Joonas Vilppunen

Koneoppivan tekoälyn hyödyntäminen reaaliaikaisessa strategiapelissä

Tietotekniikan kandidaatintutkielma

11. toukokuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Joonas Vilppunen

Yhteystiedot: joonas.v.v.vilppunen@student.jyu.fi

Työn nimi: Koneoppivan tekoälyn hyödyntäminen reaaliaikaisessa strategiapelissä

Title in English: Using machine learning in real-time strategy game AI

Työ: Kandidaatintutkielma

Sivumäärä: 21+0

Tiivistelmä: Tutkielman tavoitteena oli löytää koneoppimisen sovellusalueita RTS-pelien tekoälystä, sillä niillä olisi mahdollista parantaa pelikokemusta tekoälyä vastaan. Sovellusalueita löytyi rakennusjärjestyksen oppimisesta, vastustajan ennakoimisesta, pelaajien imitoimisesta sekä taistelutilanteiden ennakoimisesta. Tutkielmassa esiteltyjä sovelluksia on mahdollista käyttää pelikokemuksen parantamiseen tekoäkyä vastaan.

Avainsanat: koneoppiminen, tekoäly, RTS, pelit

Abstract: The purpose of this study was to find applications for machine learning in RTS-game artificial intelligence. Because these applications might help deliver better games against human players. Applications were found in build order learning, predicting the opponent, imitating other players and predicting battle outcomes. It is possible to use these applications to better the playing experience.

Keywords: machine learning, AI, RTS, games

Kuviot

Sisältö

1 Johdanto

Tässä tutkielmassa tehdään kirjallisuuskartoitus koneoppivasta (engl. machine learning) tekoälystä (engl. artificial intelligence) RTS- eli reaaliaikaisissa strategiapeleissä (engl. real-time strategy game). Tavoitteena on löytää reaaliaikaisista strategiapeleistä alueita, joihin koneoppimista voisi soveltaa siten, että tekoälyvastustaja olisi älykkäämpi ja antaisi paremman pelikokemuksen. Tarkastellaan myös sitä, miksi koneoppimista soveltavien tekoälyjen toteuttaminen peleihin on haastavaa.

Aiheen mielenkiintoa lisää se, että koneoppivan tekoälyn kehittämisen haasteet ovat hyvin samanlaisia myös muilla koneoppimisen sovellusalueilla (Lara-Cabrera, Cotta ja Fernandez-Leiva 2013). Tutkimustulokset ovat täten siirrettävissä muille koneoppimisen sovellusalueille kuten teollisuudessa tuotantolinjojen optiomointiin.

Luvussa 2 selitetään tutkielman keskeiset käsitteet, jotka ovat tekoäly, koneoppiminen ja reaaliaikastrategiapeli. Käsitteiden lisäksi luvussa kuvataan ja perustellaan tutkielmaa määrittävät näkökulmat, kiinnittäen erityistä huomiota StarCraft-nimiseen RTS-peliin.

Luvuissa 3 ja 4 perehdytään kahteen reaaliaikastrategiapelien osa-alueeseen, joita kutsutaan makromanageroinniksi ja mikromanageroinniksi. Molemmat osa-alueet puolestaan jakautuvat pienempiin osa-alueisiin koneoppimisen sovellusten mukaisesti.

2 Taustat

Tässä luvussa tullaan käsittelemään tutkielman keskeisimmät käsitteet. Perehdytään siihen mitä koneoppiminen on ja mitä tarkoitetaan RTS-pelillä.

Lara-Cabrera, Cotta ja Fernandez-Leiva (2013) huomauttavat, että tekoälytutkimus RTS-peleissä on melko uusi tutkimusala. Tästä johtuen kirjallisuuskatsaus alalle on mielenkiintoinen ja kuten mainittua tulokset ovat mahdollisesti hyödyllisiä RTS-pelien tekoälyn ulkopuolella.

2.1 Tekoäly

Russell ja Norvig (2003) määrittelevät tekoälyn (engl. artificial intelligence) rationaalisesti käyttäytyväksi tietokoneohjelmaksi. He kuvaavat myös muita näkökulmia tekoälyn määrittämiselle, mutta rationaalisesti käyttäytyvä tietokoneohjelma soveltuu parhaiten tähän tutkielmaan, sillä pelitekoälyn ei tarvitse vakuuttaa ihmisyydellään vaan riittää, että ulkoa päin katsottuna se toimii järkevällä tavalla. Perustana voidaan pitää älykästä agenttia, joka ottaessaan huomioon ympäristönsä toimisi parhaalla mahdollisella tavalla. Tästä lähtien tekoälypelaajaan viitataan sanalla agentti.

2.2 Koneoppiminen

Alpaydin (2014) kuvaa koneoppimisen seuraavasti: Koneoppiva ohjelma on sellainen ohjelma joka, pystyy optimoimaan suorituskriteerejään sille annetun tiedon tai aikaisemman kokemuksen perusteella. Tilastotieteiden malleja hyödyntäen voidaan luoda algoritmeja, jotka kykenevät tehokkaasti toteuttamaan määritelmässä esitetyt ehdot. Koneoppiminen sisältää useita tällaisia algoritmeja, joihin seuraavaksi keskitytään.

Keskeisin koneoppimisessa käytetty tilastotieteellinen malli, joka perustuu Bayesiläiseen tilastotieteeseen. Sen sisältämän Bayesin teoreeman avulla kyetään muodostamaan suhteita

havaintojen ja odotettujen tulosten välillä:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{2.1}$$

missä $P(A)$ on tapahtuman A todennäköisyys, $P(B)$ on tapahtuman B todennäköisyys ja $P(B|A)$ kertoo, mikä on A:n todennäköisyys silloin, kun B tapahtuu. Kaavasta saadaan yhteys tapahtumien A ja B välille, mikä on mahdolistaa tapahatumien tai ominaisuuksien luokittelun.

Eräs koneoppimisen muoto, *ohjattu oppiminen* (engl. supervised learning) pohjautuu juuri Bayesin teoreeman seurauksiin. Ohjattu oppiminen perustuu ohjelmalle annettavan valmiiksi luokitellun aineiston tulkintaan (Alpaydin 2014). Jotta ohjelma olisi niin sanotusti oppiva, tulisi sen kyetä oppimaan luokkien määritelmät annetusta aineistosta. Tällaista tilannetta voidaan kutsua myös luokitteluongelmaksi.

Alkuaineiston ollessa tarpeeksi kattava ohjelman tulisi kyetä luokittelemaan sille annettava aineisto. Vaarana on kuitenkin se, että alkuaineistosta riippuen algoritmi voi olla joko yli- tai alisovitettu (Alpaydin 2014). Tämä tarkoittaa sitä että, ohjelma on sopeutunut alkuaineiston luokitteluun liian hyvin tai huonosti. Yli- ja alisovittaminen voidaan estää kuitenkin helposti jakamalla alkuaineisto kahteen osaan: opetus- ja testiosaan. Näin mahdolliset sovitusvirheet on huomattavissa jo testiaineistoa läpikäydessä, eikä vasta silloin kun ohjelma on varsinaisessa käytössä.

Vahvistusoppiminen (engl. reinforcement learning) on koneoppimisen muoto, jossa tekoälyagentti oppii vuorovaikutuksestaan ympäristönsä kanssa, joko saamalla rangaistuksia tai palkintoja (Sutton ja Barto 1998). Vahvistusoppimisessa ei olla kiinnostuneita välivaiheista. Ainoastaan lopputuloksella on merkitystä.

Geneettiset algoritmit lainaavat luonnosta ja yrittävät emuloida evoluutiota (Mitchell 1998). Algoritmien toimintaa voidaan kuvata seuraavanlaisesti. Muodostetaan kromosomeja, joissa kussakin sijaitsee algoritmin toiminnalle oleellisia muuttujia. Nämä kromosomit voivat pariutua toisten vastaavien kromosomien kanssa tai mutatoitua. Kromosomien toimintaa arvioidaan jollakin ennalta määrätyillä kriteereillä, joka toimii ikään kuin luonnollisena valintana, parhaat kromosomit saavat pariutua enemmän kuin huonommin suoriutuneet. Näitä toimenpiteitä toistetaan kunnes saavutetaan haluttu toimivuus.

Jo yllä mainittujen algoritmien lisäksi on myös olemassa monia muita algoritmeja. Ne kuitenkin sivuutetaan tässä tutkielmassa, sillä niiden käsitteleminen ei ole tarpeen puuttuvien sovelluskohteiden vuoksi.

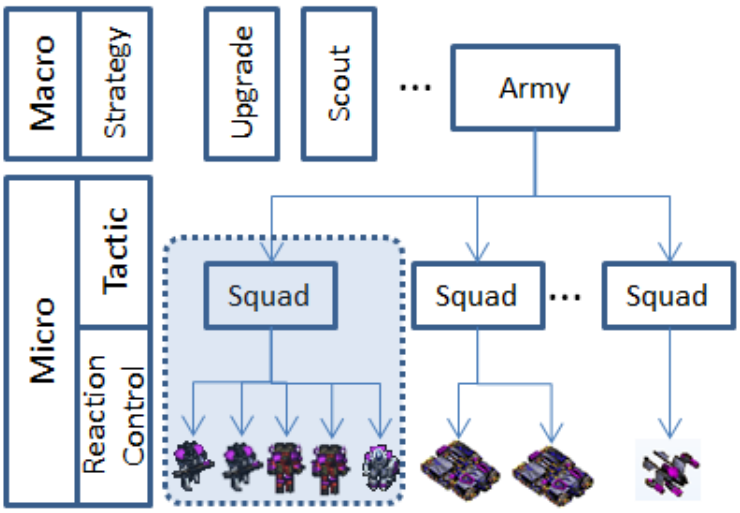
2.3 Reaaliaikainen strategiapeli

Reaaliaikainen strategia eli RTS (real-time strategy) on strategiapelien alalaji, jolle ominaista on se, että resursseja keräämällä luodaan tukikohta ja armeija, jolla yritetään tuhota vastustajan vastaava. Poiketen esimerkiksi toisesta hyvin tavallisesta koneoppivan tekoälyn tutkimuskohteesta lautapeleistä, RTS-pelissä molemmat pelaajat toimivat yhtäaikaisesti reaaliajassa, eikä kaikki informaatio ole nähtävissä. Lisäksi RTS-pelit ovat epädeterministisiä eli sama pelitapahtuma ei aina tuota samaa lopputulosta (esimerkkinä yksikköjen osumatarkkuus ei ole 100%). Näiden seikkojen vuoksi RTS-pelien tila-avaruus (engl. state space) on monta kertaa suurempi kuin esimerkiksi shakin tai gon vastaavat. Tila-avaruuden suuruudesta johtuen tavalliset menetelmät pelien ratkaisemiseen kuten pelipuun kulkeminen eivät sovellu suoraan RTS-peleihin. (Ontanon ym. 2013)

StarCraft: Brood War[1] on Blizzard Entertainmentin vuonna 1998 julkaisema RTS-peli. Pelissä on kolme erilaista pelattavaa rotua, Terran, Protoss ja Zerg, jotka eroavat pelityyliltään selkeästi. StarCraft voidaan jakaa kahteen osa-alueeseen, joita kutsutaan makro- ja mikromanageroinniksi. Kuviossa 1 on eritelty nämä osa-alueet. Makromanagerointiin kuuluu: strategia, parannukset, tiedustelu ja armeijan kokoonpano. Mikromangerointi voidaan jakaa kahteen osaan: taktiseen kontrolliin ja reaktiiviseen kontrolliin. Luku 3 käsittelee makromanagerointia ja luku 4 käsittelee mikromanagerointia.

StarCraft on valikoitunut tekoälytutkimuksen kohteeksi lähinnä siksi, että se on edelleen suosittu peli ja sitä pidetään hyvin tasapainoisena pelinä (Ontanon ym. 2013). Tekoälyn ohjaamisen helpottamiseksi on kehitetty erilaisia rajapintoja ja simulaattoreita. Esimerkkinä näistä BWAPI[2], joka antaa mahdollisuuden agentille suoraan keskustella pelin kanssa ilman tarvetta simuloida näppäimistöä ja hiirtä kuitenkin sallien ainoastaan samat toiminnot

1. http://eu.blizzard.com/en-gb/games/sc/
2. https://github.com/bwapi/bwapi



Kuvio 1. Tyypillinen tapa jakaa Starcraft (Liu, Louis ja Ballinger 2014). Makromanageroin-tiin kuuluu: strategia, parannukset, tiedustelu ja armeijan kokoonpano. Mikromangerointi voidaan jakaa kahteen osaan: taktiseen kontrolliin ja reaktiiviseen kontrolliin.

mitä ihminenkin pystyisi tekemään. Se myös mahdollistaa StarCraft-pelin uusintojen (engl. replay) muuntamisen käskylistaksi, joka sisältää kaikki pelaajien tekemät komennot, jota on helpompi tulkita koneellisesti.

3 Makromanageroinnin oppiminen

Pelin kuluessa pelaajan on tehtävä päätöksiä, jotka yhdessä muodostavat strategian, jota kutsutaan myös nimellä makromanagerointi. Nämä päätökset vaikuttavat suurelta osin pelin lopputulokseen joten on tärkeätä, että päätökset ovat järkeviä ja linjassa toistensa kanssa. Tehtäviin päätöksiin kuuluvat rakennusjärjestys, yksiköiden ja rakennusten parannukset (engl. upgrades), tiedustelu (engl. scouting) sekä armeijan hallinta. Onnistunut makromanagerointi johtaa yleensä ylivoimaiseen armeijan kokoon ja sitä kautta pelin voittoon.

3.1 Rakennusjärjestyksien oppiminen

Rakennusjärjestyksellä (engl. build order) tarkoitetaan agentin suorittamaa pelirakennuksien ja -hahmojen rakentamisjärjestystä. Rakennusjärjestys määrää hyvin pitkälti suoritettavan strategian pelin alkuvaiheessa. Kuitenkin rakennusjärjestykset voivat vaihdella hyvinkin paljon pelin edetessä vaikuttamatta pelin strategisiin päätöksiin.

Efthymiadis ja Kudenko (2013) sovelsivat vahvistusoppimista rakennusjärjestyksen oppimisessa. He loivat agentin, joka oppi suorittamaan Battlecruiser rush -nimisen rakennusjärjestyksen. Kyseisessä rakennusjärjestyksessä pyritään mahdollisimman nopeasti rakentamaan yksi Terran-rodun vahvimmista yksiköistä.

Heidän menetelmänsä perustuvat SARSA-algoritmiin (engl. State-Action-Reward-State-Action) ja STRIPS-menetelmään. SARSA koostuu viidestä vaiheesta, jotka ovat: tila, toiminta, palkinto, tila ja toiminta. Algoritmin toimintaa kuvataan kaaviossa 3.1.

Funktio Q ottaa kaksi parametria s tilan ja a toiminnan, jonka perusteella funktio laskee tilan arvon. Näitä arvoja päivitetään kaavan mukaan siten, että vanhaan arvoon lisätään oppimisnopeudella α korjattu summa, joka koostuu palkinnon r, alennuskertoimella (engl. discount factor) γ korjatun seuraavan tila-arvoparin $Q(s', a')$ arvon summasta josta poistetaan edellisen tilan arvo. Alennuskerroin vaikuttaa siihen kuinka lyhyt- tai kaukonäköinen algoritmi on oppimisen suhteen, pienet arvot laskevat tulevien oppimistilanteiden arvoa kun taas suuret kasvattavat sitä.(Alpaydin 2014) Algoritmin toimintaa voidaan parantaa ε-ahneella tutkimi-

Algorithm 1 SARSA-algoritmi

Alustetaan kaikki $Q(s,a)$ satunnaisesti

Kaikille tapahtumasarjoille tee

Alusta tila s

Valitse Q:sta toiminto a

Toista

Suorita toiminto a, havannoi palkinto r ja seuraava tila s'

Valitse Q:sta toiminto a'

Päivitä $Q(s,a)$: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$ ja $s \leftarrow s'$, $a \leftarrow a'$

Kunnes tila s on lopputila

Lopeta Kaikille

sella (engl. ε-greedy exploration), joka vaikuttaa siihen yrittääkö agentti löytää uuden tilan vai käyttääkö se jo opittua tietoa. Tämä toteutetaan yleensä todennäköisyydella $1-\varepsilon$.

Efthymiadis ja Kudenko (2013) vahvistivat tavallista SARSA-algoritmia edelleen STRIPS-menetelmällä, jossa tila-toimintoparit muutetaan suunnitelmaksi. Sen avulla voidaan laskea potentiaalit tilalle, jossa agentti on menossa ja lisätä palkinnon määrää mikäli suunta on oikea.

Kun Efthymiadis ja Kudenko (2013) vertailivat tavallista SARSA-algoritmia ja STRIPS-menetelmällä paranneltua algoritmia, he huomasivat että jälkimmäinen suoriutui merkittävästi paremmin. Sen lisäksi paranneltu algoritmi käyttää tehtävän suorittamiseen paljon vähemmän askelia, mikä tarkoitti sitä, että algoritmi kuluttaa vähemmän aikaa epäoleellisten tilojen läpikäymiseen, mikä tekee siitä tehokkaamman. Mutta he huomauttivat, että nopeutumisesta huolimatta kokeen läpikäyminen kesti noin 15 tuntia.

3.2 Vastustajan strategian ennustaminen

Vastustajan strategian ennakoiminen on tärkeää RTS -peleissä, sillä usein jokaiselle eri strategialle on olemassa yksi tai useita strategiota, jotka ovat vahvoja niitä vastaan. Täten vastustajan strategian selville saaminen antaa etulyöntiaseman. On tärkeä muistaa, että StarCraft ei ole täydellisen informaation peli kuten esimerkiksi shakki, jossa molemmat pelaajat näkevät

koko pelilaudan. StarCraft käyttää tiedon piilottamiseen niin kutsuttua *fog-of-war*-aluetta, joka estää pelaajia näkemästä suoraan toistensa tukikohtiin. Kuitenkin lähes mikä tahansa yksikkö paljastaa ympäriltään *fog-of-war*-aluetta mahdollistaen *tiedustelun*, jossa lähetetään yksikkö vastustajan tukikohtaan paljastamaan sen. Tiedustelusta saadulla tiedolla voidaan päätellä vastustajan rakennusjärjestys ja sitä kautta myös strategia.

Leece ja Jhala (2014) pyrkivät ammattilaispelaajien uusintoja tutkimalla kehittämään mallin, joka kykenee rajallisen tiedon puitteissa ennustamaan vastustajan strategian. Ongelmaa oli rajattu sisällyttämällä tutkittaviin uusintoihin vain Terran- ja Protoss-rotujen välisiä pelejä. Malli perustuu Markovin satunnaiskenttä-malliin (engl. Markov Random Field) eli MRF-malliin, joka mahdollistaa muuttujien ja niiden välisien suhteiden kuvaamisen. Leece ja Jhala (2014) määrittivät muuttujat 1253:a uusinnasta pelaajien ominaisuuksien ja tiedustelun mukaan. Malli pyrki ennustamaan strategioita määrittämällä yksikköjä ja niiden lukumääriä jonka jälkeen se kulkee mallin muuttujia ikään kuin tulevaisuuteen. Leece ja Jhala (2014) pitävät mallia lupaavana, mutta heidän mielestään malli on liian epävarma mitä kauempana ennustettava pelitila on nykyisestä tilasta, mutta se on nopea ja luotettava lyhyellä aikavälillä. He pitävät mahdollisena sitä, että vastaavanlaisesta vastustajan tilan mallintamisesta olisi merkittävää hyötyä sekä tutkijoille, pelaajille että tekoälyn kehittäjille.

Cho, Kim ja Cho (2013) lähestyivät strategian ennustamista myöskin uusintoja tutkimalla hyödyntäen päätöspuita (engl. decision tree). He kehittivät järjestelmän, joka kykeni muuttamaan uusinnat ominaisuuksilla laajennetuksi päätöspuuksi (engl. feature expanded decision tree). Tutkimuksessa käytettiin ainoastaan kahden Protoss-rodun välisiä pelejä. Uusinnoista oli kahdenlaisia versioita: sellaisia, joissa *fog-of-war*-alue oli päällä ja niitä joissa sitä ei ollut. Peleistä muodostettiin päätöspuut ja niitä verrattiin muihin yleisesti käytettyihin algoritmeihin. Päätöspuut suoriutuivat paremmin kuin muut algoritmit pelien loppuvaiheessa, mutta huonommin alkuvaiheessa. Parhaimmillaan päätöspuut onnistuivat ennustamaan strategian noin 90%:n tarkkuudella, kun taas puolestaan alkupelin heikko suorituskyky näkyy vain noin 10%:n ennustustarkkuudella. Myös *fog-of-war*-alue heikensi päätöspuiden toimintaa keskimäärin noin 20%. Cho, Kim ja Cho (2013) pitävät päätöspuita ihmisystävällisinä, mutta heikkoa alkupelin suorituskykyä tulisi parantaa tai vahvistaa käyttämällä jotakin toista algoritmia. He huomauttavat, että ennustusmallit kärsivät tekoälyn heikosta tiedustelukyvys-

tä.

4 Mikromanageroinnnin oppiminen

Mikromanagerointi on yksikköjen hienovaraista hallinnoimista taistelutilanteissa ja niiden välittömässä läheisyydessä. Liu, Louis ja Ballinger (2014) esittävät, että mikromanageroinnin voi jakaa kahteen osaan: taktiseen hallinnointiin (engl. tactical control) ja reaktiiviseen hallinnointiin (engl. reactive control). Taktisella hallinnoinnilla tarkoitetaan yksiköiden liikkeen ja sijainnin hallintaa, kun taas reaktiivisella hallinnoinnilla tarkoitetaan yksittäisten yksiköiden tarkkaa hallinnointia taistelutilanteissa. Tämä jaottelu on esitetty myös kuviossa 1. Pelin lopputuloksen kannalta onnistunut mikromanagerointi ei ole niin tärkeää kuin vahva makromanagerointi mutta onnistuneella mikromanageroinnilla on mahdollista paikata heikompaa makromanagerointia.

4.1 Pelaajien imitoiminen

Eräs tapa oppia mikromanagerointia on pelaajien imitoiminen. Young ja Hawes (2014) ja Oh, Cho ja Kim (2014) ovat tutkimalla uusintoja kehittäneet kaksi erilaista menetelmää matkia ihmis- tai tekoälypelaajien toimintoja. Young ja Hawes (2014) mainitsevat, että mikäli sovellusalueella on jo eksperttejä, ihmisiä tai tekoälyagentteja, miksi jättää niiden tiedot ja taidot hyödyntämättä. Haittapuoleksi he mainitsevat sen, että vaikka ekspertit saattavat toimia pätevästi ne eivät kuitenkaan välttämättä toimi optimaalisesti.

4.1.1 Kvalitatiiviset spatiaalisuhteet

Young ja Hawes (2014) loivat agentin, joka oppii havainnoimalla hyödyntämällä oppimismenetelmää, jonka kehittivät Montana ja Gonzalez (2011). Menetelmä toimii muodostamalla ekspertin tekemistä muutoksista ympäristöönsä oppimisjäljen jokaisella havainnointihetkellä.

StarCraft-pelin suuren tilajoukon vuoksi myös jokaisella pelin yksiköllä on suuri määrä mahdollisia tiloja, jonka vuoksi tilajoukkoa on syytä yksinkertaistaa jotenkin. Young ja Hawes (2014) sovelsivat *kvalitatiivisia spatiaalisuhteita* (engl. qualitative spatial relations) saavuttaakseen pienemmän tilajoukon ja osittain myös siksi, että yksikön liikuttaminen absoluutti-

silla koordinaateilla on herkkä virheille. Ne kutistavat useita tarpeeksi samankaltaisia tiloja yhdeksi tilaksi, jolloin tilajoukko pienenee merkittävästi. Tämän lisäksi mallin hyödyntämiseen pelin sisällä käytettiin Monte-Carlo-puuhakua palauttamaan koordinaattien *kvalitatiiviset spatiaalisuhteet* takaisin pelikoordinaateiksi.

Youngin ja Hawesin (2014) agenttia testattiin kolmessa eri mikromanagerointitilanteessa. Ensimmäisessä tilanteessa agentti kontrolloi Terran-rodun Vulture-yksikköä. Vulture-yksiköllä on pitkän kantaman hyökkäys. Sitä vastassa tilanteessa on Zerg-rodun Zergling-yksikköjä, jotka pystyvät hyökkäämään vain lähietäisyydeltä. Agentin tavoitteena on selvitä jatkuvasti lisääntyvää määrää Zergling-yksikköjä vastaan. Toisessa tilanteessa agentti kontrolloi kahta Protoss-rodun Dragoon-yksikköä kolmea Zerg-rodun Hydralisk-yksikköä vastaan. Tässä tilanteessa agentti on altavastaajan asemassa. Kolmannessa tilanteessa agentti kontrolloi Dragoon-ryhmää, joka kohtasi erilaisia vihollisryhmiä.

Heidän agenttinsa suorituskykyä verrattiin tilanteissa verrokkiagenttien lisäksi ihmispelaajiin. Oppimisaineistoa agentilla oli 150 kutakin tilannetta ja agentteja testattiin 100 kutakin tilannetta. Jokaisessa kolmessa tilanteessa verrokkiagentteja ja ihmispelaajia imitoinut agentti suoriutui heikommin kuin esimerkkinsä, mutta kuitenkin kohtuullisen hyvin. Esimerkiksi ensimmäisessä tilanteessa verrokkiagentin Vulture-yksikkö onnistui tappamaan keskiarvolta 20 Zergling-yksikköä verrattuna imitoivan agentin 17 yksikköön.

4.1.2 Vaikutuskartat

Kuten Young ja Hawes (2014), Oh, Cho ja Kim (2014) loivat agentin, joka hyödynsi uusintoja oppimiseen imitoimalla. *Kvalitatiivisten spatiaalisuhteiden* sijasta Oh, Cho ja Kim (2014) käyttivät vaikutuskarttoja (engl. influence map), jotka analysoivat yksikköjen ja kartan vaikutusta pelitilanteseen. Samassa yhteydessä pelitilanteen tapahtumat talletetaan vaikutuskartan kanssa tietokantaan. Pelissä agentti luo pelitilanteesta samanlaisen vaikutuskartan, jota se sitten vertaa vaikutuskartta-pelitapahtumatietokantaan. Mikäli agentti löytää sopivan vaikutuskartan, se lataa vaikutuskartan pelitapahtumat ja käyttää samoja komentoja mitä uusinnassa käytettiin.

Agentin tietokannassa oli 40 asiantuntijapelaajan uusintaa kahdesta eri tilanteesta. Ensim-

mäinen tilanne oli 12 vastaan 12 Dragoon-yksikköä ja toinen oli 24 vastaan 24 Dragoon-yksikköä. Agentti otteli neljää muuta agenttia vastaan, joissa se voitti keskiarvolta 83% ensimmäisen tilanteen otteluista ja 84% toisen tilanteen otteluista. Oh, Cho ja Kim (2014) pitivät agenttiaan lupaavana, sillä se voisi säästää aikaa agenttien ominaisuuksien suunnittelemisessa.

4.2 Geneettisten algoritmien hyödyntäminen

Liu, Louis ja Ballinger (2014) loivat agentin, joka hyödynsi geneettistä algoritmiä sekä vaikutus- ja potentiaalikarttoja. Potentiaalikartat eroavat vaikutuskartoista siinä, että vaikutuskartat kertovat minne yksikköjen tulisi mennä, potentiaalikartat puolestaan kertovat mitä reittiä kohteeseen pääsisi helpoiten. Vaikutus- ja potentiaalikarttojen avulla Oh, Cho ja Kim (2014) loivat geneettisen algoritmin, joka agressiivisesti suosi tehtävästään hyvin suoriutuneita kromosomeja. Tähän kromosomiin kuului karttojen lisäksi myös muita arvoja, jotka esimerkiksi määrittivät kuinka halukas se on hyökkäämään.

Agenttia testattiin kolmessa eri mikromanagerointitilanteessa tekoälykilpailuissa menestyneitä agentteja vastaan. Ensimmäisessä tilanteessa kukin agentti kontrolloi viittä Vulture-yksikköä 25 Protoss-rodun Zealot-yksikköä vastaan, joita kontrolloi StarCraft-pelin agentti. Toisessa tilanteessa agentit kontrolloivat viittä Vulture-yksikköä kuutta Vulture-yksikköä vastaan. Kolmannessa tilanteessa kukin agentti kontrolloi viittä Vulture-yksikköä viittä Vulture-yksikköä vastaan. Heidän agenttinsa suoriutui jokaisesta tilanteesta parhaiten ollen selvästi parempi toisesssa ja kolmannessa tilanteessa.

Liu, Louis ja Nicolescu (2013) lisäsivät geneettisiin algoritmeihin valmiita tapauksia keventämään algoritmin vaatimaa kuormaa. Näitä kutsutaan nimellä CIGAR: Case Injected Genetic AlgoRithm. Tässäkin työssä käytettiin vaikutuskarttoja, sekä potentiaalikenttiä määrittämään taistelutilanteita. CIGAR toimi yhtä hyvin kuin vastaava geneettinen algoritmi ilman valmiita tapauksia, ja tapauksia opittuaan CIGAR suoriutui edelleen yhtä hyvin kuin tavallinen geneettinen algoritmi, mutta löysi optimaaliset ratkaisut kaksi kertaa nopeammin.

4.3 Taistelutilanteiden ennakoiminen

Kyky ennakoida voittaja on tärkeä, sillä se auttaa välttämään taistelutilanteita joiden lopputulos on tappiollinen. Tämä kyky on tärkeä tekoälylle, sillä tappiollisiin taistelutilanteisiin joutuminen voi näyttää räikeältä virheeltä jos se pelaa ihmispelaajaa vastaan. Stanescu ym. (2013) loivat ohjelman, joka kykeni ennustamaan taistelujen lopputuloksia. Tämän saavuttamiseen käytettiin Bayesilaista verkkoa, joka on moniulotteinen ohjatun oppimisen muoto. He loivat mallin, joka kuvasi taistelutilanteen armeijan kokoa, sen tekemää vahinkoa ja kestävyyttä. Testidatan he loivat SparCraft-nimisellä simulaattorilla, joka on suunniteltu nimenomaan taistelutilanteiden simulaattoriksi. Ohjelman tarkoituksena oli selvittää voittajan lisäksi voittajan armeijan loppukokoonpano. Ohjelma ei kuitenkaan kyennyt ennustamaan voittajan armeijan tarkkaa yksikköjakoa.

5 Yhteenveto

Työssä tarkasteltiin koneoppivan tekoälyn eri sovelluskohteita RTS-peleissä. Sovelluskohteita löytyikin useilta eri pelialueilta ja niitä oltiin lähestytty monin eri näkökulmin. Tutkielmassa tutustuttiin lähemmin rakennusjärjestyksen oppimiseen, vastustajan strategian ennustamiseen, pelaajien imitoimiseen sekä taistelutilanteiden ennakoimiseen. Tutkielmassa esitelty tutkimus tuo lähemmäksi yleisen pelaaja-agentin luomisen, joka kykenisi suoriutumaan kokonaisesta pelistä ja kenties voittamaan kenet tahansa ihmispelaajan.

Efthymiadis ja Kudenko (2013) osoittivat, että vahvistusoppimisella on mahdollista luoda angentti, joka kykenee oppimaan rakennusjärjestyksen. Heidän mielestään tämä on lupaavaa, mutta vaatii työtä ennen kuin agentti pystyisi toimimaan itsenäisemmin. Ihmispelaajan näkökulmasta ajateltuna agentti, joka osaa useita rakennusjärjestyksiä on ennalta-arvaamattomampi ja sitä kautta haastavampi.

Leece ja Jhala (2014) ja Cho, Kim ja Cho (2013) onnistuivat luomaan agentteja, jotka ovat kykeneviä ennustamaan vastustajansa strategian. Vastustajan strategian ennustaminen on tärkeää, mikäli agentti haluaa näyttää uskottavalle. Strategian vaihtaminen pelin kesken tekee agentista myöskin ennalta-arvaamattomamman kuin agentin, joka ei osaisi vaihtaa strategiaansa. Nämä menetelmät kärsivät siitä, että ne tarvitsevat uusintoja peleistä, joita ei välttämättä aina ole saatavilla pelistä riippuen.

Geneettisten algoritmien hyödyntäminen mikromanageroinnissa mahdollistaa agentit, jotka käyttäytyvät enemmän kuten ihmispelaajat. Lisäksi geneettisten algoritmien iteratiivisen luonteen vuoksi yhdestä algoritmista on mahdollista saada useamman tasoisia agentteja. Mitä useampia eri tasoisia agentteja ihmispelaajilla on valittavinaan, sitä parempi pelikokemus useammalla pelaajalla on.

Stanescu ym. (2013) loivat agentin, joka pystyi ennustamaan taistelutilanteiden lopputuloksia. Taistelutilanteiden ennakoiminen puolestaan on myös tärkeä osaamisalue agentille ihmispelaajan pelikokemuksen kannalta, sillä tappiollisiin taistelutilanteisiin joutuminen on epäuskottavaa. Epäuskottavuuden lisäksi myös tämä lisää ihmispelaajan kohtaamaa haastetta pelissä agenttia vastaan. Myös tämäkin menetelmä kärsii siitä, että se tarvitsee peliuusin-

toja.

Jatkotutkimuksena voisi selvittää lisää sovelluskohteita RTS-peleissä, sekä laajentaa myös muihin peligenreihin. Jatkotutkimusta voisi myös tehdä pelien ulkopuolella siitä, kuinka esimerkiksi tässä tutkielmassa esiteltyjä menetelmiä voisi käyttää alkuperäisten sovellusalueidensa ulkopuolella, kuten teollisuuden optimoinnissa.

Lähteet

Alpaydin, Ethem. 2014. *Introduction to machine learning.* MIT press.

Cho, Ho-Chul, Kyung-Joong Kim ja Sung-Bae Cho. 2013. “Replay-based strategy prediction and build order adaptation for StarCraft AI bots”. Teoksessa *Computational Intelligence in Games (CIG), 2013 IEEE Conference on,* 1–7. Elokuu. doi:10.1109/CIG.2013.6633666.

Efthymiadis, K., ja D. Kudenko. 2013. “Using plan-based reward shaping to learn strategies in StarCraft: Broodwar”. Teoksessa *Computational Intelligence in Games (CIG), 2013 IEEE Conference on,* 1–8. Elokuu. doi:10.1109/CIG.2013.6633622.

Lara-Cabrera, R., C. Cotta ja A.J. Fernandez-Leiva. 2013. “A review of computational intelligence in RTS games”. Teoksessa *Foundations of Computational Intelligence (FOCI), 2013 IEEE Symposium on,* 114–121. Huhtikuu. doi:10.1109/FOCI.2013.6602463.

Leece, M., ja A. Jhala. 2014. “Opponent state modeling in RTS games with limited information using Markov random fields”. Teoksessa *Computational Intelligence and Games (CIG), 2014 IEEE Conference on,* 1–7. Elokuu. doi:10.1109/CIG.2014.6932877.

Liu, Siming, S.J. Louis ja C. Ballinger. 2014. “Evolving effective micro behaviors in RTS game”. Teoksessa *Computational Intelligence and Games (CIG), 2014 IEEE Conference on,* 1–8. Elokuu. doi:10.1109/CIG.2014.6932904.

Liu, Siming, S.J. Louis ja M. Nicolescu. 2013. “Using CIGAR for finding effective group behaviors in RTS game”. Teoksessa *Computational Intelligence in Games (CIG), 2013 IEEE Conference on,* 1–8. Elokuu. doi:10.1109/CIG.2013.6633652.

Mitchell, Melanie. 1998. *An introduction to genetic algorithms.* MIT press.

Montana, J, ja A Gonzalez. 2011. “Towards a unified framework for learning from observation”. Teoksessa *IJCAI Workshop on Agents Learning Interactively from Human Teachers.*

Oh, In-Seok, Ho-Chul Cho ja Kyung-Joong Kim. 2014. “Imitation learning for combat system in RTS games with application to starcraft”. Teoksessa *Computational Intelligence and Games (CIG), 2014 IEEE Conference on,* 1–2. Elokuu. doi:10.1109/CIG.2014.6932919.

Ontanon, S., G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill ja M. Preuss. 2013. “A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft”. *IEEE Transactions on Computational Intelligence and AI in Games* 5, numero 4 (joulukuu): 293–311. ISSN: 1943-068X. doi:10.1109/TCIAIG.2013.2286295.

Russell, Stuart J., ja Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach.* 2. painos. Pearson Education. ISBN: 0137903952.

Stanescu, Marius, Sergio Poo Hernandez, Graham Erickson, Russel Greiner ja Michael Buro. 2013. “Predicting Army Combat Outcomes in StarCraft”. Teoksessa *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference,* 86–92. http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/view/7381.

Sutton, Richard S, ja Andrew G Barto. 1998. *Reinforcement learning: An introduction.* MIT press.

Young, Jay, ja Nick Hawes. 2014. “Learning Micro-Management Skills in RTS Games by Imitating Experts”. *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference:* 195–201. http://www.aaai.org/ocs/index.php/AIIDE/AIIDE14/paper/view/8998.