Kanditutkielma

Tietojenkäsittelytieteen kandiohjelma

Monte Carlo hakupuu

Sami Rinkinen

1.11.2021

Matemaattis-luonnontieteellinen tiedekunta

Helsingin yliopisto

Yhteystiedot

PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

Sähkopostiosoite: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

Sisällys

1 Johdanto

Monte Carlo hakupuu (eng. Monte Carlo tree search, lyh. MCTS) on tekniikka, jolla luodaan hakupuu jollekin kohteelle käyttäen satunnaisuutta apuna. MCTS hyviä ominaisuuksia on sen yleinen käytettävyys ja kohteiden laaja kirjo. Kohteena voi olla melkein mikä tahansa ongelma joka voidaan mallintaa puuna. Suosittuja kohteita joihin MCTS on käytetty, ovat pelit ja eteenkin lautapelit kuten Go.

MCTS on aktiivisesti tutkimuksen kohteena jossa tutkitaan sen soveltuvuutta erilaisiin peleihin tekoälynä, yleisenä tekoälynä joka pystyy pelaamaan mitä vain peliä ja moneen muuhun ongelmaan joka voidaan mallintaa puuna. Osa tutkimuksista keskittyy myös itse MCTS algoritmin arviointiin, paranteluun ja erilaisiin muokkauksiin joiden saadaan parempia tuloksia joissakin kohteissa verrattuna yksinkertaiseen MCTS toteutukseen. Myös suuria saavutuksia on MCTS avulla saatu kuten AlphaGo, jossa MCTS ja koneoppimista yhdessä käyttäen saatiin aikaiseksi tekoäly joka voitti yhden parhaista Go pelaajista vuonna 2016.

Tässä aineessa on tarkoitus tutustua ja ymmärtää miten MCTS toimii, mihin se perustuu ja mihin sitä käytetään eteenkin peleissä. Tämä toteutetaan katsauksena eri tutkimuksista ja niissä käytetyistä tekniikoista ja saaduista tuloksista. Yritetään saada jonkinlainen kokonaiskatsaus tämän hetkisestä MCTS tutkimuksesta ja soveltuvuudesta peleissä pelaamaan tämän hetkisiä parhaimpia tekoälyjä vastaan ja mahdollista soveltuvuutta pelaamaan ihmistä vastaan. Selvitetään myös millaisia ongelmia ja ratkaisuja on jouduttu tekemään jotta MCTS saadaan toimimaan halutussa kohteessa ja miten tämä toteutus eroaa yksinkertaisesta MCTS toteutuksesta.

Luvussa kaksi käydään läpi hieman niitä perusteita joihin MCTS tekniikka perustuu ja miten MCTS toimii vaihe vaiheelta hakupuuta rakentaessa. Tarkoitus on saada mahdollisimman hyvä kuva MCTS ominaisuuksista ja toiminnasta. Luvussa kolme esitetään erilaisia kohteita ja millaisia muokkauksia MCTS voi vaatii jotta sitä pystytään soveltamaan näihin kohteisiin. Luvussa neljä selvitetään tutkimuksista MCTS käyttöä peleissä ja millaisia ongelmia on jouduttu ratkomaan ja millaisia tuloksia on saatu.

2 Monte Carlo hakupuu

Tässä luvussa käydään läpi MCTS toiminta ja ominaisuudet, jotka tekevät siitä kiinnostavan kohteen tekoälynä. Käydään läpi myös UCB1 ja UCT algoritmien toimintaa joita usein käytetään MCTS tutkittavien oksien valintaan. Näiden algoritmien on tarkoitus maksimoida oksista saatava hyöty kuitenkaan unohtamatta tutkia uusia oksia. Käydään kuitenkin aluksi läpi muutamia MCTS kannalta erittäin tärkeitä perusteita.

2.1 MCTS taustatietoa

Monte Carlo on menetelmä, jossa ongelmaa simuloidaan satunnaisesti ja lopputulosten keskiarvolla saadaan arvio kohteen todellisesta arvosta. Suuremmalla määrällä simulointeja saadaan keskiarvo joka on lähempänä todellista arvoa. Peleissä siirtojen arvon mahdollisimman tarkan arvioinnin takia simuloinnissa kannattaa painottaa jo hyviä tuloksia saaneita siirtoja sen sijaan, että simuloinnin tekisi täysin satunnaisesti.

Monikätisen rosvon ongelma (eng. Multi-armed bandit problem) on ongelma jossa pitää valita tarjolla olevista vaihtoehdoista toiminto joka maksimoi palkkion. Ajatellaan, että jokainen mahdollinen toiminto on pelikone ja tavoitteena on voittaa mahdollisimman suuri määrä. Jokaisen pelikoneen satunnaisjakauma on tuntematon ja ainoa tapa arvioida satunnaisjakaumaa on pelata kutakin pelikonetta ja arvioida saatujen tulosten perusteella jakaumaa. Tässä ongelmana on tasapainottaa milloin pelata uusia, mutta vielä huonoilta vaikuttavia pelikoneita ja milloin ennestään jo hyviä tuloksia antanutta pelikonetta. Tämä tasapainottelu on tärkeää, jotta parhaat pelikoneet voidaan löytää mahdollisimman nopeasti.

2.2 MCTS Toiminta

MCTS toiminta perustuu siihen, että tutkitaan siirtoja satunnaisella simuloinnilla pelin loppuun asti jonka jälkeen saatu tulos päivitetään puuhun. Usein myös painotetaan hyvältä vaikuttavien siirtojen tutkimista, jotta saataisiin mahdollisimman tarkka arvio näiden siirtojen todellisesta arvosta. Tämän takia MCTS puut ovat monesti epäsymmetrisiä. MCTS

algoritmi koostuu neljästä eri vaiheesta jotka esitellään seuraavaksi ja havainnollistetaan kuvalla.

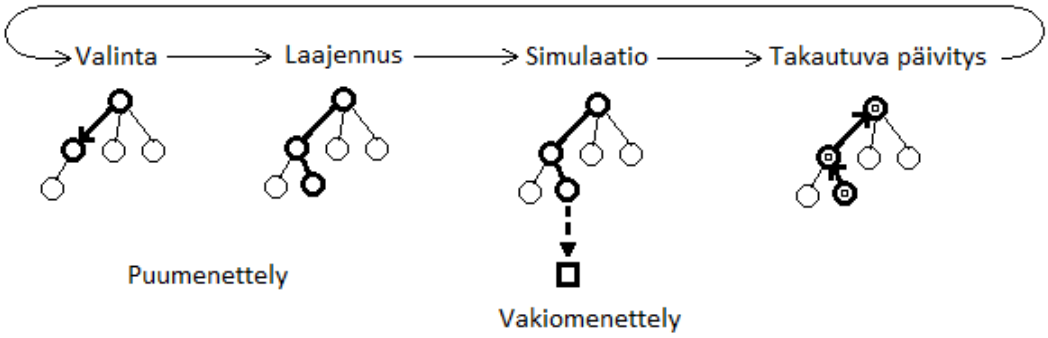
Valinta (eng. Selection): Valintavaiheessa edetään puuta läpi siirto siirrolta kunnes päästään siirtoon jota halutaan tutkia. Tämä siirron valinta voidaan ajatella monikätisen rosvon ongelmana eli pyritään valitsemaan sillä hetkellä tärkein tutkittava siirto.

Laajennus (eng. Expansion): Nyt edellisessä vaiheessa valitusta siirrosta valitaan jokin sen tarjoama uusi siirto. Tämä uusi siirto lisätään puuhun.

Simulointi (eng. Simulation): Nyt edellisessä vaiheessa lisätystä uudesta siirrosta simuloidaan peli loppuun asti. Tämä simulaatio voi olla täysin satunnainen jolloin simulaatiossa tehtyjen siirtojen taso aina pelin loppuun asti on todennäköisesti heikkotasoisia. Simulaatiossa on myös mahdollista käyttää jotain ennalta toteutettua heuristiikka jonka perusteella simulaation siirtoja valitaan. Tälläisen heuristiikan tekeminen vaatii kuitenkin usein syvällistä ymmärrystä tutkittavasta pelistä.

Vastavirta (eng. Backpropagation): Kun simulaatio on suoritettu siitä saatu tulos eli voitto, häviö tai mahdollinen tasapeli päivitetään vastavirtaan eli takautuvasti aina juureen asti. Päivitettävät siirrot ovat siis siirrot laajennusvaiheessa valitusta siirrosta valintavaiheen siirtoon ja tästä juureen asti. Päivitetty tulos eri siirroilla on kyseisen siirron siihen mennessä saatujen tulosten arvo ja siihen lisätään nyt saatu tulos.

Näillä neljällä vaiheella MCTS puu luodaan ja sen siirroille saadaan arvioita niiden todellisista arvoista. Kun jokin laskennallinen raja on saavutettu tai halutaan vain saada jokin tulos, niin MCTS suoritus keskeytetään ja saadaan paras lapsisolmu. Yleisesti hyviä valintoja parhaan lapsisolmun valintaan on maksimoiva lapsi joka siis maksimoi palkkion eli pelin tapauksessa voiton mahdollisuuden. Tutkituin lapsi on siirto jota on tutkittu eniten. Lapsisolmu joka on vierailluin ja maksimoi voiton, jos tälläistä lapsisolmua ei ole vielä lyötynyt niin jatketaan etsimistä kunnes saavutetaan tarpeeksi suuri vierailu määrä. Turvallinen lapsi on lapsisolmu joka maksimoi alemman luottamusrajan.



2.3 MCTS Ominaisuudet

MCTS on anytime-algoritmi eli sen suoritusta voidaan jatkaa niin kauan kuin halutaan. Se myös antaa aina parhaan tuloksen tutkituista siirroista vaikka suoritus olisikin keskeytetty ennen aikojaan. Pidempi suoritusaika kuitenkin tuottaa lähes aina paremman tuloksen, koska suurempi määrä simulaatioita usein antaa tarkemman arvion siirron todellisesta arvosta. Jokainen iteraatio MCTS suorittaa simulaation ja saa siitä tuloksen. Tämä saatu tulos päivitetään vastavirtaan aina juureen asti ja se takaa että kaikkien siirtojen arviot ovat ajantasalla. Koska MCTS puu on kokoajan päivitetty parhailla arvioilla, niin siitä voidaan milloin vain pyytää antamaan sen hetken parhaan arvion siirto.

MCTS ei tarvitse ennakkotietoa kohteesta tai heuristiikka jonka avulla arvottaa tilanteita. Se tekee MCTS erinomaisen valinna kohteisiin joihin ei ole tiedossa juurikaan heuristiikkaa tai mahdollinen heuristiikka ei ole kovin hyvä tasoinen vastustajiin nähden. Voidaan myös käyttää kohteisiin joihin on olemassa jo heuristiikka ja sitä voidaan jopa käyttää MCTS apuna valinta, laajennus ja simulaatio vaiheissa. Heuristiikan käytön hyöty kuitenkin on tapauskohtaista ja se usein on huomattavasti hitaampaa kuin satunnainen simulaatio. Kuitenkin heuristiikka usein tuottaa huomattavia hyötyjä tehokkuudessa. Huonona puolena heuristiikan käytössä voidaan pitää sitä, että se voi rajoittaa siirtoja joita MCTS tutkii, koska heuristiikka voi pitää joitain siirtoja huonoina vaikka todellisuudessa ne olisivat hyviä. Tämä johtuu siitä, että heuristiikkat ovat usein ihmisen tekemiä ja sisältävät oletuksia.

Epäsymmetrisyys on yleistä MCTS puissa sillä halutaan tutkia lupaavia siirtoja huomattavasti enemmän kuin siirtoja jotka ovat antaneet huonoja tuloksia. Tutkiminen myös usein painottuu lupaaviin siirtoihin sillä halutaan saada mahdollisimman tarkka arvio näiden siirtojen todellisista arvoista. Kuitenkin välillä tutkitaan myös ei niin lupaavia siirtoja

jotta voidaan löytää oikeasti parhaat siirrot. Muuten voitaisiin joutua tilanteeseen, jossa paras siirto jää tutkimatta, koska sitä ennen on löytynyt hyviä siirtoja joita jäädään tutkimaan. Näin ollen siis minkään siirron todennäköisyys tulla tutkituksi ei saa mennä nollaan. Epäsymmetrisyys siis usein antaa tietoa siitä mitkä siirrot ovat hyviä, sillä niitä on tutkittu paljon ja ne ovat silti saaneet hyviä tuloksia.

Kun mietitään millaista tekoälyä pelissä halutaan käyttää, niin usein valinta on joko MCTS tai minimax. Verrataan hieman molempien etuja tässä. Tilanne jossa pelipuu laajenee valtavasti ja ei ole olemassa heuristiikkaa on ainoa vaihtoehto MCTS, sillä minimax tarvitsee heuristiikan arvottaakseen ei lopullisia pelitiloja. Sama tilanne, mutta kohteeseen on kehitetty heurisiikkoja on silloin mahdollista valita kumpi vain. MCTS voidaan käyttää myös heuristiikan kanssa ja se usein jopa auttaa MCTS tuottamaan parempia tuloksia nopeammin vaikka simulaatioita on usein vähemmän. Minimax suoritus ja tulos puolestaan riippuu täysin käytettävissä olevan heuristiikan laadusta. Jos kohteena olevassa pelissä on mahdollisesti tilanteita joissa muutamilla siirroilla peli päättyy häviöön, niin sanotusti ansa tilanteita voi MCTS usein suoriutua heikommin, koska se usein tutkii syvälle tiettyjä siirtoja. Minimax puolestaan selviytyy ansa tilanteista usein hyvin, koska se laajenee taso kerrallaan ja tutkii jokaisen tason usein tarkemmin.

2.4 UCB1 toiminta ja hyödyt

Ylempi luottamusraja 1 (eng. Upper confidence bound 1, UCB1) on algoritmi monikätisen rosvo-ongelman ratkaisemiseen. Tämä on yksi muutamista UCB algoritmeista jotka esitellään artikkelissa Auer et al., 2002. UCB1 on myös yksinkertaisin näistä algoritmeista. Se ei myöskään tarvitse ennakkotietoa jakaumista toimiakseen mikä tekee hyvän ehdokkaan käytettäväksi MCTS tukena. Vaatimuksena jakaumalle on, että palkintojen tulee olla välillä $[0, 1]$ UCB1 käytettäessä. Tämä ei monesti ole suuri ongelma, sillä jos palkinnot poikkeavat väliltä voidaan ne usein kuitenkin skaalata sopiviksi välille.

$$\text{UCB1} = \overline{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

Aluksi jokaista pelikonetta pelataan kerran. Tämän alustus vaiheen jälkeen UCB1 pelaa pelikonetta j joka maksimoi kaavan. Kaavassa $\overline{x}_j$ on keskiarvo palkinto joka pelikoneesta j on siihen mennessä saatu. Kaikkien pelikoneiden pelauskertojen määrä on n ja jonkin tietyn pelikoneen pelauskertojen määrä on n_j.

UCB1 algoritmissa vasen puoli eli $\overline{x}_j$ suosii pelaamaan aina mahdollisimman hyviä tuloksia saanutta pelikonetta. Puolestaan oikea puoli suosii kokeilemaan vähemmän pelattuja vaihtoehtoja pelikertojen kasvaessa.

Lähteet

Auer, P., Cesa-Bianchi, N. ja Fischer, P. (2002). ”Finite-time Analysis of the Multiarmed Bandit Problem”. *Machine Learning* 47, s. 235–256. DOI: 10.1023/A:1013689704352.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. ja Colton, S. (2012). ”A Survey of Monte Carlo Tree Search Methods”. *IEEE Transactions on Computational Intelligence and AI in Games* 4.1, s. 1–43. DOI: 10.1109/TCIAIG.2012.2186810.