SATAKUNNAN AMMATTIKORKEAKOULU

Sami Aziz

2007

GENEETTINEN ALGORITMI

Tekniikka Rauma
Tietotekniikan koulutusohjelma
Tietologistiikan suuntautumisvaihtoehto

GENEETTINEN ALGORITMI

Aziz, Sami
Satakunnan ammattikorkeakoulu
Tekniikka Rauma
Tietotekniikan koulutusohjelma
Tietologistiikan suuntautumisvaihtoehto

Kesäkuu 2007
Ohjaaja: FT, yliopettaja Yrjö Auramo
Avainsanat: tekoäly, optimointimenetelmä, evoluutio
UDK-luokka: 681.327.12:159.95

Tässä opinnäytetyössä selvitetään uusinta ajatusmallia edustavaa ja suurta suosiota saavuttanutta optimointimenetelmää. Kyseessä on geneettinen algoritmi, jonka esikuva on genetiikka ja luonnossa tapahtuva evoluutio. Selvitystyössä esitetään yksityiskohtaisesti proseduurin toimintaperiaate ja muutama sovellusongelma, johon menetelmä on käytännöllinen.

Geneettinen algoritmi perustuu darwinistiseen ajatteluun, ja se jäljittelee luonnossa tapahtuvan evoluution mekanismeja. Näitä mekanismeja ovat mm. yksilöiden valinta, risteytys ja mutaatio. Algoritmi on yleensä viritetty siten, että evoluutio on hyvin nopeaa ja pyrkii kehitykseen. Kehitys ilmenee ns. rakennuspalikkahypoteesina; kun yhdistellään hyviä osaratkaisuja, saadaan vieläkin parempia osaratkaisuja. Hypoteesia toistettaessa riittävän pitkään löydetään lopulta tarpeeksi hyvä loppuratkaisu. Menetelmä on varsin tehokas vaikeisiin optimointitehtäviin, jotka sisältävät paljon parametreja ja vähän tietoa. Geneettinen algoritmi on helppo hajauttaa, joten laskentateho voidaan jakaa usealle prosessorille.

Menetelmää sovelletaan jatkuvasti yhä enemmän, ja se on onnistunut täyttämään tehtävänsä usealla eri alalla. Kun algoritmin osat on rakennettu hyvin, se on yksinkertainen, joustava, ja eikä se ajaudu lähimpään lokaaliin osaratkaisuun.

GENETIC ALGORITHM

Aziz, Sami
Satakunta University of Applied Sciences
School of Technology Rauma
Information Technology
Information Logistics

June 2007
Tutor: Yrjö Auramo, PhD, Principal Lecturer
UDC: 681.327.12:159.95
Keywords: artificial intelligence, optimization method, evolution

In this thesis an optimization method that represents the newest paradigm, which has gathered a lot of popularity, was studied. The optimization method is Genetic Algorithm, a paragon of genetics and nature's evolution. In the report a detailed principle of the procedure was introduced. A few application problems which the algorithm can be applied to were presented.

The Genetic Algorithm is based on a darwinistic perspective and it simulates mechanisms of evolution in nature. These mechanisms are, among other things, the selection of individuals, crossover, and mutation. Usually the algorithm is tuned in such a way that evolution is very rapid and aspires to development. The development appears as the so-called building block hypothesis; as a result of combining good partial solutions even better partial solutions are produced. When the hypothesis is repeated sufficiently, a fit enough final solution is found. The method is very effective for difficult optimization tasks, which contain many parameters and little information. The Genetic Algorithm is easy to decentralize in order to share computing power with many processors.

The method is adapted increasingly, and it has fullfilled its position in many fields. When the parts of the algorithm are well-engineered, it is simple, flexible and will not be confined to the nearest local partial solution.

ALKULAUSE

Tämä insinöörityö on tehty Satakunnan ammattikorkeakouluun tekniikan rauman yksikköön projektiluonteisena selvitystyönä. Projektissa tekemäni työ on ollut mielenkiintoista ja opettanut minulle paljon uutta työn aihepiiristä.

SISÄLLYSLUETTELO

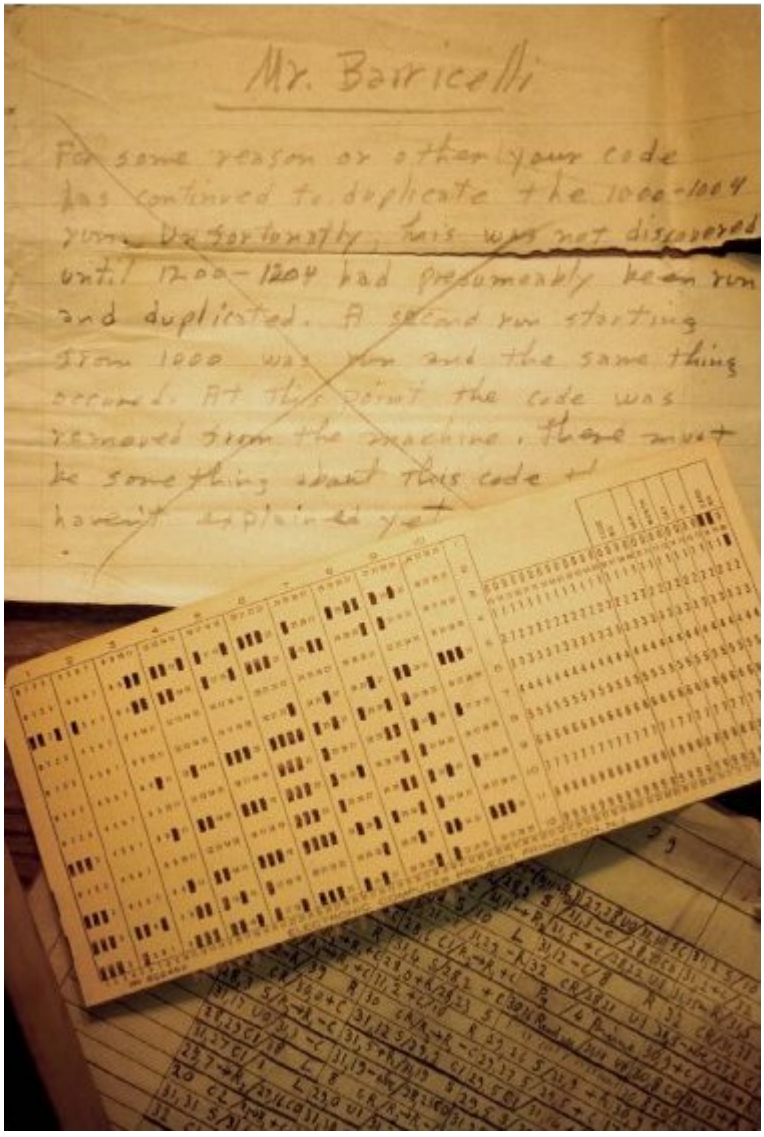
1 JOHDANTO

“Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand”

John H. Holland

Evoluutiokäsitteeseen pohjautuvien optimointimenetelmien yhteinen nimitys on *evoluutiolaskenta*. Yksi evoluutiolaskennan aloista on *geneettinen algoritmi*, joka edustaa uusinta ajatusmallia oppivissa ja älykkäissä järjestelmissä /1, s. 7/. Se luokitellaan *heuristiseksi* eli etsiväksi optimointimenetelmäksi, joka kuten myös useat *neurolaskennan* menetelmät kuuluvat satunnaishakumenetelmien joukkoon. Muita evoluutiolaskennan aloja ovat *evoluutio-ohjelmointi* ja *evoluutiostrategiat*. Näitä kaikkia yhdistää oleellisena tekijänä *populaatio* ja sen sisällä tapahtuva *valinta*, *risteytys* ja *mutaatio*. Nämä menetelmät ovat pysyneet pitkään erillisinä tutkimuskohteina, vaikka niissä on hyvin paljon samankaltaisuuksia /1, s. 7/. On huomattavaa, että geneettinen algoritmi ei tee eroa sille onko optimoitava rakenne symbolinen vai numeerinen. Geneettinen algoritmi, jota myös toisinaan evoluutioalgoritmiksi kutsutaan, perustuu siis yrityksen ja erehdyksen maailmaan, jossa hyödynnetään evoluutiobiologian tutkimuksessa löydettyjä perinnän, valinnan, risteytyksen ja mutaation periaatteita. Yrityksen voi tiivistää tapana kehittää uusia muunnelmia nykyisestä ratkaisusta ja erehdyksen testinä, joka hylkää syntyneistä ratkaisuista huonosti toimivat. Toisin sanoen tämä Charles Darwinin evoluutioteorian luonnonvalintaa jäljittelevä algoritmi ratkaisee ongelmat geenikeskeisellä prosessilla, jossa paras (sopeutuvin) ratkaisu (selviytyjä) valitaan.

2 HISTORIA



Kuva 1. Barricellin reikäkortti

Evoluutiolaskennan aloitti italialaisnorjalainen matemaatikko-biologi Nils Aall Barricelli. Vuonna 1954 hän simuloi yksinkertaista keinoelämäympäristöä käyttäen reikäkorttiohjelmointia /2/. Myös tämän jälkeen biologit ja genetiikot ovat esittäneet lukuisia teorioita keinoelämän simuloinnista aina vuoteen 1970 saakka. 1962 professori John H. Holland esitti evoluution ideoita käytettäväksi laskennallisissa menetelmissä /1, s. 7/. Vaikka Barricelli käytti evoluution simulointia yleisenä optimointimenetelmänä,

geneettinen algoritmi tuli nimetyksi ja laajalti tunnetuksi professori John H. Hollandin toimesta vasta 1970- luvun alkupuolella, joka johti 1975 ilmestyneeseen kirjaan *”Adaption in Natural and Artificial Systems”* /3/. Tutkimukset genettisistä algoritmeista pysyivät teoreettisena, kunnes pidettiin ensimmäinen kansainvälinen konferenssi aiheesta Illinoisin yliopistossa 1980- luvun puolivälissä. Kun akateeminen kiinnostus ja tietokoneiden laskentateho kasvoivat dramaattisesti, voitiin jo geneettisiä algoritmeja soveltaa käytäntöön. 1992 John R. Koza käytti geneettistä algoritmia evolvoidakseen ohjelmia suorittamaan tiettyjä tehtäviä. Hän käytti menetelmästä nimeä ”geneettinen ohjelmointi”. Ohjelmointi toteutettiin LISPillä, koska sillä voitiin tehdä sovelluksia käyttäen jäsennyspuuta /4, 5/. Nykyään GAlla on sovelluksia mm. tietojenkäsittelytieteessä, insinööritieteissä, taloustieteissä, fysiikassa ja matematiikassa /6/.

3 BIOLOGINEN TAUSTA

3.1 Kromosomit

Kaikki elävät organismit koostuvat soluista. Jokaisessa solussa on sama määrä kromosomeja. Kromosomit koostuvat *geeneistä*, perintötekijöistä eli *DNA (deoksiribonukleiinihappo)* rihmoista ja valkuaisaineesta sekä ne muodostavat vastinpareja, joista toinen peritään isältä ja toinen äidiltä. Kukin geeni sisältää aina yhden proteiinin valmistusohjeen /7/. Näin geenien toiminta vaikuttaa perimmäisellä tavalla kaikkien eliöiden ulkoasuun ja ominaisuuksiin. Yksi ominaisuuksista voi olla eliön silmien väri. Saman geenin vaihtoehtoisia muotoja kutsutaan *alleeleiksi*. Kullakin geenillä on oma *lokus* eli paikka kromosomissa. Organismin kaikkien kromosomien perintöainesta kutsutaan *genomiksi.* /8/

3.2 Risteytys

Kun puhutaan lisääntymisestä makroevolutiivisessa mittakaavassa, tarkoitetaan usein kahden yksilön risteyttämistä. Tämä tarkoittaa mikroevoluutiossa vanhempien kromosomien vastinparien tekijäinvaihtoa (eng. crossing-over) ja *rekombinaatiota* eli uudelleenyhdistämistä. Risteytys ei luo uutta geneettistä materiaalia jälkeläisille, vaan se yhdistelee jo olemassa olevaa. /9/

3.3 Populaatio

Biologiassa populaatiolla tarkoitetaan tavallisesti niiden yksilöiden joukkoa, jotka kuuluvat samaan lajiin ja elävät samanaikaisesti samalla alueella. Esimerkiksi lammessa elävät ahvenet muodostavat lammen ahvenpopulaation. Populaation ominaisuuksia ovat koko, tiheys, ikärakenne ja sukupuolijakauma. Populaation kokoon vaikuttavat syntyvyys, kuolleisuus ja muuttoliike. /10/

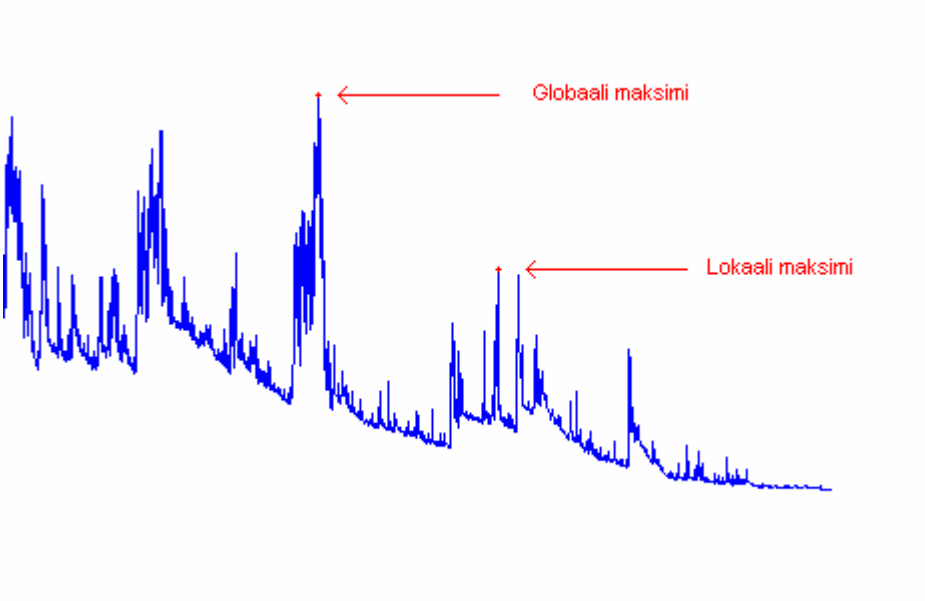
3.4 Kelpoisuus

Kelpoisuus tarkoittaa yksilön suhteellista kykyä tuottaa lisääntymiskykyisiä jälkeläisiä. Suhteellisuus tässä tarkoittaa sitä, että yksilöä verrataan kelpoisuuden osalta populaation muihin yksilöihin /11/.

3.5 Evoluutio

Evoluutio tarkoittaa muutosta populaation geenivarastossa sukupolvien myötä /12/. Kun geenivarasto kehittyy, yksilö voi sopeutua ympäristöön ja menestyä kilpailussa erilaisten ominaisuuksien vuoksi.

4 HAKUAVARUUS

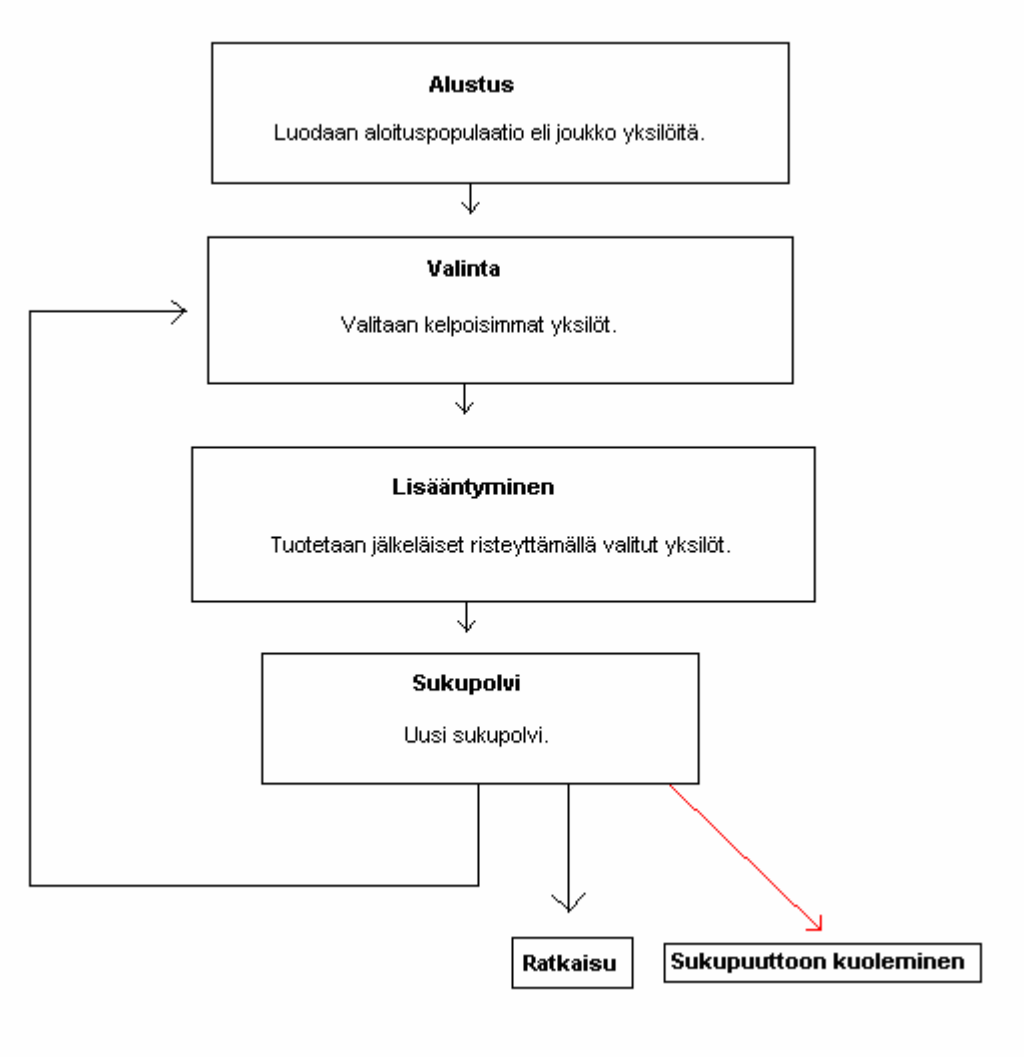


Kuva. 2. Hakuavaruus

Kun selvitetään tiettyä ongelmaa, ratkaisun tulisi olla riittävän hyvä tai paras.

Kaikkien mahdollisten ratkaisujen joukkoa kutsutaan *hakuavaruudeksi* (myös *tila-avaruudeksi*). Jokainen piste hakuavaruudessa edustaa yhtä ja tiettyä ratkaisua. Nämä mahdolliset ratkaisut voidaan “merkata” arvolla tai kelpoisuudella sen mukaan, kuinka hyvin ne ratkaisevat ongelman. Ongelma voi tulla ratkotuksi myös yhdistelemällä eri ratkaisuvaihtoehtoja. Hakuavaruudessa voi olla monta likimääräistä ratkaisua, joita kutsutaan *lokaaleiksi maksimeiksi* ja paras ratkaisu eli *globaali maksimi*. Tällöin hakuavaruus on epälineaarinen. Ongelmanratkaisussa pyritään optimiin eli globaaliin maksimiin, mutta toisinaan tyydytään likimääräiseen ratkaisuun. /13, s. III/

5 GENEETTINEN ALGORITMI



Kuva 3. Geneettisen algoritmin toiminta

On huomattavaa, että geneettinen algoritmi on monimuotoinen. Se sisältää useita eri vaiheita, joista jokainen voidaan toteuttaa monella eri tavalla. Geneettistä algoritmia käsittelevissä artikkeleissa esitellään usein jotain tehtäväkohtaisesti viritettyä versiota, jolla on pyritty tehokkuuteen ja luotettavuuteen.

5.1 Yrite kromosomin muodossa

GA:ssa yksilö koostuu keinotekoisesta kromosomista, joka sisältää ongelman komponentit ja niiden mahdolliset tilat (bio. alleelit). Kromosomia kuvataan yleensä *n*:n pituisella vektorilla, jonka komponentit X_i ovat siis geenejä /14, s. 26/.

$$<X_1,X_2.....,Xn>$$

GA:ssa geenejä voidaan koodata muutamilla eri tavoilla, joista useimmiten käytetty on kaksiaakkosellinen biinääriluku {0,1}. Kromosomi voi tällöin näyttää esimerkiksi tältä: <10110101>, jolloin kromosomin geenien määrä on 8 bittiä /14, s. 27/. Tätä bittikombinaatiota kutsutaan yritteeksi. Kromosomin jokaisella bitillä on oma paikkansa (bio. lokus) ja jokaisen bitin vaihtoehto 0 tai 1 vastaa ongelman tietyn komponentin (bio. geenin) tilaa.

Seuraavassa esimerkkikromosomissa komponentteja on 8 kpl ja mahdollisia yritteitä (bittikombinaatioita) on $2^8 =$ 256 kpl. Nämä 256 yritettä muodostavat hakuavaruuden, jonka joukossa on huonoja, parempia, hyviä ja parhaita ratkaisuja. Joukosta löytyy myös huonoin ja paras ratkaisu.

Esimerkki 1.

Huoneessa on 8 lamppua ja tarkoitus on saada huoneesta mahdollisimman valoisa. Tämä triviaalisti ratkaistaan siten, että kaikki lamput ovat päällä. Komponentteja on siis 8 kpl ja jokaisella niistä on kaksi tilaa - päällä tai pois päältä. Komponentteihin sovelletaan binäärikoodausta eli jokaisen lampun tilaa voidaan kuvata biteillä siten, että 1 vastaa päällä ja 0 pois päältä. Ratkaisu yrite on kombinaatio: <11111111>.

5.1.1 Komponenttien koodausmenetelmistä

Binäärikoodaus

Binäärikoodaus on yleisin, mutta ei kuitenkaan kaikissa ongelmissa paras mahdollinen tapa koodata komponentteja. On tärkeää koodata asianmukaisella tavalla, koska tämä on ratkaisevaa menetelmän tehokkuuden kannalta.

Esimerkissä 1. jokaisella lampulla on kaksi tilaa ja nämä tilat voidaan siis esittää binäärisenä ykkösenä tai nollana. Jos tiloja olisi enemmän esim. jokaisella lampulla olisi viisi kirkkausastetta, tarvittaisiin kolme bittiä lamppua kohden. Nämä kolme bittiä muodostavat yhden ongelman komponenteista.

Mahdolliset viisi lampun tilaa voidaan sopia seuraavasti:

1. pimeä	000
2. himmeä	001
3. valoisa	010
4. kirkkaampi	011
5. kirkkain	100

Jos esimerkissä 1. käytettäisiin lamppuja, joilla on viisi kirkkausastetta, ratkaisu yrite näyttäisi tältä:

<100 100 100 100 100 100 100 100> (välit lisätty selventämään kromosomin koostuvan 8:sta yksittäisestä komponentista)

Yhdellä bitillä voidaan kuvata kaksi tilaa, kahdella neljä, kolmella kahdeksan jne.

1 bit	2 bit	3 bit	
0	00	000	1.
1	01	001	2.
	10	010	3.
	11	011	4.
		100	5.
		101	?
		110	?
		111	?

Kuva 4. Binääritilat

Kolmella bitillä voidaan siis esittää 8 eri tilaa. Esimerkkitapauksessa lampuilla on viisi tilaa ja näihin on pakko varata vähintään kolme bittiä. Tästä seuraa se ongelma, että ylimääräisiä kombinaatioita jää 3 kpl. Optimaalinen esitys kolmella bitillä löytyisi sellaisen ongelman komponentista, joka saisi 8 eri tilaa. Koska myöhemmässä vaiheessa esitetyt kromosomin risteytys ja mutaatio voivat synnyttää kaikenlaisia bittikombinaatioita, tarvitaan keino käsitellä ylimääräisiä niistä. Yksi keino on muuttaa mahdottomat kombinaatiot mahdollisiksi mutaation kaltaisella menettelyllä. Toinen keino on käyttää sellaista kustannusfunktiota, joka sakottaa yritteitä, jotka sisältävät ylimääräisiä bittikombinaatioita. /15, s. 31/

Permutaatiokoodaus

Eräs toinen tapa koodata komponentteja on permutaatiokoodaus. Permutaatiokoodaus sopii parhaiten ongelmiin, joissa pitää järjestää komponentteja. Tällaisia ongelmia ovat mm. tehtävien ajoitus ja kauppamatkustajan ongelma. Seuraavassa esimerkissä esitetään tapa, jolla tehtävien ajoitus ongelma voidaan permutaatiokoodata kromosomiin.

Esimerkki 1.

Käytössä on kolme prosessoria ja ajoitettavana on kaksikymmentä eri tehtävää, joilla on tietty yksilöllinen deadline. Prosessorit ovat numeroitu reaaliluvuilla 1, 2 ja 3. Tehtävät ovat myös reaalilukuja 1-20. Kromosomi voidaan esittää koostuvan komponenteista (T, P), joissa T on tehtävä 1-20 ja P prosessori 1-3. /16/

Permutaatiokoodattu kromosomi voi näyttää tältä:

<(6,1) (4,2) (3,1) (15,1) (20, 2)>

Komponentteja voi olla yhtä monta kuin tehtäviäkin, mutta tässä esimerkissä niitä on 5. Toiset tehtävät voivat olla jonotuspuskurissa ja niitä ohjataan aina viiden erissä prosessoreille. Tehtävät käsitellään siinä järjestyksessä, kuin ne ovat kromosomissa. /16/

Arvoihin perustuva koodaus

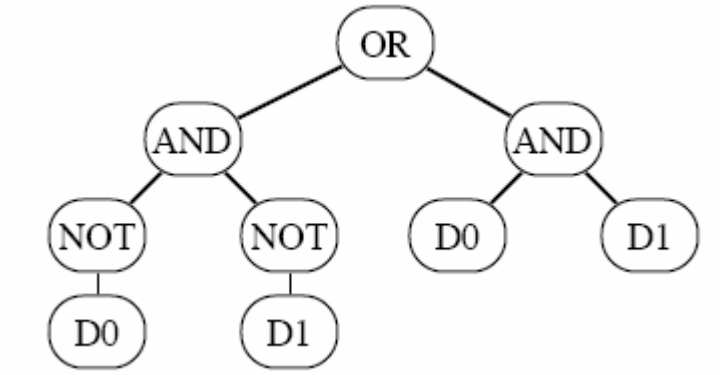
Arvoihin perustuva koodausmenetelmä käsittää sekatyyppisesti koodattuja kromosomeja. Parhaiten menetelmä soveltuu ongelmiin, joihin binäärikoodaus on vaikea toteuttaa. Osa kromosomeista voi koostua komponenteista, jotka sisältävät kirjaimia ja osa mm. lukuja, sanoja ja objekteja /13, s. X/.

Kromosomit voivat näyttää tältä:

<moi hei mitä kuuluu>
<32.00 21.12 144.2 23.22>
<2 3 3 3 4 5 6 7 3 2>

Puurakenteinen koodaus (Jäsennyspuu)

Puurakenteista koodausta käytetään pääsääntöisesti ohjelmien kehitykseen.
Nimi puurakenne tulee ajatuksesta, jonka mukaan kromosomi on koodattuna hierarkisesti kuvastaen puumaista rakennetta. Puurakenteisessa koodauksessa jokainen kromosomi voidaan kuvitella puuna, joka sisältää komponentteja, kuten objekteja, funktioita ja käskyjä. Koodaustapa on suosittu LISP ohjelmoijien keskuudessa, koska itse LISPin rakenne on esitettävissä jäsennyspuuna. /13, s. X/



Kuva 5. LISPin koodia puurakenteisena

5.2 Alkupopulaation luominen (alustus)

Geneettisen algoritmin ensimmäisessä vaiheessa luodaan alkupopulaatio eli joukko yritteitä (erilaisia kromosomeja omaavia yksilöitä). Alkupopulaation tulisi koostua yritteistä, jotka on valittu satunnaisesti ja tasaisesti koko hakuavaruuden alueelta. Populaation koko tulisi valita ratkaistavan ongelman mukaan. Mitä enemmän ongelma sisältää komponentteja, sitä suurempi aloituspopulaatio tulisi valita. Riittävän suuri aloituspopulaatio takaa *diversiteetin* eli monimuotoisuuden. Populaation koko on yleensä kymmenien tai satojen yritteiden kokoinen. Populaatio pyritään pitämään mahdollisimman pienenä, mutta toisaalta monimuotoisena, jotta algoritmin aikaa ei jatkossa tuhlaantuisi liikaa huonojen yritteiden karsintaan ja parempien yritteiden ilmestymisen odotteluun /17, s.19/. Nyrkkisääntönä voidaan pitää, että on suotuisampaa käyttää mielummin hieman suurempaa populaation kokoa kuin liian pientä, sillä näin vältytään lokaaleilta maksimeilta.

5.3 Kustannus-, kelpoisuus-, hyvyys-, fitness- funktio

Geneettisen algoritmin ratkaisu tiettyyn ongelmaan löytyy hyvistä tai parhaista yritteistä. Yritteen hyvyys mitataan ns. kustannusfunktiolla (eng. Fitness Function), joka määrittää yksilöiden kelpoisuuden laskemalla niiden onnistumisen ongelman ratkaisussa. Kustannusfunktion tarkoitus on karsia huonoja yksilöitä parempien joukosta. Funktio voi olla matemaattisesti hyvinkin monimutkainen ja sitä prosessoidaan tuhansia tai jopa miljoonia kertoja ennen kuin ratkaisu löytyy. Ideaalinen hyvyysfunktio on nopea ja se vastaa ongelman ratkaisua. /17, s. 21/

5.4 Valinta

Kun alkupopulaatio on luotu, valitaan risteytettävät vanhemmat. Nämä vanhemmat tuottavat seuraavan sukupolven. Risteytettävistä vanhemmista syntyy aina kaksi jälkeläisyritettä. Vanhempien valintaan vaikuttaa tietty proseduuri, jota käytetään. Alla on listattu viisi erilaista menetemää, joilla voidaan valita yksilöt. Jokaisessa

menetelmässä valintaa painotetaan niille yksilöille, jotka ovat kelpoisimpia. Proseduurit voidaan jakaa deterministiseen, elitistiseen valintaan ja todennäköisyyteen perustuviin menetelmiin.

5.4.1 Determistinen valinta/kokeilu (Deterministic sampling/selection)

Tämä proseduuri valitsee parhaan tai parhaat yksilöt populaatiosta. Menetelmä toimii siten, että se valitsee vain toisen vanhemmista ja sillä on paras kelpoisuus koko populaatiossa. Toinen vanhempi voidaan valita arpomalla tai todennäköisyyteen perustuvilla menetelmillä /18/.

5.4.2 Elitistinen valinta

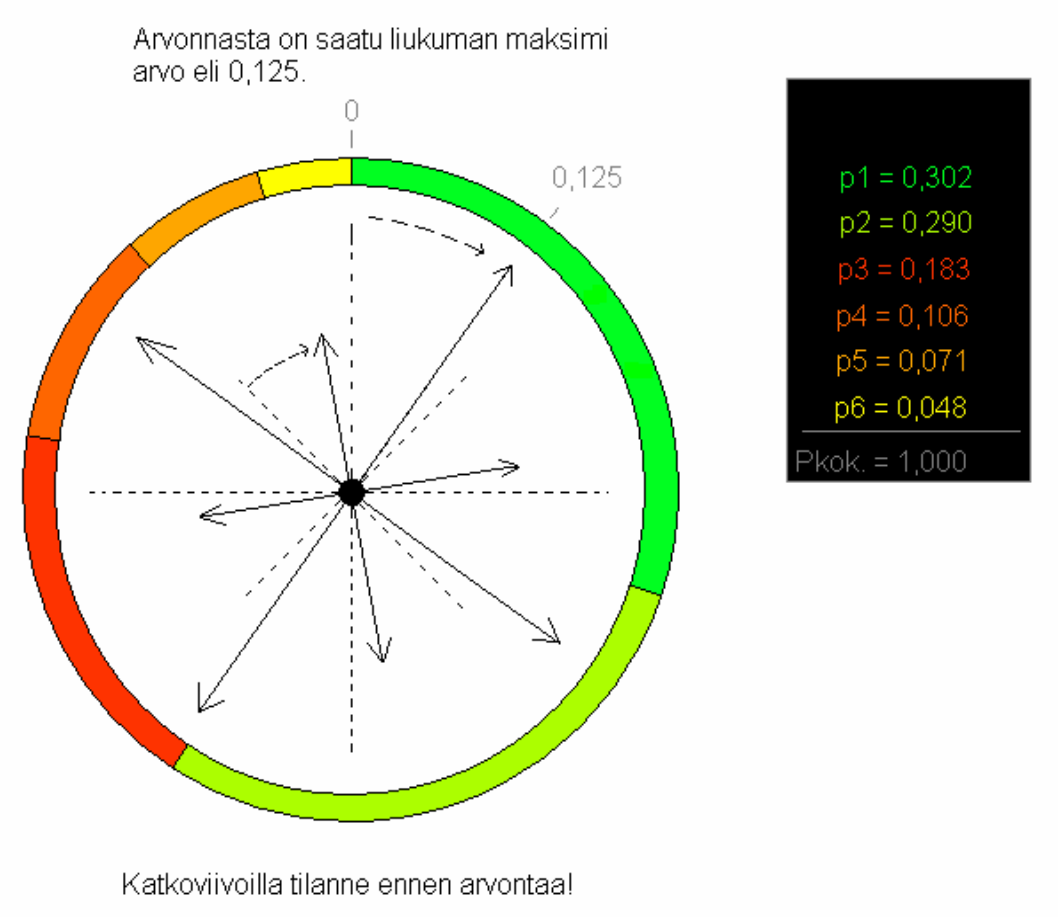
Elitistisen valinnan tarkoitus on valita yksilöt seuraavaan sukupolveen siten, että vanhan populaation huonoimmat korvataan uuden populaation parhaimmilla /15, s. 26/. Menetelmällä saadaan nopeasti uusia ja parempia yksilöitä. Valinta tuottaa nopean ratkaisun, mutta se voi olla lokaali maksimi, koska diversiteetti voi olla heikentynyt.

5.4.3 Rulettivalinta

Rulettivalinta (eng. roulette wheel selection) on suosittu todennäköisyyksiin perustuva metodi, jossa yksilöt valitaan ”pyörittämällä rulettia”. Proseduurissa annetaan yksilöille sitä suurempi todennäköisyys tulla valituksi kuin niiden fitness-arvo edellyttää. Valintatodennäköisyys on yksilön fitness-arvon suhde kaikkien yksilöiden fitness-arvojen summaan. Tämä on siis sama kuin rulettipöydässä annettaisiin paremmille ratkaisuille enemmän koloja ja triviaalisti huonommille ratkaisuille vähemmän /19/.

5.4.4 Stokastinen valinta (SUS)

Stokastinen (eng. Stochastic Universal Sampling) valinta on rulettivalintaakin suositumpi menetelmä, koska hajonta on suurempi. Stokastinen valinta toimii kuten rulettivalinta, sillä poikkeiksella, että ruletissa on monta osoitinta (pointteria), joiden etäisyys toisistaan on 1/N ja rulettia ei pyöritetä vaan sen liukuma arvotaan väliltä [0, 1/N], jossa N on pointtereiden määrä /20/. Alla olevassa kuvassa N = 8, 1/8 = n. 0.125 ja puolet pointtereista on lyhennetty selventämään tilannetta.



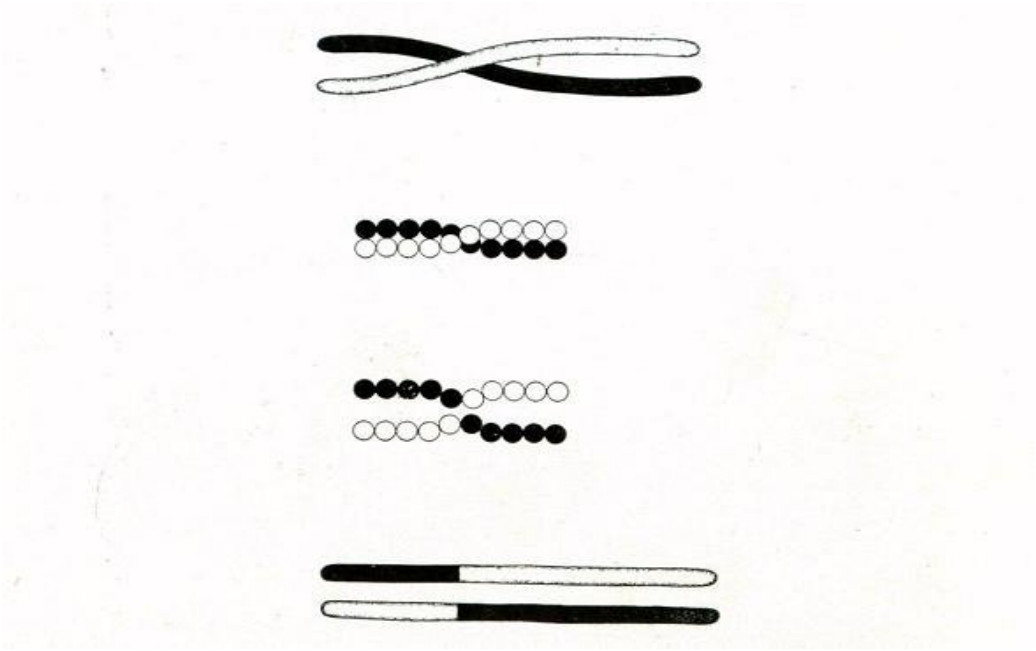
Kuva 6. Stokastinen valinta

5.4.5 Turnajaisvalinta

Turnajaisvalinnan periaatteena on valita populaatiosta satunnaisesti tietty määrä yritteitä, joista paras selviää uuteen sukupolveen. Jatkopaikasta taistelevia yritteitä valitaan yleensä kaksi. Paremmuus mitataan suoraanverrannollisesti kelpoisuuteen. Toinen tapa toteuttaa turnajaisvalinta on valita populaatiosta kaikki yritteet yksi kerrallaan ja yritteille satunnaisesti vastustajat. Kun vastustajat on valittu, tarkastellaan montako vastustajaa yrite on voittanut eli moneen vastustajaan verrattuna yritteellä on parempi kelpoisuus. Jatkoon pääsevät ne yritteet, jotka ovat menestyneet turnajaisissa parhaiten eli voittaneet eniten vastustajia. /14, s. 44/

5.5 Risteytys ja rekombinaatio

Kun vanhemmat on valittu tietyllä valintamenetelmällä, seuraa risteyttämis vaihe. Risteyttämisen tarkoitus on luoda vanhempien geneettisestä materiaalista uusia yhdistelmiä ja parempaa materiaalia omaavia jälkeläisiä. Vanhempien kromosomeille tapahtuu tekijäinvaihto eli eng. crossing-over ja rekombinaatio, jossa tekijät uudelleen sijoitetaan. Operaatio voidaan toteuttaa muutamilla eri tavoilla.

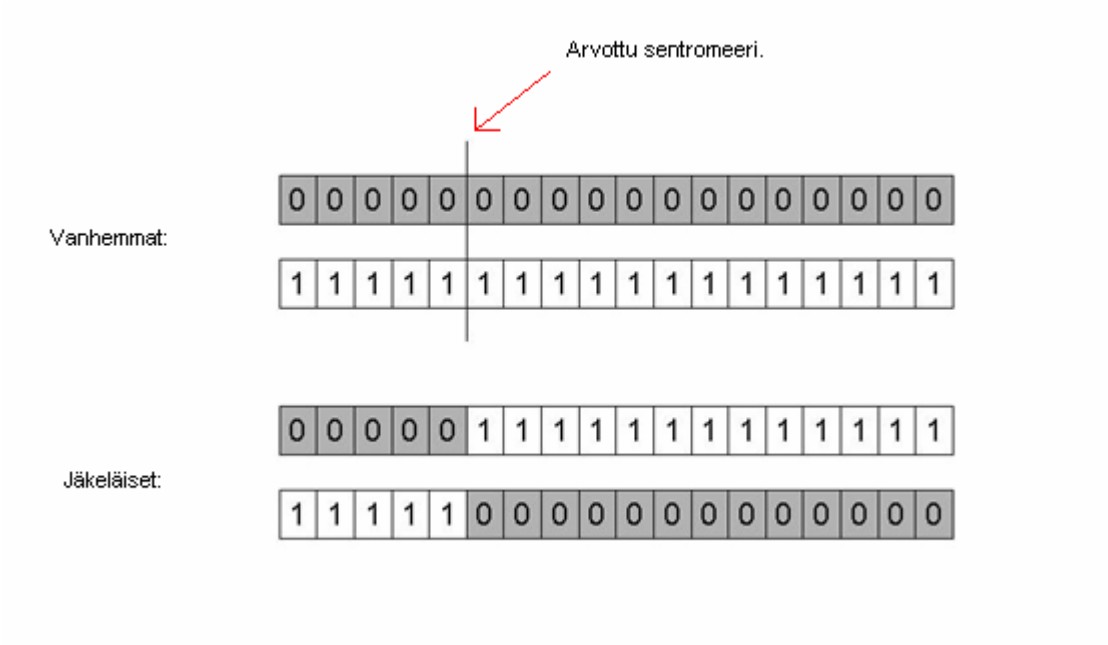


Kuva 7. Crossing-over

5.5.1 Yleiset risteytysmenetelmät

Yksipisteristeytys

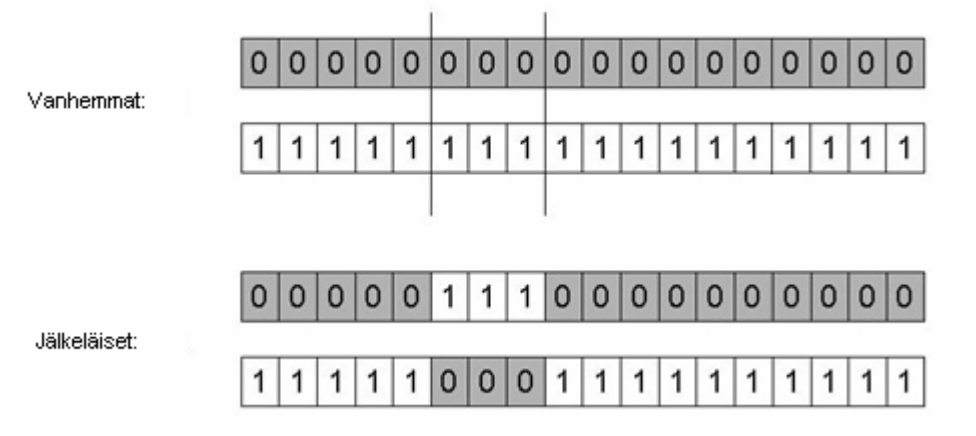
Tavallisin risteytysmenetelmä on ykspisteristeytys ja se vastaa luonnossa tapahtuvaa tekijäinvaihtoa realistisesti. Menetelmässä vanhempien yritekromosomeista valitaan satunnaisesti yksi piste (*sentromeeri*), joka erottaa kromosomin osat toisistaan. Kun osat on erotettu, ne vaihdetaan uros- ja naarasyritteen kesken siten, että uroksen ensimmäinen osa sijoitetaan naaraksen toiseen osaan ja päinvastoin. /14, 21, s. 47-48/ On huomattavaa, että molempien vanhempien kromosomeista valitaan sama piste, jotta rekombinaation jälkeen kromosomien pituudet pysyvät samana. Operaatio synnyttää kaksi jälkeläistä, joilla on molempien vanhempien geneettistä materiaalia.



Kuva 8. Yksipisteristeytys

Kaksipisteristeytys

Kaksipisteristeytys toimii muutoin samalla periaatteella kuin yksipisteristeytys sillä poikkeuksella, että kaksipisteristeytyksessä arvotaan kromosomeista kaksi risteytyskohtaa. Kun kohdat on arvottu, vaihdetaan niiden väliset geenit keskenään. Kaksipisteristeytyksen tarkoituksena on sekoitaa geneettistä materiaalia paremmin. /21, s. 47/



Kuva 9. Kaksipisteristeytys

Monipisteristeytys

Monipisteristeytys on tarkoitettu varsinaisesti erityisen pitkille kromosomeille. Tässä risteytysmenetelmässä arvotaan kolme tai useampi sentromeeri. Menetelmää on helppo soveltaa esim. binäärikoodattuihin kromosomeihin, mutta ei permutaatiokoodattuihin. Permutaatiokoodatuille kromosomeille ei voida suorittaa normaalia tekijäinvaihtoa, koska geenien järjestyksellä on merkitystä. Sama ongelma esiintyy myös yksipiste- ja kaksipisteristeytyksessä. Yksi tapa risteyttää permutaatiokoodattuja kromosomeja on käyttää seuraavassa kuvassa esitettyä menetelmää. /21, s.52, s.56, 22/

Vanhempi 1.	< 1, 2, 3, 4 | 5, 6, 7 | 8, 9, 10, 11, 12 | 13 >
Vanhempi 2.	< 12, 8, 5, 3 | 7, 4, 9 | 11, 13, 6, 10, 1 | 2 >
Jälkeläinen 1.	< 1, 2, 3, 4, 12, 8, 5, 6, 7, 9, 10, 11, 13 >
Jälkeläinen 2.	< 12, 8, 5, 3, 1, 2, 4, 7, 9, 11, 13, 6, 10 >

Kuva 10. Monipisteristeytys permutaatiokoodatuille kromosomeille

Kuvassa 5.5.3 esitetyille vanhemmille on arvottu kolme katkaisukohtaa. Jälkeläiselle 1. kopioidaan vanhemman 1. ensimmäiset neljä geeniä (numeroa). Jälkeläisen 1. ensimmäisen sentromeerin jälkeinen osa saadaan valitsemalla kolme geeniä vanhemmalta 2. siten, että yksikään geeni ei esiinny jälkeläisen 1. neljässä ensimmäisessä lokuksessa. Vanhemmat skannataan aina alusta loppuun ja kun löytyy poikkeava geeni, jota ei ole jo jälkeläisessä, valitaan se jatkoksi. Geenit valitaan siinä järjestyksessä kuin ne esiintyvät vanhemmalla 2. Toisen sentromeerin jälkeinen osa saadaan valitsemalla viisi geeniä vanhemmalta 1. siten, että yksikään geeni ei esiinny jälkeläisen 1. seitsemässä (4+3) ensimmäisessä lokuksessa. Jälkeläisen 1. viimeisen sentromeerin jälkeinen osa, jossa on vain yksi geeni, saadaan valitsemalla jäljellä oleva, puuttuva geeni vanhemmalta 2. Jälkeläinen 2. saadaan samalla periaatteella kuin jälkeläinen 1., sillä poikkeuksella, että aloitetaan vanhemmasta 2.

Satunnaisristeytys

Tämä risteytystapa arpoo satunnaisia kohtia kromosomista, jonka jälkeen ne risteytetään kyseisistä sentromeerikohdista.

Aritmeettinen risteytys

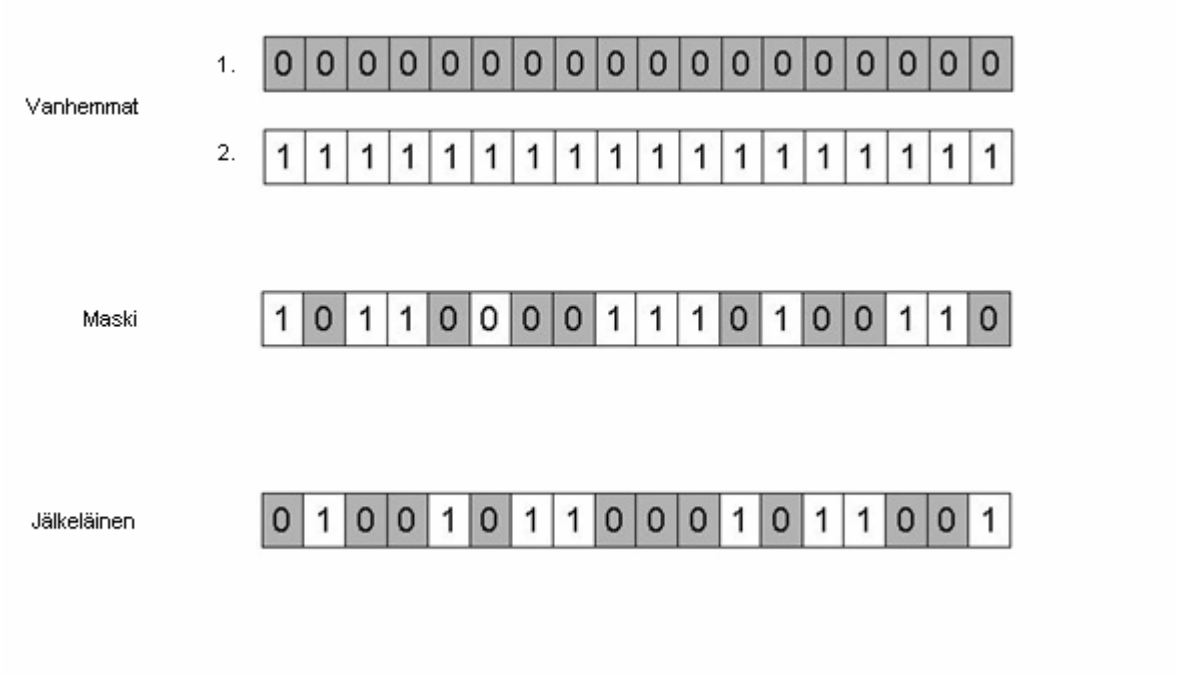
Aritmeettinen risteytys tapa käyttää aritmeettisiä operaattoreita, kuten AND, NOR, XOR ja NOT.

Spesifinen risteytys

Spesifistä risteytystapaa käytetään silloin, kun kromosomit on koodattu esimerkiksi arvoihin perustuvalla koodauksella ja muita risteytysmenetelmiä ei voida käyttää. Spesifiseen risteytykseen ei ole mitään varsinaista mallia, koska se on tapauskohtainen.

Yhtenäinen risteytys (Uniform Cross-over)

Yhtenäinen risteytys sopii varsinaisesti binäärikoodattuihin kromosomeihin. Uniformissa risteytyksessä generoidaan satunnaisesti nollia ja ykkösiä sisältävä kromosomin pituinen ns. maski. Jos maskissa on geenin kohdalla ykkönen, otetaan vastaava geeni jälkeläiselle ensimmäiseltä vanhemmalta, ja jos maskissa on nolla otetaan geeni toiselta vanhemmalta. Käytännössä maski on ennakkoon suoritettu arvonta siitä, että kummalta vanhemmalta otetaan geeni jälkeläiselle. Toinen jälkeläinen saadaan käyttämällä esim. komplementti eli käänteistä maskia tai generoimalla kokonaan uusi maski. /15, s. 38/



Kuva 11. Yhtenäinen risteytys

5.6 Mutaatio

Mutaatio on operaatio, joka suoritetaan yleensä risteyttämisen yhteydessä. Mutaation tarkoitus on lisätä diversiteettiä ja estää paikalliseen optimiin jumittuminen. Jos mutaatioita on liikaa, ne hävittävät jo löydettyjä hyviä ratkaisuja /17, s.20/. Edulliset mutaatiot yleistyvät populaatiossa risteytysten kautta, kun taas epäedulliset mutaatiot johtavat mutaation sisältävien yksilöiden menehtymiseen, ja ominaisuuden häviämiseen populaatiosta /23/. Mutaatiotekniikoita on muutamia ja toisinaan niitä pitää muokata, koska ne ovat riippuvaisia kromosomin koodaustavasta.

Yksi mutaatiotekniikoista on ns. flip bit(s) –menetelmä, jossa invertoidaan yksi tai useampi satunnaisesti valittu bitti. Esimerkiksi binäärikoodattu kromosomi 1011 voi olla mutaation jälkeen 1001 /24/. Toinen mutaatiotekniikka on swap bit(s)-menetelmä, jossa vaihdetaan satunnaisesti valitun bitin paikka. Tätä voidaan myös soveltaa useammalle eri bitille /25/.

6 ALGORITMIN HYVÄT JA HUONOT OMINAISUUDET

Geneettistä algoritmia voidaan pitää hyvänä silloin, kun sen osat on rakennettu hyvin. Kun ongelma on kombinatorinen, paljon muuttujia sisältävä ja hakuavaruudeltaan suuri, geneettinen algoritmi näyttää tehokkuudensa siinä, missä perinteisiä algoritmeja ei voida soveltaa. Perinteisillä algoritmeilla voi olla vaikea muodostaa ongelman funktio. GAn hyvänä puolena on se, että sen osat voidaan hajauttaa eri prosessoreille. GAlla voidaan suorittaaa ns. ryhmäevoluutiota (eng. co-evolution) /17/, jossa lasketaan samanaikaisesti eri populaatioita eri prosessoreilla, näin saadaan myös diversiteetti pidettyä rikkaana. Algoritmista löytyy myös huonoja ominaisuuksia, kuten sen hitaus aloituspopulaation ollessa suuri. GAn suunitteluun voi mennä paljon aikaa, sillä monimutkaisten ongelmien rakentaminen muuttujiksi, parametreiksi ja funtioksi voi olla hankalaa. Tietty perinteinen valmisalgoritmi voi ratkaista spesifisen ongelman nopeammin, tästä syystä Gata tulisi soveltaa monimutkaisempiin ongelmiin. Jos GA on suunniteltu puutteellisesti, silloin kasvaa riski, että huonolla tuurilla ratkaisu jumittuu paikalliseen maksimiin ja näin ei päädytä parhaimpaan ratkaisuun. Monimutkaiset ongelmat vaativat paljon laskentatehoa ja muistia, koska populaatiot voivat olla isoja ja raskaita. Yleensä aloituspopulaatiossa ja muutamassa seuraavassa sukupolvessa tapahtuu eniten muutoksia, jotka koskevat geneettistä materiaalia.

7 SOVELLUKSISTA

7.1 Käsin simuloitu geneettinen algoritmi

Geneettistä algoritmia voidaan soveltaa yksinkertaiseen esimerkkiin, jossa käytetään binäärikoodausta, kromosomien pituutena 8- bit ja rulettivalintaa. Kelpoisuusfunktio on binääriluvun arvo siten, että suurempi luku edustaa parempaa ratkaisua. Aloituspopulaation koko on 6 yritettä ja ne arvotaan kolikkoa heittämällä siten, että kruuna on 0 ja klaava 1. Risteytysmenetelmänä käytetään yksipisteritsteytystä ja mutaatiota sovelletaan yhden yritteen satunnaiseen komponenttiin.

Arvonta antaa aloituspopulaation yritteiksi:

Taulukko 1. Aloituspopulaation yritteet

	Kromosomi/Yrite	Kelpoisuus
1.	<00100111>	39
2.	<10010101>	149
3.	<11110110>	246
4.	<01010110>	86
5.	<00111010>	58
6.	<11101100>	236

Kelpoisuuksien keskiarvo: (814 / 6) ~ 136

Kelpoisuudet tulee suhteuttaa suurimpaan arvoon, joka on kelpoisuuksien summa, koska todennäköisyyksien summa on 1.

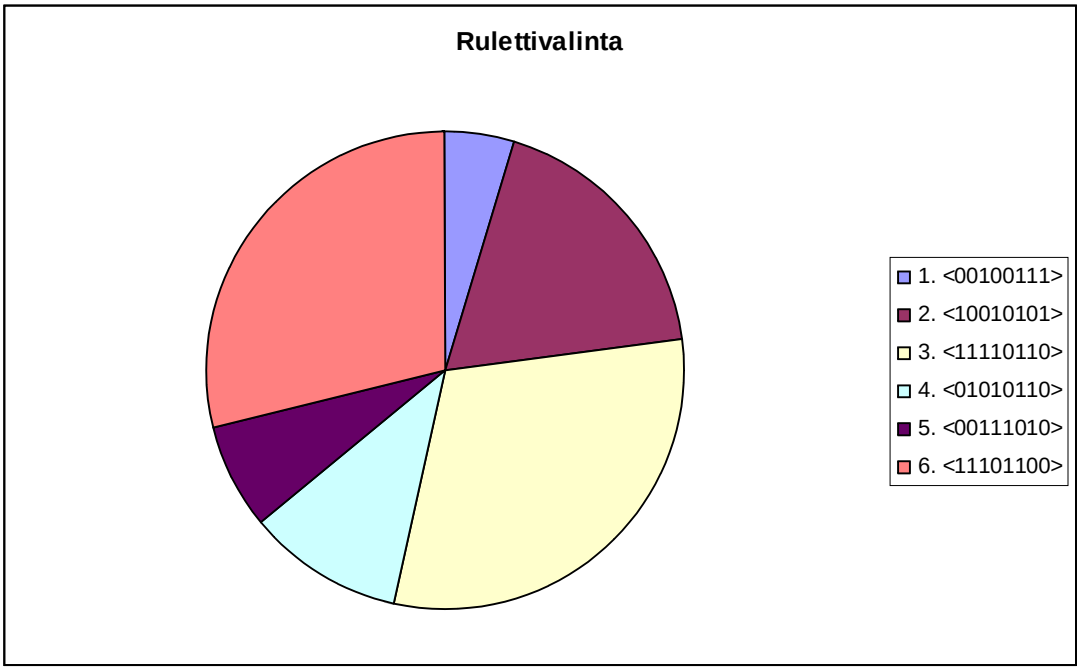
Jokaisen yritteen valintatodennäköisyys saadaan käyttämällä suhdetta ja verrantoa kullekin kelpoisuudelle. Seuraavassa kuvassa on laskettu huonoimman yritteen todennäköisyys tulla valituksi.

$$\frac{814}{39} = \frac{1}{x_p}$$

$$x_p \sim 0.048$$

Kuva 12. Huonoimman yritteen todennäköisyys

Yritteellä 3. on parhaimmat mahdollisuudet ja 1. huonoimmat tulla valituksi.



Diagrammi 1. Kaikkien yritteiden todennäköisyydet

Rulettivalinta osuu vanhempiin kuutena kierroksena seuraavasti:

Arvonnan oikea rivi: 1,6,4,3,3 ja 4.

Taulukko2. Risteytettävät yritteet

1.	<00100111>
6.	<11101100>
4.	<01010110>
3.	<11110110>
3.	<11110110>
4.	<01010110>

Risteytyskohta arvotaan kunkin parin kohdalla seuraavan taulukon mukaisesti.

Taulukko 3. Aloituspopulaation yritteet

1.	<001|00111>
6.	<111|01100>
4.	<01|010110>
3.	<11|110110>
3.	<1|1110110>
4.	<0|1010110>

Taulukko 4. Jälkeläiset yksipisteristeytyksen ja mutaation jälkeen

1. <00101100>
2. <11100111>

3. <01110110>
4. <11010110>

5. <11010110>
6. <01111110>

Taulukko 5. Jälkeläiset kaikkien geneettisten opetaatioiden jälkeen.

	Kromosomi/Yrite	Kelpoisuus
1.	<00101100>	44
2.	<11100111>	231
3.	<01110110>	118
4.	<11010110>	214
5.	<11010110>	214
6.	<01111110>	126

Kelpoisuuksien keskiarvo: 947 / 6 ~ 158.

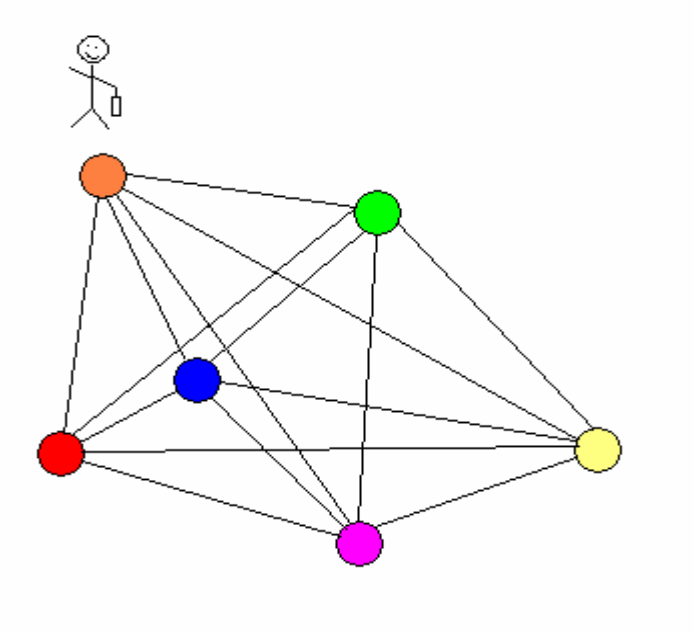
Jälkeläisten kelpoisuuksien keskiarvo on n. 14 % yksikköä suurempi, kuin vanhempien. Jatkamalla geneettistä algoritmia saadaan globaali optimi, jolloin kelpoisuuksien keskiarvo on 255.

7.2 Kauppamatkustajan ongelma (eng. Travellins Salesman Problem)

"If I could tell it, I wouldn't have to dance it"

Isadora Duncan

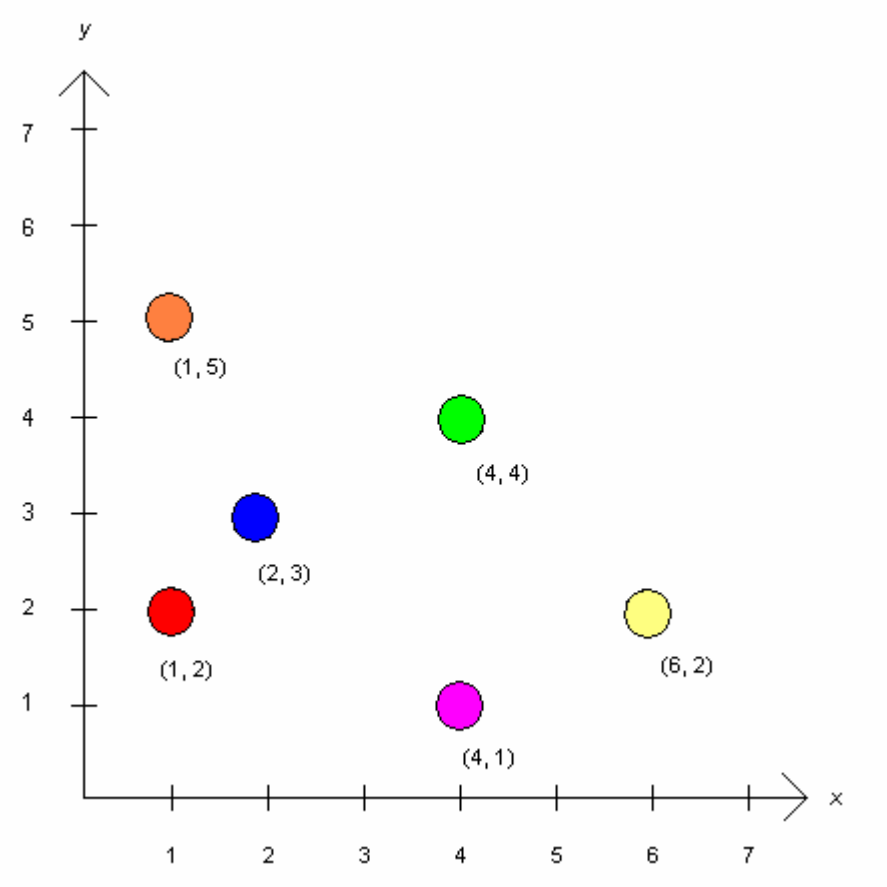
Kauppamatkustajan ongelma on kombinatorinen ja kuuluu NP (Nondeterministic Polynomial Time)- täydellisten ongelmien joukkoon eli laskennallisesti erittäin vaativiin ongelmiin. Ongelma voidaan kuvata seuraavasti: Jos kauppamatkustajalla, jonka tarkoituksena on käydä eri kaupungeissa, on käytössään kartta, aikataulu ja hän tietää kaupunkien etäisyydet ja millaisia ongelmia eri tieosuuksilla on, voiko hän laskea itselleen automaattisesti nopeimman kulkureitin, jossa palataan lopussa lähtökaupunkiin ja käydään kaikkien tarvittavien kaupunkien kautta, ilman että käydään samassa kaupungissa kahdesti, ja jos voi, miten? /27/ Kauppamatkustajan ongelmaan on kehitetty lukuisia eri laskentamenetelmiä ja algoritmeja. Tässä kappaleessa pyritään soveltamaan geneettistä algoritmia kyseiseen ongelmaan.



Kuva 13. TSP- ongelman kuvitteellinen kaupunkiasetelma

Kun tarkastellaan kuvaa 7.2, huomataan oranssista aloituspisteestä lähtevän viisi eri mahdollista reittiä, joista kukin johtaa eri kaupunkiin. Seuraavasta kaupungista lähteviä reittejä on neljä, kolmannesta kolme, neljännestä enää kaksi jne. Tämä johtaa matemaattiseen ja yleispätevään kaavaan, jossa kaupunkien määrä on *n* ja eri reittimahdollisuuksia on *(n-1)!* kpl. Kaavan mukaan kaupungeista vähennetään yksi, koska lähtökaupunkia ei lasketa matkustusvaihtoehtoihin eli nykyisestä kaupungista nykyiseen ei voi matkustaa, vaan pitää siirtyä seuraavaan. Jos kaupunkeja on kuusi, matkareittivaihtoehtoja on kaavan mukaan 120 kappaletta. Yksi mahdollisista reiteistä on lyhin ja ratkaisun pitää olla se.

Pisteiden (kaupunkien) etäisyydet toisistaan saadaan sijoittamalla ne koordinaatistoon ja laskemalla niiden väliset matkat.



Kuva 14. Kaupunkien sijaintikoordinaatit

Kahden pisteen välinen etäisyys lasketaan kaavalla: $|AB| = \sqrt{[(x_2-x_1)^2+(y_2-y_1)^2]}$, kun tunnetaan matkan (janan) päätepisteet $A(x_1, y_1)$ ja $B(x_2, y_2)$.

Soveltamalla koordinaatteja kaavaan, kaupunkien välisiksi etäisyyksiksi saadaan matkat, jotka kerrotaan sadalla.

Taulukko 6. Kaupunkien väliset matkat

Oranssi - Vihreä	316
Oranssi - Keltainen	583
Oranssi - Violetti	500
Oranssi - Punainen	300
Oranssi - Sininen	224
Vihreä - Keltainen	289
Vihreä - Violetti	300
Vihreä - Punainen	361
Vihreä - Sininen	224
Keltainen - Violetti	223
Keltainen - Punainen	500
Keltainen - Sininen	412
Violetti - Punainen	316
Violetti - Sininen	289
Punainen - Sininen	141

Kaupungit numeroidaan siten, että oranssi on 1., vihreä 2., keltainen 3., violetti 4., sininen 5. ja punainen 6. Aloituspopulaatio luodaan kymmenestä permutaatiokoodatusta kromosomista, jotka sisältävät arvottuja reittikombinaatioita. Reittikombinaatiot ovat numeroituja kaupunkeja tietyssä järjestyksessä. Reittien

kustannus määritellään funktiolla, joka suosii lyhimpiä reittejä. Hakuavaruuden koko määräytyy reittivaihtoehdoista, joita on 120.

Taulukko 7. Aloituspopulaation kromosomit ja niiden kustannus

Kromosomi		Matka
<3,4,2,5,1,6>	223+300+224+224+300+500	1771
<6,5,2,3,4,1>	141+224+289+223+500+300	1677
<2,1,4,3,5,6>	316+500+223+412+141+361	1953
<4,2,3,5,1,6>	300+289+412+224+300+316	1841
<6,3,4,5,2,1>	500+223+289+224+316+300	1852
<2,3,4,6,5,1>	289+223+316+141+224+316	1509
<6,5,4,3,1,2>	141+289+223+583+316+361	1913
<3,4,5,6,2,1>	223+289+141+361+316+583	1913
<3,2,1,4,5,6>	289+316+500+289+141+500	2035
<3,2,6,4,5,1>	289+361+316+289+224+583	2062

Aloituspopulaation kymmenen reittivaihtoehdon keskiarvo on n. 1853.

Ratkaisussa sovelletaan ensin stokastista valintaa, jonka jälkeen yksilöt risteytetään kaksipistemenetelmällä. Seuraavaksi uudesta populaatiosta valitaan kaksi parasta eli lyhintä reittiä ja risteytetään ne keskenään. Tästä syntyvät kaksi jälkeläistä sijoitetaan edellisen pupulaation kahden huonoimman eli pisimmän reitin tilalle. Näin syntyy uusi ja entistäkin parempi populaatio. Proseduuria jatketaan kunnes päästään ratkaisuun.

8 YHTEENVETO

Tässä työssä on tarkasteltu geneettistä algoritmia ja sen mekanismeja, jotka jäljittelevät luonnossa tapahtuvaa evoluutiota kehityksen suuntaan. Tärkeimpiin mekanismeihin, kuten valintamenetelmiin ja kromosomien koodaustapoihin, syvennyttiin myös yksityiskohtaisten esimerkkien muodossa. Biologisesta taustasta peruskäsitteet tuli esitettyä lyhyesti sekä tutustuttiin historiaan. Työssä tarkasteltiin sovelluksista käsin simuloitua algoritmia sekä klassista kauppamatkustajan ongelmaa. Ensimmäisessä sovellutuksessa tutustuttiin binääri- ja toisessa permutaatiokoodaukseen. Käsin simuloidun algoritmin edetessä huomattiin uuden populaation kelpoisuuksien keskiarvon nousevan, joka johti parempaan ratkaisuun. Kauppamatkustajan ongelma puettiin kirjallisesta muodosta matemaattiseen eli geneettisen algoritmin käsiteltävään muotoon ja huomattiin, että se ei ollut monimutkaista.

Jos aikaa olisi ollut enemmän käytettävissä, TSP-ongelmaa olisi voinut analysoida syvemmin ohjelmointiosuudella, jossa olisi testattu eri valintamenetelmiä ja niiden vaikutusta algoritmin tehokkuuteen. Vertailu muihin samankaltaisiin satunnaishakumenetelmiin, kuten parveilualgoritmiin (Particle Swarm Pattern) olisi ollut mielenkiintoista. GA:n toiminta herättää kysymyksiä myös sen keinotekoisuudesta, sillä algoritmin evoluutiokehitys on paljon nopeampaa kuin luonnossa tapahtuva kehitys.

LÄHTEET

1. Riihimäki, V. Lentokone-ohjus -tehtävän alkuarvauksen tuottaminen geneettisillä optimointialgoritmeilla [verkkodokumentti]. Helsinki: 2001 [viitattu 9.4.2007]. Sovelletun matematiikan erikoistyöt. Helsingin teknillinen korkeakoulu, systeemianalyysin laboratorio. 41 s. Saatavissa: http://www.sal.tkk.fi/Opinnot/Mat-2.108/pdf-files/erii01.pdf

2. Salminen, T. Geneesys – katsaus kolmiulotteiseen keinoelämään nyt ja hahmotelmia tulevaisuudesta [verkkodokumentti]. Helsinki: 2000 [viitattu 9.4.2007]. Taideteollinen korkeakoulu, medialaboratorio. 31 s. Saatavissa: http://mlab.uiah.fi/%7Etosalmin/thesis/geneesys_101_web.pdf

3. Virtanen, P. Geneettiset algoritmit ja sukupuolten taistelu [verkkodokumentti]. Helsinki: 2005 [viitattu 9.4.2007]. Sovelletun matematiikan erikoistyöt. Helsingin teknillinen korkeakoulu, systeemianalyysin laboratorio. 45 s. Saatavissa: http://www.sal.tkk.fi/Opinnot/Mat-2.108/pdf-files/evir05.pdf

4. Salo, H. Geneettiset algoritmit ohjelmoinnissa. SteP ’96 conference. Vaasa, 19.-22.8.1996. Suomen tekoälyseura (STeS), 1996. Saatavissa: http://lipas.uwasa.fi/stes/step96/step96/salo/

5. Wright, A. Introduction to Genetic Programming [verkkodokumentti]. Missoula: 2001 [viitattu 9.4.2007]. Lecture notes for October 31, 2002. Department of Computer Science, University of Montana, Missoula, MT. 4 s. Saatavissa: http://www.cs.umt.edu/CS/COURSES/CS555/lect10_31.pdf

6. Wikipedia, Geneettinen algoritmi [verkkodokumentti], 10. tammikuuta 2007 kello 12.52. [viitattu 9.4.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Geneettinen_algoritmi

7. Haila, Y. Ihmisen DNA on kartoitettu (melkein) – Entä sitten? Tiede & Edistys 2/01, s. 142–149.

8. Wikipedia, Geeni [verkkodokumentti], 15. maaliskuuta 2007 kello 22.06. [viitattu 9.4.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Geeni

9. Stenlund, O. Geneettiset algoritmit ja luonnossa tapahtuva mikroevoluutio [verkkodokumentti]. Helsinki: 2005 [viitattu 10.4.2007]. Helsingin teknillinen korkeakoulu, systeemianalyysin laboratorio. 29 s. Saatavissa: http://www.sal.tkk.fi/Opinnot/Mat-2.108/pdf-files/este05.pdf

10. Wikipedia, Populaatio [verkkodokumentti], 13. marraskuuta 2006 kello 22.44. [viitattu 10.4.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Populaatio

11. Portin, P. Biologinen evoluutio ja kulttuurin kehitys. [verkkolehti] Tieteessä tapahtuu 1/04, s. 7-11 [viitattu 12.4.2007]. Lehti ilmestyy myös painettuna. Saatavissa: http://www.tieteessatapahtuu.fi/0104/portin2.pdf

12. Grönroos, M. Evoluutio – mitä se on? [verkkodokumentti], 2001. [viitattu 12.4.2007]. Saatavissa: http://www.nic.funet.fi/~magi/artikk/evoluutio/maaritelma.html

13. Obitko, M. Genetic Algorithms [verkkodokumentti], 1998. [viitattu 12.4.2007]. Saatavissa: http://cs.felk.cvut.cz/~xobitko/ga/searchs.html

14. Lavonen, A. NeuroHaku - monikerroksisen perceptron- neuroverkon epälineaarinen optimointi [verkkodokumentti]. Jyväskylä: 2005 [viitattu 12.4.2007]. Pro- gradu työ. Jyväskylän yliopisto, tietotekniilan laitos. 74 s. Saatavissa: http://www.mit.jyu.fi/cheesefactory/documents/ProGraduArjaLavonen.pdf

15. Jalkanen, J. Geneettinen algoritmi putkipalkkikehän optimoinnissa [verkkodokumentti]. Tampere: 2004 [viitattu 12.4.2007]. Tutkimusraportti. Tampereen teknillinen yliopisto, teknillisen mekaniikan ja optimoinnin laitos. 67 s. Saatavissa: http://www.tut.fi/units/me/tme/henkilosto/jjalkanen/GA_raportti.pdf

16. Mahmood, A. A Hybrid Genetic Algorithm for Task Scheduling in Multiprocessor Real-Time Systems [verkkodokumentti], 2000. [viitattu 12.4.2007]. Saatavissa: http://pierre.ici.ro/ici/revista/sic2000_3/art05.html

17. Alander, J. Geneettisten algoritmien mahdollisuudet [verkkodokumentti]. Helsinki: 1998 [viitattu 12.4.2007]. Raportti. Teknologian kehittämiskeskus Tekes. 100 s. Saatavissa: http://www.control.hut.fi/Kurssit/AS-74.124/kirjallisuutta/Finnish1200.pdf

18. Luke, B. "Survival of the Fittest" Selection Procedures [verkkodokumentti], 2003. [viitattu 12.4.2007]. Saatavissa: http://members.aol.com/btluke/selproc.htm

19. Buckland, M. Genetic Algorithms in Plain English [verkkodokumentti], 2003. [viitattu 12.4.2007]. Saatavissa: http://www.ai-junkie.com/ga/intro/gat1.html

20. Wikipedia, Stochastic universal sampling [verkkodokumentti], 12. tammikuuta 2007 kello 12:17. [viitattu 12.4.2007]. Saatavissa: http://en.wikipedia.org/wiki/Stochastic_universal_sampling

21. A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing. Hollanti: Springer, 2003. 299 s. ISBN 3-540-40184-9

22. Soustruznk, K. Neural network optimisation using genetic algorithm [verkkodokumentti], 2004. [viitattu 12.4.2007]. Saatavisssa: http://nc25.troja.mff.cuni.cz/~soustruznik/work/notes/nnga.ps.gz

23. Vaasan yliopisto, 1999. Nokkamekanismin nokkamuodon optimointi geneettisellä algoritmilla [verkkodokumentti]. Vaasan yliopisto tiedottaa. [viitattu 12.4.2007]. Saatavissa: http://lipas.uwasa.fi/tiedotus/tiedotteet99/syys_2.html

24. Sengupta, A. An Evolutionary Algorithm for Locating the Critical Failure Surface in a Soil Slope [verkkodokumentti]. Kharagpur: 2005 [viitattu 12.4.2007]. Indian Institute of Technology. 15 s. Saatavissa: http://www.ejge.com/2005/Ppr0592/Ppr0592.doc

25. Byröd, M. An Approach to Solving the PTSP Problem: Contribution to the PTSP Challenge at GECCO 2005 [verkkodokumentti]. Göteborg: 2005 [viitattu 12.4.2007]. Department of Applied Mechanics, Chalmers University of Technology. 3 s. Saatavissa: http://cswww.essex.ac.uk/staff/sml/gecco/ptsp/entries/MartinByrod.pdf

26. Matthews, J. An Introduction to Coevolution [verkkodokumentti], 2000. [viitattu 12.4.2007]. Saatavissa: http://www.generation5.org/content/2000/coevolution.asp

27. Wikipedia, Kauppamatkustajan ongelma [verkkodokumentti], 17. maaliskuuta 2007 kello 17.28. [Viitattu 12.4.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Kauppamatkustajan_ongelma