Marko Ruotsalainen

Tekoäly reaaliaikaisissa strategiapeleissä

Tietotekniikan kandidaatintutkielma

25. toukokuuta 2015

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Marko Ruotsalainen

Yhteystiedot: marko.j.ruotsalainen@student.jyu.fi

Työn nimi: Tekoäly reaaliaikaisissa strategiapeleissä

Title in English: Artificial Intelligence in Real-Time Strategy Games

Työ: Kandidaatintutkielma

Sivumäärä: 23+0

Tiivistelmä: Reaaliaikaisille strategiapeleille ovat ominaisia monimutkaiset dynaamiset ympäristöt, joissa ei ole täsmällistä tietoa ympäristöstä. Tällaiset ympäristöt ovat haastavia tekoälylle. Aihetta kannattaa kuitenkin tutkia, sillä ongelmien ratkonta genren peleissä auttaa myös useiden vastaavien tosielämän ongelmien ratkaisemisessa. Työn tavoitteena on luoda kattava yleiskuva reaaliaikaisten strategiapelien tekoälyn ja tekoälytutkimuksen tilaan painottaen teknisiä menetelmiä. Reaaliaikaisten strategiapelien monimutkaisuudesta johtuen näitä menetelmiä on runsaasti, ja käytännössä kaikki niistä kaipaavat lisätutkimusta.

Avainsanat: tekoäly, pelit, reaaliaikaiset strategiapelit

Abstract: Complex dynamic environments with neither perfect nor complete information about the state of the environment are typical for real-time strategy games. Such environments are challenging for AI. The subject should be studied because solving the problems in games helps in solving similar problems for the similar real-life environments. The goal of the study is to create a comprehensive overall view about the state of AI and AI research among genre's games, focusing on technical methods. Due to the complexity of real-time strategy games, there are lots of these methods, practically all of which require more research.

Keywords: artificial intelligence, games, real-time strategy games

Kuviot

Sisältö

1 Johdanto

Tämän kandidaatintutkielman aiheena on tekoäly reaaliaikaisissa strategiapeleissä (engl. *real-time strategy game*, lyh. *RTS game*), joista tässä tutkielmassa käytetään englanninkielisen lyhenteen mukaisesti lyhennettä RTS-pelit. RTS-peli on peli, jossa pelaaja kerää resursseja, rakentaa tukikohtaa, kehittää teknologiaa ja hallitsee yksiköitä reaaliaikaisesti voittaakseen vastustajansa tyypillisesti jonkinlaisessa sotatilassa (Hagelbäck ja Johansson 2009). Tutkielmassa esitellään RTS-pelien tekoälyn keskeisiä osa-alueita ja haasteita pääpainopisteen ollessa tärkeimpien teknisten menetelmien yleiskatsauksessa.

Tekoälyn toteuttaminen RTS-peleissä on perinteisesti ollut haastavaa (Buro 2004), mutta tilanne on sittemmin parantunut tuntuvasti (Ontañón ym. 2013). Genrelle ovat ominaisia monimutkaiset dynaamiset ympäristöt, joissa ei ole täsmällistä tietoa ympäristöstä. Tällaiset ympäristöt ovat haastavia tekoälylle. Tosielämän esimerkkejä tällaisista ympäristöistä ovat tieliikenne, talous ja sääennusteet. RTS-pelit edustavat samankaltaisia ongelmia yksinkertaisemmassa muodossa, joten näiden ongelmien ratkaiseminen RTS-peleissä voi auttaa kyseisten ongelmien ratkaisemisessa myös muilla vastaavilla sovellusalueilla. Lisäksi ratkaisuja voidaan luonnollisesti hyödyntää peliteollisuudessa. (Ontañón ym. 2013.) Näin ollen tekoäly RTS-peleissä on tutkimusalueena kiinnostava paitsi kirjoittajan omasta mielestä myös käytännön merkityksiltään.

Herzog Zwei (Technosoft 1989) on RTS-pelien esi-isän maineessa. Sen tekoäly ei ollut järin kehittynyttä, sillä siinä viholliset jäivät jatkuvasti jumiin kunnollisen reitinhakutoiminnallisuuden puuttuessa, ja lisäksi tekoälypelaajan rakennusyksiköitä pystyi huijaamaan niin, etteivät ne pystyneet taistelemaan vastaan. Tekoäly olikin luultavasti toteutettu hyvin yksinkertaisella tilakoneella, jossa tiloja olivat rahan kerääminen, hyökkääminen ja puolustaminen. (Schwab 2009.)

Dune II: The Building of a Dynasty (Westwood Studios 1992) aloitti RTS-genren jokseenkin nykyisessä muodossaan. Pelin tekoäly ei ollut huippuluokkaa: tekoälyn eri vaiheita olivat tukikohdan rakentaminen, vastustajan tukikohdan etsiminen ja hyökkääminen. Tästä johtuen tekoälyvastustajat eivät esimerkiksi osanneet rakentaa uudestaan tuhoutuneita puolustusase-

mia. Lisäksi tekoäly huijasi suuresti. (Schwab 2009)

RTS-pelien kulta-ajalla julkaistiin monia merkittäviä pelejä kuten *Warcraft: Orcs & Humans* (Blizzard Entertainment 1994), *Command & Conquer* (Westwood Studios 1995) ja *StarCraft* (Blizzard Entertainment 1998a). Tällöin tekoälyä kehitettiin eteenpäin, ja suurin kehitys tapahtui reitinhaussa. Pelit olivat kuitenkin pääasiassa laskentatehon puutteesta johtuen edelleen täynnä porsaanreikiä, joita pelaajat pystyivät hyödyntämään tekoälyä vastaan. (Schwab 2009.)

Tutkielmassa selvitetään kirjallisuuskatsauksen keinoin, mikä RTS-pelien tekoälyn ja tekoälytutkimuksen yleinen tila on. Tutkielman tavoitteena onkin antaa kattava yleiskuva RTS-pelien tekoälyn ja tekoälytutkimuksen tilasta. Luvussa 2 esitellään RTS-pelien tekoälyn keskeiset osa-alueet ja haasteet. Luvussa 3 luodaan yleiskatsaus RTS-pelien tekoälyn keskeisiin menetelmiin.

2 Tekoälyn keskeiset osa-alueet ja haasteet RTS-peleissä

Perinteisesti RTS-pelien tekoälytutkimuksessa on tunnistettu kuusi keskeistä haastetta, jotka ovat resurssien hallinta, päätöksenteko epävarmuuden alaisuudessa, avaruudellinen ja ajallinen päättely, yhteistyö, vastustajan mallintaminen ja oppiminen sekä reaaliaikainen suunnittelu vihamielisessä ympäristössä (Buro 2004). Ontañón ym. (2013) toteavat, että osa näistä on kehittynyt huomattavasti mutta että osa on jäänyt lähes huomiotta. Ontañón ym. jatkavat edelleen, että nykyään päähaasteet ovat suunnittelu, oppiminen, epävarmuus, avaruudellinen ja ajallinen päättely, kohdealueen tietämyksen hyödyntäminen ja tehtävien ositus. Monet haasteista ovat siis edelleen samoja kuin kymmenisen vuotta sitten. Myös Millington ja Funge (2009) mainitsevat päätöksenteon yhtenä RTS-pelien tekoälyn keskeisimmistä osa-alueista, minkä lisäksi heidän mukaansa muita tärkeimpiä osa-alueita ovat reitinhaku, ryhmänä liikkuminen sekä taktinen ja strateginen tekoäly.

Kolmannen näkökannan RTS-pelien tekoälyn keskeisiin osa-alueisiin ja haasteisiin tarjoaa Schwab (2009). Hänen mukaansa RTS-peleissä tekoälyä vaativat muun muassa yksittäiset yksiköt, ekonomiset yksiköt, jotka ovat itse asiassa yksittäisten yksiköiden erikoistapaus, korkean tason tekoäly, komentajat ja muut keskitason strategiset yksiköt, tukikohdan rakentaminen, taustaelämä esimerkiksi eläimistön muodossa, reitinhaku sekä taktiset ja strategiset tukijärjestelmät, kuten esimerkiksi maastoanalyysi ja vastustajan mallintaminen. Kehityskohteiksi hän taas mainitsee oppimisen, jumiin jääneiden tekoälyn osien tunnistamisen, auttajatekoälyn, joka auttaa pelaajaa toistuvien pienten toimintojen suorittamisessa, erilaisten vastustajien persoonallisuuksien toteuttamisen ja strategisen ajattelun lisäämisen yksittäisten yksiköiden epäinhimillisen tehokkaan hallinnan sijaan.

2.1 Tehtävien ositus ja päätöksenteko

Ontañón ym. (2013) selittävät haasteita tarkemminkin. He kertovat, että tehtävien ositus tapahtuu yleensä jakamalla tehtävät strategiaan, taktiikkaan, reaktiiviseen hallintaan, maaston analysointiin ja tiedustelutiedon keräämiseen. Myös Millington ja Funge (2009) kirjoittavat jonkinlaisesta tehtävien osituksesta kertoessaan päätöksenteosta. Heidän mukaansa jo yksi-

lötasolla on jonkinasteista päätöksentekoa, kuten myös Schwab (2009) toteaa, minkä lisäksi myös keskitasolla on päätöksentekoa. Heidän mukaansa suurin osa hankalasta päätöksenteosta tapahtuu kuitenkin korkeammalla tasolla, joka käsittelee pelin kokonaistilaa: oikeiden resurssien kerääminen, teknologisen yms. kehityksen ohjaaminen ja joukkojen käskyttäminen hyökkäykseen tai puolustukseen. Millington ja Funge toteavatkin, että päätöksentekoon voi RTS-peleissä liittyä useita eri komponentteja. Edelleen Ontañón ym. toteavat, että haasteena onkin eri tehtävistä vastaavien tekoälyn osien kommunikoinnin ja yhteistyön sujuvuuden varmistaminen.

2.2 Suunnittelu ja oppiminen

Ontañón ym. (2013) kertovat, että suunnittelun haasteena on tila-avaruuden suuri koko ja käytettävissä olevien toimintojen runsaus, joten tyypilliset menetelmät kuten pelipuun haut eivät sovellu tilanteeseen kunnolla. Sen sijaan tarvitaan abstrahointia usealla eri tasolla. Suunnittelun hankaluus taas on kääntänyt huomiota kohti oppimismenetelmiä, joita on kolmenlaisia: etukäteen oppiminen esimerkiksi uusinnoista, pelin aikana oppiminen esimerkiksi vastustajasta, ja pelikertojen välillä oppiminen yhdestä pelikerrasta seuraavaan. Schwab (2009) mainitsee esimerkkejä tilanteista, joihin oppimista voisi soveltaa: Tekoäly voi lähettää useita kertoja joukkoja kuolemaan samaan tilanteeseen, mikä korjaantuisi, mikäli tekoäly kykenisi oppimaan. Vastustajan mallintaminen taas mahdollistaisi esimerkiksi pelaajan suosimien yksikkötyyppien tunnistamisen sekä pelaajan vasteajan tunnistamisen hyökkäyksen sattuessa.

2.3 Epävarmuus

Ontañón ym. (2013) kirjoittavat, että epävarmuutta RTS-peleissä luo se, ettei koko pelitilannetta yleensä ole mahdollista havainnoida. Tilannetta on kuitenkin mahdollista helpottaa tiedustelutoiminnan avulla, he jatkavat. Heidän mukaansa toinen epävarmuutta luova tekijä ovat vastustajat, sillä vastustajien toimien ennakointi on vaikeaa. Tähän luultavasti auttaisi vastustajan mallintaminen, jonka Schwab (2009) mainitsee.

2.4 Avaruudellinen ja ajallinen päättely

Avaruudelliseen päättelyyn liittyy monia maaston hyödyntämiseen liittyviä elementtejä, Ontañón ym. (2013) jatkavat: siihen liittyy rakennusten sijoittelua esimerkiksi suojaavan muurin muodostamiseksi ja tukikohdan laajentamista omaan ja vihollisten tukikohtiin nähden sopivalle alueelle. Lisäksi heidän mukaansa yksiköiden järkevä sijoittelu taistelua ajatellen on tärkeää. Ajallinen päättely taas on heidän mukaansa tärkeää hyökkäysten ja vetäytymisten sekä korkeamman tason tärkeiden päätösten ajoittamisessa, ja lisäksi tekoälyn tulee kyetä huomioimaan, etteivät kaikkien toimintojen vaikutukset ole välittömiä.

2.5 Taktinen ja strateginen tekoäly sekä keskitason komentajayksiköt

Millington ja Funge (2009) kertovat, että taktista ja strategista tekoälyä on käytetty lähinnä reitinhaun ohjaamiseen. Toinen käyttökohde on heidän mukaansa rakennusten paikkojen valinta, jossa voidaan käyttää vaikutuskarttaa (engl. *influence map*) turvallisten rakennuspaikkojen tunnistamiseen sekä esimerkiksi muurien rakentamiseen yhdessä avaruudellisen päättelemisen kanssa. Schwab (2009) mainitsee, että tukikohdan rakentamisesta päättämiseen käytetään usein jäykkiä sääntöjä, jotka eivät välttämättä sopeudu maailman muutoksiin hyvin.

Millington ja Funge (2009) jatkavat kertomalla, että taktisen analyysin käyttäminen yksiköiden suuremman mittakaavan hallintaan on ollut esillä. He antavat tästä esimerkkinä tilanteen, jossa taktisella analyysillä etsitään vihollisen muodostelmassa heikkous ja isketään siihen omilla joukoilla. Toisena esimerkkinä he antavat kaikissa RTS-peleissä käytetyn yksinkertaisen periaatteen, jonka mukaan joukot lähetetään vihollisen oletettuun sijaintiin satunnaisen etsimisen sijaan. Millington ja Funge kertovat myös, että tällä osa-alueella on mahdollista jatkaa kehitystä pidemmällekin.

Schwab (2009) kertoo, että joissain peleissä on eräänlaisia komentajayksiköitä muiden yksittäisten yksiköiden käytön tehostamiseen. Hänen mukaansa näihin yksiköihin tarvitaan keskitason tekoälyä, koska niiden tarvitsee kyetä muita yksiköitä monimutkaisempaan toimintaan, mutta toisaalta niiden toiminta ei kuitenkaan ole korkean tason strategian tasolla. Niitä voikin ajatella eräänlaisina taktisen tason yksiköinä. Schwab toteaa myös, että tämän tason toiminta

on usein puutteellisesti toteutettu pääosin koska sen toteuttaminen ja hienosäätäminen on hyvin monimutkaista. Tämä johtuu hänen mukaansa siitä, että keskitason toiminnassa tarvitaan paljon tietoa useista eri lähteistä, minkä jälkeen saatu tieto on osattava yhdistää eritasoisiin suunnitelmiin ja toimintoihin.

2.6 Reitinhaku

Millington ja Funge (2009) kertovat, että tehokas reitinhaku on ollut RTS-pelien pääasiallinen tekninen haaste. Heidän mukaansa RTS-pelien kentät ovat suuria ja ainakin pinnan alla ruutupohjaisia, haettavat reitit pitkiä ja yksiköiden määrät suuria, joten reitinhaun nopeus on hyvin tärkeää. Reittien pituuden ja yksiköiden määrän mainitsee haasteeksi myös Schwab (2009). Lisäksi Millington ja Funge jatkavat, että joissain RTS-peleissä maastonmuotoja on mahdollista muuttaa, mikä tekee reittien kattavasta etukäteen laskemisesta vaikeaa. Samantyyppisenä haasteena Schwab mainitsee muut dynaamiset elementit kuten pelaajien rakentamat muurit ja taistelun aiheuttamat rauniot. Edelleen reitinhaun haasteiksi Schwab mainitsee reitin tukkivat omat ja liittolaisten yksiköt sekä siltojen kaltaiset pullonkaulat.

2.7 Ryhmänä liikkuminen

Useimmissa RTS-peleissä käytetään nykyään jonkinlaisia muodostelmia yksittäisten yksiköiden muodostamien ryhmien liikuttamiseen, kertovat Millington ja Funge (2009). Heidän mukaansa useimmiten muodostelmassa on etukäteen määrätty määrä yksiköitä, mutta toisinaan määrä voi olla mielivaltainen. Lisäksi Millington ja Funge mainitsevat, että ympäristön rakennelmat, muodot yms. voivat vaikuttaa muodostelmaan ja että pelaajalla ei välttämättä ole tarkkaa kontrollia muodostelmaan liikkumisen aikana.

2.8 Kohdealueen tietämyksen hyödyntäminen

Kohdealueen tietämyksen hyödyntämismahdollisuudet ovat jokseenkin epäselviä tiedon suuresta määrästä huolimatta, Ontañón ym. (2013) kirjoittavat. Tyypillisiä lähestymistapoja on heidän mukaansa kaksi: olemassaolevien lähestymistapojen ohjelmointi tavalla, joka on yk-

sinkertainen toteuttaa mutta joka ei mahdollista muutoksia ajon aikana, ja automaattinen oppiminen esimerkiksi uusinnoista. Jälkimmäinen tosin on osoittautunut haastavaksi, Ontañón ym. toteavat.

3 Teknisiä menetelmiä

Tässä luvussa luodaan yleiskatsaus RTS-pelien tekoälytutkimuksen ja -kehityksen keskeisiin teknisiin menetelmiin. Joihinkin erityisen keskeisiin tai toimiviin osa-alueisiin pureudutaan myös hieman syvällisemmin.

3.1 Suunnittelu ja päätöksenteko

Chung, Buro ja Schaeffer (2005) sekä Ontañón ym. (2013) jakavat suunnittelun ja päätöksenteon kolmeen osa-alueeseen: yksittäisten yksiköiden hallinta eli yksiköiden mikrohallinta, taktinen suunnittelu eli keskitason taistelusuunnittelu ja strateginen suunnittelu eli korkean tason suunnittelu. Ontañón ym. selventävät termejä edelleen: yksiköiden mikrohallinta tarkoittaa yksiköiden oletuskäyttäytymisen, esimerkiksi resurssinkeräysyksikön automaattisen resurssin keräämisen, syrjäyttämistä omilla käskyillä. Taktinen suunnittelu tarkoittaa taistelukäyttäytymisen suunnittelua. Joissain tilanteissa voi esimerkiksi olla hyödyllistä jakaa omat joukot kahteen osaan ja hyökätä vihollisen kimppuun kahdelta puolelta. Strateginen suunnittelu tarkoittaa korkeamman tason suunnitelmia, kuten milloin rakentaa tukikohtaa ja yksiköitä, mitä yksiköitä rakentaa, milloin hyökätä jne.

Ontañón ym. (2013) toteavat, että strateginen päätöksenteko on edelleen avoin ongelma, jota on lähestytty monilla tavoilla, esimerkiksi helposti ohjelmoitavilla mutta huonosti ajon aikana sopeutuvilla tavoilla sekä suunnittelupohjaisilla ja oppimispohjaisilla lähestymistavoilla. Ontañón ym. kertovat, että Houlette ja Fu (2003) kirjoittavat äärellisten tilakoneiden käyttämisen olevan yleisin malli ohjelmoida helposti mutta ajon aikana huonosti mukautuvasti (engl. *hardcoding*), ja erityistapauksena he mainitsevat hierarkkiset äärelliset tilakoneet. Tilakoneiden yleiset periaatteet esitellään tutkielman luvussa 3.2.

Ontañón ym. (2013) jatkavat kertomalla, että asiaa on tutkittu myös suunnittelumenetelmien kautta. Esimerkiksi Ontañón ym. (2008) ovat tutkineet tekoälyn etukäteen opettamista ja tapauspohjaista suunnittelua (engl. *case-based planning*) opetetun tiedon pohjalta. Tapauskohtainen suunnittelu perustuu nimensä mukaisesti valmiiksi tiedettyihin tapauksiin, joiden pohjalta luodaan uusia suunnitelmia (Ontañón ym. 2007). Edelleen Mishra, Ontañón ja Ram

(2008) ovat jatkaneet kyseisen tutkimuksen pohjalta. Hoang, Lee-Urban ja Muñoz-Avila (2005) ovat tutkineet hierarkkisten tehtäväverkkojen (engl. *hierarchical task-networks*) avulla suunnittelemista.

Oppimisen kautta strategista päätöksentekoa ovat tutkineet esimerkiksi Weber ja Mateas (2009) soveltamalla tiedonlouhintaa sopivasti käsiteltyihin uusintoihin *StarCraft*-pelissä. Ontañón ym. (2013) kertovat, että oppimiseen liittyvää tapauspohjaista päättelyä (engl. *case-based reasoning*) ovat käyttäneet esimerkiksi Aha, Molineaux ja Ponsen (2005) ja edelleen heidän tutkimuksensa pohjalta Sun (2008).

Ontañón ym. (2013) kirjoittavat, että taktiseen päättelyyn liittyy päättelyä yksiköiden kyvyistä ryhmässä sekä päättelyä ympäristöstä ja muiden ryhmien sijainneista edun saamiseksi. Esimerkkinä tyhmästä taktisesta käyttäytymisestä he antavat kalliiden mutta haavoittuvaisten yksiköiden lähettämisen hitaita mutta voimakkaampia yksiköitä vastaan. Ontañón ym. jakavat taktisen päättelyn maaston analysointiin ja päätöksentekoon, joita molempia on tutkittu runsaasti. Heidän mukaansa päätöksenteossa on tutkittu monia erilaisia lähestymistapoja, esimerkiksi oppimista ja pelipuun hakuja. He mainitsevat myös, että tiedustelu on strategisen päätöksenteon lisäksi tärkeää myös taktisessa päätöksenteossa, mutta että asiaa ei ole tutkittu merkittävästi.

Yksittäisten yksiköiden hallinnan tavoitteena on lisätä yksiköiden tehokkuutta, Ontañón ym. (2013) kirjoittavat. He toteavat potentiaalikenttien ja vaikutuskarttojen osoittautuneen tähän tarkoitukseen sopiviksi esimerkiksi esteiden välttelyn ja vihollisen tulituksen välttelyn kautta. Samalla he kuitenkin toteavat, että potentiaalikenttien haittapuolena on suuri parametrimäärä säätämistarpeineen toivottuun käytökseen pääsemiseksi. Ongelman ratkaisemista vahvistusoppimisen kautta ovat tutkineet Liu ja Li (2008). Potentiaalikenttiä ovat tutkineet runsaasti Hagelbäck ja Johansson (2008a, 2008b, 2009). Ontañón ym. kertovat, että myös oppimispohjaiset menetelmät ovat paljon tutkittu lähestymistapa yksittäisten yksiköiden hallinnassa, samoin jossain määrin pelipuun hakuihin perustuvat menetelmät.

Vaikka suunnittelussa ja päätöksenteossa pyritäänkin usein pilkkomaan ongelma pienempiin osiin, se ei ole ainoa vaihtoehto. Ontañón ym. (2013) kertovat, että kokonaisvaltaisiakin menetelmiä on yritetty, mutta että niiden tutkiminen on ollut vähäisempää pääasiassa RTS-

pelien monimutkaisuuden takia. Ongelman pilkkominen pienempiin osiin yksinkertaisesti tuottaa useimmiten parempia tuloksia.

3.2 Äärelliset tilakoneet ja muut helposti ohjelmoitavat mutta huonosti mukautuvat menetelmät

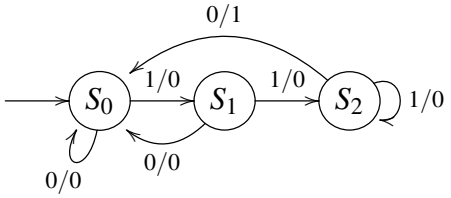
Äärelliset tilakoneet ovat hyvin yleisiä, ja muiden helposti ohjelmoitavien mutta huonosti ajon aikana mukautuvien ratkaisujen tapaan ne ovat myös tuottaneet hyviä ratkaisuja. Ideana on jakaa tekoälyn toiminta erilaisiin tiloihin, kuten hyökkäämiseen, resurssien keräämiseen ja korjaamiseen, joiden avulla käyttäytymisen hallinta on helpompaa. Näiden menetelmien ongelmana on kuitenkin dynaamisuuden ja mukautuvuuden puute sekä niiden helppo hyväksikäytettävyys. (Ontañón ym. 2013.)

Äärellinen tilakone on abstrakti kone, jossa on eri tiloja sekä tilasiirtymiä tilojen välillä. Usein tilakoneella on myös alkutila. Feldman (1992) jakaa äärelliset tilakoneet tulostaviin ja hyväksyviin tilakoneisiin. Edelleen hän jakaa tulostavat tilakoneet sellaisiin, jotka generoivat tulosteen tilasiirtymien välillä (Mooren kone), ja sellaisiin, jotka generoivat tulosteen tilasiirtymän aikana (Mealyn kone).

Feldman (1992) määrittelee tulostavan tilakoneen formaalisti kuusikkona

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0),$$

missä Q on tilojen joukko, Σ syötemerkkien joukko, Δ tulostemerkkien joukko, $\delta : Q \times \Sigma \to Q$ tilasiirtymäfunktio, $\lambda : Q \to \Delta$ funktio, joka liittää jokaiseen tilaan tulostemerkin, ja $q_0 \in Q$ tilakoneen alkutila.



Kuvio 1. Yksinkertainen Mealyn kone

Tilakoneiden formaali esittäminen on varsin raskasta. Usein on käytännöllistä esittää tilako-

ne esimerkiksi kuvana. Kuviossa 1 on esitetty yksinkertainen Mealyn kone. Kuvassa ympyrät kuvaavat tiloja ja nuolet niiden välisiä tilasiirtymiä. Nuolen vieressä olevista merkeistä ensimmäinen tarkoittaa syötettä ja toinen tulostetta. Kone laskee sille annettujen ykkösten lukumäärän kahteen asti, minkä jälkeen se jää tilaan S_2, kunnes se saa syötteen nolla. Kuviossa 1 S_0 tarkoittaa tilaa, jossa ei ole annettu yhtään ykköstä, S_1 tilaa, jossa on annettu yksi ykkönen, ja S_2 tilaa, jossa on vähintään kaksi ykköstä. Koneen tuloste on nolla kaikissa muissa tilasiirtymissä paitsi siinä, jossa kone palaa tilasta S_2 tilaan S_0.

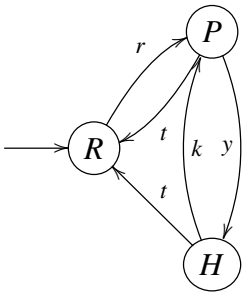
Tulostavien tilakoneiden tapaan Feldman (1992) määrittelee hyväksyvän tilakoneen formaalisti viisikkona

$$M = (Q, \Sigma, \delta, q_0, F),$$

missä tulostavan tilakoneen tapaan Q on tilojen joukko, Σ syötemerkkien joukko, $\delta : Q \times \Sigma \rightarrow Q$ tilasiirtymäfunktio ja $q_0 \in Q$ alkutila, mutta lisäksi määritelmässä on $F \subset Q$, joka on hyväksyttyjen tilojen joukko. Erona tulostaviin tilakoneisiin verrattuna on siis tulostemerkistö, tilat johonkin tulostemerkkiin liittävän funktion puute ja hyväksyttyjen tilojen joukko. Hyväksyvälle tilakoneelle voidaan antaa jokin syötemerkistön Σ mukainen merkkijono, jonka tilakone käsittelee merkki kerrallaan. Nimitys hyväksyvä tilakone tulee siitä, että mikäli merkkijonon merkkejä vastaavien tilasiirtymien jälkeen kone on jossakin hyväksytyssä tilassa, koneen sanotaan hyväksyvän merkkijonon.

Edellä esitelty hyväksyvä tilakone tunnetaan myös deterministisenä äärellisenä automaattina. Feldman (1992) esittelee sen lisäksi epädeterministisen äärellisen automaatin. Mikäli tilasiirtymäfunktiota $\delta : Q \times \Sigma \rightarrow Q$ muokataan muotoon $\delta : Q \times \Sigma \rightarrow \mathscr{P}(Q)$, missä $\mathscr{P}(Q) = \{S : S \subset Q\}$, saadaan epädeterministinen äärellinen automaatti. Epädeterministisyys ilmenee siten, että kutakin tila-syötemerkkiparia kohti tilasiirtymäfunktio antaakin joukon tiloja yhden tilan sijaan, jolloin tilakoneella voi olla samalla syötteellä useampi vaihtoehto seuraavaksi tilaksi.

Äärellisen automaatin formaali määritelmä ei välttämättä ole kovin kuvaava, ja usein myös automaattia havainnollistamaan käytetään kuvaa. Kuviossa 2 on esitetty hyvin yksinkertainen esimerkki tilakoneesta, joka voisi kuvata tekoälyn toimintaa. Kuvan lukeminen toimii kuten Mealyn koneen tapauksessa sillä erotuksella, ettei tilasiirtymiin liity tulostetta. Kuvan



Kuvio 2. Yksinkertainen deterministinen äärellinen automaatti

automaatissa on kolme eri tilaa: rakentaminen (R), puolustaminen (P) ja hyökkääminen (H). Tekoäly aloittaa rakennustilassa, jossa se rakentaa tukikohtaansa. Kun se on rakentanut tarpeeksi (r), se siirtyy puolustustilaan, jossa se voi kouluttaa uusia yksiköitä. Kun se on saanut koulutettua tarpeeksi yksiköitä (y), se siirtyy hyökkäystilaan. Toisaalta mikäli puolustustilan aikana tekoäly joutuu hyökkäyksen kohteeksi ja se huomaa menettäneensä tarpeeksi rakennuksiaan (t), se siirtyykin takaisin rakennustilaan. Edelleen hyökkäystilassa ollessaan tekoäly voi siirtyä puolustustilaan, mikäli sen joukot kärsivät liian suuria tappioita (k), tai rakennustilaan, mikäli joku muu pelaaja hyökkää tekoälypelaajan tukikohdan kimppuun ja onnistuu tuhoamaan tarpeeksi rakennuksia (t). Lisäksi kannattaa huomata, ettei kuvion 2 automaatilla ole hyväksyttyjä tiloja.

3.3 Reitinhaku

A* (Hart, Nilsson ja Raphael 1968) on suurimman osan peleistä käyttämä reitinhakualgoritmi tehokkuutensa ja helpon toteutettavuutensa vuoksi (Millington ja Funge 2009). Ontañón ym. (2013) toteavat näin olevan myös RTS-peleissä mutta jatkavat, että sen käytön suurimpia ongelmia RTS-pelien kaltaisissa ympäristöissä ovat prosessoriajan ja muistin käyttö. Koska A* on näin keskeinen RTS-peleissä, esitellään tässä lyhyesti sen perusidea.

A* on verkkotietorakenteen (engl. *graph*) reitinhakuun tarkoitettu algoritmi, joka löytää sopivissa olosuhteissa ei vain jonkin vaan peräti parhaan reitin verkon kahden solmun (engl. *node*) välillä, mikäli reitti ylipäänsä on olemassa. Sen toiminnan ytimessä on lauseke

$$f(n) = g(n) + h(n),$$

missä n on jokin verkon solmu, $f(n)$ arvioitu edullisimman reitin hinta alkusolmusta solmun n kautta tavoitesolmuun, $g(n)$ edullisimman alkusolmusta solmuun n johtavan reitin hinta ja $h(n)$ heuristisesti arvioitu hinta solmusta n tavoitesolmuun. Yksinkertainen esimerkki heuristiikasta h on kahden solmun välinen euklidinen etäisyys. (Russell ja Norvig 2009.)

Algoritmi lähtee liikkeelle alkusolmusta laskemalla sen naapurisolmujen arvioidut kustannukset ja valitsemalla seuraavaksi käsiteltäväksi solmuksi sen solmun, jonka kustannus on pienin. Edelleen algoritmi laskee kustannukset valitun solmun naapureille ja valitsee seuraavaksi käsiteltäväksi solmuksi sen käsittelemättömän solmun, jonka kustannusarvio on pienin. Mikäli algoritmi jossain vaiheessa laskee uuden kustannusarvion solmulle, jolle on jo ennestään laskettu kustannusarvio, valitsee se näistä pienemmän. Algoritmi jatkaa näin, kunnes se saavuttaa tavoitesolmun. (Russell ja Norvig 2009.)

Heuristiikka h on monotoninen, mikäli minkä tahansa solmun n kaikille seuraajille n' on kaikilla mahdollisilla toiminnoilla a voimassa

$$h(n) \leq c(n,a,n') + h(n'),$$

missä $c(n,a,n')$ on toiminnolla a solmusta n solmuun n' tapahtuvan siirtymän kustannus (Russell ja Norvig 2009). Toisin sanoen heuristiikka h on monotoninen, mikäli solmusta seuraajaan siirtyminen ei ole kalliimpaa kuin solmuun itseensä siirtyminen. Russell ja Norvig osoittavat, että mikäli käytettävä heuristiikka h on monotoninen, A* löytää aina parhaan reitin.

Millington ja Funge (2009) toteavat, että useimmissa peleissä käytetään pellin alla ruutuihin pohjautuvia ratkaisuja, joihin A* soveltuu mainiosti. Tällöin verkoksi voidaan tulkita ruutujen muodostama joukko sekä niiden väliset yhteydet, jolloin solmuiksi voidaan tulkita luonnollisesti ruudut. Toisaalta tyypillistä ruutupohjaista ratkaisua voidaan tehostaa jakamalla ruudut kolmioiksi (Demyen ja Buro 2006).

A* ei ole ainoa tapa toteuttaa reitinhaku. Eräs vaihtoehtoinen tapa on käyttää potentiaalikenttiä, kuten Hagelbäck ja Johansson (2008b) tutkimuksessaan huomauttavat. Hagelbäck ja Johansson pitävät potentiaalikenttiä reitinhaussa etukäteen laskettuja reittejä joustavampina.

3.4 Oppiminen

Russell ja Norvig (2009) kertovat, että vahvistusoppiminen (engl. *reinforcement learning*) tarkoittaa yksinkertaisesti positiivisten ja negatiivisten palkkioiden käyttämistä erilaisista toiminnoista, jotta tekoäly oppisi, mitkä toiminnot ovat hyviä ja mitkä huonoja. Näin ollen vahvistusoppimisessa tekoälyn ei tarvitse alussa tietää edes, mitä sen käytössä olevat toiminnot tekevät, sillä se oppii toimintojen käyttötilanteet palautteen avulla. Russell ja Norvig antavat tiivistettynä esimerkkinä vahvistusoppimisesta uuden pelin, jonka säännöt ovat tuntemattomia. Lopulta vaikkapa sadan pelatun vuoron kuluttua vastapelaaja ilmoittaa pelaajan hävinneen, mikä on selvä negatiivinen palaute.

Szita (2012) kertoo pelien olevan suosittu tutkimusalue vahvistusoppimiselle. Hän toteaa vahvistusoppimisen olevan useissa tapauksissa kilpailukykyinen vaihtoehto muille tekoälymenetelmille ja jopa ihmisille, mutta kaikissa tapauksissa näin ei vielä ole. Samalla Szita kuitenkin toteaa, että vahvistusoppiminen peleissä on nopeasti kehittyvä tutkimuskohde.

Konkreettisena esimerkkinä Wender ja Watson (2012) testasivat vahvistusoppimisella oppivaa tekoälyä *StarCraft: Brood War* -pelissä (Blizzard Entertainment 1998b). Testitapauksena oli monta vihollisen heikompaa yksikköä ja yksi tekoälyn yksikkö, jonka nopeus, kantama ja jossain määrin myös tulivoima olivat parempia kuin yksittäisellä vihollisyksiköllä. Useaa vihollisyksikköä vastaan tekoälyn tehokkaampi yksikkö kuitenkin normaalisti häviää selvästi. Tavoitteena oli saada tekoäly oppimaan taktiikka, jossa se pysyttelee vihollisen kantaman ulkopuolella ampuen itse turvallisen etäisyyden päästä. Yli tuhannen testitapauksen testeissä vahvistusoppimista hyödyntävä tekoäly oppi voittamaan pelin sisäänrakennetun tekoälyn noin 100 prosentissa tapauksista kaikilla testatuilla algoritmeilla. Toisaalta oppimisen nopeuttaminen johti parhaassakin tapauksessa 40 prosentin heikentymiseen suorituskyvyssä, joten Wender ja Watson toteavat aiheen kaipaavan lisää tutkimusta.

Myös Madeira, Corruble ja Ramalho (2006) ovat tutkineet aihetta. He kehittivät korkeamman tason tekoälyn, joka hyödynsi vahvistusoppimista yhdistettynä muihin menetelmiin. Muutaman tuhannen pelikerran jälkeen heidän kehittämänsä tekoäly paitsi voitti pelin sisäänrakennetun kaupallisen tekoälyn selvästi, myös pystyi kilpailemaan keskivertoihmispelaajan kanssa pisteissä mitattuna. Madeira, Corruble ja Ramalho toteavatkin, että vahvistusoppimi-

nen voi toimia kohtuullisessa ajassa muihin tekniikoihin yhdistettynä ja että se on lupaava tutkimussuunta laajojen strategiapelien tekoälyssä.

4 Yhteenveto

RTS-pelien tekoälyn keskeisimmät osa-alueet ja samalla haasteet ovat suunnittelu, oppiminen, epävarmuus, avaruudellinen ja ajallinen päättely, kohdealueen tietämyksen hyödyntäminen sekä tehtävien ositus. Suunnittelu jaetaan karkeasti yksittäisten yksiköiden hallintaan, taktiseen suunnitteluun ja strategiseen suunnitteluun, joista kutakin voidaan tutkia monilla eri tavoilla. Muita keskeisiä osa-alueita ovat reitinhaku, ryhmänä liikkuminen, yksittäisten yksiköiden toiminta sekä taktiset ja strategiset tukijärjestelmät. Muihin kehityskohteisiin taas lukeutuvat jumiin jääneiden tekoälyn osien tunnistaminen, pelaajaa pienten ja toistuvien toimintojen suorittamisessa auttava tekoäly, erilaisten vastustajien persoonallisuuksien toteuttaminen ja strategisen ajattelun lisääminen sekä yksittäisten yksiköiden epäinhimillisen tehokkaan hallinnan vähentäminen.

Erilaisia käytettäviä menetelmiä ovat muun muassa tilakoneet ja muut helposti ohjelmoitavat mutta huonosti ajon aikana mukautuvat menetelmät, tapauspohjainen suunnittelu ja päättely, hierarkkiset tehtäväverkot, tiedonlouhinta, pelipuun haut, potentiaalikentät ja vahvistusoppiminen. Lisäksi kokonaisvaltaisia lähestymistapoja on yritetty, mutta tyypillisesti tehtävien ositus tuottaa kuitenkin parempia tuloksia. Reitinhaussa käytetyin menetelmä on huonoine puolineenkin A*, jota voidaan tarvittaessa tehostaa esimerkiksi pilkkomalla pelialueen ruutuja kolmioiksi.

Kaiken kaikkiaan RTS-pelien tekoälytutkimus on hyvin monipuolista ja moniulotteista. Lisäksi se on hyvin haastavaa RTS-pelien monimutkaisuudesta johtuen. Aihetta on tutkittu paljon aivan viime vuosiin saakka eikä vaikuta olevan syytä olettaa, että asiaan olisi tulossa muutosta.

RTS-pelien tekoäly on kehittynyt genren syntymän jälkeen valtavasti, ja vielä viimeisten kymmenen vuoden aikanakin kehitystä on tapahtunut huomattavasti. Tutkimuksen monipuolisuuden ja moniulotteisuuden sekä RTS-pelien monimutkaisuuden takia tutkittavaa lienee vielä pitkäksi aikaa lähes jokaisella osa-alueella, sillä millään osa-alueella ei tunnu olevan yhteisymmärrystä mistään yksittäisestä parhaasta ratkaisusta. Opiskelijan näkökulmasta RTS-pelien tekoälystä voi helposti saada useitakin aiheita pro gradu -tutkielmaa varten.

Lähteet

Aha, David W., Matthew Molineaux ja Marc Ponsen. 2005. “Learning to Win - Case-Based Plan Selection in a Real-Time Strategy Game”. *Case-Based Reasoning Research and Development:* 5–20.

Blizzard Entertainment. 1994. *Warcraft: Orcs & Humans.*

———. 1998a. *StarCraft.*

———. 1998b. *StarCraft: Brood War.*

Buro, Michael. 2004. “Call for AI Research in RTS Games”. Teoksessa *Proceedings of the 4th Workshop on Challenges in Game AI, San Jose.*

Chung, Michael, Michael Buro ja Jonathan Schaeffer. 2005. “Monte Carlo Planning in RTS Games”. *CIG.*

Demyen, Douglas, ja Michael Buro. 2006. “Efficient Triangulation-Based Pathfinding”. Teoksessa *AAAI,* 6:942–947.

Feldman, Yishai A. 1992. “Finite-State Machines”. Teoksessa *Encyclopedia of Computer Science and Technology,* nide 25. Marcel Dekker, Inc.

Hagelbäck, Johan, ja Stefan J Johansson. 2008a. “The Rise of Potential Fields in Real Time Strategy Bots”. *AIIDE* 8:42–47.

———. 2008b. “Using Multi-Agent Potential Fields in Real-Time Strategy Games”. Teoksessa *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems,* 2:631–638. International Foundation for Autonomous Agents ja Multiagent Systems.

———. 2009. “A Multiagent Potential Field-Based Bot for Real-Time Strategy Games”. *International Journal of Computer Games Technology* 2009:4.

Hart, P.E., N.J. Nilsson ja B. Raphael. 1968. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. *Systems Science and Cybernetics, IEEE Transactions on* 4, numero 2 (heinäkuu): 100–107. ISSN: 0536-1567. doi:10.1109/TSSC.1968.300136.

Hoang, Hai, Stephen Lee-Urban ja Héctor Muñoz-Avila. 2005. “Hierarchical Plan Representations for Encoding Strategic Game AI.” *AIIDE:* 63–68.

Houlette, R., ja D. Fu. 2003. “The Ultimate Guide to FSMs in Games”. Teoksessa *AI Game Programming Wisdom,* 2:283–302.

Liu, Liang, ja Longshu Li. 2008. “Regional Cooperative Multi-agent Q-learning Based on Potential Field”. Teoksessa *Natural Computation, 2008. ICNC'08. Fourth International Conference on Natural Computation,* 6:535–539. IEEE.

Madeira, Charles AG, Vincent Corruble ja Geber Ramalho. 2006. “Designing a Reinforcement Learning-based Adaptive AI for Large-Scale Strategy Games.” Teoksessa *AIIDE,* 121–123.

Millington, Ian, ja John Funge. 2009. *Artificial Intelligence for Games.* Second Edition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 0123747317, 9780123747310.

Mishra, Kinshuk, Santiago Ontañón ja Ashwin Ram. 2008. “Situation Assessment for Plan Retrieval in Real-Time Strategy Games”. Teoksessa *Advances in Case-Based Reasoning,* 355–369. Springer.

Ontañón, Santiago, Kinshuk Mishra, Neha Sugandh ja Ashwin Ram. 2007. “Case-Based Planning and Execution for Real-Time Strategy Games”. Teoksessa *Case-Based Reasoning Research and Development,* 164–178. Springer.

———. 2008. “Learning from Demonstration and Case-Based Planning for Real-Time Strategy Games”. Teoksessa *Soft Computing Applications in Industry,* 293–310. Springer.

Ontañón, Santiago, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill ja Mike Preuss. 2013. “A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft”. *Computational Intelligence and AI in Games, IEEE Transactions on* 5 (4): 293–311.

Russell, Stuart, ja Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach.* 3rd Edition. Upper Saddle River, NJ, USA: Prentice Hall Press. ISBN: 0136042597, 9780136042594.

Schwab, Brian. 2009. *AI Game Engine Programming.* 2nd Edition. Game Programming. Course Technology/Cengage Learning. ISBN: 9781584505723.

Sun, Chuen-Tsai. 2008. “Building a Player Strategy Model by Analyzing Replays of Real-Time Strategy Games”.

Szita, István. 2012. “Reinforcement Learning in Games”. Teoksessa *Reinforcement Learning,* 539–577. Springer.

Technosoft. 1989. *Herzog Zwei.*

Weber, Ben George, ja Michael Mateas. 2009. “A Data Mining Approach to Strategy Prediction”. Teoksessa *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on,* 140–147. IEEE.

Wender, Stefan, ja Ian Watson. 2012. “Applying Reinforcement Learning to Small Scale Combat in the Real-Time Strategy Game StarCraft: Broodwar”. Teoksessa *Computational Intelligence and Games (CIG), 2012 IEEE Conference on,* 402–408. IEEE.

Westwood Studios. 1992. *Dune II: The Building of a Dynasty.*

———. 1995. *Command & Conquer.*