Monte Carlo -menetelmä ja pelitekoälyt

Kalle Viiri

Referaatti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 18. syyskuuta 2015

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Kalle Viiri

Työn nimi — Arbetets titel — Title

Monte Carlo -menetelmä ja pelitekoälyt

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Referaatti	18. syyskuuta 2015	3

Tiivistelmä — Referat — Abstract

Tiivistelmä.

Avainsanat — Nyckelord — Keywords
Monte Carlo, puuhaku, tekoäly

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Peliteoria ja pelipuu

Pelillä tarkoitetaan peliteoriassa ongelmaa, jossa yksi tai useampi toimija vaikuttavat päätöksillään lopputulokseen, jolla on jokin määritelty arvo kullekin pelaajista. Peli on kombinatorinen, kun se täyttää seuraavat ehdot [1]:

i. Nollasummaisuus, eli kaikkien mahdollisten lopputulosten arvo pelaajille on yhteensä nolla. Tämä tarkoittaa, että yhden pelaajan saadessa paremman lopputuloksen muiden pelaajien lopputulos huononee.

ii. Täydellinen informaatio, eli tarkka pelitilanne on jatkuvasti kaikkien pelaajien tiedossa.

iii. Determinismi, eli peliin ei sisälly satunnaistekijöitä.

iv. Vuorottainen, eli pelaajat valitsevat ja suorittavat toimintansa vuorotellen.

v. Diskreetti, eli peliä pelataan erillisissä vuoroissa reaaliaikaisuuden sijaan.

Esimerkkejä kombinatorisista peleistä ovat *shakki* ja *go*. Kombinatoriset pelit ovat tekoälylle erinomainen tutkimus- ja soveltamisala, sillä niissä on usein yksinkertaiset säännöt, joiden päälle rakentuu kuitenkin edistynyttä strategiaa [1].

Kombinatorinen peli on helppo mallintaa pelipuuna. Pelipuun juurisolmu kuvaa pelin alkutilannetta, ja jokaisen solmun lapset pelitilanteita jotka solmun kuvaamasta pelitilanteesta voi saavuttaa. Pelin lehtisolmut ovat lopputilanteita, joissa pelin lopputulos määräytyy. Pelipuuna mallinnettua peliä voi käsitellä tavallisilla puuhakualgoritmeilla, kuten leveyssuuntaisella haulla [2].

Perinteinen tapa toteuttaa pelitekoälyjä on *minimax*-haku, jossa tekoäly olettaa kaikkien pelaajien pyrkivän maksimoimaan oman voittonsa toisten pelaajien pelatessa optimaalisesti [2]. Tällainen tekoäly valitsisi siis mielummin varmasti tasapeliin johtavan siirron kuin sellaisen, jolla voi vastustajan valinnoista riippuen joko voittaa tai hävitä.

Minimax-haun ongelmana on se, että vain puun juurisolmujen arvo pelaajille on tiedossa. Yksinkertaisissa peleissä, kuten 3x3-ristinollassa tämä

ei ole vielä ongelma, mutta vaikkapa shakin pelipuu haarautuu aivan liian runsaasti jotta hakua voitaisiin jatkaa juurisolmuihin asti [2]. Jotta seuraava siirto saataisiin valittua järkevässä ajassa on haku katkaistava ennen lopputilannetta ja käytettävä arviota keskeneräisen pelin edullisuudesta pelaajille eli heuristiikkaa [2].

Yksinkertainen heuristiikka shakkipeliin voi olla esimerkiksi nappuloiden lukumäärän laskeminen. Tätä heuristiikkaa käyttävä tekoäly osaisi valita tilanteita, joissa se saa materiaalietumatkaa, mutta edistyneemmät heuristiikat osaisivat painottaa tilanteita myös nappuloiden arvon ja sijainnin mukaan. Heuristiikkojen ongelmana on, että hyvän heuristiikan muodostaminen vaatii kehittäjältään paljon ymmärrystä pelistä. Joihinkin peleihin, esimerkiksi gohon, ei tunneta heuristiikkoja joilla tekoäly olisi kilpailukykyinen vahvoja ihmispelaajia vastaan [1].

2 Monte Carlo -puuhaku

Monte Carlo -puuhaussa (*Monte Carlo Tree Search*, jatkossa MCTS) pelipuuta laajennetaan iteratiivisesti ja käsitellään leveyssuuntaisen etsinnän sijaan simuloimalla satunnaisilla siirroilla tilanteesta pelin etenemistä loppuun saakka [1]. Menetelmän perusajatus on, että tarpeeksi suurella satunnaisotoksella voidaan arvioida kunkin siirron todellista hyötyä pelaajalle [1].

Menetelmässä puun solmut pitävät yllä tietoa odotusarvostaan, eli niiden kautta kulkeneiden simulaatiopelien tuloksista, ja siitä kuinka usein niitä on tutkittu [1]. Pelipuuta laajennetaan valitsemalla solmu, jolla on vielä tutkimattomia lapsia. Solmun valintaa voi painottaa sen odotusarvon ja tutkimiskertojen mukaan, jolloin algoritmi käyttää vähemmän aikaa jo todennäköisesti huonojen tapausten tutkimiseen ja keskittyy uusien, kannattavien peliliikkeiden etsimiseen [1].

Kun valittua solmua laajennetaan, sen lapsi tai lapset liitetään pelipuuhun, ja samassa yhteydessä simuloidaan satunnaisilla siirroilla peli uudesta lapsitilasta loppuun saakka. Pelin lopputilanteen arvo pelaajalle palautetaan pelipuussa takaisin juureen. Jokainen matkalla juureen oleva solmu käyttää simulaation tulosta oman odotusarvonsa päivittämiseen [1].

MCTS valitsee siirron, kun haun laskenta-aika on kulutettu loppuun tai haku halutaan keskeyttää muusta syystä [1]. Tämän jälkeen on vielä valittava, mikä siirto lopuksi valitaan. Eri malleja seuraavan siirron valintaan on useita. Voidaan valita esimerkiksi siirto, jolla on korkein odotusarvo, tai siirto jota on tutkittu eniten ja jonka odotusarvo on siten todennäköisesti lähimpänä totuudenmukaista [1].

3 Erityisiä hyötyjä

MCTS on algoritmina erittäin helppo soveltaa monenlaisiin ongelmiin. Koska se ei tarvitse heuristiikkaa, tekoälyn ohjelmoijan ei tarvitse tietää pohjalla olevasta pelistä muuta kuin perussäännöt [1]. Heuristiikan liittämisellä pelitilanteiden arviointiin voidaan kuitenkin saavuttaa parempia tuloksia myös MCTS:n kanssa [1].

MCTS ylläpitää arviota siirtojen kannattavuudesta reaaliaikaisesti [1]. Haun voi siis keskeyttää koska tahansa ja saada silti kelvollisen ratkaisun. Pidemmän laskenta-ajan antaminen MCTS:lle parantaa kuitenkin tuloksen laatua, sillä algoritmilla on enemmän aikaa suorittaa simulaatioita ja siten etsiä parempia toimintamalleja [1]. Ajankäyttö on myös tehokasta, sillä puuta voi laajentaa epätasaisesti painottaen lupaavia solmuja [1].

Lähteet

[1] Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. ja Colton, S.: *A Survey of Monte Carlo Tree Search Methods.* Computational Intelligence and AI in Games, IEEE Transactions on, 4(1):1–43, March 2012, ISSN 1943-068X.

[2] Russell, Stuart J. ja Norvig, Peter: *Artificial intelligence : A Modern Approach.* Pearson Education, Upper Saddle River, N.J., cop. 2010., ISBN 978-0-13-207148-2. Previous ed.: Upper Saddle River, N.J.: Prentice Hall, 2003.