja/tai toiminnan simulaattori.

Geneettisen algoritmin toimintaperiaate on esitetty kuvassa 7.1. Tehtävän ratkaiseminen alkaa aloituspopulaation tuottamisesta. Yleensä yritteitä luodaan puhtaasti satunnaisesti, mutta tässä vaiheessa kannattaa toki käyttää hyväksi kaikkea ratkaisusta tiedettävää, jotta aikaa ei jatkossa tuhlaantuisi pelkästään huonojen vaihtoehtojen karsintaan ja hyvien osaratkaisujen ilmestymisen odotteluun. Erityisesti kannattaa kiinnittää huomiota populaation diversiteettiin eli monimuotoisuuteen: parametrien arvojen pitäisi olla suhteellisen tasaisesti jakautuneena hakualueella.

1. **Aloitus:** luo (satunnainen) joukko yritelmiä *(aloituspopulaatio)*.
2. **Valinta:** karsi huonoimmat yritelmät.
   Jos riittävän hyvä ratkaisu on löytynyt tai aika on loppunut, niin **lopeta**.
3. **Yhdistely:** yhdistele ratkaisuja satunnaisesti (**risteytys**)
   ja lisää tarvittaessa satunnaisuutta (**mutaatio**).
4. **Toisto:** palaa askeleeseen 2 (uusi **sukupolvi**).

Kuva 7.1: Geneettisen algoritmin runko.

7.1.1 Rekombinaatio

Ratkaisujen parametrien yhdistely tapahtuu ns. **crossover** -menetelmällä eli tekijöiden vaihdolla (kuva 7.2), joka matkii vastaavaa genetiikan ilmiötä: Valitaan tyypillisesti kaksi ratkaisuparametrien yhdistelmää (vanhemmat; yritteet) satunnaisesti käsillä olevasta joukosta eli populaatiosta ja vaihdetaan niiden kesken satunnaisista kohdista parametreja (kuva 7.2). Tällöin saadaan kaksi uutta parametrien kombinaatiota (jälkeläiset), joista toinen tai

a	B	C	d	e	F	G	0. vanha parametrikombinaatio
H	i	j	K	L	m	n	1. vanha parametrikombinaatio
			$\Downarrow$				
0	**1**	**1**	**0**	**0**	**1**	**1**	***crossover indeksi*** X
			$\Downarrow$				
a	i	j	d	e	m	n	0. uusi parametrikombinaatio
H	B	C	K	L	F	G	1. uusi parametrikombinaatio

Kuva 7.2: Crossover eli tekijöiden rekombinaatio.

molemmat otetaan uusiksi populaation jäseniksi. Ohjelmointiteknisesti rekombinaatio voidaan toteuttaa esimerkiksi indeksoimalla (C-kieli):

```
for (i=0;i<n; i++) B[k][i] = P[(X[i]?p1:p2)][i];
```

missä `B` on uusi ja `P` vanha yrite, `X` on rekombinaatioindeksivektori (alkiot satunnaisesti generoitu 0 tai 1), `i` (geenin) juokseva indeksi, `p1` ja `p2` vanhempien indeksit, `k` jälkeläisen indeksi sekä `n` kromosomin eli yritevektorin pituus. Usein samalla kertaa/vaivalla luodaan myös toinen jälkeläinen, joka on ensimmäisen komplementti:

```
for (i=0;i<n; i++) B[k+1][i] = P[(X[i]?p2:p1)][i];
```

Käytännössä erillinen `B`-taulukko ei yleensä ole välttämätön, koska tulos viedään joka tapauksessa populaatioon eli taulukkoon `P`.

Luonnossa vanhemmilta saadut kromosomit yhdistyvät samanlaisen tekijöiden vaihdon kautta. Tällöin yleensä vaihto alkaa vain parista kohtaa kromosomia. Miksi näin vähän risteyskohtia on toistaiseksi varsin epäselvää. Luontevin selitys on ehkä se, että luonnossa tilanne on varsin hyvin tasapainossa, joten kovin suurta painetta dramaattisesti uusien rekombinaatioiden generoimiseksi ei ole. Evoluutiohan tapahtuu luonnossa, kuten jo aiemmin todettiin, kovin hitaasti.

GA:ssa käytetään yleensä hyvin satunnaista tekijöiden vaihtoa, eli paljon risteämispisteitä (ns. uniform crossover), koska halutaan nopeasti kehittyvä hyvin monipuolinen joukko uusia yritteitä. Tällöin vaihto voidaan suorittaa jopa biteittäin, jolloin rekombinaatio luo esimerkiksi uusia kokonaislukuja bittivektoreita rekombinoimalla.

7.1.2 Mutaatio

Toinen geneettinen operaatio on **mutaatio**, jolla lisätään satunnaisesti diversiteettiä populaation geeniainekseen. Ilman mutaatiota valinta vähitellen karsii pois kaikki muut vaihtoehdot paitsi toistaiseksi parhaan löydetyn yksilön geenit eli etsintä pysähtyisi vääjäämättä paikalliseen optimiin. Liiallinen mutaatio taas hävittää löydettyjä hyviä elinkelpoisia kombinaatioita, joten mutaation todennäköisyys on käytännössä oltava suhteellisen pieni ainakin optimoinnin loppuvaiheessa.

Mutaatiota pidetään yleisesti luonnon evoluution perustekijänä valinnan ohella. Todellisuudessa monet mekanismit solussa pyrkivät aktiivisti korjaamaan DNA:han tulleita virheitä. Itse asiassa nämä mekanismit ovat niin tehokkaita, että geneettisen informaation merkittävin piirre on sen pysyvyys. Kun tietokoneen levyke voi kymmenessä vuodessa muuttua lukukelvottomaksi, niin DNA:sta voidaan löytää jaksoja, jotka ovat miljoonia vuosia säilyneet lähes muuttumattomina, huolimatta miljoonista kopioitumisista ja niiden aikana vääjäämättä ja moneen kertaan tapahtuneista mutaatioista!

Luonnossa useimmat mutaatiot ovat haitallisia ja karsiutuvat yleensä hyvin nopeasti.

7.1.3 Kustannusfunktio

Yritteiden hyvyys (optimaalisuus) arvioidaan ns. hyvyys- tai kustannusfunktion (engl. fitness function) avulla, joka siis on optimoitava funktiomme. Se voi olla numeerinen funktio tai hyvinkin monimutkaisen prosessoinnin tulos. Optimoitava voi olla myös testattava laite tai jopa ihmisen antama arvio. Jälkimmäistä tapaa on käytetty lähinnä tuotettaessa tietokonegrafiikkaa: Tietokoneen ehdottamista fraktaali- tms. kuvioista, joilla esimerkiksi yritetään jäljitellä vaikka puun kaarnaa, ihminen valitsee ne, jotka eniten muistuttavat kohdetta.

Yleensä kustannusfunktio on optimoinnin eniten laskentaa vaativa osa, jota joudutaan laskemaan tuhansia tai jopa miljoonia kertoja toivotun lopputuloksen saamiseksi.

7.1.4 Valinta

Etsinnän kohdistaminen lupaavimmille alueille tapahtuu karsimalla kustannusfunktion mukaan huonoimmat yritteet populaatiosta. Valintatapoja on useita, mutta tehokkainta lienee valita noin 20-70% parhaista yritteistä jatkoon. Tätä tapaa kutsutaan elitistiseksi, koska se takaa aina parhaan ratkaisun pääsyn jatkoon.

Elitismi on ehdottoman hyvä, jos optimoitava funktio on laakea ja mahdollisesti vain yksihuippuinen. Monen huipun tapauksessa vaarana on ennenaikainen konvergenssi (engl. premature convergence) paikalliseen optimiin, koska hyvä arvo voi nopeasti vallata populaation ja täten tuhota monimuotoisuuden.

7.2 Analyysi

Tarkastellaan sitten geneettisen algoritmin tärkeimpiä ominaisuuksia ja analysointimenetelmiä.

7.2.1 Ominaisuuksia

Menetelmän etuina on riippumattomuus optimoitavan funktion ominaisuuksista ja kyky löytää hyviä ratkaisuvaihtoehtoja vaikeissakin tehtävissä. Se on myös helppo hajauttaa laskettavaksi usealla prosessorilla eikä toteuttaminen vaadi yleensä kovin paljon ohjelmointia.

Vastaavasti menetelmän heikkouksia ovat sen ominaisuuksien vajavainen tunteminen ja suhteellisen pitkä prosessointiaika, joka kuitenkin kuluu lähes kokonaan optimoitavan funktion laskentaan, ei itse optimointialgoritmiin, joka on perusmuodossaan hyvin yksinkertainen ja sopii jopa joihinkin onlinesovelluksiin, mikäli hyvyysfunktio pystytään laskemaan riittävän nopeasti.

Pääosa GA:n sovelluksista on enemmän tai vähemmän todellisia tekniikan, luonnontieteiden ja taloustieteiden optimointiongelmia, joihin on ollut vaikea löytää muita yhtä yleispäteviä ratkaisumenetelmiä. Mikään yleispätevä kaikki ongelmat ratkaiseva “viisastenkivi” geneettinen algoritmikaan ei kuitenkaan ole vaan sen menestyksellinen käyttö vaatii tiettyä yleistä optimointiajattelutavan hallintaa.

7.2.2 Matemaattinen analyysi

Vaikka geneettisen algoritmin toimintaperiaate onkin hyvin yksinkertainen, niin silti sen matemaattinen analysointi ei ole ollenkaan helppoa. Perussyynä tähän on menetelmän epälineaarisuus ja diskreettisyys. Aihetta ei myöskään

hyvää	*huonoa*
robusti	huonosti tunnettu
yleinen	hidas
joustava	vaatii optimointitietämystä
yksinkertainen	ei-analyyttinen
ratkaisuja jatkuvasti saatavilla (ns. anytime algoritmi)	stokastinen

Taulukko 7.1: Geneettisen algoritmin tärkeimmät ominaisuudet.

ole vielä kovin paljon tutkittu poikkeuksena muutama alaan erikoistunut teoreetikko. Erityisesti Markovin ketjuja on käytetty analysoinnissa. Koska teoreettisilla tuloksilla ei vielä näytä olevan suurempaa merkitystä käytännön sovellusten kannalta, emme tässä enempää puutu GA:n teoriaan. Kiinnostunut lukija voi aloittaa teoriaan tutustumisen esimerkiksi seuraavien lähteiden avulla:

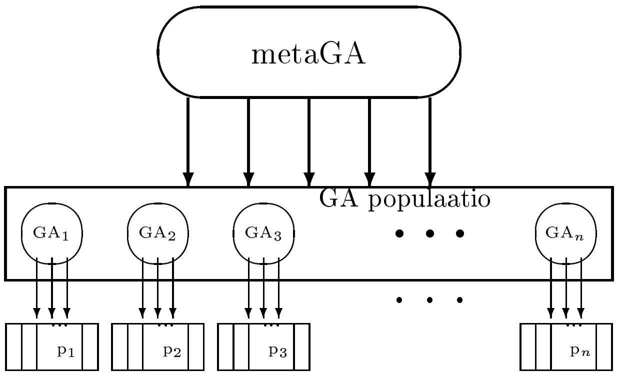
Kirjallisuutta [34, 87, 155, 169, 38, 37, 75, 116, 137, 144, 143, 154, 175]
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaTHEORYbib.ps.Z`

7.2.3 Kokeellisia tuloksia

Koska geneettisten algoritmien matemaattinen analysointi on vaikeaa, jäljelle jää tutkia niitä kokeellisesti käytännön sovelluksia silmällä pitäen. Tätä onkin harrastettu kohtalaisen innokkaasti jopa niin, että tyypillinen soveltaja mielellään ottaa kantaa GA:n toimintaan ja sen parametrien arvoihin usein varsin vaatimattomaan aineistoon tukeutuen, mikä kannattaa huomata kirjallisuuteen perehtyessä.

Seuraavassa esitetään kirjoittajan aikoinaan tekemää koetta, jossa geneettinen algoritmi optimoi geneettisen algoritmin tärkeimpiä parametreja. Vastaavia kokeita on tehty muitakin, mutta mitään seuraavassa esitettävästä dramaattisesti poikkeavaa ei ole ilmennyt.

Koejärjestelyssä populaatio koostui geneettisistä algoritmeista tai oikeammin niiden parametreista. Hyvyys alemman tason GA:lle laskettiin sen mukaan, miten nopeasti ne keskimäärin ratkaisivat niille annetun yksinkertaisen tehtävän (kaikille sama muutaman solmun kauppamatkustajan ongelma) (kuva 7.3). Optimoitavat parametrit ja niiden vaihteluvälit on esitetty taulukossa 7.2.



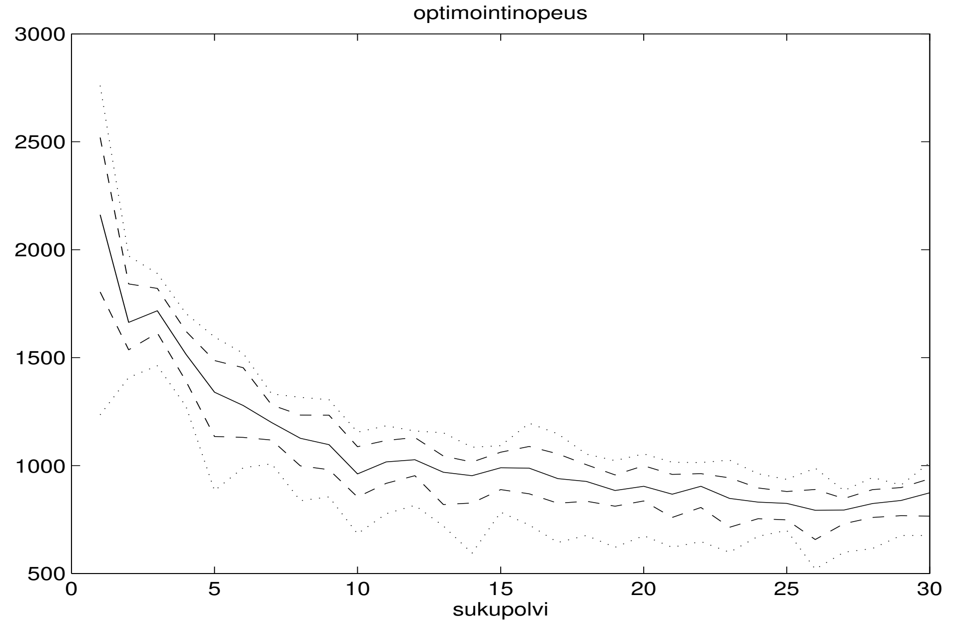
Kuva 7.3: Geneettinen algoritmi (metaGA) optimoimassa geneettisen algoritmin (GA_i) parametreja (p_i).

parametri	*ala-raja*	*ylä-raja*	*bittiä*
populaation koko	0	127	7
mutaatiotiheys	0	15	4
crossingtiheys	0	31	5
liikkuvuus	0	15	4
minimi Hamming-etäisyys	0	15	4
alipopulaatiot	1	8	3
elitismi	25%	100%	2
yhteensä			29

Taulukko 7.2: Geneettisen algoritmin optimoidut parametrit ja niiden arvoalueet.

Optimointinopeus

Kuvassa 7.4 on esitetty optimoitavan geneettisen algoritmin optimointinopeuden (hyvyysfunktio) kehittyminen. Kuten nähdään kehitystä tapahtuu, alussa nopeastikin, mikä ei ole sinänsä mikään ihme, koska optimointi aloitettiin täysin satunnaisesta populaatiosta. Oleellista on kuitenkin, että mitään kovin dramaattista kehitystä ei näytä tapahtuvan, kun alkuhämmennyksestä on selvitty. Kääntäen tämän voi tulkita niin, että menetelmä toimii kohtuullisen hyvin jopa alkutilanteen keskimääräisillä satunnaisilla parametrien arvoilla eli, että se ei ole erityisen herkkä parametriensa arvoille. Tämä on hyvä asia käytännössä, koska se merkitsee, että käyttäjän ei yleensä tarvitse olla kovin huolissaan geneettisen algoritminsa parametrien arvoista, ellei kyse ole aivan kriittisestä sovelluksesta.



Kuva 7.4: GA:n keskimääräisen optimointinopeuden kehittyminen sukupolvien funktiona. Tässä ja muutamassa seuraavassa kuvassa käytetyt merkinnät: — keskiarvo, - - - keskimääräinen poikkeama, $\cdots$ minimi ja maksimi.

Crossing- ja mutaatiotiheys

Kuvassa 7.5 on esitetty crossover-tiheyden kehittyminen sukupolvien mukana. Kuten huomataan, mitä enemmän risteämispisteitä (= uniform crossover) sitä parempi. Tämä optimiarvo löytyy varsin nopeasti.

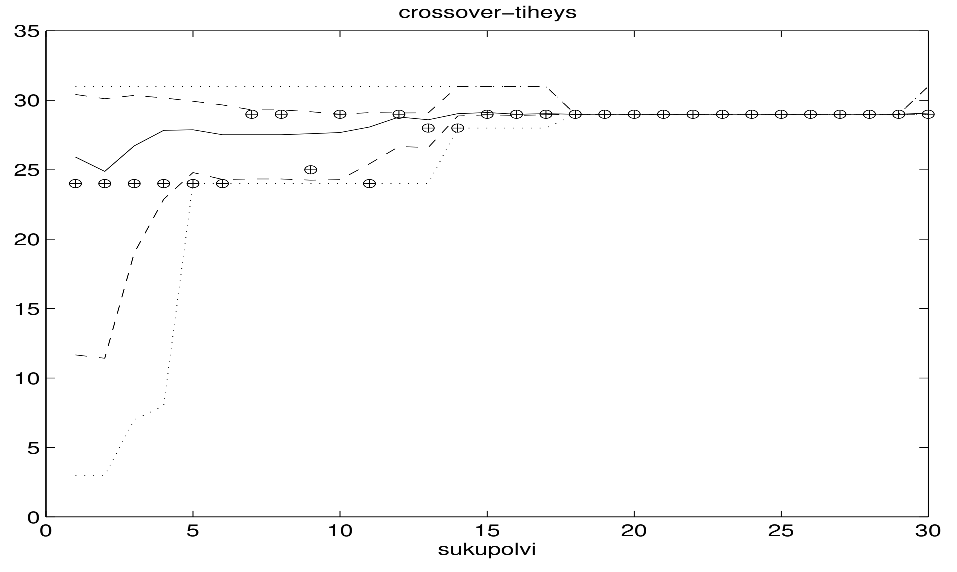
Kuvassa 7.6 on esitetty vastaava mutaatiotiheyden kehittyminen. Mutaatiotiheys pienenee nopeasti optimoinnin edetessä, mikä vastaa hyvin ennakkoarviota.

Muut parametrit

Muiden samassa työssä testattujen parametrien merkitys lienee vähäisempi, koska niiden arvot suppenivat tehdyissä kokeissa hitaammin.

Vaihtuvuus

Kuvassa 7.7 on esitetty (vasemmalla) yksilöiden iän kehittyminen: yksilöiden vaihtuvuus on populaatiossa melkoista. Keski-ikä on vain hieman yli yksi sukupolvi ja maksimissaankin luokkaa viisi sukupolvea. Menetelmä ei



Kuva 7.5: Crossover-tiheyden kehittyminen. Merkinnät kuten edellisessä kuvassa; lisäksi $\oplus$ sukupolven paras yksilö.

näytä jäävän paikalliseen optimiin, mikä olisikin epätodennäköistä, koska optimoitava funktio (siis se miten kauan optimoitava GA kulloinkin ratkaisee annettua tehtävää) on epädeterministinen eli vaihtelee satunnaisesti laskentakerrasta toiseen.

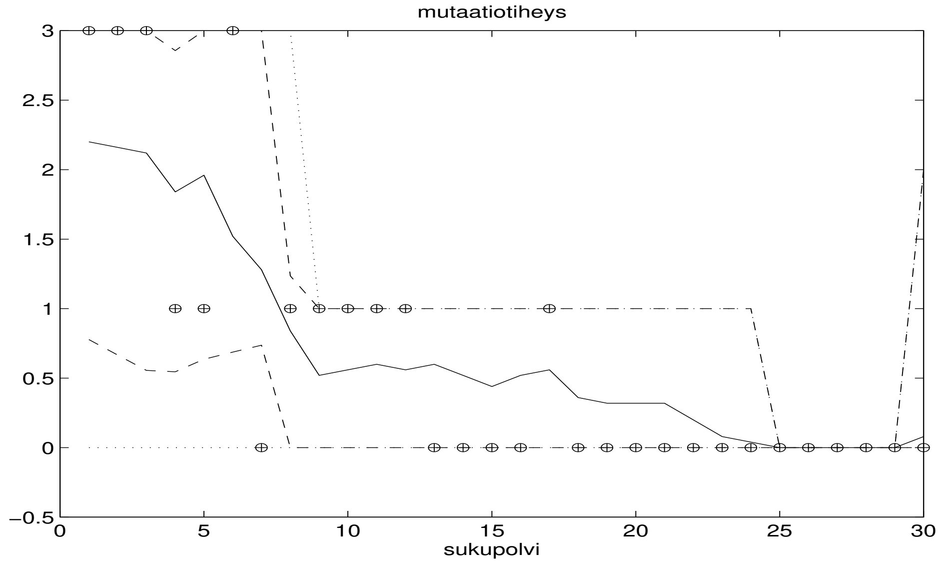
Diversiteetti

Kuvassa 7.7 oikealla on esitetty populaation jäsenten välisen Hamming-etäisyyden summan kehittyminen. Hamming-etäisyydellä tarkoitetaan yksilöiden toisistaan eroavien bittien lukumäärää. Sitä voidaan käyttää populaation monimuotoisuuden eli diversiteetin mittana.

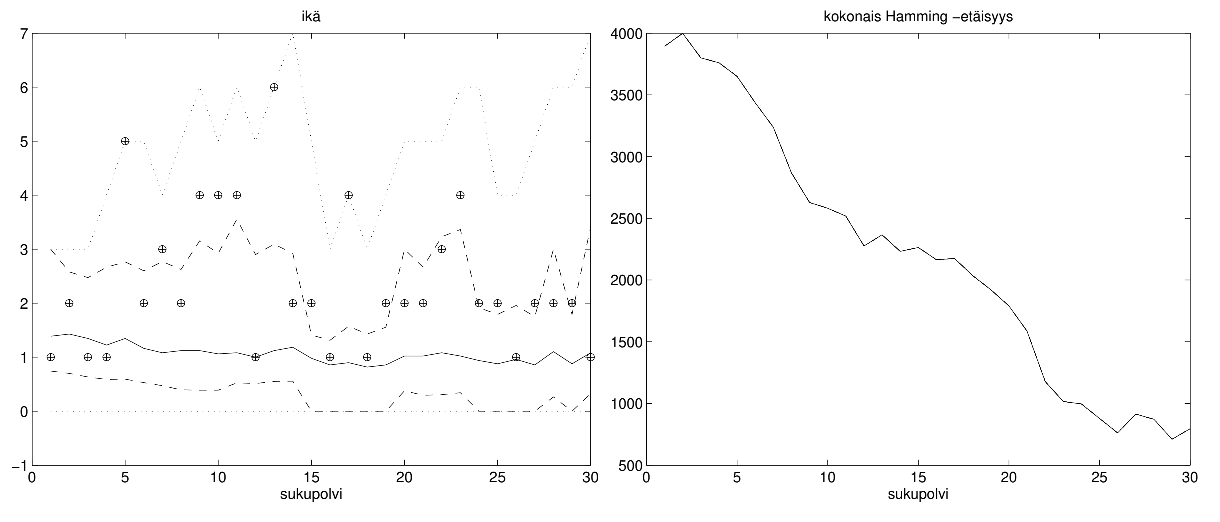
Kuten havaitaan, populaation diversiteetti pienenee jatkuvasti optimoinnin kuluessa eli yksilöt alkavat yhä enemmän muistuttaa toisiaan. Lopulta kaikki yksilöt ovat hyvin samanlaisia, eli lähellä löydettyä optimia. Siinä tilanteessa uusille alueille päästään ainoastaan mutaation avulla.

Populaation koko

Kuvassa 7.8 on esitetty populaation koon optimaalisen arvon (noin 30) löytyminen. Käytännössä koko määräytyy monen tekijän, niiden joukossa ratkaistavan ongelman vaativuuden mukaan, mutta nyrkkisääntönä voidaan todeta, että muutaman kymmenen yksilön populaatio on useimmiten järkevä.

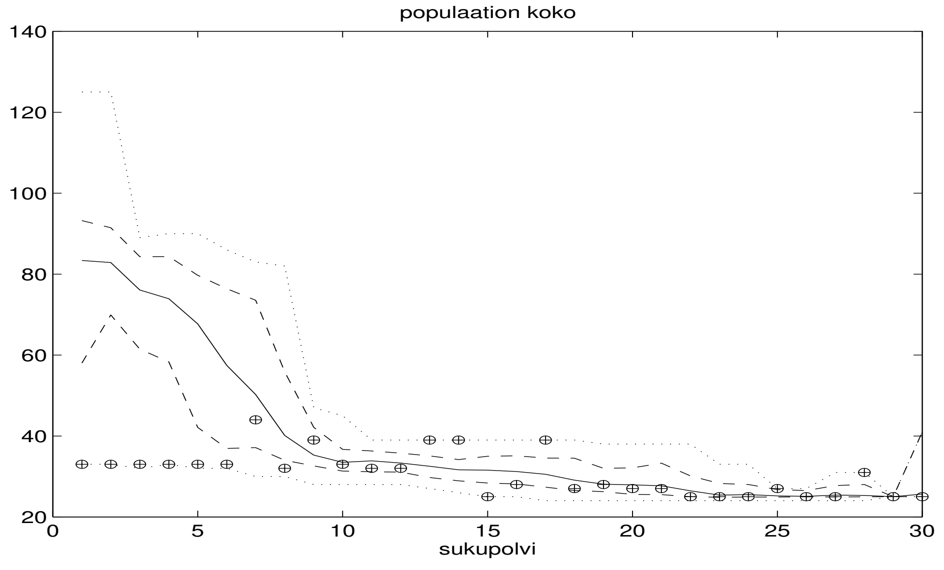


Kuva 7.6: Mutaatiotiheyden kehittyminen. Merkinnät kuten edellä.



Kuva 7.7: Iän ja Hamming-etäisyyden summan kehittyminen.

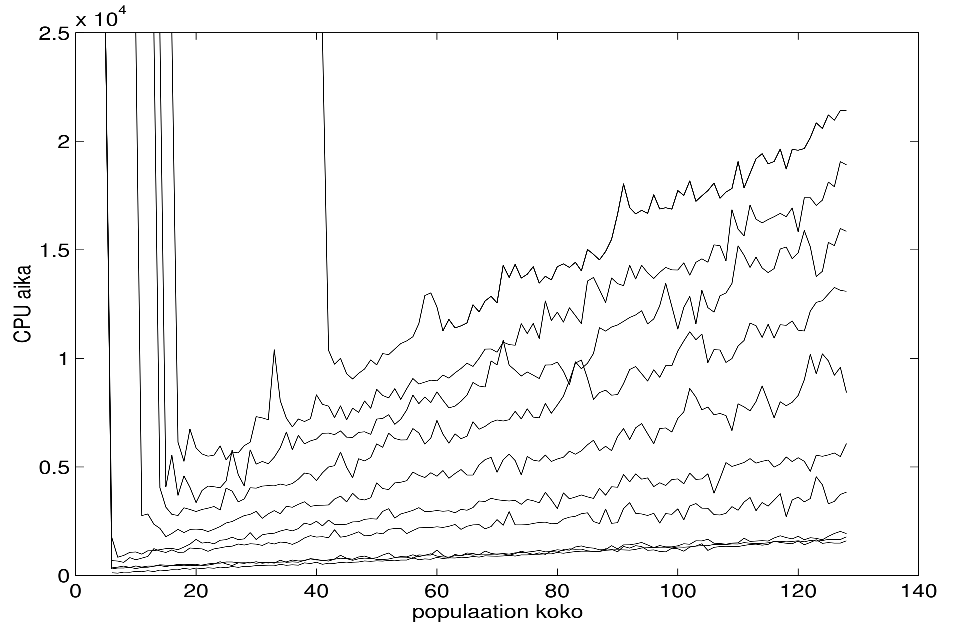
Tyypillinen tutkimuksissa käytetty populaation koko on ollut 50, jota voikin suositella tyypillisessä optimointitehtävässä, jossa optimoitava ei ole erityisen hankala. Vaikean ongelman tapauksessa on syytä kasvattaa populaation kokoa riittävän diversiteetin takaamiseksi. Vastaavasti laakean ja sileän funktion tapauksessa hyvinkin pieni populaatio on riittävä. Jopa pari yritettä saattaa olla riittävä [27]. Tällöin geneettisen algoritmin toiminta on hyvin lähellä satunnaishakua tai termodynaamista menetelmää.



Kuva 7.8: Optimaalinen populaation koko.

Populaation koon riippuvuutta tehtävän vaikeudesta tai monimutkaisuudesta kokeiltiin etsimällä tiettyä bittikuviota ja muuttamalla bittijonon pituutta. Hyvyysfunktiona oli oikeiden bittien lukumäärä. Kuvassa 7.9 on esitetty näin saatu käyräjoukko. Kuten huomataan, hyvin pienillä populaatioilla tehtävä vaatii huomattavan kauan aikaa (tai ei valmistu ollenkaan, kun diversiteetti ei ole riittävä; kehitys on täysin satunnaisen evoluution varassa). Populaation kasvattaminen taas puolestaan lisää laskenta-aikaa suurin piirtein lineaarisesti, kun käytetään yhtä prosessoria. Näiden kahden alueen välille muodostuu varsin selvä optimaalisen populaation koon alue, jonka sijaintia on kuitenkin hieman vaikea arvioida, koska se on riskialttiisti välittömästi hyvin huonon alueen (liian pieni populaatio) vieressä. Kannattaa siis varmuuden vuoksi käyttää mieluummin hieman suurempaa populaation kokoa kuin liian pientä. Jos käytettävissä on useampia prosessoreita, populaation koon kasvattamisesta aiheutuva ylimääräinen laskenta-aika saadaan pienennettyä ja optimin alue on paljon laakeampi ja täten myös laajempi ja siitä syystä vähemmän herkkä populaation koolle.

Mitä edellä esitetyt kokeet kertovat? Lähinnä, että geneettisen algoritmin parametreilla näyttäisi olevan tietyt optimiarvot, mutta toisaalta se ei onneksi ole kovinkaan herkkä näiden parametrien vaihteluille (kuva 7.4). Lisäksi GA näyttää sopivan kohtuullisen helposti esimerkiksi tämän tyyppisten



Kuva 7.9: Laskenta-ajan riippuminen populaation koosta ja tehtävän kompleksisuudesta. Tehtävänä oli etsiä 8 (alin käyrä) , 10, 12, 14, 16, 20, 24, 26, 28 ja 32 (ylin käyrä) pituisia bittijonoja. Hyvyysfunktiona oli oikeiden bittien lukumäärä.

optimointitehtävien ratkaisemiseen, siis vaikka optimoimaan omia parametrejaan.

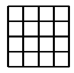
On huomattava, että optimaalinen populaation koko riippuu myös varsinaisen geneettisen algoritmin ytimen ja hyvyysfunktion laskennan suhteellisesta vaativuudesta ja että suoritetuissa testeissä ohjelmassa oli mukana mm. erilaista tilastointiin liittyvää toimintoa, jotka myös osaltaan vaikuttavat ohjelman ajankäyttöön.

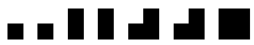
Kirjallisuutta [1, 3, 4, 5, 220]

7.3 Esimerkki: Sijoittelutehtävän ratkaiseminen

Tarkastellaan geneettisen algoritmin yksinkertaista sovellusta ja käydään läpi kaikki oleelliset vaiheet, joita tyypillisessä optimointitehtävän ratkaisemisessa on.

Tehtävä

Peitetään neliön muotoinen 4×4 ruudun alue  seuraavilla 7 palalla eli hienommin polyominoilla:



Tehtävä on siis varsin yksinkertainen kombinatorinen optimointi, jonka pienikin LEGO™-palikoilla leikkivä lapsi pystyy nopeasti ratkaisemaan. Se on valittu lähinnä helpon visualisoinnin vuoksi. Toki tämän tyyppisiä optimointitehtäviä löytyy myös tuotantotoiminnasta, kun esimerkiksi halutaan leikata levymateriaalia niin, että hukka on mahdollisimman pieni tai vaikka optimoidaan sävykuvan rasterointia painotekniikassa.

Tietorakenne

Geneettistä algoritmia, ja yleensäkin tietokonekäsittelyä varten, pitää tehtävä "koodata" eli suunnitella tietorakenne, joka kuvaa ongelmaa ja joka on lisäksi helppo tietokoneella toteuttaa. Käytetään seuraavaa varsin luonnollista kuvaustapaa: kunkin palan vasemman alanurkan koodinaateista ($x = (0, \dots, 3), y = (0, \dots, 3)$) ja palan suunnasta koordinaattiakselien suhteen ϕ (1 (pienet neliöt eli monominot ▪ ja iso neliö eli tetromino ■), 2 (suorakaiteet eli dominot ▮) ja 4 (haat eli kulmatrominot ▟) vaihtoehtoa) muodostetaan vektori (yhteensä $3 \times 4 + 2 \times (1 + 4) + 2 \times (2 + 4) = 34$ bittiä). Tällä koodaustavalla meillä on periaatteessa $2^{34} \approx 16 \times 10^9$ mahdollista vaihtoehtoa. Vastaavasti käypiä ratkaisujakin on koko joukko, koska palasia voidaan vaihtaa keskenään useilla tavoilla. Karkean arvion mukaan ratkaisuja on ilmeisesti ainakin noin 7000 eli ratkaisun löytymisen todennäköisyys pelkällä satunnaishaulla on luokkaa 10^{-6}.

Kukin yrite kuvataan kokonaislukutaulukkona siten, että kutakin palaa varten on oma alkio (kuvan 7.10 taulukoiden 2. sarake). Yhtä hyvin yrite voitaisiin kuvata 34 bitin mittaisella bittijonolla eli yhdellä pitkällä kokonaisluvulla. Rekombinaation yhteydessä kuvassa 7.10 on esitetty neljän yritteen tiedot ja palojen peittämä alue graafisesti.

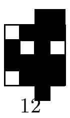
Hyvyysfunktio

Seuraava tehtävä on suunnitella hyvyysfunktio, joka kertoo karkeasti miten lähellä ratkaisua ollaan eli optimoitava funktio. Esimerkkimme tapauksessa luonteva sijoittelun kriteeri on peitetyn alueen pinta-ala. Tässä pitää ottaa huomioon kaksi erikoistapausta: kun pala on osittain peitettävän alueen ulkopuolella tai kaksi palasta on ainakin osittain päällekkäin. Voimme esimerkiksi sakottaa alueelta poistumisesta negatiivisella pinta-alalla samoin kuin päällekkäisyydestä. Sovitaan kuitenkin, että hyvyysfunktio saadaan yksinkertaisesti laskemalla alueen peitettyjen ruutujen määrä. Hyvyys eli peitto vaihtelee silloin välillä 1 (= kaikki palaset ruudukon oikeassa ylänurkassa) ja 16 (= kaikki ruudut peitetty), joka on tehtävämme ratkaisu. Näistä rajoista laskien hyvyyden vaihtelualueen keskiarvo on 8,5.

Kuten usein kombinatorisissa tehtävissä, hyvyysfunktio on varsin karkea, esimerkissämme 16 mahdollista arvoa, joilla luokitellaan lukemattomat parametrikombinaatiot. Esimerkiksi nollavektorin hyvyys on 4 ja tällöin kaikki palat ovat ruudukon vasemmassa alanurkassa peittäen (2×2) alueen.

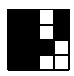
Rekombinaatio

Tarkastellaan kuvan 7.10 tilannetta, jossa kaksi sijoittelua (peitot 12 ja 11) rekombinoidaan indeksivektorin [0,0,1,0,0,1,1] mukaan, jolloin saadaan kaksi uutta yritettä, joiden peitot ovat 10 ja 13. Toinen jälkeläinen sattui tässä olemaan parempi ja toinen vastaavasti huonompi kuin kumpikaan vanhemmista, mutta tämä oli puhdasta sattumaa.



12

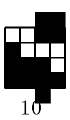
i	des	bin	ϕ	x	y
1	7	01 11	-	1	3
2	12	11 00	-	3	0
3	1	0 00 01	0	0	1
4	4	0 01 00	0	1	0
5	26	01 10 10	$-\pi/2$	2	2
6	44	10 11 00	$-\pi$	3	0
7	11	10 11	-	2	3



11

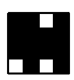
i	des	bin	ϕ	x	y
1	15	11 11	-	3	3
2	4	01 00	-	1	0
3	14	0 11 10	0	3	2
4	0	0 00 00	0	0	0
5	41	10 10 01	$-\pi$	2	1
6	18	01 00 10	$-\pi/2$	0	2
7	2	00 10	-	0	2

$$X = [0, 0, 1, 0, 0, 1, 1]$$



10

i	des	bin	ϕ	x	y
1	15	11 11	-	3	3
2	4	01 00	-	1	0
3	1	0 00 01	0	0	1
4	0	0 00 00	0	0	0
5	41	10 10 01	$-\pi$	2	1
6	44	10 11 00	$-\pi$	3	0
7	11	10 11	-	2	3



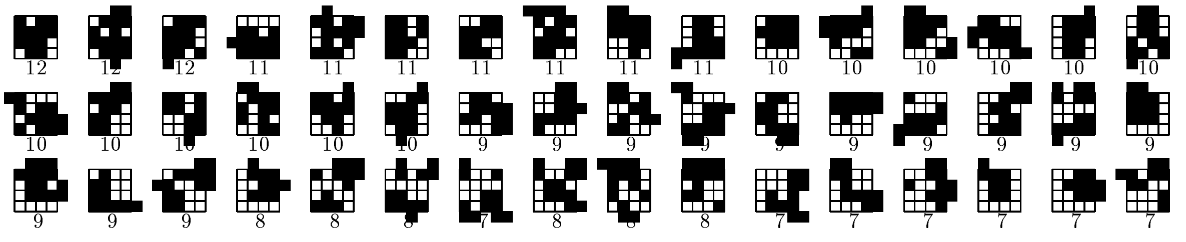
13

i	des	bin	ϕ	x	y
1	7	01 11	-	1	3
2	12	11 00	-	3	0
3	14	0 11 10	0	3	2
4	4	0 01 00	0	1	0
5	26	01 10 10	$-\pi/2$	2	2
6	18	01 00 10	$-\pi/2$	0	2
7	2	00 10	-	0	2

Kuva 7.10: Kahden sijoittelun rekombinaatio. Taulukoiden sarakkeet: i = palan indeksi, des = palan parametrit kokonaislukuna, bin = parametrit binäärisenä, ϕ = rotaatiokulma, x ja y koordinaatit.

Populaatio

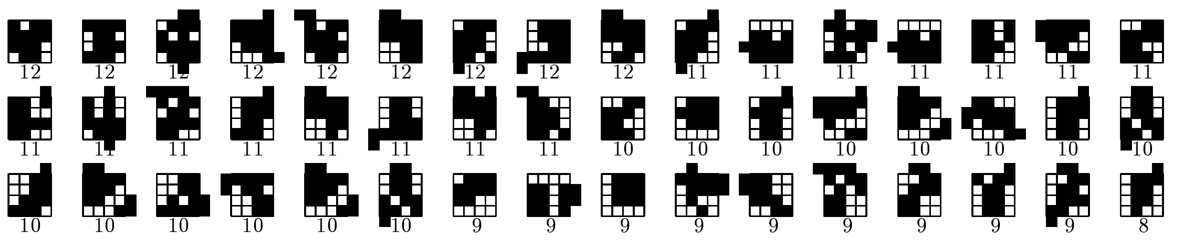
Sovitaan vielä, että populaation koko on 48 yksilöä, joista 16 jatkaa seuraavaan sukupolveen (elitismi = 33,3 %). Voimme nyt aloittaa arpomalla populaatioon satunnaislukuja 48 yritettä varten. Aloituspopulaatio, jonka keskimääräinen hyvyys on 9,3125, on esitetty kuvassa 7.11. Valinnan helpottamiseksi yritteet on järjestetty paremmuusjärjestykseen.



Kuva 7.11: Täysin satunnainen aloituspopulaatio; keskimääräinen peitto = 9,3125.

Valinta ja sukupolvet

Valitaan seuraavan sukupolven vanhemmiksi 16 parasta yritettä ja risteytetään näitä satunnaisesti keskenään, jolloin populaatiossa on jälleen 48 yritettä (16 vanhaa ja 32 uutta; keskimääräinen peitto = 10,4583) (kuva 7.12).

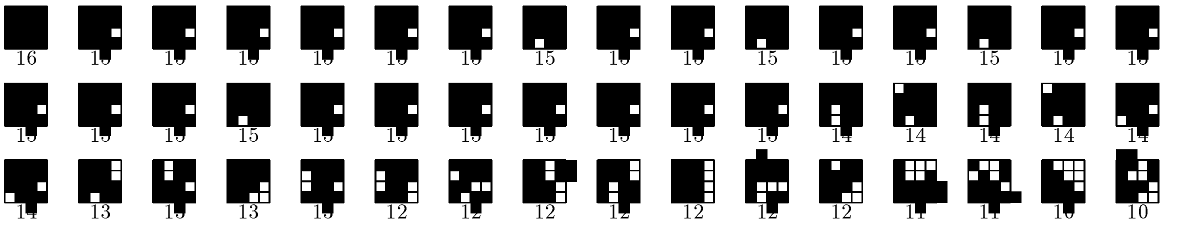


Kuva 7.12: Ensimmäinen sukupolvi; keskimääräinen peitto = 10,4583

Toistetaan, kunnes ratkaisu löytyy.

Ratkaisun löytyminen risteytyksen ja mutaation avulla on kuitenkin hidasta. Miksi? Siksi, että vähitellen, kun lähes koko alue on peittynyt tasaisesti, tyhjän ruudun löytyminen satunnaisesti indeksejä arpomalla on varsin epätodennäköistä. Kuitenkin jokaisessa yritteessä on tarkka tieto siitä, missä kukin palanen on. Tämä tieto voidaan ottaa käyttöön siten, että mutaation lisäksi vaihdetaan geenejä eli saman palasen parametreja keskenään, jolloin varsin todennäköisesti ei siirretä palasta ruudussa jo olevan toisen palasen päälle, koska lopputilanteessa päällekkäisiä palasia on enää hyvin vähän, vaan vaihdossa löytyy lähes aina ainakin yksi tyhjä ruutu.

Kun tämä uusi operaatio, vaihto, otetaan mukaan, niin ratkaisu sattuu löytymään testiajomme sukupolvessa 67 (kuva 7.13), jolloin oli ehditty laskemaan 2192 yritettä, joista 476 oli mutatoitu ja 569 oli kohdannut vaihdon. Kuten kuvasta nähdään menetelmällä on ollut ilmeisiä ongelmia löytää oikea rotaation arvo oikeassa alanurkassa olevalle haanmuotoiselle palalle.



Kuva 7.13: Sukupolvi 67: ratkaisu löytyi; keskimääräinen peitto = 13,9167

Pohdintaa

Huomaa, että toisilla parametrien arvoilla ratkaisu löytyisi nopeammin ja joillakin toisilla taas vastaavasti hitaammin, mutta joka tapauksessa sukupolvien kuluessa tulos vähitellen paranee ja ratkaisu lopulta löytyy. Jos voimme ottaa huomioon ongelmakohtaista tietoa, kuten edellä esitetyn palasten parametrien vaihto-operaation, niin menetelmä etenee kohti ratkaisua keskimäärin nopeammin, koska yritteiden keskimääräinen laatu on parantunut huomattavasti.

Kaikkiaan esimerkkiohjelmamme C++:lla ohjelmoituna on noin 300 riviä, jonka ohjelmoimiseen (lajittelualiohjelma (noin 30 riviä) oli valmiina) kului noin yksi työpäivä ja -ilta.

Selvyyden vuoksi todettakoon, että esimerkkimme ratkaisemiseen on monia mahdollisuuksia. Voidaan ajatella sijoittelussa olevan kyse palojen permutaatiosta, kun paloja pinotaan systemaattisesti vapaina oleviin ruutuihin. Kun otetaan huomioon, että paloja on 7 ja että kuusi niistä muodostaa parin, jotka voidaan vaihtaa keskenään ja että neljää voidaan rotatoida saadaan kaikkiaan $7!2^2 4^2/2^3 = 40320$ erilaista permutaatiota, joiden joukossa ovat myös ne noin 7000 ratkaisua. Tällä tavalla on läpikäytävien yritteiden määrä saatu pudotettua rajusti samalla kun käypien ratkaisujen määrä on kuitenkin pysynyt samana, eli yritteiden keskimääräinen laatu on selvästi parantunut. Tämä johtuu siitä, että on käytetty hyväksi tehtävän rakennetta: sen sijaan että arvottaisiin koordinaatteja ne lasketaan kulloisestakin sijoittelusta. Puhtaassa GA-menetelmässä tunnettiin vain peitto eli hyvyysfunktion arvo kullakin parametrikombinaatiolla, mutta ei mitään muuta ruudukon tilanteesta, eli kyseessä on täysin ns. blackbox -tyyppinen menetelmä.

Tehtävän sisäinen rakenne voidaan huomioida esim. käyttämällä ns. peruutus (engl. backtracking) -menetelmää, jossa palasia sijoitetaan ruutuun peräjälkeen vapaisiin paikkoihin, kunnes huomataan, että ei voida jatkaa vaan peruutetaan ja yritään uutta sijoitteluvaihtoehtoa. Menetelmä on helppo ohjelmoida rekursiota käyttäen, se on deterministinen ja johtaa lopulta aina ratkaisuun, jos sellainen on olemassa. Ongelmana on kuitenkin se, että vaihtoehtojen määrän lisääntyessä laskenta-aika kasvaa nopeasti mahdottoman pitkäksi, vaikka vaihtoehtojen määrää on pystytty karsimaan tehtävän rakennetta hyödyntämällä (kombinatorinen räjähdys). Tehtävien ratkaiseminen Prolog-kielellä perustuu tällaiseen peruutusmekanismiin, joka on ko. kieleen sisäänrakennettuna.

Kirjallisuutta [90, 95]
Esimerkkiohjelman C++-kielinen versio löytyy `ftp`:llä: `ftp.uwasa.fi:cs/report97-4/polyomino.cxx`.

7.4 Hajautus

Geneettinen algoritmi on helppo hajauttaa, sillä kukin yrite voidaan yleensä aina laskea muista riippumatta. Populaatio on yhteinen, mutta sekään ei aseta kovin suuria vaatimuksia synkronoinnille. Yritettä laskeva prosessi voi jopa "kaatua" ajonaikaiseen virheeseen ilman, että sillä on juuri merkitystä muiden prosessien kannalta. Tämä on hyvä ominaisuus, kun hyvyysfunktio on epästabiili eli sen laskennassa saattaa ilmetä ongelmia.

Hajautus voidaan toteuttaa monella tavalla:

- Hienojakoisesti (engl. fine-grained parallelism) eli jakamalla tehtävä usealle prosessorille, käyttäen rinnakkaislaskentaan kehitettyjä menetelmiä ja kieliä.
- Saarimallilla (engl. island model), jossa populaatio on jaettu useaan osapopulaatioon, joita optimoidaan omilla GA:lla. Aina silloin tällöin käydään vaihtamassa geeniainesta osapopulaatioiden välillä.
- hajautus yrite/prosessi periaatteella, jossa laskenta hajautetaan yhden tai useamman prosessorin kesken.

Ainakin niissä tapauksissa, joissa laskenta vaatii huomattavan paljon aikaa tai hyvyysfunktio on epästabiili, hajautetuksesta on hyötyä. Hajautuksesta löytyy runsaasti artikkeleita. Suosituin koneympäristö on ollut Transputer, joka sopii myös varsin hienojakoiseen hajautukseen. Karkean hajautuksen voi tehdä helposti PC- tai työasemaverkossa esim. PVM (Parallel Virtual Machine) aliohjelmakirjastolla tai vaikka yksinkertaisesti turvautumalla jaettuun levymuistijärjestelmään, jossa useammalla koneella on pääsy yhteiseen hakemistoon.

Kirjallisuutta [12, 151, 162, 173, 174, 52, 194].
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaPARALLELbib.ps.Z`

Luku 8

Geneettisten algoritmien sovelluksia

Tässä luvussa esitetään ensin lyhyt katsaus suosituimmista sovellusalueista, jonka jälkeen tarkastellaan tarkemmin muutamaa esimerkkisovellusta.

8.1 Suosituimmat sovellusalueet

Taulukkoon 8.1 on kerätty suosituimmat sovellusalueet Vaasan GA-viitetietokannasta. Huomaa, että sama viite voi kuulua useampaan luokkaan (keskimäärin 1,5 avainsanaa / viite).

8.1.1 Hermoverkot

Lähes 10% artikkeleista käsittelee muodossa tai toisessa hermoverkkosovelluksia, joista tärkeimmät ovat verkkojen rakenteen optimointi, verkkojen opettaminen sekä erilaiset GA:n ja hermoverkkojen hybridit erilaisissa sovelluksissa.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaNNbib.ps.Z`

8.1.2 Optimointi-intensiiviset alat

Geneettisiä optimointimenetelmiä on luonnollisesti sovellettu innokkaasti niillä tekniikan aloilla, joilla muutenkin on totuttu käyttämään erilaisia optimointimenetelmiä.

Operaatiotutkimukseksi (engl. operations research) kutsuttu ala on erikoistunut erilaisten kombinatoristen käytännön optimointitehtävien ratkaise-

aihe	*julk.*	*aihe*	*julk.*
tekniikka (engineering)	1050		
hermoverkot	773		
geneettinen ohjelmointi	503		
optimointi	480		
ohjaus ja säätö	461		
rinnakkaiset ja hajautetut GA:t	391		
töiden ajoitus (mm. JSS)	359		
kuvankäsittely	350		
evoluutio strategiat	297		
GA:n analysointi ja teoria	281		
robotiikka	261		
menetelmien vertailut	254		
koneoppiminen	240		
toteutukset	224		
CAD	219		
hybridit	197		
signaalinkäsittely	181		
katsaukset	160		
ns. classifier systems	151		
ohjaimet	136		
sumeat systeemit	135		
crossover	115		
kauppamatkustajan ongelma	115		
taloussovellukset	112		
VLSI	109		
keinoelo	101		
kemia	96		
suodattimet	95		
evoluution mallitus	93		
koodaus	80		
luokittelijat	78		
elektroniikka	76		
bibliografiat	76		
proteiinit	75		
tuotantojärjestelmät (manufact.)	75		
hahmontunnistus	73		
populaation koko	72		
elektromagnetiikka	71		
reititys	68		
graafit	68		
valinta (selection)	67		
sijoittelun suunnittelu	66		
tutoriaalit	64		
lääketiede	62		
simulointi	61		
fysiikka	60		
parametrien estimointi	58		
sääntöpohjaiset järjestelmät	56		
telekommunikaatio	53		
VLSI suunnittelu	53		
tietokannat	52		
testaus	50		
mutaatio	50		
proteiinit	45		
optiikka	44		
ryhmittely (clustering)	43		
bin-packing	41		
antennit	38		
geofysiikka (mm. seismologia)	36		
soluautomaatit	35		
tietokonegrafiikka	34		
rekombinaatio	31		
pelit	31		
asiantuntijajärjestelmät	30		
logistiikka	30		
muodon suunnittelu	30		

Taulukko 8.1: Suosituimmat GA:n sovellusalueet. Tietokannassa oli yli 9000 luokiteltua viitettä, kun tämä lista koottiin. Huomaa, että sama julkaisu voi kuulua useampaan ryhmään (esimerkiksi tekniikka sisältää mm elektroniikan ja VLSI suunnittelun).

miseen. Näistä on seuraavaksi tarkasteltu töiden ajoittamista ja ns. kauppamatkustajan ongelmaa.

Töiden ajoitus

Tuotantotoiminnan ohjauksessa tärkeä ongelma on ajoittaa eli järjestää (engl. schedule) työt mahdollisimman optimaalisesti. Ongelmaa ratkotaan päivittäin tuotannossa eikä siten olekaan ihme, että ajoitustehtävät ovat vilkkaan tutkimuksen kohteena. Ne ovat vaikeita kombinatorisia optimointiongelmia, jotka lisäksi vaativat erikoisjärjestelyjä, kuten palikkaesimerkissämme (sivu 124), jotta sitä olisi GA:lla järkevää ratkaista. Ongelmana on, että tavanomainen lineaarinen koodaus ja rekombinaatio sopii huonosti yhteen ajoituksen tapaisen permutaatiotyyppisen ongelman kanssa. Vastaavanlaiseen ongelmaan törmätään myös esimerkiksi seuraavaksi esiteltävän kauppamatkustajan ongelman yhteydessä, jossa etsitään optimaalista solmupisteiden permutaatiota.

Kirjallisuutta [45]
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaORbib.ps.Z`

Kauppamatkustajan ongelma

Tyypillinen vaikea optimointitehtäväklassikko, on ns. kauppamatkustajan ongelma (engl. Travelling Salesman Problem; lyh. TSP), jossa pyritään löytämään lyhyin reitti n solmupisteen kautta, kun solmujen välisten reittien pituudet tunnetaan. Ongelman ratkaisulla on käyttöä monilla aloilla, ei pelkästään logistiikassa. Ongelmaa on tutkittu todella paljon ja sille on kehitetty varsin hyviä (heuristisia) ratkaisumenetelmiä.

Kauppamatkustajan ongelma on permutaatiotyyppinen ongelma. Siinä etsitään kaikkien solmupisteiden kautta kulkevaa reittiä, joka voidaan esittää solmupisteiden permutaationa.

Kuten jo aiemmin todettiin permutaatiotyyppinen ongelma vaatii yleensä sopivaa tietorakennetta ja/tai operaatioita, kun sitä ratkotaan GA:lla. Lisäksi kauppamatkustan ongelmaan löytyy hyviä ratkaisumenetelmiä, joille perusmuotoinen GA ei mitenkään pärjää. Kokonaan toinen asia on tehdä hybridi, jossa GA:n ja TSP:n erikoisratkaisijan, kuten klassisen Lin-Kernighan heuristiikan, hyvät puolet yritetään yhdistää.

Kirjallisuutta [140, 164, 50]
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaTSPbib.ps.Z`

Voimantuotanto

Perinteisistä tekniikan aloista kaksi on ylitse muiden, kun tarkastellaan GA:n soveltamista. Toinen on energian tuotanto, jossa on muutamia erittäin tärkeitä optimointitehtäviä. Tärkein näistä on päättää, millä yksiköillä milloinkin tarvittava teho kannattaa tuottaa sähköverkkoon. Tämä ns. unit commitment ongelma on paitsi vaikea, myös taloudellisesti erittäin merkittävä.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaPOWERbib.ps.Z`

Rakenteiden suunnittelu

Toinen perinteinen tekniikan ala, jossa optimointi on ollut merkittävä suunnittelijan käyttämä apuväline on rakenteiden suunnittelu. Miten rakentaa silta tai muu vastaava rakenne pienimmin kustannuksin ja kuitenkin niin, että annetut tekniset vaatimukset ja turvanormit tulevat täytetyksi?

Ohjaus ja säätö

Perinteinen säätö pyrkii myös eksplisiittisesti optimaaliseen tulokseen, joten erilaisten optimointimenetelmien käyttö myös tällä sovellusalueella on luontevaa.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaCONTROLbib.ps.Z`

VLSI

Korkean teknologian alana, jossa suunnittelu perustuu täysin tietokonepohjaisiin menetelmiin ja sisältää paljon optimointiongelmia, VLSI-tekniikka on mitä oivallisin sovellusalue myös geneettisille algoritmeille.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaVLSIbib.ps.Z`

8.1.3 Robotiikka

Poikkitieteellisenä alana robotiikka on kiitollinen monenmoisille tekniikoille niin myös optimoinnille. Robotiikkaan liittyviä ongelmia ovat liikkeiden suunnittelu, toimintojen koordinointi, robottien ohjelmointi sekä erilaiset hahmontunnistustehtävät.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaROBOTbib.ps.Z`

8.1.4 Signaalinkäsittely

Suodattimien optimointi erilaisia tarkoituksia varten ja erilaisin kriteerein on GA:lle otollinen sovellusalue, jossa kilpailevia menetelmiä saa etsiä. Tyypilli-

sesti kohteena ovat olleet suodattimet, joiden optimointi muilla menetelmillä on hankalaa suodattimien matemaattisten ominaisuuksien takia. Esimerkiksi mediaanisuodattimet ovat jo toimintaperiaatteensa takia epälineaarisia ja siten matemaattisesti erittäin vaikeita hallita. Toinen motivaatio suodattimien optimoimisessa on niiden toiminnan tehostaminen eli laskenta-ajan minimointi. Annetusta laitteistosta halutaan saada "kaikki irti" mahdollisimman vähin kustannuksin erityisesti viihde-elektroniikassa, joka on digitaalisten suodattimien suurkuluttaja. On myös olemassa potentiaalisia sovelluksia, jotka tulevat mahdollisiksi toteuttaa, kunhan piireistä ja algoritmeista saadaan riittävän nopeita. Täten optimointi on digitaalisten ja tietysti myös analogisten suodattimien suunnittelussa tärkeää ja suuren mielenkiinnon kohteena niin teoriassa kuin käytännössä.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaSIGNALbib.ps.Z`

8.2 Esimerkkisovelluksia

Seuraavassa on esitelty tarkemmin joitakin GA:n teknisiä sovelluksia. Osa perustuu kirjoittajan tutkimusprojekteihin osa on valittu alan laajasta kirjallisuudesta. Tarkoituksena on osoittaa millaisia tyypilliset sovellukset ovat ja toisaalta miten laajalla kirjolla erilaisia sovelluksia esiintyy.

Esitetyistä sovelluksista voi ottaa mallia, mutta kannattaa kuitenkin muistaa, että perimmältään kysymys on aina siitä, että ongelma on tai se voidaan kohtuullisella työllä muuttaa optimointiongelmaksi, jolle ei ainakaan toistaiseksi löydy muita tehokkaampia ratkaisumenetelmiä. On myös aina mietittävä yksittäisen ongelman ratkaisemisen kustannuksia. Perinteisen menetelmän ja GA:n hybridi voi myös olla hyvä idea niin teknisesti kuin taloudellisesti.

8.2.1 Hissien ryhmäohjaus

Kun hissin kyytiin pyritään isossa ja monikerroksisessa rakennuksessa, jossa on useita hissejä ryhmänä, kutsunappin painaminen johtaa varsin monimutkaiseen päättelyjen ketjuun hissiryhmäohjaimessa. Mikä hissi muuttaa ajoreittiään poimiakseen kutsujan? Ongelma on varsin vaikea, eikä sitä pystytä kombinatorisen räjähdyksen vuoksi optimaalisesti ratkaisemaan perinteisin hissiryhmäohjauksen menetelmin, joita ei ole alunperin suoranaisesti optimointiin tarkoitettu. GA:lla voidaan ongelma pukea selkeästi optimointiongelmaksi, joka ainakin suoritettujen simulointikokeiden mukaan, voidaan ratkaista selvästi paremmin kuin perinteisin menetelmin, jotka perustuvat deterministisiin sääntöihin, joita sitten pyritään soveltamaan esimerkiksi hissiryhmän ja liikennetilanteen mukaan. Eksplisiittisesti ohjelmoimalla on mahdotonta etukäteen varautua kaikkiin mahdollisiin tilanteisiin.

Koska kyseessä on optimointiongelma päätettiin kokeilla sen ratkaisemista sellaisena geneettisen algoritmin ja hissisimulaattorin avulla. Kun simulaattorista oli karsittu pois grafiikka ja muu manuaaliseen seurantaan tarkoitettu käyttöliittymä, saatiin kohtuullisen tehokas ja optimointiin sopiva apuväline.

Menetelmän periaatteena on, että geneettinen algoritmi generoi erilaisia hissien allokointeja eli suunnittelmia hissien ajoreiteiksi. Simulaattorin avulla sitten lasketaan kunkin vaihtoehdon kustannukset eli esimerkiksi matkustajilta kuluva odotusaika. Muutaman kierroksen kuluttua parhaimmat reitit ovat selvinneet ja hissit ajavat sitten niiden mukaan, ellei myöhemmin havaita parempia vaihtoehtoja tai tilanne ole oleellisesti muuttunut. Koska optimointi suoritetaan kerran sekunnissa, ovat suunnitelmat aina ajantasalla.

Tehtyjen varsin laajojen ja tilastollisin testein varmistettujen kokeiden mukaan GA pystyi selvästi parempaan tulokseen, kun optimointikriteerinä käytettiin matkustajien keskimääräistä odotusaikaa. Menetelmä mukautuu jatkuvasti (kerran sekunnissa) olemassa olevaan liikennetilanteeseen, joten se on riippumaton esimerkiksi liikennetilanteen ennusteista, joihin nykyisin käytössä olevat menetelmät usein perustavat päätöksentekonsa.

Menetelmän toinen huomattava etu on siinä, että optimointikriteerejä voidaan helposti muuttaa talon ja tilanteen mukaan. Esimerkiksi hissien poissaolo huollon aikana ei vaadi kovinkaan paljon ylimääräistä ohjaimen ohjelmointia. Riittää, kun simulaattori on ajantasalla todellisesta tilanteesta.

Työssä verrattiin tilastollisesti kolmea hissiohjainta: kahta perinteistä ja yhtä GA-pohjaista. Vertailu perustui simulaattorilla suoritettuihin ajoihin ja siinä todettiin, että GA-optimointiin perustuva menetelmä antoi noin 10% lyhyemmän keskimääräisen odotusajan kuin perinteiset menetelmät, jotka nekin oli pyritty suunnittelemaan mahdollisimman optimaalisiksi.

Kirjallisuutta [6, 29, 13, 78, 167, 156, 207, 25, 14, 200]

8.2.2 Muodon optimointi

Voima-anturi

Gordon Robinson National Physical Laboratoriosta (NPL) Englannista on käyttänyt geneettistä algoritmia väitöskirjassaan kalibrointivoima-anturin runko-osan muodon optimointiin. NPL vastaa Englannissa voiman mittayksikön newtonin realisoinnista ja siirtostandardeista. Voima-anturi muodostuu pyörähdyssymmetrisestä runko-osasta ja sen keskikohtaan kiinnitetyistä venymäliuska-antureista (kuva 8.1). Muodon optimoinnilla pyrittiin siihen, että anturi olisi mahdollisimman riippumaton kuormituksen jakautumisesta, joka on vaikeasti hallittavissa kalibrointitilanteessa. Samalla pyrittiin löytämään myös mahdollisimman matala ratkaisu, koska osassa kalibroitavista koneista (vanhat 3MN betonin testauslaitteet) on hyvin matala työalue (200mm).

Optimointia varten laadittiin ohjelma, joka FEM-rutiinien avulla laski runko-osan jännitysjakautuman, kun GA generoi sille profiilin Bezier-käyrinä. Optimoinnin tuloksena immuunisuus kuorman jakautumalle kasvoi EN-standardin mukaan simulointitulosten avulla laskien 26 kertaiseksi ja PCCU-standardinkin mukaan 10,5 kertaiseksi. Laskelmat tarkistettiin valmistamalla 300kN-3MN:n anturi 826M40 teräksestä (kuva 8.1, korkeus 185mm). Se läpäisikin helposti vaaditun EN10002-3 luokan 00 testin.

Geneettisen algoritmin populaation koko eli kerralla muistissa olevien ratkaisuvaihtoehtojen määrä oli 100 ja yhden parametrin eli geenin pituus oli 10 bittiä. Optimointi lopetettiin 50 sukupolven jälkeen. Edellä esitetyt parametrien arvot ovat varsin tyypillisiä sovellukselle, jossa optimoitava, tässä siis FEM-analyysin tulos, on suhteellisen raskas laskea, mutta jossa optimoitava muuten käyttäytyy varsin leppoisasti.

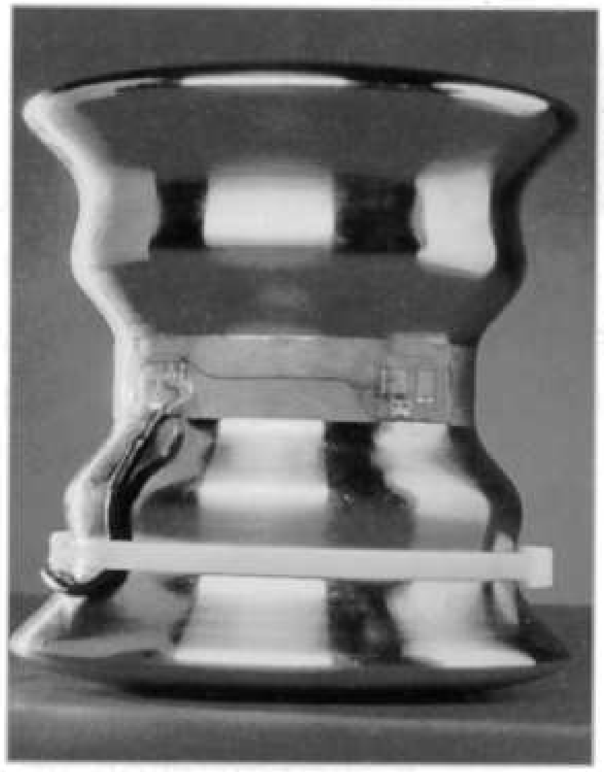
Kirjallisuutta [177, 178, 41, 42, 203]

Venttiilinokkien muodon optimointi

Erilaiset nokkamekanismit ovat tärkeitä kone-elimiä monissa automaateissa ja koneissa. Nokan avulla muutetaan, yleensä tasainen tai lähes tasainen, pyörimisliike tarkoin ajoitetuksi lineaariseksi liikkeeksi. Yksi tärkeä nokkamekanismien (kuva 8.3), sovellusalue ovat polttomoottorien venttiilimekanismit joissa nokan muodon avulla määrätään moottorin tärkeimmät ominaisuudet.

Vaasan yliopiston Tietotekniikan ja tuotantotalouden laitoksella on yhdessä Wärtsilä Diesel NSD Oy:n kanssa optimoitu suurten dieselmoottoreiden venttiili- ja pumppunokkien profiilien muotoja. Muodon esitys on vaativampi kuin edellä esitetyssä voima-anturin tapauksessa, koska nokan profiilin on oltava jatkuva myös ensimmäisen ja toisen derivaatan suhteen. Tämän lisäksi profiiliin kohdistuu muitakin nokkamekanismin ja sen geometrian aiheuttamia rajoitteita. Profiili kuvattiin jatkuvuusehtojen takia kuudennen asteen paloittain jatkuvana polynomina eli splinikäyrästöllä, jonka ohjauspisteitä (yhteensä 160 kpl) GA optimoi muuttamalla niiden arvoja radiaalisuunnassa. Ohjauspisteet olivat kulmamuuttujan suhteen tasavälisesti ja kiinteillä paikoilla (kuva 8.4).

Profiilin hyvyys laskettiin simuloimalla mekanismin toimintaa annetut rajoitteet huomioon ottaen. Käytännössä simulointi vaatii melkoisen paljon



Kuva 8.1: Geneettisellä algoritmilla optimoitu punnitusanturi. [[177] G. M. Robinsonin luvalla]

aikaa, mutta lopputulos oli kuitenkin tavoitteen mukainen: ennen suunnittelijalta viikon vienyt työvaihe saatiin automatisoitua ja valmistumaan vuorokaudessa eli käytännössä "yön yli ajona". Kuvassa 8.5 on esimerkki geneettisellä algoritmilla optimoidusta nokkaprofiilista. Kuvassa 8.6 on esitetty nokan hyvyyden kehittyminen sukupolvien funktiona. Kuten nähdään, tulos paranee dekadilla aina noin 10.000 simulointikerran eli noin 3000 sukupolven kuluttua, kun populaation koko oli 30. Käytännössä populaation koko vaihteli eri kokeissa välillä 30-100 ilman merkittävää vaikutusta nopeuteen tai tarkkuuteen.

Kirjallisuutta [142, 147, 15, 16, 21, 139, 20]

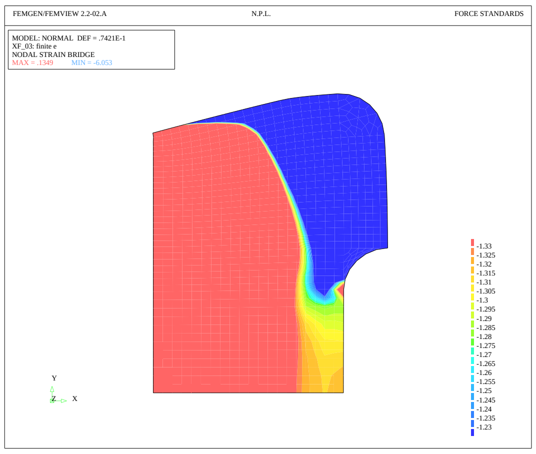


Figure 9.9: Contours of bridge strain in conventional load cell: Convex pad: Friction=0.3

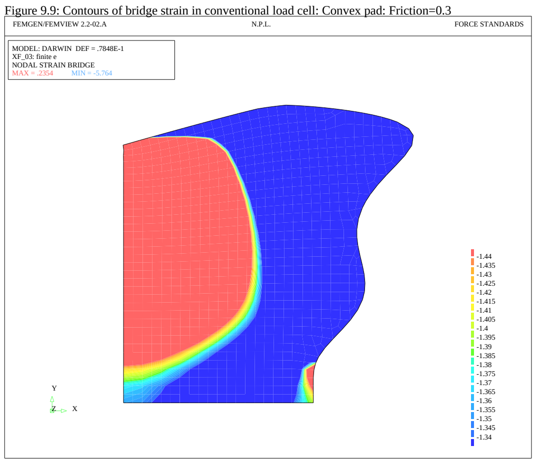


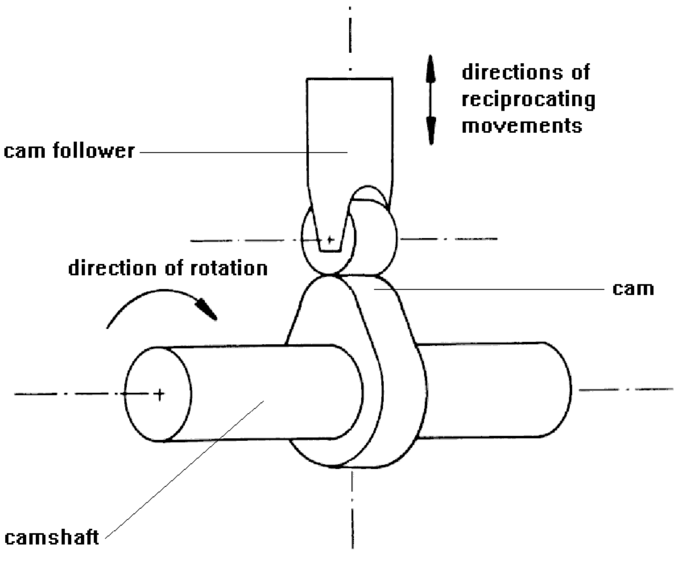
Figure 9.13: Contours of bridge strain in GA optimised load cell: Convex pad: Friction=0.3

Kuva 8.2: Tavanomaisen anturin rungon jännitysjakauma (vasemmalla) ja GA:lla optimoidun anturin rungon jännitysjakautuma (oikealla) FEM-analyysin mukaan. [[177] G. M. Robinsonin luvalla]

8.2.3 Optinen pinnoite

Optiikka on mm. tarkkoja mittausmenetelmiä tarjoava tekniikan ala, jossa optimointimenetelmät ovat perinteisesti olleet oleellinen osa suunnittelua. T. Eisenhammer *et al* ovat soveltaneet geneettistä algoritmia aurinkokerääjän puoliläpäisevän peilin pinnoitteen suunnitteluun. Peilin tarkoituksena on päästää mahdollisimman paljon auringon säteilyenergiaa kerääjän absorbaattoriin, mutta pitää mahdollisimman tarkoin kerääjän sisällä oleva lämpösäteily kerääjässä (kuva 8.7).

Puoliläpäisevä peili koostuu lasin pinnalle höyrystetyistä ohuista metalli (hopea, kulta tai kupari) ja vahvasti valoa taittavista dielektrisistä kerrok-



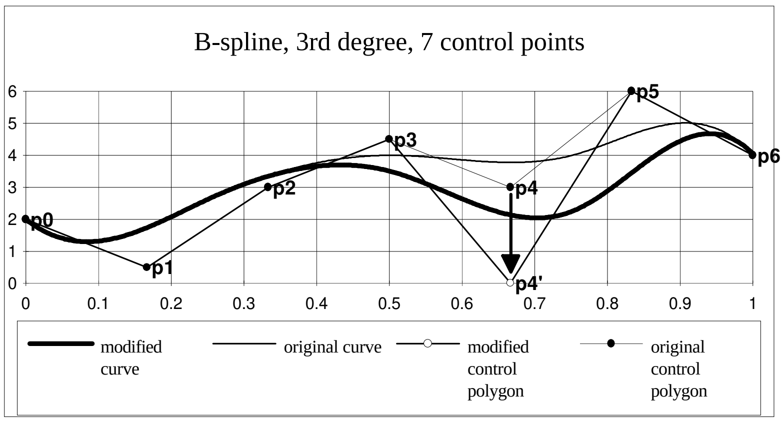
Kuva 8.3: Nokkamekanismi.[17]

sista (7 erilaista materiaalia). Geneettisellä algoritmilla optimoitiin kerrosten paksuudet ja materiaalit. Viiden kerroksen tapauksessa optimaalinen rakenne oli (TiO_2 (33,3nm) /Ag (7,15nm) /TiO_2 (73,0nm)/Ag (7,2nm)/Y_2O_3 (58,7nm)). Populaation koko oli 150 ja sukupolvia laskettiin 300-500 eli kaikkiaan hyvyysfunktion arvoja laskettiin 45.000-75.000. Kromosomin pituus oli 55 bittiä. Valinnassa käytettiin elitismiä eli parhaat yritteet siirtyivät aina seuraavaan polveen. Geneettisellä algoritmilla saadut tulokset vastasivat laadultaan muilla menetelmillä saatuja tuloksia. Itse asiassa tekijät saivat kokeen aikana tiedon vastaavasta rakenteesta, joka oli jo kaupallisessa käytössä.

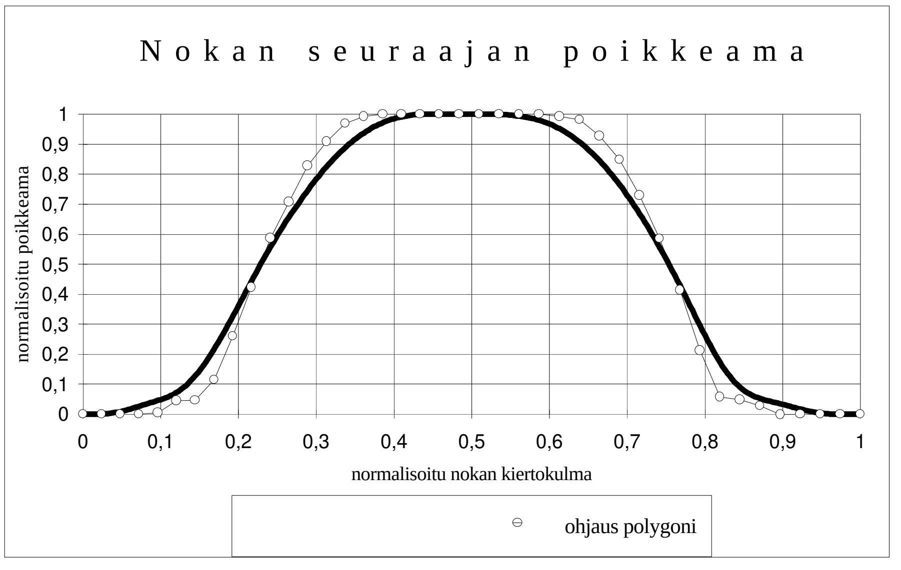
Kirjallisuutta [62, 61, 221]

8.2.4 Tietoliikenneverkon optimointi

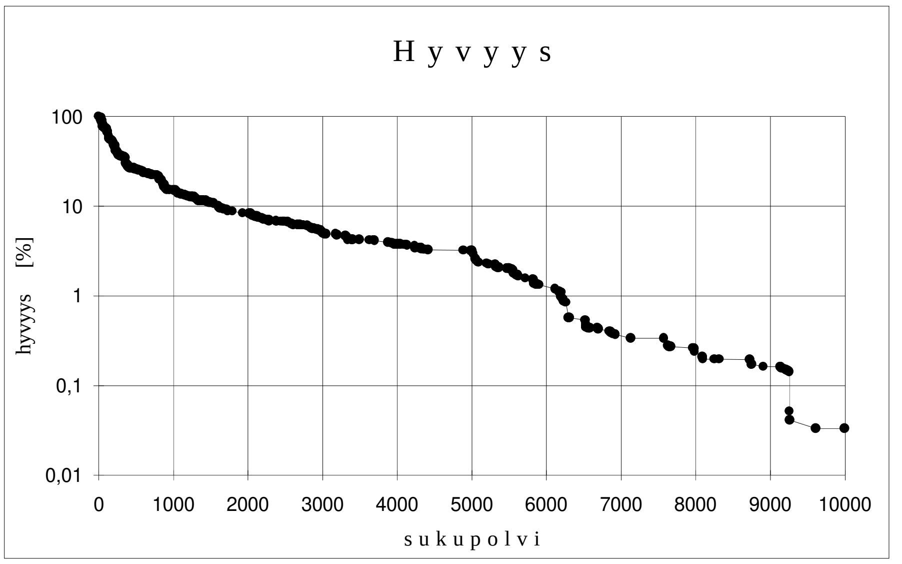
T. Carpenter *et al* ovat optimoineet SONET (Synchronous Optical Network) renkaan solmujen kapasiteettia geneettisellä algoritmilla. SONET -verkon muoto on kuvan 8.8 mukaisesti rengas, jossa jokainen solmu voi olla yhteydessä mihin tahansa muuhun solmuun kahta toisistaan riippumatonta reittiä, myötä- tai vastapäivään. Solmut ovat kytketty toisiinsa optisella kuidulla. Verkon rakenteen vuoksi yhden solmun häiriö ei ole ongelmallinen, koska signaalit muista solmuista voivat silloin kulkea reittiä, joka toimii riippumattomasti häiriöön joutuneesta solmusta. Verkon solmut ovat tarvittavan



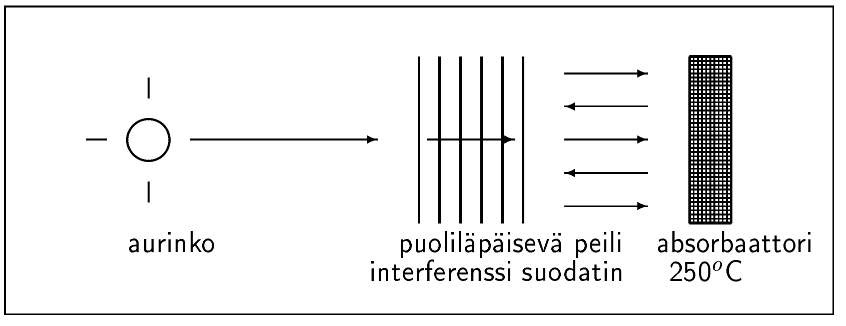
Kuva 8.4: Esimerkki muodon esittämisestä B-splini-käyrän (paksu viiva) ohjauspisteiden ($p0$, $p1, \ldots, p6$) avulla. Yhtä ohjauspistettä ($p4$) on muutettu, mikä näkyy käyrän muodon paikallisena muutoksena (ohut viiva). Huomaa että kuvan käyrä on kolmatta astetta, kun taas venttiilinokan muoto esitetään kuudennen asteen splininä, mutta periaate on sama; ainoastaan ohjauspisteitä tarvitaan enemmän. [138]



Kuva 8.5: Nokkaprofiili 40 ohjauspisteen avulla esitettynä.[19]



Kuva 8.6: Nokan hyvyyden kehittyminen.[19]

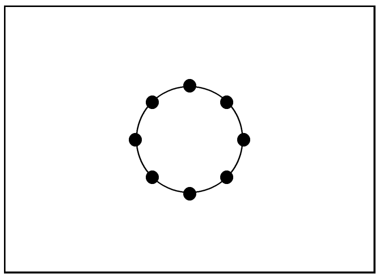


Kuva 8.7: Aurinkokerääjän toimintaperiaate.

kapasiteetin suhteen erilaisia ja suunnittelun yksi tavoite onkin määritellä solmuille minimikapasiteetit, paitsi normaalitoiminnassa, myös yhdestä virheestä toipumistilanteessa.

Verkon kuormitusongelma (engl. loading problem) on nyt seuraava: miten viestien reitittäminen (myötä/vastapäivään) pitää jakaa niin, että kaikkien solmujen maksimikuormat ovat mahdollisimman alhaiset? Voidaan todistaa, että tämä ongelma on NP-täydellinen.

T. Carpenter *et al* ratkaisivat kuormitusongelmaa geneettisellä algoritmilla ja vertasivat tuloksia kahteen muuhun menetelmään, joista toinen oli



Kuva 8.8: SONET verkko, jossa on 8 solmua.

CPLEX sekalukuohjelmoinnin ja heuristisen menetelmän yhdistelmä ja toinen oli Cosaresin ja Sanieen 2-vaiheinen ahne heuristiikka. Kokeet suoritettiin 10 ja 25 solmun verkoilla.

10 solmun tapauksessa kaikki menetelmät toimivat hyvin. 25 solmun tapauksessa CPLEX:n rajoitukset tulevat useissa tapauksissa vastaan ennen kuin ratkaisu löytyy. Ongelmana on se, että menetelmä joutuu käymään läpi branch and bound -tyyppisesti (peruutus) suuren joukon vaihtoehtoja (kombinatorinen räjähdys). Geneettisen algoritmin tulokset olivat useimmiten parhaimmat.

Geneettinen algoritmi oli helppo soveltaa ja se vaati suurin piirtein saman laskenta-ajan kuin mikä oli verkonsuunnitteluohjelmiston (SONET Toolkit) kaksivaiheisella menetelmällä. Koodaus on yksinkertainen ja luonnollinen: 1 = myötäpäivään, 0 = vastapäivään.

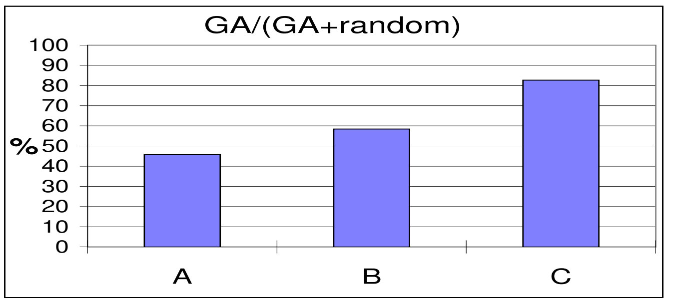
Rekombinaatio tapahtui yhden pisteen crossover -operaatiolla. Geneettinen algoritmi oli D. Whitleyn Genitor [218] (yksi tekijöistä on ollut aikaisemmin D. Whitleyn GA-tutkimusryhmässä).

Kirjallisuutta [117, 119]

8.2.5 Ohjelmien testaus

Ohjelmien testaus on ohjelmistotuotannon työläimpiä vaiheita, joka helposti jää ohjelmistoprojektin muiden kiireiden takia lapsipuolen asemaan. Seurauksena on ohjelmistojen usein luvattoman huono laatu, joka ei ole omiaan herättämään asiakkaiden luottamusta sen paremmin ohjelmistotuotteisiin kuin niiden tekijöiden ammattitaitoon. Teoreettisesti voidaan osoittaa, ettei millään testauksella voi pitävästi todistaa ohjelman toimivan moitteettomasti kaikissa mahdollisissa olosuhteissa. Kuitenkin on parempi yrittää testata ohjelma mahdollisimman kattavasti ennen kuin se luovutetaan asiakkaalle. Asiakkaan tekemä testaus toki on ilmaista useimpien ohjelmien tekijöille, mutta on pidemmän päälle varsin huonoa mainosta. Sulautettujen ohjelmistojen tapauksessa virhe voi johtaa jopa merkittäviin välittömiin aineellisiin vahinkoihin.

Joihinkin testausongelmiin voidaan soveltaa geneettistä algoritmia. Erityisen potentiaalinen sovellus on ohjelman suorityskyvyn ääriarvojen etsintä, mikä on optimointiongelma, joskin monimutkainen sellainen. Millä syötteiden kombinaatiolla ohjelma vaatii pisimmän prosessointiajan tms? Ohjelman voidaan ajatella olevan kombinatorinen ongelma, joka tuottaa tietyillä syötteillä tietynlaisen vasteen. Riippuvuus on vaihteva, mutta jotakin korrelaatiota esiintyy ja tietyt syötteet ilmeisesti aiheuttavat ainakin lähes samanlaisena toistuvan vasteen. Tilanne on lähes ideaalinen GAn kannalta: Rakennuspalikkahypoteesi näyttäisi pätevän ainakin osittain, kilpailevia menetelmiä on vähän ja ongelma olisi pystyttävä jotenkin ratkaisemaan kohtuullisessa ajassa. Lisäksi ohjelman testaukseen on yleensä käytettävissä suhteellisen runsaasti aikaa. Jos ohjelma "kaatuu" tms testin aikana, niin samalla vaivalla on tullut löydettyä virhe, joka olisi joka tapauksessa pitänyt korjata.



Kuva 8.9: Geneettisen algoritmin ja satunnaisen testidata generaattorin testitapausten suhteen vertailu. A = ei-kriittinen alue, B = välialue ja C = aikakriittinen alue eli pisimmät vasteajat.

Ohjelmien testaus GA:lla on varsin uusi keksintö eikä sitä ole vielä tutkittu paljoakaan. Huomattavasti enemmän menetelmää on käytetty vastaavassa ongelmassa VLSI-piirien testauksessa, jossa ei riitä että suunnitelma testataan, vaan että myös jokainen valmistettu piiriyksilö pitäisi testata, jotta siinä mahdollisesti olevat valmistuksen aikana syntyneet yksilölliset valmistusvirheet paljastuisivat.

Kuvassa 8.9 on esitetty geneettisen algoritmin ja satunnaisen testidatan aiheuttaman suoritusajan jakautuminen eräässä simuloidussa sulautetussa ohjelmassa. Kuten näkyy GA on pystynyt tuottamaan selvästi enemmän (yli 4 kertaa) sellaisia tapauksia, joiden suoritusaika on pitkä ja siten aika-kriittisen ohjelman potentiaalisia ongelmatilanteita.

Kirjallisuutta [68, 30, 44, 52, 51, 53, 82, 158, 114, 115, 46, 181, 23, 22, 94]

8.2.6 Plasmaleikkaus robotilla

T. Shibata *et al* esittävät artikkelissaan menetelmän, jolla parannetaan robotisoidun plasmaleikkauksen laatua hienosäätämällä ihmisen alustavasti ohjelmoimaa työstörataa. Tämänkin sovelluksen ydin on kustannusfunktio. Se on heuristinen eli se suunniteltiin käyttäen hyväksi leikkauslaitteen kokeneita ohjelmoijia, joilla oli hyvä tuntuma prosessiin. Heidän kokemuksensa mukaan robotin akselien tulisi liikkua mahdollisimman hitaasti eli konfiguraatioiden singulariteeteista johtuvia paikallisia yllättäviä liikkeitä pitäisi välttää. Yksinkertainen ja järkevä sääntö, joka oli jo valmiiksi optimointiin sopivassa muodossa. Ongelmana tässä helposti ohjelmoitavassa säännössä on vain se, että siitä saatava funktio on epälineaarinen ja siten jälleen kohtuullisen vaikea optimoitava tavanomaisilla menetelmillä. Geneettiselle algoritmille (tai muille vastaaville menetelmille) tapaus on kuitenkin varsin helppo eikä menetelmä rajoita ratkaisujen etsintää.

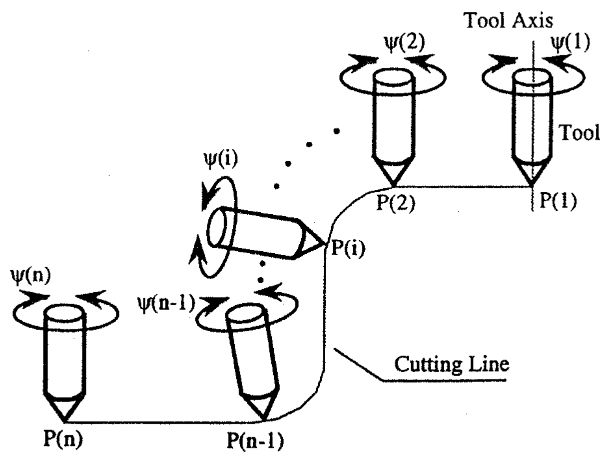
Koska robotissa oli viisi vapausastetta ja leikkaamiseen tarvittiin niistä yhtä aikaa vain neljää, niin optimointia varten jäi yksi vapausaste, jonka arvolla voitiin "pelata" radan jokaisessa pisteessä, jotta singulaariset pisteet voitiin välttää.

Kokeissaan T. Shibata *et al* käyttivät populaation kokoa 8 eli hyvin pientä populaatiota. Optimointi aloitettiin operaattorin suunnittelemasta radasta. Vertailemalla kokeneen ja kokemattoman operaattorin ratojen optimoinnin kehittymistä saatiin käsitys menetelmän käyttökelpoisuudesta. Havaittiin, että noin 200 sukupolven eli 1600 optimoitavan funktion laskennan jälkeen GA:lla optimoitu kokemattoman operaattorin rata vastasi kokeneen operaattorin rataa ja noin 500 sukupolven kuluttua sekä kokeneen että kokemattoman ohjelmoijan radat olivat optimoituneet suurinpiirtein yhtä hyviksi ja selvästi myös kokeneen ohjelmoijan esittämää rataa paremmiksi (kuva 8.11). Kuvassa 8.12 on esitetty leikatut koekappaleet ja kuvassa 8.10 on esitetty kappaleen malli työstöratoineen.

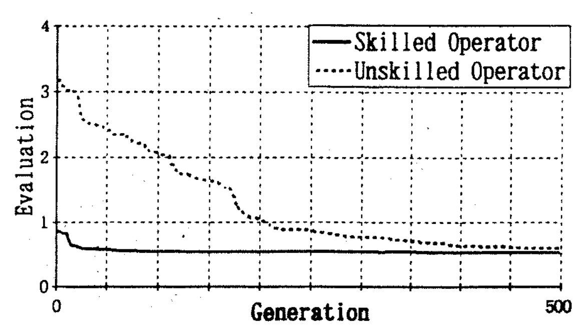
Jatkohankkeessa kirjoittajat käyttivät fuzzy ID3 algoritmia kriteereiden etsintään.

(Huom! Singulariteettiongelma on yleisesti tunnettu ja se johtuu robotin geometriasta. Tekijät etsinnevät muita ongelmatilanteita.)

Kirjallisuutta [198, 199].



Kuva 8.10: Leikattavan kappaleen malli ja työstörata [[199] T. Shibatan luvalla]

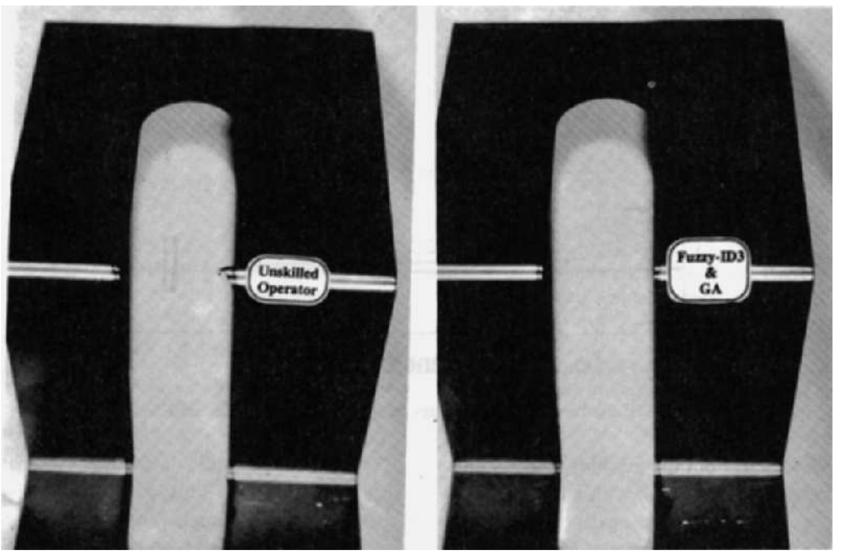


Kuva 8.11: Plasmaleikkauksen laadun paraneminen sukupolvien mukana kokeneen (—) ja kokemattoman operaattorin suunnittelemasta radasta lähtien (- - -). [[199] T. Shibatan luvalla]

8.2.7 Piirien suunnittelu

Piirien, niin integroitujen kuin erillisistä komponenteista koottujen piirilevyjen, suunnittelussa on monia optimointivaiheita, joista useimmat ovat hyvin vaikeita, mutta taloudellisesti merkittäviä. Esimerkiksi piirin pinta-ala on lähes suoraan verrannollinen valmistuskustannuksiin ja johdotuksen pituus verrannollinen viiveisiin ja vikatodennäköisyyteen.

Piirien suunnittelun ongelmat ovat pääsääntöisesti laskennalliseen geo-

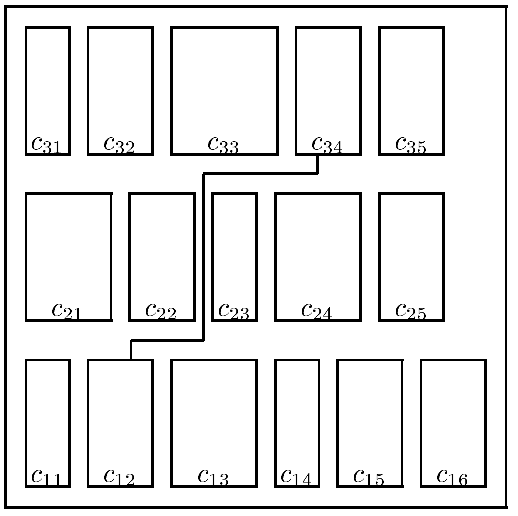


Kuva 8.12: Plasmaleikatut koekappaleet: vasemmalla kokemattoman operaattorin ohjelmoima ja oikealla, kun rataa on geneettisellä algoritmilla optimoitu. [[199] T. Shibatan luvalla]

metriaan kuuluvia, ja niitä on käytännössä aina yksinkertaistettu rajoittamalla tarkastelu vain koordinaattiakseleiden suuntaisiin suorakulmioihin, joiden muotoisista osista sitten kaikki piirin osat, transistorit, vastukset, johtimet yms koostuvat. Piirien valmistus ei sinänsä tätä rajoitusta välttämättä vaatisi. Esimerkiksi johtimet voisivat kulkea kahden pisteen välistä lyhintä reittiä "linnuntietä", mutta laskennallisista tai paremminkin algoritmisista syistä johtimet vedetään koordinaattiakselien suuntaisesti, jolloin ne useimmiten ovat ainakin hieman linnuntietä mitattua (teoreettista) minimipituuttaan pidempiä.

Tällaisia rajoituksia tehden optimointiongelma saadaan yksinkertaisemmaksi ratkaista. Samalla joudutaan kuitenkin myös jonkin verran tinkimään absoluuttisesta optimista.

Tyypillinen optimointitehtävä VLSI-tekniikassa on mm. suunnitella ns. standardisolujen sijoittelu, joka johtaa pienimpään mahdolliseen piirin pinta-alaan. Standardisolu on digitaalipiirin itsenäinen rakenneosa esimerkiksi 4-sisäänmenoinen NAND-veräjä, jonka sisäänmenot ja ulostulot ovat piirin ylä- ja alareunalla. Kaikki tietyn "teknologian" standardisolut ovat yhtä korkeita suorakaiteen muotoisia alueita (tästä niiden nimi), mutta niiden leveys vaihtelee. Piiri rakennetaan näistä soluista valitsemalla niille sopiva järjestys ja johdottamalla ne kuvan 8.13 mukaisesti standardisolurivien väliin jäävää aluetta käyttäen.



Kuva 8.13: Johdottaminen ns. standardisolutekniikassa. Esimerkkinä solujen c_{12} ja c_{34} yhdistäminen.

Kirjallisuutta [122, 121, 196, 197, 179, 182, 195]

8.2.8 Tuotannon järjestelyt

Töiden ajoitus

Töiden ajoitus (engl. scheduling), niin tehtaassa kuin vaikka moniprosessorisessa tietokoneessa, on tunnetusti vaikea ongelma ja ns. operaatiotutkimuksen (engl. operations research) perinteisiä tutkimuskohteita. Se on myös käytännön kannalta mielenkiintoinen suuren taloudellisen merkityksensä takia, koska esimerkiksi tehtaan käyttöaste määräytyy sen mukaan miten tehokkaasti sen resursseja pystytään hyödyntämään eli töitä ajoittamaan ja sijoittamaan toistensa suhteen tilanteen vielä yleensä jatkuvasti muuttuessa. Yksinkertaisimmillaan ongelmana on etsiä yhdelle työstökoneelle, prosessorille tms. tuleville n työlle optimaalinen järjestys eli permutaatio. Kuten jo kombinatorisen räjähdyksen esimerkissä sivulla 109 totesimme, niin mahdollisten permutaatioiden määrä kasvaa todella nopeasti ($= n!$) eikä kaikkia vaihtoehtoja voida käydä läpi nopeimmillakaan tietokoneilla, kun $n > 20$. Tarvitaan siis menetelmää, jolla kelvollisia ratkaisuja saadaan käymättä kuitenkaan kaikkia vaihtoehtoja läpi.

Geneettinen algoritmi on yksi sellainen menetelmä, vaikka se ei olekaan perusmuodossaan otollisin ratkaisemaan permutaatiotyyppisiä ongelmia. Syynä tähän on se, että yksinkertainen rekombinaatio kahdesta permutaatiosta ei yleensä ole permutaatio. Eli GA tuottaa paljon kelvottomia ratkaisuehdotuksia, jotka pitää karsia tavalla tai toisella. Ratkaisuna ongelmaan on korvata yksinkertainen rekombinaatio sellaisella, joka tuottaa kahdesta lähtöpermutaatiosta uuden permutaation. Tällaisia on esitetty kirjallisuudessa useitakin [148]. Toinen vaihtoehto on käyttää koodausta, joka sallii yksinkertaisen rekombinaation. Ongelman voidaan nimittäin katsoa olevan yhtä hyvin tiedon esitysmuodossa kuin siihen kohdistuvissa operaatioissa.

Tuotantolinjan suunnittelu

Markus Ettl ja Markus Schwehm ovat soveltaneet geneettistä algoritmia ja jonoverkkoanalyysiä (engl. queueing networks) tuotantolinjan optimaaliseen ositukseen JOT-tuotannon ja kanban-järjestelmän yhteydessä.

Ongelmana oli minimoida investointikustannukset, kuitenkin niin, että annettu kapasiteetti täytetään jakamalla linja osiin ja allokoimalla kanbankorttien määrät niille.

Geneettisen algoritmin antamaa tulosta on arvioitu vertaamalla sitä täydelliseen hakuun, joka vaati noin kahden miljoonan vaihtoehdon läpikäynnin. Hyvä ratkaisu löytyi nopeasti eli noin 20 sukupolvessa eli noin 1000-2000 vaihtoehdon laskennan jälkeen, jolloin tulos oli yleensä parin prosentin päässä globaalista optimista.

Kirjallisuutta [65, 67]

8.2.9 Laskennallinen kemia

Makromolekyylien kuten DNA:n, RNA:n ja proteiinien rakenteen laskemisessa geneettisen algoritmin voi ajatella palanneen juurilleen. Ei olekaan ihme, että GA on ollut suhteellisen suosittu kemiassa ja erityisesti bio- ja rakennekemian eli biologisten makromolekyylien tutkimuksessa.

DNA-ketjussa on yksiselitteinen proteiinin rakennekaava: kolme peräkkäistä nukleotidia määrää aina yhden aminohapon, joita on kaikkiaan kaksikymmentä erilaista. Proteiini koostuu sitten kymmenistä tai jopa sadoista peräkkäin, kuin helminauhaksi, järjestetyistä aminohapoista DNA:n täsmälleen määräämässä järjestyksessä.

Varsinainen ongelma on siinä, että pitkä ketjumainen proteiini vetäytyy "sykkyrään" eli laskostuu tietyllä tavalla saavuttaakseen aktiivisen tilansa. Se miten tämä laskostuminen toteutuu määrää puolestaan täysin proteiinin kemialliset ominaisuudet esimerkiksi entsyyminä. Aktiivinen eli siis laskostunut tila vastaa kutakuinkin molekyylin vapaan energian minimiä. Täten proteiinin rakenteen laskenta voidaan yrittää ratkaista optimointiongelmana. Vapaita parametreja aminohappoa kohden on ketjussa kaksi kulmaa, joiden ympäri jokainen aminohappomolekyyli voi enemmän tai vähemmän vapaasti pyöriä. Kaikkiaan vapausasteita proteiinimolekyylissä on siis useista kymmenistä satoihin ketjun pituudesta riippuen. Ongelman proteiinien laskostuksen voidaan ajatella olevan pitkälti geometrinen: erimuotoisia helmiä sisältävä helminauha pyritään pakkaamaan mahdollisimman pieneen tilaan.

Jos proteiinien laskostusongelma saadaan joskus ratkaistua, se tulee merkitsemään vähintään mikropiiritekniikan veroista vallankumousta materiaalitekniikan, lääketieteen ja biokemian aloilla, koska silloin tarjoutuu keino tuottaa mm halutunlaisia entsyymejä, rakennemateriaaleja (kuituja), (täsmä)lääkkeitä, reagensseja ja ties mitä nanotekniikan tuotteita.

Toistaiseksi ongelmaa ei ole ratkaistu, mutta alan asiantuntijat ovat optimistisia ja optimointialgoritmeilla tullee olemaan oma merkityksensä ongelman ratkaisemisessa.

Tätä raporttia kirjoitettaessa ilmestyi Science-lehdessä artikkeli [54] jossa laskostusongelma on ilmeisesti ratkaistu ns. sinkkisormen (n. 30 aminohappoa) tapauksessa. Ratkaisu perustui optimoinnin osalta termodynaamiseen menetelmään, mutta vastaavissa ongelmissa on myös käytetty geneettistä algoritmia.
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaPROTEINbib.ps.Z`

8.2.10 Geneettinen ohjelmointi

Oman ryhmänsä evoluutio-optimointimenetelmiä muodostaa ns. geneettinen ohjelmointi, jossa populaation muodostaa joukko tietokoneohjelmia. Erityisesti LISP on ollut suosittu kieli yksinkertaisen rakenteensa vuoksi. John Koza on ollut tämän tekniikan innokas tutkija Stanfordissa.

Menetelmä on sama kuin perinteinen bitti- tai muilla vektorityyppisillä yritteillä tehtävä geneettinen algoritmi, sillä erolla, että yritteinä ovat tietokoneohjelmat tai niiden osat. Konekielistä ohjelmaa käytettäessä menetelmät eivät oleellisesti eroa tietorakenteidensa puolesta, mutta muiden ohjelmointikielten osalta eroa on. Esimerkiksi LISP-kielen funktiot on yksinkertaisinta esittää lausekepuun muodossa. Puun lehtisolmut ovat muuttujia tai vakioita, kun taas sisäsolmut ovat funktioita.

Mutaatio lausekepuussa tarkoittaa solmun arvon muuttamista toiseksi esimerkiksi funktio `sin` muutetaan funktioksi `+`.

Rekombinaatio tapahtuu valitsemalla lausekepuista alipuut, jotka vaihdetaan keskenään (kuva 8.14).

Jokainen, joka on joskus vähänkään yrittänyt ohjelmoida, ymmärtää, että tällainen satunnainen ohjelmointi tuottaa paljon toimimattomia ohjelmia. Voiko satunnaisella menetelmällä ylipäätään tuottaa kelvollisia ohjelmia? Kyllä voi, kunhan yritteitä on riittävästi eikä yritä luoda liian monimutkaista ohjelmaa vaan ainoastaan hyvin rajoitetun tehtävän suorittavan funktion, kuten esimerkiksi satunnaislukugeneraattorin.

Tulokset ovat varsin hämmästyttäviä: Kozan menetelmää esittelevät kirjat sisältävät koko joukon perinteisiä tekoälyn tms. ongelmia, jotka ratkeavat kunhan ohjelmoija kertoo mitä (alkeis)funktioita kannattaa yhdistellä ja antaa hyvyysfunktion, joka kertoo miten lähellä kelpoista ratkaisua ollaan. Yhdisteltävien (alkeis)funktioiden joukossa voi olla mielivaltaisen monimutkaisia funktioita, jotka tyypillisesti sitten sisältävätkin erikoisalueen tietämystä.

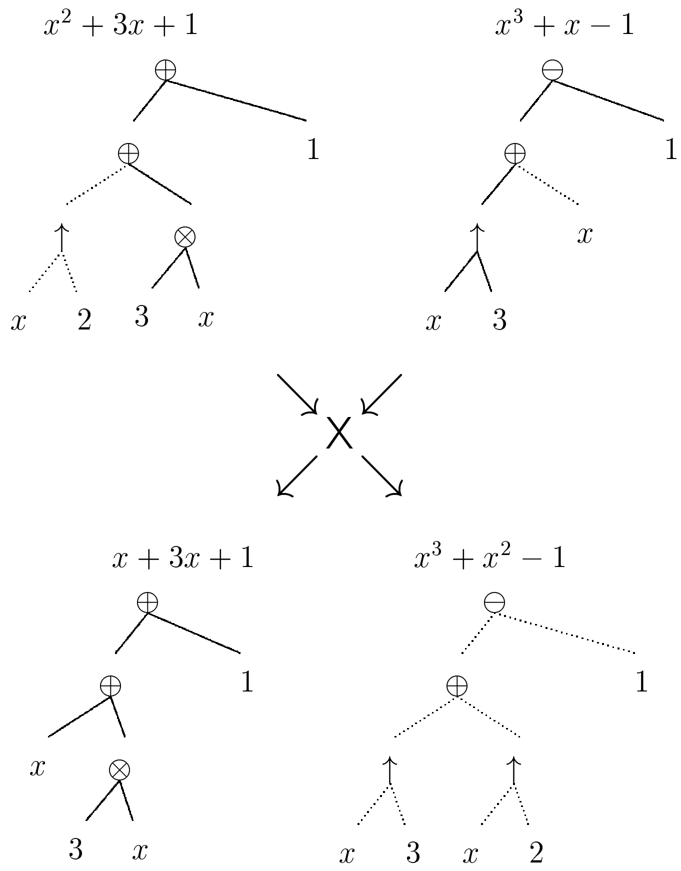
Koza *et al* esittää artikkelissa [130] neljä esimerkkiä suunnittelutehtävistä, joissa geneettinen ohjelmointi on tuottanut paremman tuloksen, kuin mihin muilla menetelmillä on pystytty. Nämä ongelmat olivat:

- 1D soluautomaatin majoriteettiluokittelija
- membraaniproteiinien transmembraani sekvenssien tunnistus
- ja pari lähinnä passiivisista komponenteista rakennettua analogista signaalien suodatinta

Viimemainitussa tapauksessa populaation koko oli melkoinen: 640.000! Muut parametrit olivat tavanomaisempia. Kumpaa kannattaa säästää koneen muistia ja aikaa vai suunnittelijan työtä?

Ehkä kaikkein silmiinpistävin GP sovellus on ollut tietokonegrafiikan tuottaminen sen avulla. Tämän tekniikan uranuurtaja on ollut Karl Sims ja William Latham, joista viime mainitun GP:lla luodun grafiikan näyttely on ollut esillä mm. Ateneumissa. URL:`http://www.nemeton.com/axis-mutatis/latham.html`

Kirjallisuutta [201, 133, 127, 102, 141, 130]
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaGPbib.ps.Z`



Kuva 8.14: Kahden lausekepuun rekombinaatio

Luku 9

Muita heuristisia menetelmiä

Geneettisiä algoritmeja on sovellettu varsin paljon erilaisiin enemmän tai vähemmän käytännöllisiin optimointitehtäviin. Tästä käytännöllisestä lähestymistavasta on tietysti seurannut se, että menetelmää on pyritty vertaamaan myös muihin vastaaviin, etupäässä heuristisiin, menetelmiin.

Yleensä GA on menestynyt hyvin, joskaan testit eivät yleensä ole olleet kovinkaan kattavia. Tietotekniikan alalla ei myöskään ole ollut juuri tapana julkaista negatiivisia tuloksia. Tuloksia esiteltäessä lähinnä vain todetaan menetelmien periaatteellinen toimivuus myös käytännössä. Suhtautuminen muilla tekniikan ja tieteen aloilla on ollut kriittisempää ja ehkä samalla myös hieman objektiivisempaa. Tähän, osittain tietotekniikan käytännöllisyyteen nähden paradoksaaliseen ja yleisesti vaivaavaan ongelmaan on viime aikoina kiinnitetty huomiota [109, 145, 146]. Tietotekniikan puolustukseksi todettakoon, että negatiivisesta tuloksesta eli siitä, että menetelmällä ei ole saatu ennakoituja tuloksia ei voi oikeastaan päätellä paljoakaan: menetelmä voi olla huono, mutta yhtä hyvin jotakin muuta on voinut mennä vikaan, mikä on varsin yleistä ongelmia ohjelmoiden ratkottaessa. Sen sijaan hyvien tulosten saaminen sattumalta tai viallisella ohjelmalla tai algoritmilla on tietysti varsin epätodennäköistä.

Kirjallisuutta [103, 113, 149]
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaSAbib.ps.Z` ns. simulated annealing menetelmän ja GA:n vertailuja.

9.1 Keinoelo

Viimeaikoina elämän simulointi eli keinoelo on ollut eräiden biologien ja tekoälytutkijoiden mielenkiinnon kohteena. Biologien kiinnostus selittyy sillä, että tietokoneella voidaan helposti tehdä kokeita, jotka laboratorio-oloissa olisivat vaikeita tai mahdottomia ja joita on vaikea matemaattisesti mallittaa ja analysoida. Tietokoneella voidaan myös simuloida täysin teoreettisia ja kuviteltuja eliöitä ja niiden muodostamien populaatioiden evoluutiota. Mahdollisuuksia on rajattomasti ja vain ohjelmoinnin työläys ja vaadittu laskentakapasiteetti asettavat rajat.

Tietotekniikan puolella keinoelossa nähdään mahdollisuus ratkoa mm. tekoälyn ongelmia lähtien liikkeelle hyvin yksinkertaisista perustoiminnoista. Vähin erin toivotaan kehityksen johtavan yhä "älykkäämpiin" ratkaisuihin.

Hyvänä esimerkkinä tästä on ns. muurahaisalgoritmi (engl. ant algorithm), jossa simuloidaan muurahaisyhteiskunnan kollektiivista ongelmanratkaisukykyä. Menetelmä osoittautui suoritetuissa alustavissa kokeissa hyvin kilpailukykyiseksi muiden globaalien optimointimenetelmien kanssa. Ehkäpä muurahaisten menestys evoluutiossa johtuu niiden ongelmanratkaisukyvystä [108].
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaALIFEbib.ps.Z`

9.2 Soft computing

Geneettiset algoritmit kuuluvat yhtenä osana ns. soft computing -menetelmiin, joista tunnetuimmat muut menetelmät ovat hermoverkot ja sumea logiikka. GA:lla voidaan numeeristen parametrien lisäksi käsitellä myös monimutkaisempia tietorakenteita, jopa ohjelmia, jolloin kyseessä on ns. geneettinen ohjelmointi, jonka avulla voidaan tuottaa automaattisesti lyhyitä mutta vaikeasti ohjelmoitavia funktioita, kuten satunnaislukugeneraattoreita.

Lotfi Zadeh on antanut yhteisen nimen soft computing menetelmille:

- hermoverkot (neural networks)
- sumea logiikka (fuzzy logic)
- evoluutioalgoritmit yms (evolution strategies, genetic algorithms, genetic programming, immune systems,...)

Termi kuvaa näiden menetelmien vastakohtaisuutta verrattuna deterministisiin tekoälymenetelmiin, jotka pohjautuvat esimerkiksi matemaattiseen formaaliin logiikkaan ja sen sovellutuksiin.

Menetelmät ovat heuristisia ja tämä on inspiroinut myös termin **meta-heuristiikat** (meta-heuristics) käyttöön, joka kuitenkaan ei ainakaan vielä ole saavuttanut suurempaa suosiota. Termin alle on kerätty soft computing menetelmien lisäksi joukko muitakin heuristiikkoja kuten tabu-search.

Kirjallisuutta [83, 84, 159].

9.3 Hybridit ja globaali optimointi

Koska geneettinen algoritmi on parhaimmillaan globaalissa optimoinnissa eikä lokaalissa, niin usein käytetäänkin, jos vain mahdollista, hyväksi paikallista menetelmää esimerkiksi alussa esiteltyä rinteen kapuamista tai vaikka Newtonin menetelmää.

Silminpistävän vähän on ollut yhteyksiä toisaalta geneettisiä algoritmeja harrastavien ja toisaalta optimointialan tutkijoiden välillä.

Kirjallisuutta [174]

9.4 Lähialueita

9.4.1 Algoritmit ja kemia

Nobelisti Manfred Eigen on suorittanut optimointia käyttämällä hyväksi RNA synteesiä ja siinä tapahtuvia mutaatioita. Sittemmin on samankaltaisella menetelmällä ratkaistu täysin kemiallisesti ns. kauppamatkustajan ongelma.

Walter Fontana on esittänyt kemiaa muistuttavan laskentamenetelmän ns. algoritmisen kemian.

Immuunijärjestelmä on selkärankaisten kemiallinen puolustusmenetelmä ulkoisia tunkeutujia (myrkyt, virukset, bakteerit, yms) vastaan. Se perustuu vasta-aine vaihtoehtojen satunnaiseen generointiin ja näiden vaihtoehtojen karsintaan. Periaatetta on jonkin verran sovellettu laskentaan GA:n tapaan.

Kirjallisuutta [190, 71, 76, 77, 202, 43, 74]

9.4.2 Termodynaaminen menetelmä

Termodynaaminen tai simuloitujäähdytys menetelmä (engl. simulated annealing) on nimensä mukaisesti myös satunnaisuuteen perustuva optimointimenetelmä. GA:sta se eroaa erityisesti siinä, että kulloinkin tarkastellaan vain yhtä ratkaisua, jonka annetaan vaeltaa ratkaisuavaruudessa optimoitavan funktion ja lämpötilaparametrin määräämissä rajoissa. Aluksi lämpötila on korkea ja ratkaisu voi vaeltaa lähes riippumatta funktion arvoista. Vähitellen kuitenkin lämpötilaa alennetaan, jolloin yrite voi liikkua yhä rajoitetummin, kunnes päädytään pisteeseen, jonka ympäristö sulkee yritteen sisäänsä.

On tultu ääriarvoon, joka hyvällä onnella voi olla globaali. Normaalisti etsintä toistetaan useita kertoja peräkkäin ja valitaan löytyneistä ratkaisuista paras.

Kirjallisuutta [210, 166, 168]

Luku 10

Tietolähteitä

Lääkäri: "Minkälaisen ongelma vuoksi tulitte?"
Potilas: "En tiedä, kun en ole lääkäri."

Osmo Saarelma, *Hyvä Terveys 7/97*

10.1 Ohjeita

Tässä kappaleessa annetaan lyhyt lista asioista, jotka on hyvä tarkistaa ennen kuin ryhtyy soveltamaan evolutionäärisiä tai yleensä optimointimenetelmiä jonkin tehtävän ratkaisemiseen.

1. **Optimointiongelma?** Onko tehtävä optimointi- tai etsintätyyppinen tai voidaanko se mahdollisesti muokata sellaiseksi? Ellei näin ole niin, saattaa olla, että jokin muu lähestymistapa kuin optimointi on paremmin paikallaan.

2. **Olemassa olevat menetelmät** Onko ongelman ratkaisemiseksi jo olemassa hyviä ja tehokkaita menetelmiä? Jos on, niin käytä ensin niitä ja palaa evolutionäärisiin menetelmiin, vain jos todellista tarvetta edelleen ilmenee.

3. **Kustannusfunktio olemassa?** Onko tehtävän ratkaisulle löydettävissä kustannusfunktio? Ideaalinen kustannusfunktio on jo valmis funktio tai ohjelma, josta ehkä vain pienin muutoksin saadaan tarvittava funktio. Kustannusfunktio voi olla myös esimerkiksi laitteisto tai vaikka ihminen, joka arvioi ratkaisujen laatua. Jos kustannusfunktiota ei ole. Mieti miten suuri työ sellaisen tekeminen on.

4. **Kustannusfunktion ominaisuudet?** Jos kustannusfunktio on yksinkertainen kuten (lähes)lineaarinen tai vain parin parametrin funktio, niin tarkista edellisen kohdalla lineaarisen ohjelmoinnin mahdollisuudet ja jälkimmäisessä tapauksessa perinteisten numeeristen optimointimenetelmien. Jos taas kustannusfunktiossa on hyvin paljon (tuhansia) ei-lineaarisia parametreja, niin tuskin sen ratkaiseminen onnistuu millään nykytekniikan keinolla.

5. **Hybridi?** Onko olemassa jo joku ratkaisumenetelmä, joka kuitenkin usein jää paikalliseen optimiin. Tällöin tuon menetelmän ja GA:n hybridi voi olla suhteellisen tehokas ja varmempi ratkaisumenetelmä.

6. **Ongelman ominaispiirteiden hyödyntäminen** Mieti mikä on oleellista ja miten tehtävän erityispiirteitä voitaisiin hyödyntää. Onko esimerkiksi olemassa jokin menetelmä, joka ei kuitenkaan aina ja kaikissa tapauksissa anna hyviä tuloksia, mutta jonka voisi liittää yhteen GA:n kanssa?

7. **Ennakkotapauksia?** Ennen kuin ryhtyy todella ratkaisemaan ongelmaa GA:lla tai muulla vastaavalla menetelmällä kannattaa tarkastaa kirjallisuudesta onko kukaan ratkonut vastaavaa ongelmaa. (Tämä käy helpoiten Vaasan GA-bibliografioiden avulla `ftp.uwasa.fi:cs/report94-1/ga*bib.ps.Z`, joissa on tiedot tuhansista sovelluksista.) Myös muualta tietoa löytyy ja alan asiantuntijat vastaavat yleensä mielellään omaa alaansa ja sen sovelluksia koskeviin kyselyihin.

10.2 Optimointikysely

Jotta olisi saatu karkea yleiskuva teollisuuden optimointitekniikoiden tuntemuksesta ja hyödyntämisestä jatkotoimia ja suosituksia varten, tehtiin yksinkertainen kyselytutkimus. Liitteen A mukaisia kyselylomakkeita postitettiin maksetun vastauskuoren kera noin 760 yritykselle, joiden osoitetiedot saatiin Kompass-tietokannasta. Yritykset valittiin teollisuuden ja tietoliikenteen ryhmästä siten, että mukaan otettiin ne, joilla oli suurin liikevaihto. Vastauksia palautettiin noin 110 kpl eli varsin tarkkkaan 1/7 (yli 14%). Yhteenveto vastauksista on esitetty taulukossa 10.1.

Ensimmäinen kysymys koski optimoinnin tärkeyttä yrityksen tuotesuunnittelutoiminnassa. Lähes täsmälleen puolet vastaajista totesi optimoinnin olevan yrityksensä tuotekehityksen tai tuotannonohjauksen kannalta tärkeää. Näin tietysti on mutta ehkäpä ne, jotka näin arvelivat olivat aktiivisempia vastaajia, kuin ne joiden mielestä optimointi ei ole tärkeää. Osa yrityksistä oli

kyllä	*ei osaa sanoa*	*ei*	*kysymys*
54	28	26	Optimointi tärkeää tuotekehityksessä
11	14	**83**	Käytetään automaattisia menetelmiä
8	24	**73**	Saatu koulutus riittävä
76	13	18	Kiinnostunut lisätiedoista
30	**47**	29	Kiinnostunut alan kurssituksesta

Taulukko 10.1: Optimointikyselyn tulokset.

tutkinto	*lkm*
teknikko	5
ins.	**42**
DI	**33**
tekn. lis.	1
kauppatiet. kand.	1
ekon.	8
M.Sc.	1
FL	1
valtiot. m.	1
muu	5
Yhteensä	108

Taulukko 10.2: Vastaajien koulutustausta (tutkinnot)

puhtaasti maahantuojia, joilla ei ollut omaa tuotekehitystä tai suunnittelua, vaikka kysely pyrittiin kohdistamaan tuotannollisiin yrityksiin.

Toisessa kysymyksessä tiedusteltiin yleisesti onko yrityksessä käytössä automaattisia optimointimenetelmiä. Noin kymmenessä prosentissa yrityksistä eli joka viidennessä, joissa optimointi koettiin tärkeäksi, näin oli asianlaita. Melkein sama prosenttiosuus vastasi saamansa optimointimenetelmien koulutuksen olleen riittävän. Yhdessä nämä vastaukset kertovat erittäin selvästi, että suomalaisella teollisuudella on todella tarvetta nostaa optimointitekniikoiden osaamisen ja soveltamisen tasoa. Tämä on haaste myös suunnittelijoiden koulutukselle, jossa optimointimenetelmät pitäisi tuoda paremmin esille. Muutamalle vastaajalle jopa optimointi terminä oli vieras!

Tiedon tason vaatimattomuus näkyi myös kahden seuraavan kysymyksen vastauksissa. Kysyttiin oliko vastaaja kiinnostusta lisätiedoista ja mahdollisesta kurssituksesta. Lisätiedoista oli kiinnostunut yli 2/3 vastaajista ja yli 1/4 oli kiinnostunut kurssituksesta. Ehkä lomakkeen olivat ahkerimmin palauttaneet optimoinnista todella kiinnostuneet, mutta joka tapauksessa kentällä näkyy olevan tarvetta hankkia lisää tietoa optimointitekniikoista.

Vastaajista noin 2/3 oli tekniikan alan koulutus eli olivat pääosin insinöörejä tai diplomi-insinöörejä. Lopun 1/3 koulutus oli varsin heterogeeninen (taulukko 10.2). Kysely lienee tavoittanut varsin hyvin suunnittelusta vastaavat henkilöt.

Kiitokset kaikille vastanneille.

10.3 Suosituksia toimenpiteiksi

Tietotekniikan ja yleensäkin korkeatasoisen tekniikan tehokas hyödyntäminen lienee ainoa keino, jolla suomalainen tuotantoelämä pidemmän päälle menestyy kovassa kansainvälisessä kilpailussa. Erityisen tärkeä on pystyä tuottamaan parempilaatuisia ja halvempia tuotteita, kuin kilpailijat, usein hyvinkin eriytyneillä markkinoilla. Tämän vuoksi kaikkien optimointimenetelmien, ei vain evolutionääriseen periaatteeseen perustuvien, tuntemus ja soveltaminen on ensiarvoisen tärkeää. Kokemus on kuitenkin osoittanut, että optimointi on jo terminä teollisuuden suunnittelijoille usein vieras, vaikka kaikki pyrkivätkin optimoimaan tuotteitaan.

10.3.1 Täydennyskoulutus

Ehdotankin, että teollisuuden suunnittelijoiden optimointitietämyksen tasoa pyritään nostamaan järjestämällä aiheesta seminaareja ja koulutustilaisuuksia, joissa suunnittelijat voisivat asiantuntijoiden opastuksella perehtyä menetelmiin ja saada niistä myös kokemusperäistä tietoa.

Nämä kurssit tulisi pääsääntöisesti ottaa osaksi suunnittelijoiden tutkintoihin tähtäävää täydennyskoulutusta (ins $\rightarrow$ DI, DI $\rightarrow$ TkT, FK $\rightarrow$ FT, ekon. $\rightarrow$ KTT).

Kurssin runko voisi olla esimerkiksi seuraavanlainen:

- johdanto & kertausta optimointiin
- tärkeimmät optimointimenetelmät
- ohjelmistojen esittelyjä ja demonstraatioita
- sovelluksia
- harjoitustyö

10.3.2 Tekesiin optimointitekniikan ohjelma?

Optimoinnin suuren taloudellisen merkityksen sekä selvän tarpeen vuoksi olisi aiheellista pohtia optimointitekniikoihin ja niiden kehittämiseen ja tietämyksen levittämiseen pyrkivän teknologiaohjelman tarpeellisuutta.

Paitsi geneettisiä algoritmeja, ohjelman pitäisi sisältää myös muita menetelmiä kuten kokeiden suunnittelua (Taguchi) ja perinteisempiä optimointimenetelmiä.

10.4 Ohjelmistot

GA:n periaate on niin yksinkertainen, ettei menetelmän toteutus aiheuta vaikeuksia tietotekniikan taitajalle. Sen lisäksi on olemassa joukko ohjelmapaketteja, joiden avulla menetelmään tutustuminen käy helposti. Internet uutisryhmän `comp.ai.genetic`:n FAQ (Frequenly Asked Questions) sisältää mm. varsin hyvän listan ns. public domain GA-ohjelmia. Tämän FAQ-dokumentin lisäksi alla olevissa `ftp` palvelimissa on runsaasti myös muuta GA aineistoa:

```
ftp lumpi.informatik.uni-dortmund.de
cd /pub/EA/docs
binary
get hhgtec-2.2.ps.gz (gzipped postscript)

ftp.Germany.EU.net
cd /pub/newsarchive/news.answers/ai-faq/genetic/
mget part? (text version)

http://www.cis.ohio-state.edu/hypertext/faq/usenet/ai-faq/genetic/top.html
```

Kirjallisuutta [32, 176, 96]
→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaIMPLEbib.ps.Z`

10.5 Patentit

Ohjelmien patentointi on tunnetusti vaikeaa. Täten on ymmärrettävää, ettei kovinkaan monta geneettisiin algoritmeihin liittyvää patenttia ole olemassa. Taulukkoon 10.3 on koottu Vaasan bibliografiasta löytyvät patentit, jotka ovat pääosin peräisin joko Goldbergin tai Kozan bibliografioista [89, 128].

viite	*patentin nimi*
[126]	A non-linear genetic algorithms for solving problems
[129]	A non-linear genetic algorithms for solving problems
[134]	A non-linear genetic process for data encoding and for solving problems using automatically defined functions
[136]	A non-linear genetic process for problem solving using spontaneously emergent self-replicating and self-improving entities
[106]	Adaptive computing system cabable of learning and discovery
[81]	Machine learning procedures for generating image domain feature detectors
[70]	Method and apparatus for training a neural network using evolutionary programming
[107]	Method of controlling a classifier system
[124]	Non-linear genetic algorithms for solving problems
[125]	Non-linear genetic algorithms for solving problems by finding a fit composition of functions
[135]	Non-linear genetic process for data encoding and for solving problems using automatically defined functions
[131]	Non-linear genetic process for use with co-evolving populations
[132]	Non-linear genetic process for use with plural co-evolving populations
	yhteensä 12 patenttia

Taulukko 10.3: GA patentit Vaasan GA bibliografian mukaan.

10.6 EvoNet

Alan tutkijat Euroopan Unionin alueella ovat järjestäytyneet Unionin rahoittamaksi ns. network of excellence organisaatioksi, jonka nimi on EvoNet. Tämä verkosto pyrkii edistämään alan tutkimusta ja menetelmien soveltamista käytäntöön mm. julkaisemalla tiedotuslehteä **EvoNews** ja ylläpitämällä erilaisia evoluutiomenetelmiä koskevia www ja ftp-palveluja (`http://www.dcs.napier.ac.uk/evonet/evonet.htm`).

Taulukossa 10.4 on lueteltu EvoNetin työryhmät ja niiden e-mail osoitteet.

10.7 Konferenssit

Alalla on parisen kymmentä kansainvälistä konferenssia vuosittain. Lisäksi sovelluksia esitellään lukuisissa muiden alojen kokouksissa.

- FOGA (Foundations of Genetic Algorithms) nimensä mukaisesti GA:n teoriaa käsittelevä pieni mutta korkeatasoinen konferenssi.
- ICGA (International Conference on Genetic Algorithms) parittomina vuosina USA:ssa järjestettävä alan merkittävin ja vanhin konferenssi.
- PPSN (Parallel Problem Solvin from Nature) Euroopassa parillisina vuosina järjestettävä GA:ta ja lähialoja käsittelevä konferenssi.

e-mail	*nimi*
`evonet.exec`	Execute Committee
`evonet.rc`	Research Committee
`evonet.tc`	Training Committee
`evonet.ilc`	Industrial liason Committee
`evonet.ecc`	Electronic Communications Committee
`ectel`	WG on Evolutionary Computing in Telecommunications
`evogp`	WG on Genetic Programming
`evoelec`	WG on Evolutionary Electronics
`evostim`	WG on Scheduling and Time-tabling
`evorob`	WG on Evolutionary Robotics

Taulukko 10.4: EvoNetin komiteat ja työryhmät. E-mail osoitteen osoiteosa on kaikille yhteinen = `@dcs.napier.ac.uk`

- NWGA (Nordic Workshop on Genetic Algorithms) Pohjoismainen jokavuotinen työkokous, jossa on perinteisesti ollut suomalaisten osallistujien lisäksi Pohjoismaiden ulkopuolelta tulleita tutkijoita.

- EA (Evolution Artificielle) ranskalainen GA konferenssi

- IEEE-ICEC yksi monista IEEE:n massakonferensseista; yhdessä hermoverkkojen ja sumean logiikan vastaavien konferenssien kanssa.

- MENDEL, Tsekin tasavallassa vuosittain järjestettävä GA yms konferenssi munkki Mendelin asuinsijoilla.

- ICANNGA (International Conference on Artificial Neural Networks and Genetic Algorithms) parittomina vuosina järjestettävä GA:a, hermoverkoja sekä niiden yhdistelmiä käsittelevä Eurooppalainen konferenssi.

- GP (Annual International Genetic Programming Conference) Amerikassa John Kozan ja kumppanien järjestämä jokavuotinen geneettistä ohjelmointia käsittelevä konferenssi. Paikalla lähes kaikki alan aktiivit.

10.8 Geneettisten algoritmien kirjallisuus

"Jos akateemisessa maailmassa julkaistaan uusi tutkimustulos luonnontieteiden tai humanististen tieteiden alalta tiedon eturintamasta, tutkimuksen julkaisija voi saada palkinnokseen professuurin tai ylennyksen, vaikka tulos myöhemmin osoittautuisi vääräksi. ...
Ihmiskunnan asiat ovat paljon paremmin sitten, kun palkitsemisen perusteet muuttuvat niin, että valintapaineet uralla suosivat yhtä lailla tiedon lajittelua kuin hankintaakin."

Nobel-fyysikko Murray Gell-Mann, *Kvarkki ja jaguaari [80, s. 406]*

Kirjoittaja on viime vuosina kerännyt systemaattisesti kaiken GA alan kirjallisuuden sekä itse artikkelit sikäli kuin ne ovat olleet helposti saatavilla (TKK:n pääkirjasto, `ftp` jne) että viittaukset yleisimmistä indeksijulkaisuista. Kuvassa 10.1 on esitetty artikkelien (sisältää myös muun aineiston) vuotuinen määrä. Kuten kuvasta voi nähdä, niin viimeisten vuosien aikana julkaisujen määrä on kasvanut varsin tasaista noin 40% vuosivauhtia eli kaksinkertaistunut joka toinen vuosi. Kirjallisuus vuoteen 1993 asti on kerätty kirjaseen [7]. Sittemmin olen koonnut perusaineistosta suppeampia aihealueita kattavia erikoisbibliografioita, jotka ovat saatavilla `ftp`:llä (taulukko 10.5).

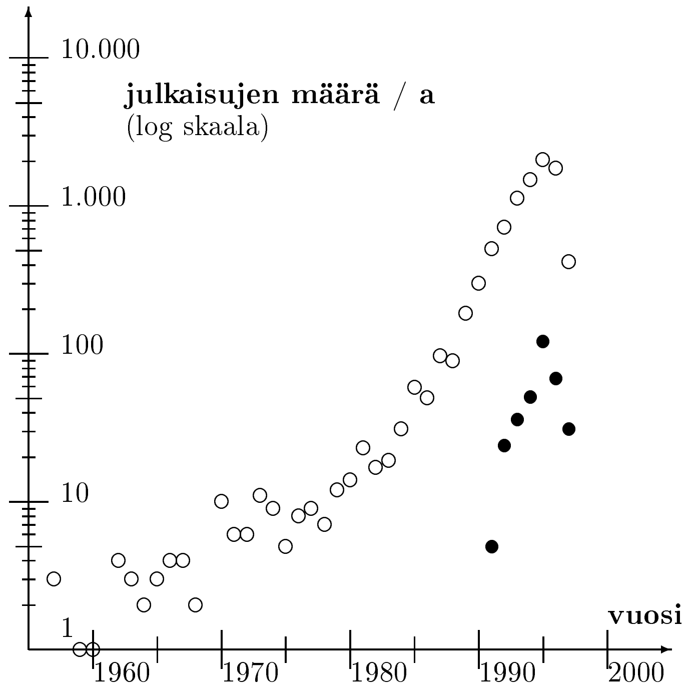
Taulukossa 10.6 on esitetty GA artikkelien maantieteellinen jakautuminen. Yli 90% viitteistä on luokiteltu tekijöiden koti- tai oleskelumaan mukaan. Tutkimus on ollut selvästi vilkkainta Yhdysvalloissa (noin 1/3 kaikista julkaisuista). Tuhannen julkaisun rajan ovat ylittäneet Englanti ja Japani, joka ilmeisesti on kovaa vauhtia ottamassa kakkossijan sitä pitkään ja selvästi hallinneelta Englannilta. Neljännellä sijalla on Saksa noin 8%:n osuudella. Sitten tulevatkin Suomi, Italia, Ranska, Australia ja Kiina parin prosentin osuuksillaan. Kaikkiaan julkaisuja on noin 60 maasta. Suomen sijalukua selittää osittain se, että suomalaiset tutkimukset on saatu varsin tarkasti mukaan tietokantaan, mutta joka tapauksessa tutkimusaktiviteetti on ollut melkoinen kansainvälisestikin mitattuna.

Taulukossa 10.7 on esitetty lehdet joissa on julkaistu eniten geneettisiin algoritmeihin liittyviä tutkimuksia. Yhteensä artikkeleja on lähes 2500 kaikkiaan vajaasta 10.000 viitteestä eli noin 25% kaikista julkaisuista.

Taulukossa 10.8 on esitetty julkaisujen määrän mukaan aktiivisimmat geneettisten algoritmien tutkijat ja heidän tutkimusalansa.

tiedosto	*sisältö*
ga90bib.ps.Z	GA vuonna 1990
ga91bib.ps.Z	GA vuonna 1991
ga92bib.ps.Z	GA vuonna 1992
ga93bib.ps.Z	GA vuonna 1993
ga94bib.ps.Z	GA vuonna 1994
ga95bib.ps.Z	GA vuonna 1995
ga96bib.ps.Z	GA vuonna 1996
ga97bib.ps.Z	GA vuonna 1997
gaAIbib.ps.Z	GA ja tekoäly
gaALIFEbib.ps.Z	GA ja keinoelo
gaARTbib.ps.Z	GA taiteessa ja musiikissa
gaAUSbib.ps.Z	GA Australiassa
gaBASICSbib.ps.Z	GA:n perusteita
gaBIObib.ps.Z	GA biotieteissä (mm. lääketiede)
gaCADbib.ps.Z	GA ja tietokoneavusteinen suunnittelu
gaCHEMPHYSbib.ps.Z	GA kemiassa ja fysiikassa
gaCONTROLbib.ps.Z	GA ohjauksessa ja säädössä
gaCSbib.ps.Z	GA tietotekniikassa (mm. tietokannat ja GP)
gaECObib.ps.Z	GA taloustieteissä
gaENGbib.ps.Z	GA tekniikassa
gaESbib.ps.Z	Evoluutio strategiat
gaFAR-EASTbib.ps.Z	GA Kauko-Idässä (mm. Japani ja Kiina)
gaFRAbib.ps.Z	GA Ranskassa ja ranskaksi
gaFTPbib.ps.Z	`ftp`:n ja `www`:n avulla saatavissa olevat GA julkaisut
gaFUZZYbib.ps.Z	GA ja sumea logiikka
gaGERbib.ps.Z	GA Saksassa ja saksaksi
gaGPbib.ps.Z	GP (geneettinen ohjelmointi)
gaIMPLEbib.ps.Z	GA implementointeja
gaISbib.ps.Z	immuunisysteemit
gaJOURNALbib.ps.Z	lehdissä julkaistut artikkelit
gaLOGISTICSbib.ps.Z	GA logistiikassa
gaMANUbib.ps.Z	GA tuotantotoiminnassa
gaMEDITERbib.ps.Z	GA Välimeren alueella
gaNNbib.ps.Z	GA ja hermoverkot
gaNORDICbib.ps.Z	GA Pohjoismaissa (pääasiassa Suomessa)
gaOPTIMIbib.ps.Z	GA ja optimointi (vain muutama)
gaOPTICSbib.ps.Z	GA optiikassa ja kuvankäsittelyssä
gaORbib.ps.Z	GA ja operatiotutkimus
gaPARAbib.ps.Z	Rinnakkaiset ja hajautetut GA
gaPOWERbib.ps.Z	GA sähkö- ja energiatekniikassa
gaPROTEINbib.ps.Z	GA ja proteiinit
gaROBOTbib.ps.Z	GA robotiikassa
gaSAbib.ps.Z	GA ja termodynaaminen menetelmä (SA)
gaSIGNALbib.ps.Z	GA signaalien käsittelyssä
gaTHEORYbib.ps.Z	GA:n teoriaa ja analysointia (mm. vertailuja)
gaTOP10bib.ps.Z	Ahkerimmat GA tutkijat (yli 10 paperia julkaisseet)
gaUKbib.ps.Z	GA Englannissa
gaVLSIbib.ps.Z	GA VLSI-piirien suunnittelussa ja testauksessa

Taulukko 10.5: Erikois GA -bibliografioita `ftp.uwasa.fi` palvelimella hakemistossa `cs/report94-1`.



Kuva 10.1: Koko Maailman (○) ja pohjoismaiset (•) geneettisten algoritmien julkaistut tutkimukset vuosittain. Huomaa, että pari viimeistä vuotta ovat vielä varsin puutteellisesti tiedossa.

10.8.1 Perusteita

Hyviä geneettisten algoritmien yleisesityksiä ovat mm. seuraavat artikkelit [47, 72, 73, 39, 40, 86, 205]. Näiden lisäksi aiheesta on kirjoitettu oppikirjoja [170, 104, 191, 192, 204, 85, 49, 56, 105, 128, 153, 172, 206, 48], joista ehkä David Goldbergin kirja [85] on käytetyin, joskin se on Goldbergin itsensäkin mukaan jo auttamattomasti vanhentunut.

Geneettisten algoritmien suosio sai alkunsa lähinnä Goldbergin käytännöllisistä sovelluksista [88] ja hänen ilmeisesti niiden innoittamana kirjoittamasta kirjasta. Ehkäpä myös hermoverkkojen esimerkki on ollut osaltaan rohkaisemassa tutkijoita ainakin Suomessa, jossa hermoverkkojen tutkimus on ollut vilkasta ja menestyksellistä jo pitkään.

Varsinaisena alan perusteoksena voidaan pitää John Hollandin kirjaa [104], johon usein viitataan, mutta joka kuitenkin on varsin teoreettinen ja käytännön sovelluksille kaukainen.

→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaBASICSbib.ps.Z`

Maa	*julk.*	*%*	*Maa*	*julk.*	*%*
Yhteensä	9449	100.00	Yhdysvallat	2901	30.70
Englanti	1004	10.63	Japani	1004	10.63
tuntematon	876	9.27	Saksa (ml. DDR)	772	8.17
Suomi	278	2.94	Italia	243	2.57
Ranska	233	2.47	Kiina (ml. Hong Kong)	211	2.23
Australia	204	2.16	Etelä-Korea	152	1.61
Taiwan	148	1.57	Kanada	129	1.37
Hollanti	112	1.19	Intia	97	1.03
Espanja	90	0.95	Belgia	86	0.91
Tsekin tasavalta	66	0.70	Puola	56	0.59
Sveitsi	56	0.59	Israel	52	0.55
Singapore	49	0.52	Venäjä	43	0.46
Itävalta	38	0.40	Brasilia	33	0.35
Kreikka	32	0.34	**Ruotsi**	30	0.32
Tanska	29	0.31	Romania	23	0.24
Unkari	21	0.22	Saudi-Arabia	20	0.21
Slovenia	20	0.21	Turkki	20	0.21
Portugali	16	0.17	Irlanti	15	0.16
Slovakia	12	0.13	Kuwait	11	0.12
Meksiko	11	0.12	Venezuela	11	0.12
Kroatia	8	0.08	Bulgaria	7	0.07
Iran	7	0.07	Thaimaa	7	0.07
Ukraina	7	0.07	Valkovenäjä	6	0.06
Uusi-Seelanti	6	0.06	Jugoslavia	5	0.05
Etelä-Afrikka	4	0.04	Argentiina	3	0.03
Kypros	3	0.03	Marokko	3	0.03
Norja	3	0.03	Chile	2	0.02
Egypti	2	0.02	Arabiemiirikunnat	2	0.02
Azerbaidžan	1	0.01	Kolumbia	1	0.01
Jordania	1	0.01	Libanon	1	0.01

Taulukko 10.6: GA tutkimusten maantieteellinen jakauma Vaasan GA-bibliografiatietokannan mukaan.

10.9 Suomalaisia GA projekteja

Lähes kaikissa Suomen yliopistoissa ja korkeakouluissa on tutkittu GA:n käyttöä. Seuraavassa listassa muutamia tärkeimpiä esimerkkejä:

- perusteita: sekä teoriaa että menetelmien empiiristä testausta [1, 3, 33, 69, 187, 188, 189]

Lehti	*artikkelia*
Evolutionary Computation	39
Electronics Letters	38
Complex Systems	38
Computers & Operations Research	36
Computers & Industrial Engineering	35
IEEE Transactions on Systems, Man, and Cybernetics	30
Nippon Kikai Gakkai Ronbunshu C Hen	27
IEEE Transactions on Magnetics	26
Pattern Recognition Letters	25
IEEE Transactions on Power Systems	25
Machine Learning	24
IEEE Transactions on Neural Networks	22
Biological Cybernetics	22
IEEE Expert	21
AI Expert	21
yhteensä lehtiartikkeleita noin tuhannessa lehdessä	2421

Taulukko 10.7: Tärkeimmät alan lehdet

- säätö ja ohjaus: hissien ryhmäohjaus [28, 29, 13, 25, 14, 156], PID-säätäjä [209]
- muodon optimointi: dieselmoottorin nokka-akselit [26, 18, 139] lentokoneen siipiprofiilien optimointi [208]
- sähkömoottorin optimointi [163]
- energian tuoton optimointi [150]
- laitetason toteutukset [13, 101, 100, 99]
- kemiallisen reaktorin suunnittelu [12]
- hiukkaskiihdyttiminen suunnittelu [180].
- signaalin käsittely ja suodattimet [152, 213, 111, 123, 157, 214]
- hermoverkot [98, 36, 35]
- tietokantakyselyt [184, 183, 185, 186]
- graafien piirtäminen [63]
- ohjelmien testaus [24]

julk.	*nimi*	*tutkimuskohteet*	*laitos*
9394	*kaikki tutkijat yhteensä*		
127	Goldberg, David E.	sovellukset	U. Illinois at Urbaba-Champaign
104	Alander, Jarmo T.	sovellukset, bibliografiat	Vaasan yliopisto
85	Fogel, David B.	evolutionary computing	Natural Selection, Inc.
68	Fogarty, Terence C.	sovellukset	Napier University
63	Koza, John R.	geneettinen ohjelmointi (GP)	Stanford U.
61	Fukuda, Toshio	robotiikka	Nagoya University
59	Whitley, Darrell	teoria	Colorado State U.
53	Karr, Charles L.	sumeat säätäjät	U. Alabama
52	Schwefel, Hans-Paul	evoluutiostrategiat	U. Dordmund
48	Grefenstette, John J.	sotilassovellukset	US Navy
46	Dorigo, Marco	robotiikka	U. Brussels, IRIDIA
46	Jong, Kenneth A. De	perusteet	George Mason U.
46	Michalewicz, Zbigniew	optimointi	U. North Carolina
45	Mühlenbein, Heinz	breeder GA	GMD, Saksa
44	Bäck, Thomas	perusteet	U. Dordmund
39	Banzhaf, Wolfgang	teoria	U. Dordmund
39	Forrest, Stephanie	soluautomaatit yms.	Santa Fe Instituutti
39	Garis, Hugo de	hermoverkot	ATR Labs, Kioto
38	Deb, Kalyanmoy	teoria	U. Illinois at Urbana-Champaign
37	Holland, John H.	perusteet	San Fe Instituutti
35	Parmee, Ian C.	robotiikka	U. Sussex
34	Harvey, Inman	robotiikka	U. Sussex
33	Shibata, Takanori	robotiikka	MITI
32	Gen, Mitsuo	optimointi	Ashikaga Inst. of Tech.
32	Uchikawa, Yoshiki	sovellukset	Nagoya University
31	Herrera, Francisco	sumea logiikka	U. Granada
31	Husbands, Philip	robotiikka	U. Sussex
30	Davis, Lawrence	sovellukset	Tica Associates
30	Fogel, Lawrence J.	evolutionary computing	Natural Selection, Inc.
30	Iba, Hitoshi	säätö, GP	Tokyo Inst. of Tech.
30	Lucasius, Carlos B.	kemia	U. Nijmegen
30	Reeves, Colin R.	OR	U. Coventry
30	Voigt, Hans-Michael	perusteet	TU Berlin

Taulukko 10.8: Aktiivisimmat GA-tutkijat Vaasan GA-bibliografian mukaan.

→ Vaasan GA bibliografia: `ftp.uwasa.fi:cs/report94-1/gaNORDICbib.ps.Z`

10.10 Tämän raportin GA-osuus Internetissä

Tämän raportin GA-osuuteen liittyy oleellisena osana elektronisessa muodossa oleva tieto eli

- tämän raportin GA-osuuden päivitetty versio PostScript^{TM}-muodossa `ftp.uwasa.fi:cs/report97-4/Finnish.ps.Z` tai `html`-muodossa URL:`http://www.uwasa.fi/GA/Finnish.html`.
- geneettisten algoritmien erikoisbibliografioita, joihin on koottu perusbibliografiatietokannasta eri alojen indeksoituja koosteita (taulukko 10.5 sivulla 165): `ftp.uwasa.fi:cs/report94-1/ga*bib.ps.Z`.
- Vaasan GA-bibliografian www kyselyliittymä URL:`http://www.uwasa.fi/GA/biblio`.
- lisäksi hakemistossa `ftp.uwasa.fi:cs/` on joukko alihakemistoja, joissa puolestaan on joukko geneettisiä algoritmeja ja niiden sovelluksia käsitteleviä artikkeleita (katso `README` tiedosto kussakin hakemistossa).

Kirjallisuutta

[1] Jarmo T. Alander. On finding the optimal genetic algorithms for robot control problems. In *Proceedings IROS '91 IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, volume 3, pages 1313–1318, Osaka, 3.-5. November 1991. IEEE Cat. No. 91TH0375-6. .

[2] Jarmo T. Alander, editor. *Geneettiset algoritmit – Genetic Algorithms*, number TKO-C53. Helsinki University of Technology (HUT), Department of Computer Science, 1992. (Proceedings of a GA Seminar held at HUT).

[3] Jarmo T. Alander. On optimal population size of genetic algorithms. In Patrick Dewilde and Joos Vandewalle, editors, *CompEuro 1992 Proceedings, Computer Systems and Software Engineering, 6th Annual European Computer Conference*, pages 65–70, The Hague, 4.-8. May 1992. IEEE Computer Society, IEEE Computer Society Press. .

[4] Jarmo T. Alander. Optimal GA. In Alander [2], pages 164–179. also as [1].

[5] Jarmo T. Alander. Population size. In Alander [2], pages 180–189. also as [3].

[6] Jarmo T. Alander. Editorial of the 2nd finnish workshop on genetic algorithms and their applications. In *Proceedings of the Second Finnish Workshop on Genetic Algorithms and their Applications* [8], pages iii–iv. (`ftp.uwasa.fi:cs/report94-2/main.ps.Z`).

[7] Jarmo T. Alander. *An indexed bibliography of genetic algorithms: Years 1957-1993*. Art of CAD Ltd., Vaasa (Finland), 1994. (over 3000 GA references).

[8] Jarmo T. Alander, editor. *Proceedings of the Second Finnish Workshop on Genetic Algorithms and their Applications*, volume Report Series 94-2, Vaasa (Finland), 16.-18. March 1994. University of Vaa-

sa, Department of Information Technology and Industrial Economics. (`ftp.uwasa.fi:cs/report94-1/*.ps.Z`).

[9] Jarmo T. Alander, editor. *Proceedings of the First Nordic Workshop on Genetic Algorithms and their Applications (1NWGA)*, Proceedings of the University of Vaasa, Nro. 2, Vaasa (Finland), 9.-12. January 1995. University of Vaasa. (`ftp.uwasa.fi:cs/1NWGA/*.ps.Z`).

[10] Jarmo T. Alander, editor. *Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*, Proceedings of the University of Vaasa, Nro. 11, Vaasa (Finland), 19.-23. August 1996. University of Vaasa. (`ftp.uwasa.fi:cs/2NWGA/*.ps.Z`).

[11] Jarmo T. Alander, editor. *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*, Helsinki (Finland), 18.-22. August 1997. Finnish Artificial Intelligence Society (FAIS). (`ftp.uwasa.fi:cs/3NWGA/*.ps.Z`).

[12] Jarmo T. Alander, Antti Autere, Janne Mäntykoski, and Kari I. Keskinen. Distributed genetic algorithm for fitting of model parameters of chemical reaction kinetics. In Alander [8], pages 115–126. (`ftp.uwasa.fi:cs/report94-2/Alander3.ps.Z`).

[13] Jarmo T. Alander and Juha Herajärvi. Geneettinen algoritmi vauhdittaa hissiryhmän toimintaa [Genetic algorithm in elevator control]. *Tekniikan näköalat*, (2):44–45, 23. May 1995. (in Finnish).

[14] Jarmo T. Alander, Juha Herajärvi, Ghodrat Moghadampour, Tapio Tyni, and Jari Ylinen. Genetic algorithms in elevator allocation problem. In George D. Smith?, editor, *Artificial Neural Nets and Genetic Algorithms, Proceedings of International Conference (ICANNGA97)*, page ?, Norwich (UK), April 1997. Springer-Verlag, Wien. (to appear; `ftp.uwasa.fi:cs/report97-2/ICANNGA97-171.ps.Z`).

[15] Jarmo T. Alander and Jouni Lampinen. Shape optimization of diesel engine camshaft by genetic algorithm. In Ošmera [160], pages 5–10. .

[16] Jarmo T. Alander and Jouni Lampinen. Improving design of cam shape used in valvetrain of internal-combustion engine using a genetic algorithm. In Pavel Ošmera, editor, *Proceedings of the MENDEL'96*, pages 11–17, Brno (Czech Republic), June 1996. Technical University of Brno. .

[17] Jarmo T. Alander and Jouni Lampinen. Shape optimization of diesel fuel injection cam by genetic algorithm. In Alander [10], pages 195–204. (`ftp.uwasa.fi:cs/2NWGA/Lampinen.ps.Z`).

[18] Jarmo T. Alander and Jouni Lampinen. Shape optimization of internal-combustion engine valvecam by genetic algorithm. In B. Katalinic, editor, *Proceedings of the 7th International DAAAM Symposium*, pages 5–6, Vienna (Austria), 17.-19. October 1996. Danube Adria Association for Automation & Manufacturing International, Vienna (Austria). `ftp.uwasa.fi:cs/report96-x/Wien96.ps.Z`.

[19] Jarmo T. Alander and Jouni Lampinen. Cam shape design by genetic algorithm. In C. Poloni and D. Quagliarella, editors, *Proceedings of the EuroGen 1997*, page ?, Trieste (Italy), 28. November-5. December 1997. John Wiley & Sons, New York. (in press).

[20] Jarmo T. Alander and Jouni Lampinen. A distributed implementation of genetic algorithms for cam shape optimization. In B. H. V. Topping, editor, *Advances in Computational Mechanics with Parallel and Distributed Processing (Proceedings of the First Euro-Conference on Parallel and Distributed Computing for Computational Mechanics)*, pages 209–217, Lochinver (Scotland), 26. April-1. May 1997. Civil-Comp Press, Edinburgh. .

[21] Jarmo T. Alander and Jouni Lampinen. On implementing CAD-systems based on existing simulators and optimization by genetic algorithm. In Ošmera [161], pages 7–12. .

[22] Jarmo T. Alander, Timo Mantere, and Ghodrat Moghadampour. Testing software response times using a genetic algorithm. In Alander [11], pages 293–298. (`ftp.uwasa.fi:cs/3NWGA/Alander1.ps.Z`).

[23] Jarmo T. Alander, Timo Mantere, and Pekka Turunen. Genetic algorithm based software testing. In George D. Smith?, editor, *Artificial Neural Nets and Genetic Algorithms, Proceedings of International Conference (ICANNGA97)*, page ?, Norwich (UK), April 1997. Springer-Verlag, Wien. (to appear; `ftp.uwasa.fi:cs/report97-2/ICANNGA97-172.ps.Z`).

[24] Jarmo T. Alander, Timo Mantere, Pekka Turunen, and Jari Virolainen. GA in program testing. In Alander [10], pages 205–210. (`ftp.uwasa.fi:cs/2NWGA/Alander.ps.Z`).

[25] Jarmo T. Alander, Ghodrat Moghadampour, and Jari Ylinen. Comparison of elevator allocation methods. In Alander [10], pages 211–214. (ftp.uwasa.fi:cs/2NWGA/Ghodrat1.ps.Z).

[26] Jarmo T. Alander, Mikael Nordman, and Henri Setälä. Register-level hardware design and simulation of a genetic algorithm using VHDL. In Ošmera [160], pages 10–14. .

[27] Jarmo T. Alander and Matti Ryynänen. Magnetic field refinement: Genetic algorithms vs. hill-climbing? In Alander [11], pages 333–340. (ftp.uwasa.fi:cs/3NWGA/Alander4.ps.Z).

[28] Jarmo T. Alander, Tapio Tyni, and Jari Ylinen. Optimizing elevator group control parameters using distributed genetic algorithm. In Alander [8], pages 105–113. (ftp.uwasa.fi:cs/report94-2/Alander2.ps.Z).

[29] Jarmo T. Alander, Jari Ylinen, and Tapio Tyni. Optimizing elevator group control parameters using distributed genetic algorithms. In D. W. Pearson, N. C. Steele, and R. F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 400–403, Alès (France), 19.-21. April 1995. Springer-Verlag, Wien New York. .

[30] Enrique Alba and José M. Troya. Genetic algorithms for protocol validation. In Voigt et al. [212], pages 870–879. .

[31] R. F. Albrecht, C. R. Reeves, and N. C. Steele, editors. *Artificial Neural Nets and Genetic Algorithms*, Innsbruck, Austria, 13. -16. April 1993. Springer-Verlag, Wien. .

[32] Anon. Evolver^{TM} 2.0 A genetic algorithm for spreadsheets. *Computers & Mathematics with Applications*, 26(12):94, 1993. .

[33] Antti Autere. An empirical study of population and non-population based search strategies for optimizing a combinatorial problem. In Albrecht et al. [31], pages 375–384. .

[34] Haldun Aytug, Siddartha Bhattacharrya, and Gary J. Koehler. A Markov chain analysis of genetic algorithms with power of 2 cardinality alphabets. *European Journal of Operational Research*, 96(1):195–201, 10. January 1996. .

[35] Barbro Back, Teija Laitinen, and Kaisa Sere. Neural networks and genetic algorithms for bankruptry predictions. In J. K. Lee, J. Liebowitz,

and Y. M. Chae, editors, *Critical Technology: Proceedings of the Third World Congress on Expert Systems*, volume 1, pages 123–130, Seoul (South Korea), 5.-9. February 1996. Cognizant Communication Corp., Elmsford. .

[36] Barbro Back, Kaisa Sere, and Michiel C. van Wezel. Choosing the best set of bankruptcy predictors. In Alander [9], pages 285–301. (ftp.uwasa.fi:cs/1NWGA/Back.ps.Z).

[37] Thomas Bäck. The interaction of mutation-rate, selection, and self-adaptation within a genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 85–94, Brussels, 28.-30. September 1992. Elsevier Science Publishers, Amsterdam. .

[38] Thomas Bäck. Optimal mutation rates in genetic search. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–8, Urbana-Champaign,IL, 17.-21. July 1993. Morgan Kaufmann, San Mateo, CA. .

[39] David Beasley, David R. Bull, and Ralph R. Martin. An overview of genetic algorithms. 1. fundamentals. *University Computing*, 15(2):58–69, 1993. (ralph.cm.cf.ac.uk:/pub/GAs/ga_overview1.ps or alife.santafe.edu:/pub/YSER-AREA/EC/GA/papers/over93.ps.gz).

[40] David Beasley, David R. Bull, and Ralph R. Martin. An overview of genetic algorithms. 2. research topics. *University Computing*, 15(4):170–181, 1993. (ralph.cm.cf.ac.uk:/pub/GAs/ga_overview2.ps or alife.santafe.edu:/pub/USER-AREA/EC/GA/papers/over93-2.ps.gz).

[41] Peter J. Bentley. *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*. PhD thesis, University of Huddersfield, Huddersfield, UK, 1996. (URL:http://users.aol.com/pbent1ey/).

[42] Peter J. Bentley and Jonathan P. Wakefield. Conceptual evolutionary design by genetic algorithms. *Engineering Design and Automation Journal*, 2(3):?, ? 1996. .

[43] Hugues Bersini. Immune network and adaptive control. In R. Trappl, editor, *Cybernetics and Systems Research 92, Proceedings of the 11th European Meeting on Cybernetics and System Research*, volume 1, pages 111–118, Vienna (Austria), 21. -24. April 1992. World Scientific Publishing Company, Pte., Ltd., Singapore. .

[44] George Bilchev and Ian C. Parmee. Constraint handling for the fault coverage code generation problem: An inductive evolutionary approach. In Voigt et al. [212], pages 880–889. .

[45] Jacek Błżewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):1–33, 23. August 1996. .

[46] Edward B. Boden and Gilford F. Martino. Testing software using order-based genetic algorithms. In John R. Koza, David Goldberg, David Fogel, and Rick L. Riolo, editors, *Proceedings of the GP-96 Conference*, page ?, Stanford, CA, 28.-31. July 1996. MIT Press, Cambridge, MA. .

[47] David G. Bounds. New optimization methods from physics and biology. *Nature*, 329(?):215–219, 17. September 1987. .

[48] Bill P. Buckles and Frederick E. Petry, editors. *Genetic Algorithms.* Electronica Books Ltd., Middlesex (UK), 1993. .

[49] Innesa L. Bukatova, Yu. I. Mikhasev, and A. M. Sharov. *Evoinformatics: Theory and Practice of Evolutionary Simulation.* Nauka Publishers, Moscow, 1991. (in Russian).

[50] Sangit Chatterjee, Cecilia Carrera, and Lucy A. Lynch. Genetic algorithms and traveling salesman problems. *European Journal of Operational Research*, 93(3):490–510, 20. September 1996. .

[51] Fulvio Corno, Paolo Prinetto, Maurizio Rebaudengo, and Matteo Sonza Reorda. Comparing topological, symbolic and GA-based ATPGs: an experimental approach. In *Proceedings of the 1996 International Test Conference*, pages 39–47, Washington, DC, 20.-25. October 1996. IEEE Computer Society Press, Los Alamitos, CA. .

[52] Fulvio Corno, Paolo Prinetto, Maurizio Rebaudengo, and Matteo Sonza Reorda. Exploiting competing subpopulations for automatic generation of test sequences for digital circuits. In Voigt et al. [212], pages 792–800. .

[53] Fulvio Corno, Paolo Prinetto, Maurizio Rebaudengo, and Matteo Sonza Reorda. A parallel genetic algorithm for automatic generation of test sequences for digital circuits. In H. Liddell, A. Colbrook, B. Hertzberger, and S. Sloot, editors, *Proceedings of the International Conference*

and Exhibition HPCN EUROPE 1996, pages 454–459, Brussels, Belgium, 15.-19. April 1996. Springer-Verlag, Berlin (Germany). .

[54] Bassil I. Dahiyat and Stephen L. Mayo. De novo protein design: Fully automated sequence selection. *Science*, 278(5335):82–87, October 1997.

[55] Charles Darwin. *On The Origin of Species by Means of Natural Selection.* Avenel Books, New York, (reprint of the 1859 edition) edition, 1979.

[56] Lawrence Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991. .

[57] Richard Dawkins. *The Selfish Gene.* Oxford University Press, Oxford, 1976.

[58] Richard Dawkins. *The Extended Phenotype: The Gene as a Unit of Selection.* Oxford University Press, Oxford, 1982.

[59] Richard Dawkins. *The Blind Watchmaker.* Oxford University Press, Oxford (England), 1986. .

[60] Richard Dawkins. *Sokea kelloseppä.* WSOY, Juva, 1989.

[61] Thomas Eisenhammer, M. Lazarov, M. Leutbecher, U. Schöffel, and R. Sizmann. Optimization of interference filters with genetic algorithms applied to silver-based heat mirrors. *Applied Optics*, 32(31):6310–6315, 1. November 1993. .

[62] Thomas Eisenhammer, M. Lazarov, and R. Sizmann. Optimization of silver based heat mirrors using a genetic algorithm. In A. Hugotlegoff, C. G. Granqvist, and C. M. Lampert, editors, *Optical Materials Technology for Energy Efficiency and Solar Energy Conversion XI*, volume SPIE-1727, pages 194–202, Tolouse Labege (France), 18. May 1992. The International Society for Optical Engineering. .

[63] Timo Eloranta. Geneettisten algoritmien soveltaminen suuntaamattomien verkkojen piirtoon [Applying genetic algorithms to drawing undirected graphs]. Report Series B B-1996-1, University of Tampere, Department of Computer Science, 1996. (in Finnish, also as [64]).

[64] Timo Eloranta. Geneettisten algoritmien soveltaminen suuntaamattomien verkkojen piirtoon [Applying genetic algorithms to drawing undirected graphs]. Master's thesis, University of Tampere, Department of Computer Science, 1996. (in Finnish, also as [63]).

[65] Markus Ettl and Markus Schwehm. A design methodology for Kanban-controlled production lines using queuing networks and genetic algorithms. Interne Bericht IMMD VII - 15/94, Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung, 1994. (also as [66]).

[66] Markus Ettl and Markus Schwehm. A design methodology for Kanban-controlled production lines using queuing networks and genetic algorithms. In ?, editor, *Proceedings of the Tenth International Conference on Systems Engineering (ICSE'94)*, pages 307–314, ?, ? 1994. Coventry University Press, Coventry (UK). .

[67] Markus Ettl and Markus Schwehm. Determining the optimal network partition and Kanban allocation in JIT production lines. In Jörg Biethahn and Volker Nissen, editors, *Evolutionary Algorithms in Management Application*, pages 139–152. Springer-Verlag, Berlin, 1995. .

[68] R. Ferguson and B. Korel. Software test data generation using the chaining approach. In ?, editor, *Proceedings of the International Test Conference*, pages 703–709, Washington, DC, USA, 21.-25. October 1995. Int. Test Conference, Altoona, PA, USA. .

[69] Patrik B. J. Floréen and Joost N. Kok. Tracing the moments of distributions in genetic algorithms. In Alander [8], pages 51–60. (`ftp.uwasa.fi:cs/report94-2/Floreen.ps.Z`).

[70] David B. Fogel and Lawrence J. Fogel. Method and apparatus for training a neural network using evolutionary programming, 1992. (U. S. patent no. 5,214,746. Issued May 25 1993).

[71] Walter Fontana. Algorithmic chemistry. In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Artificial Life II, Proceedings of the Workshop on Artificial Life Held February, 1990 in Santa Fe, New Mexico*, Proceedings Volume X, Santa Fe Institute Studies in the Sciences of Complexity, pages 159–209. Addison-Wesley, Reading, MA, 1992. .

[72] Stephanie Forrest. Genetic algorithms - principles of natural selection applied to computation. *Science*, 261(5123):872–878, 13 August 1993. .

[73] Stephanie Forrest. *Chapter 24. Genetic algorithms*, pages 557–571. CRC Press, Boca Raton, FL, 1996. (a version of [72]).

[74] Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, October 1997. .

[75] Stephanie Forrest and Melanie Mitchell. Relative building-block fitness and the building-block hypothesis. In Whitley [217], pages 109–126. (`ftp.santafe.edu:/pub/Users/mm/Forrest-Mitchell-FOGA.ps.Z`).

[76] Stephanie Forrest and Alan S. Perelson. Genetic algorithms and the immune system. In Schwefel and Männer [193], pages 320–325. .

[77] Stephanie Forrest and Alan S. Perelson. Computation and the immune system. *SIGBIO Newsletter*, 12(2):52–57, 1992. .

[78] A. Fujino, T. Tobita, K. Segawa, K. Yoneda, and A. Togawa. An elevator group control system with floor attribute control method and system optimization using genetic algorithms. In *Proceedings of the 21st International Conference on Industrial Electronics, Control, and Instrumentation (IECON'95)*, volume 2, pages 1502–1507, Orlando, FL, 6.-10. November 1995. IEEE, New York. .

[79] Murray Gell-Mann. *The Quark and the Jaguar*. W. H. Freeman and Company, New York, 1994. .

[80] Murray Gell-Mann. *Kvarkki ja jaguaari*, pages 370–373. Werner Söderström Osakeyhtiö, Juva (Finland), 1996. (translation of [79]; in Finnish).

[81] Andrew McGilvary Gillies. Machine learning procedures for generating image domain feature detectors, 1985. (U. S. patent no. 4,821,333. Issued Apr. 11 1989).

[82] P. Girard, C. Landrault, S. Pravossoudovitch, and B. Rodriguez. A diagnostic ATPG for delay faults based on genetic algorithms. In *Proceedings of the 1996 International Test Conference*, pages 286–293, Washington, DC, 20.-25. October 1996. IEEE Computer Society Press, Los Alamitos, CA. .

[83] F. Glover. Tabu search - part I. *ORSA J. on Computing*, 1(?):190–206, 1989.

[84] F. Glover. Tabu search - part II. *ORSA J. on Computing*, 2(?):4–32, 1990.

[85] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989. .

[86] David E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119, March 1994. .

[87] David E. Goldberg, Bradley Korb, and Kalyanmoy Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989. (Also TCGA Report 89003).

[88] David E. Goldberg and Chie Hsiung Kuo. Genetic algorithms in pipeline optimization. *Journal of Computing in Civil Engineering*, 1(2):128–141, April 1987. .

[89] David E. Goldberg, Kelsey Milman, and Christina Tidd. Genetic algorithms: A bibliography. IlliGAL Report 92008, University of Illinois at Urbana-Champaign, 1992. .

[90] Soloman W. Golomb. *Polyominoes, Puzzles, Patterns, Problems, and Packings.* Princeton University Press, Princeton, New Jersey, 2 edition, 1994.

[91] Stephen Jay Gould. *Wonderful Life. The Burges Shale and the Nature of History.* W. W. Norton & Company, Inc., 1989.

[92] Stephen Jay Gould. *Livet är underbart.* Ordfronts förlag, Stockholm, 1990.

[93] Stephen Jay Gould. *Ihmeellinen elämä.* Art House OY, Helsinki, 1991.

[94] K. Gunavathi and A. Shanmugam. Genetic algorithm based performance analysis of two new distributed MAC protocols for LAN. In Ošmera [161], pages 59–64. .

[95] B. H. Gwee and M. H. Lim. Polyominoes tiling by a genetic algorithm. *Comput. Optim. Appl. (Netherlands)*, 6(3):273–291, 1996. .

[96] Juha Haataja. Geneettisten algoritmien simulointi Matlab 4.0:lla [On simulation of genetic algorithms by Matlab 4.0]. *SuperMenu*, ?(2):21–25, 1993. (in Finnish).

[97] Juha Haataja. *Optimointitehtävien ratkaiseminen [On solving optimization problems]*, chapter 9, Erikoismenetelmiä [Special methods]. CSC-Tieteellinen laskenta Oy, Espoo, 1993. (in Finnish).

[98] Juha Hakkarainen, Anne Jumppanen, Jari Kyngäs, and J. Kyyrö. An evolutionary approach to neural network design applied to sunspot prediction. Report A-1996-3, University of Joensuu, Department of Computer Science, 1996. .

[99] Timo Hämäläinen. *Implementation and Algorithms of a Tree Shape Parallel Computer*. PhD thesis, Tampere University of Technology, 1996. .

[100] Timo Hämäläinen, Harri Klapuri, Jukka Saarinen, Pekka Ojala, and Kimmo Kaski. Accelerating genetic algorithm computation in tree shaped parallel computer. *Journal of Systems Architecture*, 42(1):19–36, August 1996. .

[101] Timo Hämäläinen, Jukka Saarinen, Pekka Ojala, and Kimmo Kaski. Implementing genetic algorithms in a tree shape computer architecture. In Alander [9], pages 259–284. (`ftp.uwasa.fi:cs/1NWGA/Hamalainen.ps.Z`).

[102] Simon G. Handley. The genetic planner: The automatic generation of plans for a mobile robot via genetic programming. In *Proceedings IEEE International Symposium on Intelligent Control*, pages 190–195, Chicago, IL, 25.-27. August 1993. IEEE, New York. .

[103] Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolution strategies: Similarities and differences. In Schwefel and Männer [193], pages 455–471. .

[104] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975. .

[105] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, 1992. .

[106] John H. Holland and A. W. Burks. Adaptive computing system cabable of learning and discovery, 1985. (U. S. Patent no. 4,697,242. Issued Sep. 29 1987).

[107] John H. Holland and A. W. Burks. Method of controlling a classifier system, 1989. (U.S. Patent 4,881,178. Issued Nov. 14 1989).

[108] B. Hölldobler and E. O. Wilson. *The Ants*. Springer-Verlag, Berlin, 1990.

[109] J. N. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, March-April 1994.

[110] Chris R. Houck, Jeff A. Joines, and Mike G. Kay. Comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems. *Computers & Operations Research*, 23(6):587–596, 1996. .

[111] Heikki Huttunen, Pauli Kuosmanen, and Jaakko Astola. Optimal RASF filtering. In Edward R. Dougherty, Jaakko T. Astola, and Harold G. Longbotham, editors, *Nonlinear Image Processing VII*, volume SPIE-2662, pages 154–165, San Jose, CA, 29. -30. January 1996. The International Society for Optical Engineering, Bellingham, WA. .

[112] *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield (UK), 12.-14. September 1995. IEEE. .

[113] Lester Ingber and Bruce Rosen. Genetic algorithms and very fast simulated annealing: A comparison. *Mathematical and Computer Modelling*, 16(11):87–100, November 1992. .

[114] B. F. Jones, H.-H. Sthamer, and D. E. Eyres. Generating test data for ADA procedures using genetic algorithms. In IEE/IEEE Sheffield '95 [112], pages 65–70. .

[115] B. F. Jones, H.-H. Sthamer, and D. E. Eyres. Automatic structural testing using genetic algorithms. *Software Engineering Journal*, 11(5):299–306, September 1996. .

[116] Hillol Kargupta. Information transmission in genetic algorithm and Shannon's second theorem. IlliGAL Report 93003, University of Illinois at Urbana-Champaign, 1993. .

[117] Nachimuthu Karunanithi and Tamra Carpenter. A ring loading application of genetic algorithms. Technical Report TM-ARH-023337, Bellcore, Bell Communications Research, Morristown, NJ, 1993. (also as [118]).

[118] Nachimuthu Karunanithi and Tamra Carpenter. A ring loading application of genetic algorithms. In ?, editor, *Proceedings of the ACM Symposium on Applied Computing*, pages 227–231, ?, ? 1994. ACM. .

[119] Nachimuthu Karunanithi and Tamra Carpenter. SONET ring sizing with genetic algorithms. *Computers & Operations Research*, 24(6):581–591, June 1997. (also as [118]).

[120] Stuart A. Kauffman. *The Origins of Order, Self-Organization and Selection in Evolution.* Oxford University Press, New York, 1993. .

[121] R.-M. King and P. Banerjee. Optimization by simulated evolution with applications to standard cell placement. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 20–25, Orlando, FL, 24.-28. June 1990. IEEE, New York. .

[122] R.-M. Kling and P. Banerjee. ESP: A new standard cell placement package using simulated evolution. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pages 60–66, ?, ? 1987. IEEE, New York. .

[123] Pertti Koivisto, Heikki Huttunen, and Pauli Kuosmanen. Training-based optimization of soft morphological filters. *Journal of Electronic Imaging*, 5(3):300–322, July 1996. .

[124] John R. Koza. Non-linear genetic algorithms for solving problems, 1990. (U. S. Patent no. 4,935,877. Filed May 20 1988 and issued June 19 1990).

[125] John R. Koza. Non-linear genetic algorithms for solving problems by finding a fit composition of functions, 1990. (U.S. Patent application filed Mar. 28 1990).

[126] John R. Koza. A non-linear genetic algorithms for solving problems, 1991. (Australian patent 611,350. Issued Sep. 21 1991).

[127] John R. Koza. A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In *Proceedings of the IJCNN International Joint Conference on Neural Networks*, volume IV, pages 310–318, Baltimore, MD, 7.-11. June 1992. IEEE, New York. .

[128] John R. Koza. *Genetic Programming: On Programming Computers by Means of Natural Selection and Genetics.* The MIT Press, Cambridge, MA, 1992. .

[129] John R. Koza. A non-linear genetic algorithms for solving problems, 1992. (Canadian patent 1,311,561. Issued Dec. 15 1992).

[130] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane. Four problems for which a computer program evolved by genetic programming is competitive with human performance. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, page ?, Nagoya, Japan, 20.-22. May 1996. IEEE, Piscataway, NJ. .

[131] John R. Koza and James P. Rice. Non-linear genetic process for use with co-evolving populations, 1990. (U.S. Patent application filed Sep. 18 1990).

[132] John R. Koza and James P. Rice. Non-linear genetic process for use with plural co-evolving populations, 1990. (U. S. Patent 5,148,513. Filed Sep. 18 1990. Issued Sep. 15 1992).

[133] John R. Koza and James P. Rice. Automatic programming of robots using genetic programming. In *AAAI-92 Proceedings Tenth National Conference on Artificial Intelligence*, pages 194–201, Jan Jose, California, 12. - 16. July 1992. AAAI Press/ The MIT Press. .

[134] John R. Koza and James P. Rice. A non-linear genetic process for data encoding and for solving problems using automatically defined functions, 1992. (U.S. patent application filed May 11 1992).

[135] John R. Koza and James P. Rice. Non-linear genetic process for data encoding and for solving problems using automatically defined functions, 1992. (U. S. patent Application. Filed May 11 1992).

[136] John R. Koza and James P. Rice. A non-linear genetic process for problem solving using spontaneously emergent self-replicating and self-improving entities, 1992. (U.S. patent application filed June 16 1992).

[137] Vladik Kreinovich, Chris Quintana, and Olac Fuentes. Genetic algorithms: what fitness scaling is optimal? *Cybernetics and Systems*, 24(1):9–26, January-February 1993. .

[138] Jouni Lampinen. Choosing a shape representation method for optimization of 2D shapes by genetic algorithms. In Alander [11], pages 305–320. (`ftp.uwasa.fi:cs/3NWGA/Lampinen.ps.Z`).

[139] Jouni Lampinen. Diesel-moottorin nokka-akselin muodon optimointi geneettisellä algoritmilla [Diesel engine cam shape design by genetic

algorithm]. Master's thesis, University of Vaasa, Department of Information Technology and Production Economics, 1997. (in Finnish; abstract in English and Swedish).

[140] G. Laporte. The traveling salesman problem: an overview of exact and approximative algorithms. *European Journal of Operations Research*, 59(2):231–247, 1992.

[141] William Latham, Karl Sims, Stephen Todd, and Michael Tolson. The applications of evolutionary and biological processes to computer art and animation. In *20th Annual SIGGRAPH Computer Graphics Proceedings*, pages 389–390, Anaheim, CA, 1.-6. August 1993. ACM SIGGRAPH. .

[142] Prabhat Hajela Jongsoo Lee. Genetic algorithms in multidisciplinary rotor blade design. In ?, editor, *Technical Papers of the 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, volume Pt. 4, pages 2187–2197, New Orleans, LA, 10.-13. April 1995. American Institute of Aeronautics and Astronautics, Washington, DC. .

[143] Yee Leung, Yong Gao, and Zong-Ben Xu. Degree of population diversity – a perspective on premature convergence in genetic algorithms and its Markov chain analysis. *IEEE Transactions on Neural Networks*, 8(5):1165–1176, September 1997. .

[144] Yee Leung, Yong Gao, and Zongben Xu. A Markov chain analysis of premature convergence in genetic algorithms. In *Proceedings of the International Conference on Neural Information Processing*, volume 2, pages 1330–1334, Hong Kong, 24.-27. September 1996. Springer-Verlag, Singapore. .

[145] Paul Lukowicz, Ernst A. Heinz, Lutz Prechelt, and Walter F. Tichy. Experimental evaluation in computer science: A qualitative study. Technical Report 17/94, University of Karlsruhe, Department of Informatics, 1994. (also as [146]; `ftp.ira.uka.de:/pub/uni-karlsruhe/papers/techreports/1994/1994-17.ps.Z`).

[146] Paul Lukowicz, Ernst A. Heinz, Lutz Prechelt, and Walter F. Tichy. Experimental evaluation in computer science: A qualitative study. *Journal of Systems and Software*, ?(?):?, January 1995. (to appear in; also as [145]; `ftp.ira.uka.de:/pub/uni-karlsruhe/papers/techreports/1994/1994-17.ps.Z`).

[147] Raino A. E. Mäkinen and Jari Toivanen. Parallel solution of optimal shape design problem governed by Helmholtz/potential flow equations. Report 5, University of Jyväskylä, Laboratory of Scientific Computing, 1995. .

[148] Bernard Manderick and Piet Spiessens. Computational intelligence imitating life. chapter How to select genetic operators for combinatorial optimization problems by analyzing their fitness landscape, pages 170–181. IEEE Press, New York, 1994. .

[149] Vittorio Maniezzo, Alberto Colorni, and Marco Dorigo. ALGODESK: an experimental comparison of eight evolutionary heuristics applied to the QAP problem. *European Journal of Operational Research*, 81(1):188–204, January 1995. .

[150] Timo Mantere. Sähkön hankinnan optimointi [Economic dispatch problem, a comparison of some solution methods]. Master's thesis, University of Vaasa, Department of Information Technology and Production Economics, 1996. (in Finnish).

[151] Janne Mäntykoski. Parallel implementation of genetic algorithms. Master's thesis, Helsinki University of Technology, Department of Electrical Engineering, Laboratory of Signal Processing and Computer Technology, 1994. .

[152] Janne Mäntykoski and Jarkko Vuori. Non-linear filter design using genetic algorithm. In Alander [8], pages 89–95. (`ftp.uwasa.fi:cs/report94-2/Mantykoski.ps.Z`).

[153] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence. Springer-Verlag, New York, 1992. .

[154] John A. Miller, Walter D. Potter, Ravi V. Gandham, and Chito N. Lapena. An evaluation of local improvement operators for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(5):1340–1351, September/October 1993. .

[155] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In Francisco J. Varela and Paul Bourgine, editors, *Toward a Practice of Autonomous System: Proceedings of*

the First European Conference on Artificial Life, pages 245–254, Paris, 11.-13. December 1991. MIT Press, Cambridge, MA. (`ftp.santafe.edu:/pub/Users/mm/sfi-91-10-046.ps.Z`).

[156] Ghodrat Moghadampour. Hissien ohjausjärjestelmien tilastollinen vertailu [A statistical comparison of two elevator call allocation method]. Master's thesis, University of Vaasa, Department of Information Technology and Production Economics, 1996. (In Finnish; abstract in English).

[157] Alexandru Murgu and Risto Lahdelma. Tracking statistics in communication traffic control using a genetic algorithm. In Alander [9], pages 241–258. (`ftp.uwasa.fi:cs/1NWGA/Murgu.ps.Z`).

[158] M. J. O'Dare and T. Arslan. Hierarchical test pattern generation using a genetic algorithm with a dynamic global reference table. In IEE/IEEE Sheffield '95 [112], pages 517–523. .

[159] Ibrahim H. Osman and James P. Kelly, editors. *Meta-heuristics: Theory and Applications*, Breckenridge, CO, 22.-26. July 1996. Kluwer Academic Publishers, Norwell, MA. (Proceedings of the 1995 Meta-Heuristics International Conference (MIC).

[160] Pavel Ošmera, editor. *Proceedings of the MENDEL'95*, Brno (Czech Republic), 26.-28. September 1995. Technical University of Brno. .

[161] Pavel Ošmera, editor. *Proceedings of the 3rd International Mendel Conference on Genetic Algorithms, Optimization problems, Fuzzy Logic, Neural networks, Rough Sets (MENDEL'97)*, Brno (Czech Republic), 25-27June 1997. Technical University of Brno. .

[162] Moloud Oussaidène, Bastien Chopard, and Marco Tomassini. Programmation évolutionniste parallèle. In Dekeyser, Lebert, and Manneback, editors, *Proceedings of the RenPar'7, Actes des 7es Recontres Francopones du Parallélisme*, page ?, ?, 30. May- 2. June 1995. PIP-FPMs Mons, Belgium. .

[163] Sakari Palko. *Structural optimization of induction motor using genetic algorithm and finite element method.* PhD thesis, Helsinki University of Technology, 1996. .

[164] Christos H. Papadimitriou. The complexity of the Lin-Kernighan heuristics for the traveling salesman problem. *SIAM Journal of Computing*, 21(3):450–465, 1992.

[165] A. S. Perelson and S. A. Kauffman. *Molecular Evolution on Rugged Landscapes: Proteins, RNA and the Immune System.* Addison-Wesley, Reading, MA, 1991.

[166] Mauro Piccioni. A combined multistart-annealing algorithm for continuous global optimization. *Computers & Mathematics with Applications*, 21(6/7):173–179, ? 1991.

[167] Dan Pietrasik. On the use of genetic programming in elevator control design. In John R. Koza, editor, *Genetic Algorithms at Stanford 1995*, page ?, Stanford, CA, 1995. Stanford Bookstore. .

[168] F. M. Rabinowitz. Algorithm 744: a stochastic algorithm for global optimization with constraints. *ACM Transactions on Mathematical Software*, 21(2):194–213, June 1995.

[169] Nicholas J. Radcliffe. The algebra of genetic algorithms. Technical Report TR-92-11, Edinburgh Parallel Computing Centre, 1992. (to appear in *Annals of Mathematics and Artificial Intelligence*; `ftp.epcc.ed.ac.uk:/pub/tr/92/tr9211.ps.Z`).

[170] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technisher Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart, 1973. (reprint in [171]).

[171] Ingo Rechenberg. *Evolutionsstrategie '94*. Frommann-Holzboog-Verlag, Stuttgart (Germany), 1994. (in German; includes also [170]).

[172] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Publications, Oxford, 1993. .

[173] A. Reinefeld and V. Schnecke. Portability versus efficiency? Parallel applications on pvm and parix. In ?, editor, *Proceedings of the Workshop on Parallel Programming and Computation (ZEUS'95) and the 4th Nordic Transputer Conference (NTUG'95)*, pages 35–49, Linköping, Sweden, ? 1995. IOS Press 1995, Amsterdam, Netherlands. .

[174] Jean-Michel Renders and Stéphane P. Flasse. Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 26(2):243–258, April 1996. .

[175] David Reynolds and Jagannathan Gomatam. Stochastic modelling of genetic algorithms. *Artificial Intelligence*, 82(1-2):303–330, April 1996. .

[176] José L. Ribeiro Filho, Philip C. Treleaven, and Cesare Alippi. Genetic-algorithm programming environments. *Computer*, 27(6):28–45, June 1994. .

[177] Gordon M. Robinson. *Genetic Algorithm Optimisation of Load Cell Geometry by Finite Element Analysis.* PhD thesis, City University, London, School for Engineering, 1995. .

[178] Gordon M. Robinson. Load cell shape optimisation using genetic algorithms and finite element analysis. In A. Bizmajer, editor, *Proceedings of the 14th International IMEKO TC3 Conference: State of the Art in Force and Mass Measurement*, pages 167–172, Warzawa (Poland), September 1995. IMEKO, Budapest (Hungary). .

[179] Simon Ronald. Practical handbook of genetic algorithms. volume 1, Applications, chapter 11. Routing and scheduling problems, pages 367–430. CRC Press, Boca Raton, FL, 1995. .

[180] Matti Ryynänen. Optimization of permanent magnets ordering of hybrid X-ray wiggler for MAX-II with a chained pairing and genetic algorithm. In Alander [8], pages 61–70. (`ftp.uwasa.fi:cs/report94-2/Ryynanen.ps.Z`).

[181] Daniel G. Saab, Youssef G. Saab, and Jacob A. Abraham. Automatic test vector cultivation for sequential VLSI circuits using genetic algorithms. *IEEE Transactions on Computer Aided Design Integr. Circuits Syst.*, 15(10):1278–1285, October 1996. .

[182] Sadiq M. Sait and Habib Youssef. chapter 4. Placement, pages 186–191. McGraw-Hill, New York, 1995. .

[183] Sam Sandqvist. On finding optimal potential customers from a large marketing database – A genetic algorithm approach. In Eero Hyvönen, Jouko Seppänen, and Markku Syrjänen, editors, *STeP-92 Tekoälyn uudet suunnat*, volume 3, pages 35–38, Espoo, 9.-11. June 1992. Finnish Artificial Intelligence Society (FAIS). .

[184] Sam Sandqvist. GA and database queries. In Jarmo T. Alander, editor, *Proceedings of the First Finnish Workshop on Genetic Algorithms and their Applications*, volume TKO-A30 of *Research Reports*. Espoo (Finland), 4.-5. November 1992 1993. .

[185] Sam Sandqvist. On finding optimal potential customers from a large marketing database – A genetic algorithm approach. Licenciate thesis,

Helsinki University of Technology, Department of Computer Science, 1993. .

[186] Sam Sandqvist. On finding optimal potential customers from a large marketing database – A genetic algorithm approach. In Albrecht et al. [31], pages 528–535. .

[187] Sam Sandqvist. A temporal logic model of the genetic algorithm. In Alander [8], pages 139–150. (`ftp.uwasa.fi:cs/report94-2/Sandqvist.ps.Z`).

[188] Sam Sandqvist. Modelling the time behaviour of genetic algorithms. In Alander [9], pages 205–214. (`ftp.uwasa.fi:cs/1NWGA/Sandqvist.ps.Z`).

[189] Sam Sandqvist. Some aspects of time in genetic algorithms. In Alander [10], pages 267–278. (`ftp.uwasa.fi:cs/2NWGA/Sandqvist.ps.Z`).

[190] Andreas Schober, Marcel Thuerk, and Manfred Eigen. Optimization by hierarchical mutant production. *Biological Cybernetics*, 69(5-6):493–501, ? 1993. .

[191] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser Verlag, Basel and Stuttgart, 1977. (in German; in English as [192]).

[192] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley, Chichester, 1981. also as [191].

[193] Hans-Paul Schwefel and R. Männer, editors. *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, Dortmund (Germany), 1.-3. October 1991. Springer-Verlag, Berlin. (Proceedings of the 1st Workshop on Parallel Problem Solving from Nature (PPSN1)).

[194] Markus Schwehm. *Globale optimierung mit massiv parallelen genetischen Algorithmen*. PhD thesis, Universität Erlangen-Nürnberg, Institut für Matematische Maschinen und Datenverarbeitung (Informatik), 1997. (Arbeitsberichte des IMMD, Band 30, Nr. 1).

[195] Markus Schwehm, Thilo Opaterny, and Karlheinz Kirsch. Plazierung von Makrozellen durch genetische Algorithmen auf verteilten und massiv parallelen rechnern. *PARS-Mitteilungen*, ?(13):69–74, ? 1994. (in German).

[196] Khushro Shahookar and Pinaki Mazumder. A genetic approach to standard cell placement using meta-genetic parameter optimization. *IEEE Transactions on Computer-Aided Design*, 9(5):500–511, May 1990. .

[197] Khushro Shahookar and Pinaki Mazumder. VLSI cell placement techniques. *ACM Computer Surveys*, 23(2):143–220, June 1991. .

[198] Takanori Shibata, Tamotsu Abe, Kazuo Tanie, and Matsuo Nose. Skill based motion planning in hierarchical intelligent control of a redundant manipulator. *Robotics and Autonomous Systems*, 18(?):65–73, ? 1996. .

[199] Takanori Shibata, Tamotsu Abe, Kazuo Tanie, and Matsuo Nose. Motion planning by genetic algorithm for a redundant manipulator using a model of criteria of skilled operators. *Information Sciences*, 102(1-4):171–186, 1997. .

[200] Marja-Liisa Siikonen. *Planning and control models for elevators in high-rise buildings*. PhD thesis, Helsinki University of Technology, Department of Computer Science, 1997.

[201] Karl Sims. Artificial evolution for computer graphics. *Computer Graphics*, 25(4):319–328, July 1991. .

[202] Robert Elliot Smith, Stephanie Forrest, and Alan S. Perelson. Population diversity in an immune system model: Implications for genetic search. In Whitley [217], pages 153–166. .

[203] Chee Kiong Soh and Jiaping Yang. Fuzzy controlled genetic algorithm search for shape optimization. *Journal of Computing in Civil Engineering*, 10(2):143–150, April 1996. .

[204] Branco Soucek and the IRIS Group, editors. *Dynamic, Genetic, and Chaotic Programming*. Sixth Generation Computer Technologies. John Wiley & Sons, New York, 1992. .

[205] M. Srinivas and Lalit M. Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–27, June 1994. .

[206] Joachim Stender, editor. *Parallel Genetic Algorithms*. IOS Press, Amsterdam, 1993. .

[207] Toshimitsu Tobita, Atsuya Fujino, Kazuhiro Segawa, Kenji Yoneda, and Yoshiaki Ichikawa. Parameter tuning method using genetic algorithms for an elevator group control system. In *Proceedings of the 1996*

IEEE 22nd International Conference on Industrial Electronics, Control, and Instrumentation (IECON), volume 2, pages 823–828, Taipei (Taiwan), 5.-10. August 1996. IEEE Computer Society Press, Los Alamitos, CA. ga96aTobita.

[208] Jari Toivanen, Raino A. E. Mäkinen, and Risto Lahdelma. The reconstruction of an airfoil in 2D potential flow using a genetic algorithm on a parallel computer. In Alander [9], pages 229–240. (`ftp.uwasa.fi:cs/1NWGA/Toivanen.ps.Z`).

[209] Pasi Törmänen. Geneettisten algoritmien käyttö tutkittaessa sumean logiikan hyötyä PID-säädössä - esimerkkinä taajuusmuuttajan sisäiset säätäjät [Evaluating the benefit of fuzzy logic for PID-control by means of genetic algorithms - case: frequency controller]. Master's thesis, University of Vaasa, Department of Information Technology and Production Economics, 1997. (`ftp.uwasa.fi:cs/report97-3/part*.ps.Z`).

[210] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1988.

[211] Sami Viitanen. *Some new global optimization algorithms*. PhD thesis, Åbo Akademi University, Department of Computer Science, 1997.

[212] Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors. *Parallel Problem Solving from Nature – PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, Berlin (Germany), 22.-26. September 1996. Springer-Verlag, Berlin. .

[213] Jarkko Vuori. Identification of nonlinear systems with evolutionary computation. In Alander [8], pages 97–103. (`ftp.uwasa.fi:cs/report94-2/Vuori.ps.Z`).

[214] Jarkko Vuori. Genetic algorithm for codebook design in spread spectrum correlator receiver. In Ošmera [160], pages 165–168. .

[215] Jonathan Weiner. *The Beak of the Finch*. Knopf, ?, 1994.

[216] Jonathan Weiner. Evolution made visible. *Science*, 267(5194):30–33, 6. January 1995.

[217] Darrell Whitley, editor. *Foundations of Genetic Algorithms — 2 (FOGA-92)*, Vail, CO, 24.-29. July 1992 1993. Morgan Kaufmann: San Mateo, CA. .

[218] Darrell Whitley and J. Kauth. GENITOR: A different genetic algorithm. Technical Report No. CS-88-101, Colorado State University, Department of Computer Science, Fort Collins, 1988. also as [219].

[219] Darrell Whitley and J. Kauth. GENITOR: A different genetic algorithm. In *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, CO, ? 1988. ? also as [218].

[220] Darrell L. Whitley, Soraya Rana, John Dzubera, and Keith E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2):245–276, August 1996. .

[221] Nobukazu Yoshikawa, Masahide Itoh, and Toyohiko Yatagi. Quantized phase optimization of two-dimensional Fourier kinoforms by a genetic algorithm. *Optics Letters*, 20(7):752–754, 1. April 1995. .

Liite A

Kyselylomake

OPTIMOINTIKYSELY / Vaasan yliopisto
Tietotekniikan ja tuotantotalouden laitos
PL 700, 65101 Vaasa

Optimointi on yrityksemme tuotekehityksessä tärkeää:
☐ *kyllä* ☐ *ei* ☐ *en osaa sanoa*

Yrityksessämme käytetään automaattisia optimointimenetelmiä:
☐ *kyllä* ☐ *ei* ☐ *en osaa sanoa*

Mitkä ovat tärkeimmät optimointiongelmat yrityksessänne? (*tarvittaessa jatka kääntöpuolelle*):

Koulutukseni optimointitekniikoista on antanut riittävät valmiudet optimointiongelmien ratkaisemiseen:
☐ *kyllä* ☐ *ei* ☐ *en osaa sanoa*

Koulutustaustani on (ympyröi): *ins. DI TKT LuK FK FT ekon. muu* (mikä?).

Haluaisimme lisätietoja yleisimmistä optimointimenetelmistä:
☐ *kyllä* ☐ *ei* ☐ *en osaa sanoa*

Olemme kiinnostuneita osallistumaan optimointitekniikoiden täydennyskoulutuskursseille, jos sellaisia järjestetään:
☐ *kyllä* ☐ *ei* ☐ *en osaa sanoa*

Optimointiasioissa yhteyshenkilömme on
Puhelin
Osoite