Geneettiset algoritmit ohjausjärjestelmissä

Tapio Tyni ja Jari Ylinen,
Kone Elevators Research Center

Pehmeää laskentaa ja heuristiikkaa

Niin kutsuttuun pehmeään laskentaan (soft computing) luetaan mm. sumea logiikka, hermoverkot ja evoluutiomenetelmät. Viimeksimainituille on yhteistä esikuvan löytyminen luonnosta. Hollandin vuonna 1975 [Hol75] esittelemät geneettiset algoritmit (GA) ovat yksi evoluutiomenetelmien haara.

Geneettiset algoritmit voidaan luokitella myös heuristiseksi menetelmäksi. Kombinatoristen ongelmien yhteydessä heuristisuudella tarkoitetaan menetelmää, joka etsii hyviä ratkaisuvaihtoehtoja kohtuullisella laskentatyöllä, mutta ei takaa ratkaisujen mielekkyyttä tai optimaalisuutta, eikä välttämättä pysty kertomaan kuinka lähellä optimia löytynyt ratkaisu sijaitsee [Ree95].

Kombinatoriset optimointitehtävät

Käytännön optimointitehtävät ovat usein niin monimutkaisia, että matemaattisten menetelmien soveltaminen on mahdotonta. Analyyttiset ratkaisumenetelmät edellyttävät optimoitavalta kohdefunktiolta tiettyjä ominaisuuksia joita voi olla vaikeata täyttää - epälineaarista tehtävää ei esimerkiksi pystytä palauttamaan lineaariseksi tai kohdefunktiossa voi olla epäjatkuvuuskohtia. Reaalimaailmassa optimointikohdetta ei välttämättä ole edes olemassa matemaattisessa muodossa, vaan se voi olla vaikkapa tietokonepohjainen simulointimalli kohdejärjestelmän käyttäytymisestä.

Kombinatoristen ongelmien tapauksessa voidaan ajatella käytettävän raakaa laskentakapasiteettia: lasketaan suoraviivaisesti kaikki mahdolliset vaihtoehdot ja valitaan niistä paras. Käytännössä tätä voidaan käyttää vain yksinkertaisilla optimointitehtävillä. Kombinatorisille ongelmille on luonteenomaista ratkaisujoukon räjähdys: mahdollisten ratkaisujen lukumäärä kasvaa hyvin nopeasti - vähintäinkin eksponentiaalisesti - ongelman koon kasvaessa. Edes nykyisten tietokoneiden tarjoama laskentakapasiteetti ei mahdollista ongelman ratkaisua hyväksyttävässä ajassa.

Tarvitaan siis muita keinoja tämäntyyppisien ongelmien käsittelemiseksi. Satunnaishaulla voidaan esimerkiksi haarukoida hakuavaruutta generoimalla ratkaisuvaihtoehtoja tasaisesti jakautuneilla satunnaisluvuilla ja valitsemalla syntyneistä ratkaisuista paras tai jatkamalla hakua kunnes tyydyttävä ratkaisu löytyy tai käytettävissä oleva aika loppuu. Menetelmä on yksinkertainen, mutta koska se ei mitenkään hyödynnä kohdefunktion ominaisuuksia, on se usein tehoton.

Geneettinen algoritmi

Luonnossa tapahtuva evoluutio on synnyttänyt kaikki eliöt ja se muokkaa parhaillaankin maanpäällistä elämää: lajit kehittyvät, uusia lajeja syntyy ja vanhoja katoaa. Vaikka Darwin esitteli evoluutioteoriansa jo vuonna 1859 [Dar1859] on sen toiminta luonnossa pystytty todistamaan tieteellisesti vasta aivan hiljattain. Lähteessä [Wei94] kerrotaan tästä mielenkiintoisesta tutkimustyötä Galapagos-saarilla. Yli kahdenkymmenen vuoden työn tuloksena tutkijat pystyivät näyttämään toteen evoluution vaikutukset saarilla eristetyksissä elävissä sirkkuyhdyskunnissa.

Geneettisissä algoritmeissa simuloidaan tietokoneella kooltaan *pienen populaation* käyttäytymistä kun populaatioon vaikuttaa samat voimat kuin luonnonkin evoluutiossa: *valintapaine*, *risteytys ja mutaatio*. Geneettisen algoritmin toimintaa selitetään nk. *rakennuspalikka-hypoteesilla*: yhdistelemällä hyviä ratkaisuja on mahdollista saada vielä parempia ratkaisuja. Myös luonnossa näyttää mekanismi olevan tällainen: vanhempien ominaisuudet periytyvät ja voivat yhdistyä niin, että jälkeläiset pystyvät paremmin kilpailemaan rajallisista resursseista ja sen myötä todennäköisemmin selviytymään ja tuottamaan jälkeläisiä. Näin edulliset geenit selviytyvät ja leviävät koko populaatioon.

Kuvassa 1 on esitetty geneettisen algoritmin toimintaperiaate. Algoritmi on helppo toteuttaa millä tahansa ohjelmointikielellä. Toiminta koostuu viidestä pääkohdasta. Geneettisen haun lähtökohdaksi luodaan *satunnainen populaatio*. Tämä vastaa edellä kerrottua satunnaishakua, sillä tavoitteena on tuottaa ratkaisuvaihtoehtoja tasaisesti ympäri hakuavaruutta. Edellä kuvatun satunnaishaun vaiheet loppuvat *ratkaisuvaihtoehtojen evaluointiin*. Geneettinen algoritmi sen sijaan saa tietoa myös kohdefunktion ominaisuuksista *valit-*

```
geneettinen_algoritmi
begin

  g = 0; /*aloitussukupolvi*/
  luo_satunnainen_populaatio(g)

  repeat

     evaluoi_ratkaisut(g)

     g = g + 1 /* seuraava sukupolvi */
     valitse_parhaat_ratkaisut(g)
     risteytä_valittuja_ratkaisuja(g)
     mutatoi_joitakin_geenejä(g)

  until(lopetus)

  palauta_ongelman_ratkaisu(g)

end
```

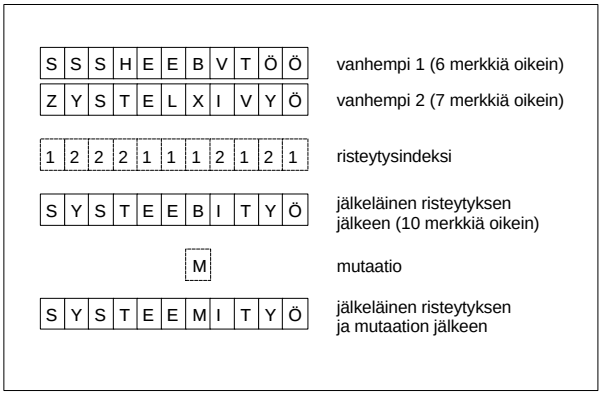
Kuva 1. Geneettinen algoritmi

semalla parhaat ratkaisuvaihtoehdot eli kromosomit ja luomalla seuraavan sukupolven *risteyttämällä* näitä kromosomeja. Uusi jälkeläinen luodaan valitsemalla kromosomin sisältämiä geenejä satunnaisesti molemmilta vanhemmilta. *Mutatoimalla* pienellä todennäköisyydellä populaation geenejä tuotetaan populaatioon diversiteettiä ja palautetaan mahdollisesti puuttuvia tai kadonneita geenien arvoja. Tämä on tärkeää heti etsinnän alkuvaiheen jälkeen, koska populaatio homogeenisoituu eikä risteytys yksin pysty tuottamaan uusia ratkaisu-vaihtoehtoja.

Kuvassa 2 on populaatiosta otettu kaksi eri ratkaisuvaihtoehtoa eli kromosomia. Kromosomi edustaa siis kohdefunkton parametrivektoria, ts. sen sisältämät geenit ovat ongelman parametreja. Esimerkin kuvassa etsitään oikeaa merkkijonoa. Risteytysoperaattori valitsee satunnaisesti geenin jommalta kummalta vanhemmalta.

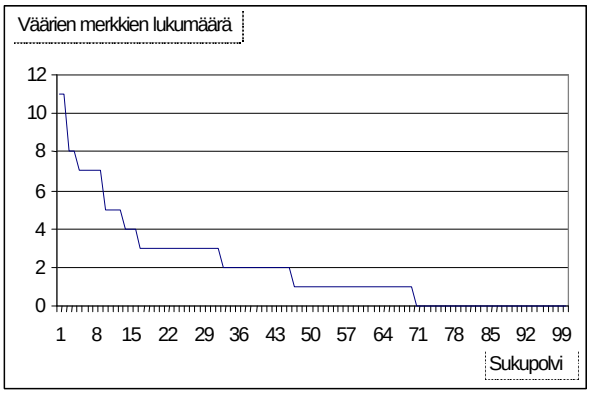
Nämä vanhemmat ovat olleet sukupolvensa parhaimmistoa, koska
ne ovat olleet siinä joukossa joka on valittu muodostamaan seuraavaa sukupolvea.

Risteytyksen jälkeen on jälkeläisessä jo 10 merkkiä oikein. Onnekas mutaatio asettaa vielä viimeisenkin virheellisen merkin kohdalleen. Esimerkkimme on luonnollisesti turhan onnekas, mutta antaa kuvan geneettisten operaatioiden toiminnasta: ratkaisut jalostuvat sukupolvien kuluessa valinnan, risteytyksen ja mutaatioiden myötävaikutuksesta.



Kuva 2. Risteytyksen ja mutaation vaikutus

Kuvassa 3 näkyy geneettisen haun eteneminen. Kyseessä on kuvan 2 merkkijonon etsintäesimerkki. Tavoitteena on siis minimoida väärien merkkien lukumäärää. Algoritmille on ominaista haun hyppäyksittäinen eteneminen: populaatio voi olla pitkiäkin aikoja juuttuneena paikalliseen ääriarvoon ennen kuin se onnekkaan mutaation ja risteytyksen tuloksena pääsee etenemään seuraavaan ääriarvoon.



Kuva 3. Geneettisen haun eteneminen

Huolimatta näennäisen yksinkertaisesta toimintaperiaatteesta ei algoritmia hallita teoreettisesti kovinkaan hyvin. Tyypillisesti esitetyt teoriat pätevät rajoitetuissa olosuhteissa ja sisältävät yksinkertaistuksia ja oletuksia - se kaiken kokoava teoria odottaa yhä löytäjäänsä. Alan julkaisuille on myös ominaista sovelluspainotteisuus ja koesuuntautuneisuusGeneettisten algoritmien käyttö ohjausjärjestelmissä

Vaikka geneettisillä algoritmeilla saadaankin ratkaistua hankalia kombinatorisia ongelmia tarvitaan silti edelleen huomattava määrä laskentaa. Etsinnän aikana evaluoitavien ratkaisuvaihtoehtojen lukumäärä saadaan suoraan kertomalla populaation koko tarvittavien sukupolvien lukumäärällä. Tyypillinen populaation koko on sata kromosomia. Kun sitä kehitetään esimerkiksi sadan sukupolven ajan joudutaan tekemään kaikkiaan 10000 kohdefunktion evaluointia. Olettakaamme, että kohdefunktion evaluointi kestää yhden millisekunnin. Tällöin koko etsintä kestää noin 10 sekuntia. Tästä on helppo nähdä, että vasteaikavaatimukset rajoittavat algoritmin soveltamista reaaliaikaisiin ohjaus- ja säätöjärjestelmiin.

Tarvittavan laskennan takia ylivoimaisesti suurin osa geneettisten algoritmien sovellutuksista ovat erilaiset off-line –tyyppiset sovellukset, joissa optimointi tehdään kerran eikä laskenta-aika ole kriittinen tekijä. Tyypillisiä sovelluksia ovat esimerkiksi materiaalin kulutuksen optimointi, kappaleen muodon optimointi, aikataulutukseen liittyvät optimointitehtävät, sijoittelutehtävät, jakelutehtävät jne.

Vaikka reaaliaikaisuus onkin suhteellinen käsite, niin vasteaikavaatimusten vuoksi geneettisten algoritmien rooli ohjaus- ja säätöjärjestelmissä on yleensä olla varsinaisen ohjaus- tai säätösilmukan ulkopuolella. Sopivia kohteita geneettiselle algoritmille ohjaus- ja säätöjärjestelmissä ovat esimerkiksi kohdejärjestelmän identifiointi, parametrien optimointi, ohjausjärjestelmän rakenteen optimointi, kohdejärjestelmän mallin rakenteen suunnittelu (esim. hermoverkon rakenteen optimointi), säätölakien suunnittelu, säätöratojen suunnittelu, laitteiston sijoittelu laitokselle jne.

Jos algoritmia pystytään käyttämään suoraan osana ohjaus- tai säätösilmukkaa, voi se tuottaa kyseessä olevalle kohdejärjestelmälle fysikaalisesti mahdottomia tai muuten käyttökelvottomia ratkaisuja. Näitä voidaan eliminoida kahdella tavalla. Ratkaistavan kohdefunktion parametrien arvojoukkoja voidaan yksinkertaisesti rajoittaa ohjattavan järjestelmän vaatiman tilanteen mukaan. Tämä ei aina auta, sillä silti on mahdollista, että etsinnän aikana generoituu ratkaisuvaihtoehtoja jotka rikkovat kohdefunktion rajoitteita. Tällaisissa tapauksissa voi kohdefunktio palauttaa sopivan sakon, jolloin geneettinen algoritmi tulkitsee kyseisen kromosomin huonoksi ratkaisuvaihtoehdoksi ja se karsiutuu pois tulevista sukupolvista.

Haun tuloksena syntyvien päätösten tai ohjaussuureiden stabiilisuus voi muodostua ongelmaksi. Tavallisesti ohjaus- ja säätöjärjestelmien laskentasilmukka toimii tietyllä näytteenottotaajuudella: tasaisin aikavälein tuotetaan järjestelmän tilan ja asetusarvojen perusteella uudet ohjaussuureiden arvot. Kun ohjattava järjestelmä on monimutkainen niin sen tila-avaruudessa ei välttämättä ole yhtä selkeätä globaalia ääriarvoa, vaan sen sijaan monia, yhtä hyviä tai lähes yhtä hyviä, sinänsä käyttökelpoisia lokaaleja ääriarvoja. Koska geneettisen algoritmin toiminta on stokastista ei mikään takaa, että tällaisessa tilanteessa toistuvassa silmukassa tehty etsintä päätyisi joka kerta samaan ääriarvoon eli samaan ongelman ratkaisuun. Jos asiaan ei kiinnitetä huomiota on tällainen järjestelmä epävakaa eikä toimi odotetulla tavalla.

Parempaan vasteaikaan pyrittäessä voidaan geneettistä algoritmia nopeuttaa rinnakkaistamalla laskentaa. Populaation evaluointivaiheessa voidaan laskenta jakaa useamman prosessorin kesken, jolloin tarvittava laskenta-aika pienenee lähes prosessoreiden lukumäärään verrannollisena. Itse algoritmia on myös kehitetty monin tavoin. Monet esitetyistä menetelmistä tuovat kuitenkin itse algoritmiin sovellusriippuvuuksia, mitä ei aina voi pitää toivottavana asiana. Lähteessä [TY99a] on esitetty sovellusriippumaton keino nopeuttaa hakua. Menetelmä lyhentää etsintään kuluvaa aikaa seuraavaksi esiteltävässä ohjaussovelluksessa noin 65 prosenttia, ollen avaintekijä kyseisen sovelluksen käytännön implementoinnille.

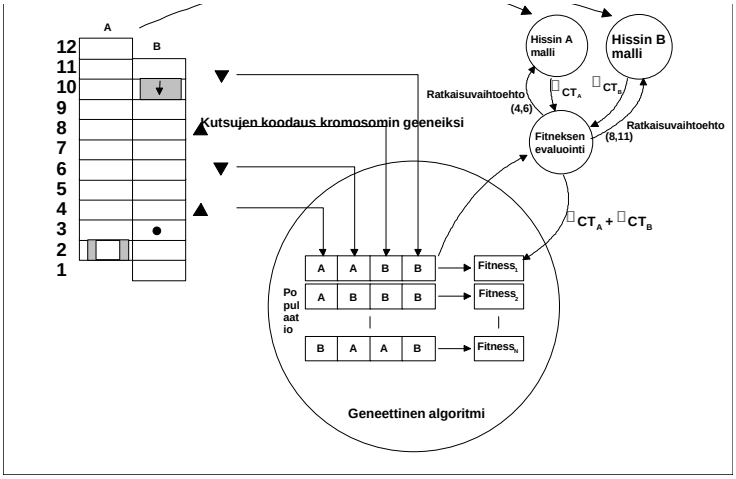
Esimerkki: hissiryhmän ohjaus

Eräs ensimmäisiä ja toistaiseksi harvalukuja geneettisiin algoritmeihin perustuvia teollisia reaaliaikaisia ohjaussovelluksia on Kone Elevatorsilla kehitetty hissiryhmän ohjaus [TY99b]. Esimerkissä näkyy erinomaisesti myös geneettisen algoritmin käyttöön liittyvä ongelman koodausperiaate. Algoritmi itsessään on sovellusriippumaton: GA-ydin käsittelee ”neutraaleja” kromosomeja ja kohdefuntion palauttamaa, kromosomin edustaman ratkaisuvaihtoehdon hyvyyttä. Käsillä oleva ongelma pitää pystyä koodaamaan mielekkäällä tavalla kromosomin geeneiksi. Geneettsisen algoritmin parametrit (populaation koko, sukupolvien lukumäärä, mutaatiotodennäköisyys jne.) ovat sovellusriippuvia, mutta menetelmä on onneksi varsin robusti niiden suhteen. Lähteessä [Ala98] on käsitelty algoritmin parametrien vaikutusta ja valintaa.

Kerroksessa odottava matkustaja tilaa hissin ylös- tai alasnapeilla riippuen siitä, kumpaan suuntaan hän on matkalla. Hissiryhmän ohjaimen tehtävänä on allokoida matkustajien painamat kutsut hisseille niin, että matkustajat saisivat mahdollisimman hyvää palvelua. Ilman erityisjärjestelyjä ei yksittäisistä matkustajista saada tietoa ja matkustajan odotusajan sijasta voidaan minimoida kutsuaikoja. Yksinkertaiselta näyttävä tehtävä paljastuu hankalaksi kombinatoriseksi ongelmaksi, jossa eri reittivaihtoehtojen lukumäärä kasvaa eksponentiaalisesti kutsujen lukumäärän mukaan. Kaikki reittivaihtoehdot pystytään laskemaan systemaattisesti vain aivan pienissä hissiryhmissä.

Kuvassa 4 on esitetty ohjauksen toimintaperiaate kun hissien optimaaliset reitit etsitään geneettisellä algoritmilla.

Kuvan esittämässä tilanteessa hissi A seisoo ovi auki kerroksessa 2 ja hissi B on kerroksessa 10 matkalla alaspäin kerrokseen 3, jonne sillä on matkustajan antama korikutsu. Kerroksissa 4 ja 8 ovat mat-



Kuva 4. Hissiryhmän ohjaus geneettisellä olgoritmilla.

kustajat antaneet kutsut ylöspäin ja kerroksissa 11 ja 6 on alaspäin meneviä matkustajia.

Reitinmuodostusongelma on koodattu seuraavasti: geenin sijainti kromosomissa (*lokus* GA-terminologiassa) kertoo mistä kutsusta on kysymys kun taas geenin arvo (*alleeli*) kertoo mikä hissi kyseisen kutsun palvelee. Kukin kromosomi eli ratkaisuvaihtoehto evaluoidaan hissiryhmän simulaatiomallissa. Kuvassa populaation ensimmäisessä kromosomissa on ratkaisuvaihtoehto {A,A,B,B}. Tämä tulkitaan siten, että hissi A palvelee kutsut 4 ja 6 ja hissi B palvelee kutsut 8 ja 11. Kromosomin edustama ratkaisuvaihtoehto dekoodataan ja viedään hissimalleille, jotka ajavat tarjotut kutsut käyttäytyen mahdollisimman todenmukaisesti lähtien liikkeelle hissiryhmän nykytilasta. Geneettisen algoritmin ytimelle palautetaan ratkaisun hyvyyttä (*fitness*) vastaava kokonaiskutsuaika *CT*. Kun populaatioon sovelletaan aikaisemmin esiteltyjä geneettisiä operaatioita on ongelman ratkaisu saatavissa viimeisen sukupolven parhaasta kromosomista, joka olisi kuvan tapauksessa {B,B,A,A}.

Saatujen tulosten mukaan uusi ohjausmenetelmä lyhentää matkustajien odotusaikoja, rakennuksesta ja liikennetilanteesta riippuen, 10-20 prosenttia verrattuna tähän asti käytössä olleeseen ohjaimeen. Tulosta voidaan pitää merkittävänä parannuksena. Lisäksi korit muun muassa liikkuvat keskimäärin tyhjempinä, mikä osaltaan myös parantaa matkustusmukavuutta.

Kohdejärjestelmän mallin avulla työskenneltäessä ongelman abstraktiotaso nousee. Tämän ansiosta järjestelmää on myös helpompi muokata ja ylläpitää. Sen sijaan, että yritettäisiin *a priori* laatia mutkikasta päätöksenteon sääntökantaa selviytymään lukuisista kohdejärjestelmäspesifisistä tilanteista voidaan keskittyä tekemään mahdollisimman hyvin järjestelmän käyttäymistä kuvaava simulaatiomalli ja antaa geneettisen algoritmin kaivaa tämän mallin avulla ongelman ratkaisu esiin.

Malliin voidaan lisätä myös useampia optimointikohteita. Hissiryhmäohjausesimerkissämme malli voi palauttaa mainitun kutsuajan lisäksi vaikkapa ratkaisuvaihtoehdon vaatiman energiakulutuksen. Tämän avulla voidaan pyrkiä ohjaamaan hissiryhmää siten, että energian kulutus ja matkustajien palvelu ovat sopivalla tavalla tasapainossa. Geneettiset algoritmit soveltuvatkin hyvin myös tämäntyyppisiin monitavoiteoptimointitehtäviin.

Yhteenveto

Kiinnostus evoluutiolaskentaan ja geneettisiin algoritmeihin lisääntyy kaiken aikaa. Syyt tähän ovat ilmeisiä: geneettinen algoritmi on yksinkertainen implementoida, robusti käyttää ja tehokas löytämään hyviä ratkaisuja vaikeisiinkin optimointitehtäviin.

Tähän asti valtaosa geneettisten algoritmien sovelluksista on ollut erilaisia offline sovelluksia, mutta tarjolla olevan laskentakapasiteetin kasvun, GA-tietämyksen leviämisen ja itse geneettisten menetelmien kehittymisen myötä nähdään tulevaisuudessa varmasti myös enemmän sellaisia reaaliaikaisia ohjaus- ja säätösovelluksia, joissa geneettisillä algoritmeilla on aktiivinen rooli varsinaisessa ohjauksessa ja säädössä.

Edellytykset geneettisen algoritmin käytölle ovat hyvät ja menetelmän käyttöä kannattaa harkita, jos ongelma on selkeästi kombinatorinen ja siitä on olemassa tai muodostettavissa laskentamalli, jolla menetelmän luomat ratkaisuvaihtoehdot voidaan evaluoida.

Reaaliaikaisissa järjestelmissä pitää kiinnittää erityistä huomiota laskennallisesti tehokkaaseen toteutukseen kaikilla osa-alueilla: ongelman koodaukseen kromosomiksi sekä GA-ytimen että järjestelmän mallin tehokkaaseen toteutukseen. Päätösten stabiilisuus voi myös aiheuttaa erikoisjärjestelyjä - ongelma voi olla sen luonteinen, että geneettinen haku ei takaa kaikissa tilanteissa vakaita päätöksiä tai ohjaussuureita.

Viitteet ja lisätietoa

[Ala98] Alander J. T., 1998. *Geneettisten algoritmien mahdollisuudet*. Teknologiakatsaus 59/98. Tekes, Helsinki.

[Dar1859] Darwin C., 1859. *On the origin of species by means of Natural Selection*. Avenel Books, New York. Reprint of the 1859 edition, 1979.

[Hol75] Holland J. H., 1975. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor.

[Ree95] Reeves C. R., editor, 1995. *Modern Heuristic Techniques for Combinatorial problems*. McGraw-Hill, London.

[TY99a] Tyni T. and Ylinen J., 1999. *Improving the Performance of Genetic Algorithms with a Gene Bank*. Proceedings of Eurogen99, Jyväskylä, Finland (to appear).

[TY99b] Tyni T. and Ylinen J., 1999. *Procedure for allocating landing calls in an elevator group*. Australian patent AU698715, dated 18 February 1999.

[Wei94] Weiner J., 1994. *Darwinin linnut*. Otavan kirjapaino, Keuruu 1997. Englanninkielinen alkuteos *The beak of the Finch – A Story of Evolution in Our Time*.

[Emm91] Emmeche C., 1991. *Tekoelämä*. Art House. Tanskankielinen alkuteos *Det Levende Spil: Biologisk Form og Kunstigt Liv*.

[Gol89] Goldberg D. E., 1989. *Genetic Algorithms in Search, Optimisation & Machine Learning*. Addison-Wesley, Reading.

[Haa95] Haataja J., 1995. *Optimointitehtävien ratkaiseminen*. CSC – Tieteellinen laskenta Oy, Otaniemi.

[Mit97] Mitchell M., 1997. *Introduction to Genetic Algorithms*. The MIT Press, Cambridge.

[TJ99] Taipale O. ja Jurva E., 1999. *Muuttujien valinta ja mittausaineiston muodostaminen*. Teknologiakatsaus 66/99. Tekes, Helsinki.

http://liiwww.ira.uka.de/bibliography/Ai/genetic.html

Tapio Tyni ja Jari Ylinen,
Kone Elevators Research Center