

Memeettiset algoritmit

Jyri Leskinen
jyril@cc.jyu.fi

22.4.2006

Tiivistelmä

Memeettiset algoritmit ovat eräs stokastisten globaalien etsintäheuristiikkojen luokka, jossa perinteisiä evoluutiopohjaisia algoritmeja tehostetaan lokaaleilla hauilla. Tässä raportissa esitellään memeettiset algoritmit, sekä niihin läheisesti liittyvät käsitteet evoluutioalgoritmit ja lokaalit haut. Esimerkitapauksessa käydään läpi QAP-ongelman ratkaisu memeettisen algoritmin avulla ja todetaan, että memeettiset algoritmit ovat tehokkaita ratkaisemaan tämäntyyppisiä ongelmia.

1 Johdanto

Memeettiset algoritmit ovat eräs stokastisten globaalien etsintäheuristiikkojen luokka, jossa perinteisiä evoluutiopohjaisia algoritmeja tehostetaan lokaaleilla hauilla. Memeettiset algoritmit ovat osoittautuneet erittäin tehokkaiksi monilla optimoinnin osa-alueilla. [3]

Memeettisille algoritmeille on kirjallisuudessa annettu lukuisia erilaisia nimiä, kuten **hybridiset geneettiset algoritmit** (*hybrid GAs*), **baldwinilaiset evoluutioalgoritmit** (*Baldwinian EAs*), **lamarckilaiset evoluutioalgoritmit** (*Lamarckian EAs*) ja **geneettiset lokaalit etsintäalgoritmit** (*genetic local search algorithms*). [3]

2 Evoluutioalgoritmeista

Tässä luvussa esitellään memeettisten algoritmien historiaa ja niihin liittyviä käsitteitä.

2.1 Historiaa

Evoluutioalgoritmit perustuvat Charles Darwinin evoluutioteorian soveltamiseen tietojenkäsittelyssä. Jo vuonna 1948 Alan Turing esitteli **geneettisen eli evoluutiohaun** ja vuonna 1962 Hans-Joachim Bremermann toteutti tietokoneella optimointitehtäviä evoluution ja rekombinaation avulla. 1960-luvulla evoluutioalgoritmeja kehitettiin edelleen. Kolme eri ryhmää Yhdysvalloissa ja Saksassa kehittivät **evoluutio-ohjelmoinnin** (*evolutionary programming*), **geneettiset algoritmit** (*genetic algorithms*), sekä **evoluutios-trategiat** (*evolutionary strategies*). Nämä alueet kehittyivät toisistaan erillään, kunnes 1990-luvun alkuun mennessä niitä alettiin pitää saman teknologian, **evoluutiolaskennan** (*evolutionary computation*) eri esitysmuotoina. 1990-luvun alussa kehitettiin vielä neljäs menetelmä, **geneettinen ohjelmointi** (*genetic programming*). [3]

Evolutiivisten järjestelmien toiminta voidaan kiteyttää kahteen perusvaikutukseen, jotka ohjaavat evoluutiota: 1) Variaatio (rekombinaatio ja mutaatio) synnyttää tarvittavan diversiteetin, joka puolestaan mahdollistaa uudet ratkaisut. 2) Valinta ja rajoitteet, jotka varmistavat sen, että hyvät ratkaisut tulevat vallitseviksi. [3]

Begin

```
ALUSTA populaatio satunnaisilla ratkaisuehdokkailla;
```

```
TESTAA jokaista ehdokasta;
```

```
Repeat Until ( LOPETUSEHTO on saavutettu ) Do
```

```
  VALITSE vanhemmat;
```

```
  RISTEYTÄ vanhemmista muodostetut parit;
```

```
  MUTATOI jälkeläistä;
```

```
  TESTAA uusia ehdokkaita;
```

```
  VALITSE yksilöt seuraavaan sukupolveen;
```

```
EndDo
```

End.

Kuva 1: Evoluutioalgoritmin toimintaperiaate.

2.2 Toimintaperiaate ja käsitteet

Evoluutioalgoritmissa ongelman ratkaisuja kutsutaan yleensä **fenotyypeiksi**. Ne on koodattu **genotyypeihin**, joita evoluutioalgoritmi käsittelee.

Evoluutioalgoritmin **kuvaus** (*representation*) voidaan esittää relaationa $R = (P, G, M)$, joka määrittelee fenotyypin P ja genotyypin G välisen suhteen. M on funktio, jonka määrittelyjoukko on G ja maalijoukko on P . Se toimii kuvauksen ”tulkkina”. Esimerkkinä valittakoon fenotyypin P reaalilukuavaruus \mathbf{R} . Reaaliluvut voidaan koodata binäärisiksi genotyypin $G = \{0, 1\}^+$. Funktio M on tällöin Graykoodaus, joka tulkitsee binäärijonot reaaliarvoiksi.

Alkuperäisen ongelmanratkaisun yhteydessä **ratkaisuehdokas**, fenotyyppi ja **yksilö** ovat pisteitä mahdollisten ratkaisujen avaruudessa, **fenotyypin avaruudessa**. Evoluutioalgoritmin puolella genotyyppi, **kromosomi** ja yksilö ovat pisteitä **genotyypin avaruudessa**, jossa evolutiivinen etsintäoperaatio toteutetaan. Yksilön elementeillä on useita nimityksiä, kuten **muuttuja** tai **geeni**. Muuttujan arvoa kutsutaan usein **alleeliksi**.

Evaluointifunktio (*evaluation function*) toimii valinnan perusteena, ja se määrää yksilön laadun. Samalla se määrittää, mitä yksilön parantamisella tarkoitetaan. Evoluutiolaskennassa sitä kutsutaan usein **hyvyysfunktioksi** (*fitness function*). Optimointiongelmaa ratkaistessa kyseessä on tyypillisesti **objektifunktio**.

Populaatio on mahdollisen ratkaisujen joukko. Useimmissa evoluutioalgoritmeissa alkupopulaatio koostuu satunnaisesti muodostetuista yksilöistä. Populaation **diversiteetillä** tarkoitetaan populaation erilaisten ratkaisujen määrää.

Parinvalinnan tarkoituksena on valita populaatiosta paremmat yksilöt seuraavan sukupolven vanhemmiksi. **Vanhemmalla** tarkoitetaan yksilöä, joka on valittu tuottamaan uusi jälkeläinen. Evoluutiolaskennassa vanhemman valinta perustuu yleensä todennäköisyyksiin, eli paremmilla yksilöillä on suurempi todennäköisyys tulla valituiksi vanhemmiksi. Usein kuitenkin huonommillekin yksilöille annetaan pieni mahdollisuus, sillä muuten algoritmi saattaa jäädä jumiin paikalliseen optimiin diversiteetin kuihtuessa.

Luonnonvalinnan avulla valitaan laadun perusteella ne yksilöt, jotka siirtyvät seuraavaan sukupolveen. Valintaa sovelletaan sen jälkeen, kun valittujen vanhempien jälkeläiset on muodostettu. Koska populaation koko on yleensä vakio, vain osa yksilöistä valitaan. Toisin kuin parinvalinnan tapauksessa, luonnonvalinta on tyypillisesti ennalta määrättyä.

Variaatio-operaattoreilla muodostetaan uusia yksilöitä vanhoista. Operaattoreita on kaksi, **mutaatio** ja **rekombinaatio**. Mutaatiota sovelletaan joka kerta vain yhteen yksilöön kerrallaan ja se on aina satunnaista. Rekom-

binaatiota (risteytystä) sovelletaan yleensä kahteen yksilöön kerrallaan. Mutaation tapaan rekombinaatio perustuu satunnaisuuteen, eli ei ole ennalta määrätty, kumman vanhemman geeni siirtyy jälkeläiseen. [3]

3 Lokaaleista hauista

Lokaali haku on etsintämetodi, joka iteratiivisesti käy läpi nykyisen ratkaisun naapurustoa ja korvaa ratkaisun sen paremmalla naapurilla, jos sellainen löytyy. Lokaali haku voidaan jakaa kolmeen pääkomponenttiin: [3]

- **Tukisääntö** (*pivot rule*) määrää kriteerin, jonka mukaan uusi ratkaisu valitaan. Esimerkiksi mäennousumenetelmässä uudeksi ratkaisuksi valitaan piste, jonka arvo on naapuruston suurin.
- **Syvyys** (*depth*) määrää, kuinka pitkälle iteraatiota jatketaan.
- **Naapuruston muodostava funktio** (*neighborhood generating function*) määrää pisteet, joihin lokaali haku voi siirtyä.

4 Memeettiset algoritmit

Puhtaat evoluutioon perustuvat algoritmit perustuvat kokonaan mutaatioon ja valintaan, mistä johtuen ne usein konvergoivat hitaasti kohti optimiratkaisua. Tästä syystä on kehitetty lukuisia erilaisia hybridimenetelmiä, joissa evoluutioalgoritmeihin yhdistetään muita optimointi- ja etsintämenetelmiä. **Memeettiset algoritmit** on eräs uusimmista tällaisista menetelmistä. Termin ”memeettinen algoritmi” esitti Pablo Moscato vuonna 1989 ja se viittaa algoritmin yhtäläisyyksiin meemi-käsitteen kanssa. [1] [3]

4.1 Meemi

Kirjassaan *Geenin itsekkyyks* (The Selfish Gene, 1976) Richard Dawkins esitteli termin **meemi**, joka tarkoittaa ”jäljittelyn yksikköä”. Esimerkiksi sävelmiä, ideoita, iskulauseita tai vaatemuotia voidaan pitää meemeinä, jotka siirtyvät ihmisen mielestä toiseen. Meemi on joiltain osin analoginen geenin kanssa, eli sitä voidaan pitää eräänlaisena ”kulttuurin geeninä”. Toisin kuin geenit, meemit eivät kehity pelkästään satunnaisesti mutaation ja luonnonvalinnan avulla, vaan niiden muodostumiseen liittyy myös informaatiota itse ongelmasta ja ne kykenevät siirtämään hankittuja ominaisuuksia jälkeläisilleen (lamarckismi). Tämä mahdollistaa huomattavasti nopeamman kehityksen

verrattuna geeneihin, mikä on selvästi havaittavissa verrattaessa ihmisen kulttuurin kehitysnopeutta luonnolliseen evoluutioon. [1]

4.2 Memeettisten algoritmien toimintaperiaate

```
Begin
  ALUSTA populaatio;
  TESTAA jokaista kandidaattia;
  Repeat Until ( LOPETUSEHTO on saavutettu ) Do
    VALITSE vanhemmat;
    RISTEYTÄ ne jälkeläistä varten;
    TESTAA jälkeläistä;
    PARANNA jälkeläistä paikallisella haulla;
    MUTATOI jälkeläistä;
    TESTAA jälkeläistä;
    PARANNA jälkeläistä paikallisella haulla;
    VALITSE yksilöt seuraavaa sukupolvea varten;
  EndDo
End.
```

Kuva 2: Yksinkertainen memeettinen algoritmi.

Meemin käsite soveltuu parhaiten sellaisiin hybridievoluutioalgoritmeihin, joissa yhteen tai useampaan vaiheeseen sovelletaan yksittäisen yksilön tehostamista. Yksinkertaisimmillaan tämä tarkoittaa paikallisen haun soveltamista yksilöihin, jotka on muodostettu mutaation tai rekombinaation avulla. [3]

Toisin kuin perinteisten evoluutioalgoritmien tapauksessa, jossa genotyyppit edustavat ratkaisuja tiettyyn ongelmaan, meemit edustavat ”strategioita” ratkaisujen parantamiseksi. Memeettisten algoritmien vahvuus piilee meemien kyvyssä parantaa olemassaolevia ratkaisuehdokkaita. [3]

Vaikka memeettisiin algoritmeihin voidaan sisällyttää heuristiikoita ja ongelma-kohtaista tietoa, mitään ”maagisia ratkaisuja” kaikkiin ongelmiin ne eivät ole, vaan niiden suunnittelussa tulee olla huolellinen. [3]

Algoritmin ennen aikainen konvergenssi johonkin paikalliseen optimiin on tyypillinen evoluutioalgoritmien ongelma. Memeettisille algoritmeille, jotka käyttävät apunaan paikallista hakua, ongelma voi olla erityisen hankala. Siksi onkin kehitetty lukuisia menetelmiä, joilla populaation diversiteetti ja kyky löytää uusia optimeja voidaan säilyttää. Esimerkiksi rajoittamalla hyvien yksilöiden määrää populaatiossa varmistetaan se, etteivät ratkaisut ole liian lähellä toisiaan. [3]

Luultavasti tärkein osa memeettisen algoritmin suunnittelussa on siirtymäoperaattori, joka määrää mihin naapuriin paikallinen haku voi siirtyä. Oikean operaattorin valinnalla saadaan algoritmin tehokkuutta parannettua huomattavasti. [3]

5 Malliesimerkki

Tässä luvussa verrataan erään memeettisen algoritmin tehokkuutta kolmeen muuhun evoluutioalgoritmiin QAP-ongelman ratkaisemisessa.

5.1 QAP (Quadratic Assignment Problem)

QAP-ongelma on seuraavanlainen:

Olkoon valittu n toimijaa ja n paikkaa, joihin toimijat voidaan sijoittaa. Jokaiselle paikkaparille on määritetty keskinäinen etäisyys ja virtaus (esim. tavaroiden kulku palvelujen välillä). Ongelmana on sijoittaa kaikki toimijat eri paikkoihin niin, että etäisyyksien ja virtausten keskinäisten tulojen summa saadaan minimoitua. [4]

5.2 Algoritmi

```
procedure QAP_MA;
begin
  initialize population P;
  foreach individual i e P do i := Local-Search(i);
  repeat
    for i := 1 to #recombinations do
      select two parents i_a, i_b e P randomly;
      i_c := Recombine(i_a, i_b);
      i_c := Local-Search(i_c);
      add individual i_c to P;
    endfor;
    P := Select(P);
    if P converged then
      foreach individual i e P \ {best} do i := Local-Search(Mutate(i));
    endif
  until terminate = true;
end;
```

Kuva 3: Esimerkin tehtävässä käytetty algoritmi. [2]

5.3 Yksilöiden ja populaation esitystapa

Valitussa tapauksessa permutaatio on koodattu genotyyppiin niin, että alleeli j paikassa i tarkoittaa, että toimija j on sijoitettu kohtaan i . Yksilö A kuvassa 4 esittää ratkaisua, jossa toimija 2 on sijoitettu paikkaan 1, toimija 4 paikkaan 2 jne. Alkuperäinen populaatio koostuu satunnaisesti valituista ratkaisuista. [2]

5.4 Lokaali haku

Ongelman ratkaisussa käytettiin niin kutsuttua 2-opt-heuristiikan varianttia, jossa kaksi elementtiä vaihdetaan keskenään. [2]

Yksilö A:	2	4	7	1	8	9	3	5	6
Yksilö A':	2	4	9	1	8	7	3	5	6

Kuva 4: Haun eteneminen.

5.5 Rekombinaatio-operaattori

Paikat:	1	2	3	4	5	6	7	8	9
Vanhempi A:	2	4	7	1	8	9	3	5	6
Vanhempi B:	7	4	5	8	3	9	1	2	6
	-	4	-	-	-	9	-	-	6
	2	4	7	-	-	9	-	5	6
	2	4	7	8	3	9	1	5	6
Jälkeläinen:	2	4	7	8	3	9	1	5	6

Kuva 5: Jälkeläisen muodostaminen rekombinaatiolla.

Rekombinaatio toteutettiin seuraavasti: Ensimmäiseksi ne toimijat, jotka sijaitsevat samoissa paikoissa molemmissa vanhemmissa, siirretään suoraan jälkeläiseen (eli toimijat 4, 9 ja 6). Seuraavaksi, aloittaen satunnaisesti valitusta paikasta (tässä tapauksessa paikasta 3) toimija valitaan, jälleen satunnaisesti, jommastakummasta vanhemmasta ja siirretään jälkeläiseen (esimerkissä toimija 7). Nyt muut toimijat tulee valita niin, ettei epäsuoraa mutaatiota pääse tapahtumaan. Asettamalla vanhemman A toimija 7 kohtaan 3 estetään vanhemman B toimijan 5 asettaminen samaan kohtaan kuin A:ssa. Asettamalla toimija kohtaan 5 vaatii sen, että myös toimija 2 tulee sijoittaa

jonnekin, eli tässä tapauksessa genomissa kohtaan 1. Koska B:n kohdassa 1 toimija on 7, joka on jo sijoitettu jälkeläiseen, voidaan valita seuraava toimija jälkeläisen oikeanpuolimmaiseen tyhjään paikkaan. Esimerkissä vanhemman B toimija 8 on sijoitettu jälkeläiseen. Mutaation välttämiseksi toimija 1 sijoitettiin kohtaan 7 ja toimija 3 kohtaan 5. Algoritmi päättyy, sillä kaikki toimijat on sijoitettu jälkeläiseen. [2]

5.6 Mutaatio-operaattori

Mutaatio tapahtuu vaihtamalla toimijoita pareittain keskenään, kunnes saavutetaan tietty, ennalta määritelty etäisyys vanhemman ja jälkeläisen välille. [2]

Vanhempi:	2	4	7	1	8	9	3	5	6
Vaihdetaan 9 ja 4:	2	9	7	1	8	4	3	5	6
Vaihdetaan 4 ja 1:	2	9	7	4	8	1	3	5	6
Vaihdetaan 1 ja 3:	2	9	7	4	8	3	1	5	6
Jälkeläinen:	2	9	7	4	8	1	3	5	6

Kuva 6: Mutaation eteneminen.

5.7 Valinta ja diversiteetin kasvattaminen

Parinvalinta suoritettiin täysin satunnaisesti ilman parempien yksilöiden suosintaa. Yksilönvalinta suoritettiin valitsemalla parhaat jälkeläiset vanhempien ja jälkeläisten poolista huolehtien samalla siitä, että uudessa populaatiossa sama fenotyyppi esiintyy vain kerran. [2]

5.8 Tulosten vertailu

Algoritmia verrattiin kolmeen erittäin tehokkaaseen evoluutioalgoritmiin, robustiin tabuetsintäalgoritmiin (Ro-TS), reaktiiviseen tabuetsintäalgoritmiin (Re-TS) ja nopeaan muurahaisyhdyskunta-algoritmiin (FANT). Testauksessa havaittiin, että esitelty memeettinen algoritmi on useimmissa tapauksissa tehokkain. Lisäksi menetelmä osoittautui hyvin robustiksi, sillä se löysi parhaat tunnetut ratkaisut kaikissa testiajoissa lyhyillä keskimääräisillä suoritusajoilla. [2]

6 Yhteenveto

Memeettiset algoritmit ovat lupaava evoluutioalgoritmien luokka, joka on osoittautunut hyvin tehokkaaksi monilla evoluutiolaskennan osa-alueilla. Koska kyseessä on kuitenkin heuristinen menetelmä, joka hyödyntää ongelmas- ta saatavaa informaatiota ongelmanratkaisussa, on tärkeää, että algoritmien hyvään suunnitteluun kiinnitetään riittävästi huomiota tehokkaan ratkaisun aikaansaamiseksi.

Viitteet

- [1] Pablo Moscato, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts. Towards Memetic Algorithms*, saatavilla WWW-muodossa <URL: <http://www.densis.fee.unicamp.br/moscato/papers/bigone.ps>>, viitattu 23.4.2006.
- [2] Peter Mertz, Bernd Freisleben, *A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem*, Proceedings of the 1999 International Congress of Evolutionary Computation, 6.-9.7.1999, saatavilla WWW-muodossa <URL: <ftp://ftp.informatik.uni-siegen.de/pub/papers/pmerz/cec99.ps.gz>>, viitattu 23.4.2006.
- [3] W.E. Hart, N. Krasnogor, J.E. Smith, *Memetic Evolutionary Algorithms*, kirjassa Recent Advances in Memetic Algorithms (W.E. Hart, N. Krasnogor, J.E. Smith, toim.), Springer-Verlag, 2004, s. 3–27.
- [4] Useita kirjoittajia, *Quadratic assignment problem*, Wikipedia, saatavilla WWW-muodossa <URL: <http://en.wikipedia.org>>, viitattu 23.4.2006.