Jonni Pitkänen

AlphaZero shakkikoneena

Tietotekniikan kandidaatintutkielma

1. toukokuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Jonni Pitkänen

Yhteystiedot: jopepitk@student.jyu.fi

Työn nimi: AlphaZero shakkikoneena

Title in English: AlphaZero as a chess engine

Työ: Kandidaatintutkielma

Sivumäärä: 26+0

Tiivistelmä: DeepMindin koneoppiva go:ta, shogia ja shakkia pelaava AlphaZero yllätti shakkimaailman vuoden 2017 lopussa omalaatuisella ihmisläheisellä pelityylillään ja kiistattomalla tehokkuudellaan. Tässä tutkielmassa haluttiin selvittää AlphaZeron rakennetta sekä sen taustalla olevia menetelmiä. Syy AlphaZeron menestykseen todettiin olevan sen ihmisistä riippumaton syvä vahvistettu oppiminen, sekä lupaaviin variaatioihin keskittyvä Monte-Carlo -puuhaku. Tiedon pohjalta pääteltiin, että AlphaZeron pelitilanteita analysoiva neuroverkko sekä liikkeitä etsivä puuhaku vastaavat yllättävän tarkasti perinteisten shakkikoneiden kaksiosaista mallia, mutta kummankin osan toteutus vaikuttaa olevan perinteisiä funktioita tehokkaampi).

Avainsanat: AlphaZero, shakkikone, vahvistettu oppiminen, koneoppiminen, Monte-Carlo -puuhaku, neuroverkko, syväoppiminen

Abstract: The world of chess was surprised in late 2017 by DeepMind's machine learning go-, shogi- and chess engine AlphaZero with it's unique human-like playstyle and it's undisputed efficiency. The objective of this thesis was to study the structure of AlphaZero and the methods used to complement it. According to the information gathered, the key to AlphaZero's success was it's human-independent deep reinforcement learning and it's Monte-Carlo Tree Search, that is able to concentrate on more promising variations. From these finds it was derived, that structure-wise AlphaZero resembles the traditional chess engine surprisingly well, but it seems AlphaZero's components are more effective in their tasks.

Keywords: AlphaZero, chess engine, reinforcement learning, machine learning, Monte-Carlo Tree Search, neural network, deep learning

Kuviot

Sisältö

1 Johdanto

Shakki on ikivanha ja ikivihreä peli, jonka saloja ollaan pyritty ratkaisemaan tietokoneiden avulla niin kauan kuin tietokoneita on ollut olemassa. Jo vuonna 1956 shakkikone MANIAC onnistui voittamaan noviisitason pelaajan yksinkertaistetuilla Los Alamos -säännöillä, jossa pelataan pienemmällä laudalla ilman lähettejä. Ihmiskunnan parhaimmiston voittaminen oli kuitenkin vielä pitkän matkan päässä, mutta kisa oli jo alkanut. Vielä vuonna 1996 hallitseva maailmanmestari Garry Kasparov onnistui voittamaan sen hetkisen vahvimmaksi shakkikoneeksi uskotun Deep Bluen kuuden pelin sarjassa lupaavasti pistein 4-2. Jatkuva kehitys oli kuitenkin shakkikoneiden puolella, ja voimatasapaino oli nopeasti muttumassa. Jo seuraavana vuonna pelatussa Kasparovin ja Deep Bluen välisessä revanssissa paranneltu Deep Blue voitti Kasparovin pistein 3,5-2,5. Vaikka Kasparov protestoikin pelin olosuhteista, shakkikoneiden hidas mutta varma kehittyminen maailmanluokan shakinpelaajien ohitse alkoi näyttää selvältä. Vuonna 2005 ukrainalaisesta suurmestarista Ruslan Ponomarovista tuli historian viimeinen korkeatasoisen shakkikoneen turnaussäännöillä voittanut ihmispelaaja, kun hän voitti shakkikone Fritzin Mansaaren turnauksessa. Seuraavan vuosikymmenen aikana pelattiin useita ihmisten ja shakkikoneiden välisiä sarjoja ja turnauksia, joista ihmispelaajat eivät onnistuneet voittamaan ensimmäistäkään. Ihmisten aikakausi shakin kuninkaina oli ohitse.

Shakkikoneiden kehityksen myötä monet shakin pelaajat alkoivat ottaa vaikutteita parhaiden shakkikoneiden tekemistä siirroista, kuten myös hyödyntämään niitä omien siirtojensa analysoinnissa. Ammattilaisten piireissä puhuttiin, että ihmiset alkoivat pelata samaan tapaan kuin shakkikoneet, kun niiden kylmää ja laskelmoivaa tyyliä pyrittiin soveltamaan ihmispelaajien toimesta. Vuoden 2017 lopulla shakkimaailman katseet kääntyivät kuitenkin uudenlaiseen shakkikoneeseen Googlen omistaman DeepMind-yhtiön esitellessä projektinsa itseoppivasta shakkitekoälystä AlphaZerosta. Tämä DeepMindin koneoppiva tekoäly vaikutti pelaavan shakkia täysin omalla tyylillään eroten huomattavasti perinteisten shakkikoneiden pelityyleistä. AlphaZero teki vaikutuksen myös tehokkuudellaan, kun se onnistui voittamaan maailman parhaaksi shakkikoneeksi vuonna 2016 kruunatun Stockfishin.

Tässä tutkielmassa selvitetään miten AlphaZeron shakkitekoäly on toteutettu, miten se pe-

laa shakkia, ja mitkä ovat sen ratkaisevat eroavaisuudet muista shakkikoneista. Luvussa 2 pyritään luomaan ymmärrys AlphaZeron oppimisen taustalla olevista menetelmistä. Tämän pohjalta tutkitaan AlphaZeron neuroverkon rakennetta ja muita rakennusosasia luvussa 3, sekä tarkastellaan AlphaZeron opetusprosessia ja eroavaisuuksia perinteisiin shakkikoneisiin verrattuna. Lopuksi luvussa 4 analysoidaan täysin oppineen AlphaZeron tehokkuutta sekä pelityyliä, ja demonstroidaan AlphaZeron havaittuja vahvuuksia AlphaZeron ja Stockfishin välisessä shakkipelissä. Tutkielma toteutetaan kirjallisuuskatsauksena.

2 AlphaZeron oppiminen

Tässä luvussa käsitellään AlphaZeron oppimiseen liittyviä käsitteitä. Alustavana tietona voidaan sanoa, että AlphaZero oppii käyttämällä vahvistettua oppimista (Marcus 2018), joka on koneoppimisen alaluokka. Luvussa avataan ensin hieman koneoppimista yleisesti, jonka jälkeen käsitellään neuroverkkoja ja syväoppimista. Näiden aiheiden pohjalta käsitellään lopuksi vahvistettua oppimista.

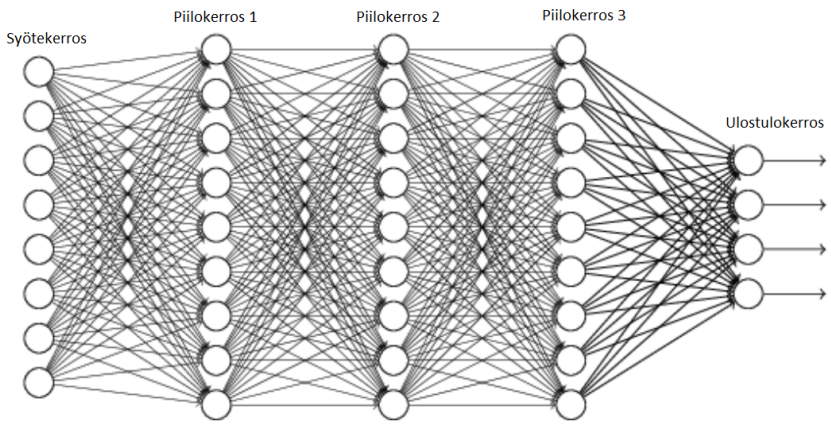
2.1 Koneoppiminen

Koneoppiminen voidaan määritellä ilmiöksi, jossa ohjelma oppii esimerkin tai kokemuksen kautta tekemään jotain, mitä sille ei alunperin oltu suoraan ohjelmoitu (Mohammed, Khan ja Bashier 2017). Sutton ja Barto (2018) ovat jakaneet koneoppimisen kolmeen alaluokkaan: ohjattu oppiminen (engl. supervised learning), ohjaamaton oppiminen (engl. unsupervised learning), sekä vahvistettu oppiminen (engl. reinforcement learning). Tässä tutkielmassa keskitytään enimmäkseen vahvistettuun oppimiseen, mutta koneoppimista alustetaan hieman ohjatun oppimisen kautta.

Ohjatussa oppimisessa konetta pyritään opettamaan antamalla sille opetusdatana (x, y) -pareja, joissa jokaisella syötteellä x on vastaus y (Mohammed, Khan ja Bashier 2017). Esimerkiksi autoja tunnistavan ohjatun oppimisen algoritmin opetusdata sisältäisi kuvan x, sekä selityksen y, joka kertoo onko kuvassa oleva asia auto vai ei. Opetusdatan tiedot tallennetaan algoritmille muistiin, jolloin se pystyy jatkossa hyödyntämään samaansa tietoa päätöksenteossaan (LeCun, Bengio ja Hinton 2015). Täten algoritmi oppii datan perustella paremmaksi autojen tunnistajaksi. Opetuksen tavoitteena on saada algoritmin tietämys opetettavasta aiheesta yleistettyä opetusdatan ulkopuolelle (Alpaydin 2016), jolloin se voitaisiin ottaa käyttöön tunnistamaan autoja vaikkapa autotallin oven eteen. Koneoppivan algoritmin toteutukseen on useita vaihtoehtoja, kuten neuroverkko, päätöspuu tai lineaarinen erotteluanalyysi (engl. linear discriminant analysis) (Mohammed, Khan ja Bashier 2017). AlphaZeron tunnetaan käyttävän oppimiseen neuroverkkoa (Silver ym. 2017), joten tässä tutkielmassa keskitytään siihen.

2.2 Neuroverkot ja syväoppiminen

Samalla tavalla kuin ihmisen aivot koostuvat neuroneista, jotkä siirtävät tietoa toistensa välillä synapsien avulla, myös keinotekoiset neuroverkot koostuvat kerroksista neuroneita ja niiden välisistä synapseista (Caterini ja Chang 2018). Jokaisella verkon neuroneita toisiinsa liittävällä synapsilla on painoarvo, joka määrittää edellisen neuronin vaikutuksen uuteen neuroniin datan kulkiessa neuronien lävitse (Alpaydin 2016). Neuroverkon ensimmäinen kerros on syötekerros (input layer), jonka kautta neuroverkko vastaanottaa tietoa (Mohammed, Khan ja Bashier 2017). Riippuen mitä neuroverkolla halutaan toteuttaa, voi tämä kerros olla esimerkiksi ääntä tai kuvaa lukeva sensori. Syötekerroksen alla on yhdestä useampaan piilokerrosta (engl. hidden layer), joissa syötteen prosessointi tapahtuu (Alpaydin 2016). Neuroverkon vastaanottaessa syötteen sen syötekerroksen neuronit aktivoituvat ja lähettävät syötteensä synapsien kautta alempien kerrosten neuroneille (Mohammed, Khan ja Bashier 2017). Synapsien kautta neuroneissa liikkuva data muokkaantuu synapseissa määrättyjen painoarvojen mukaisesti, kunnes se saapuu ulostulokerrokseen, jonka aktivoituneet neuronit edustavat neuroverkon arviota annetusta syötteestä (Nielsen 2015). Neuroverkon rakenne on havainnollistettu kuviossa 1, jossa eri kerroksissa olevat pallot kuvastavat verkon neuroneita ja neuroneiden väliset viivat kuvastavat synapseja.



Kuvio 1: Havainnollistava kuva neuroverkosta, jolla on kolme piilokerrosta (Nielsen 2015, s. 169)

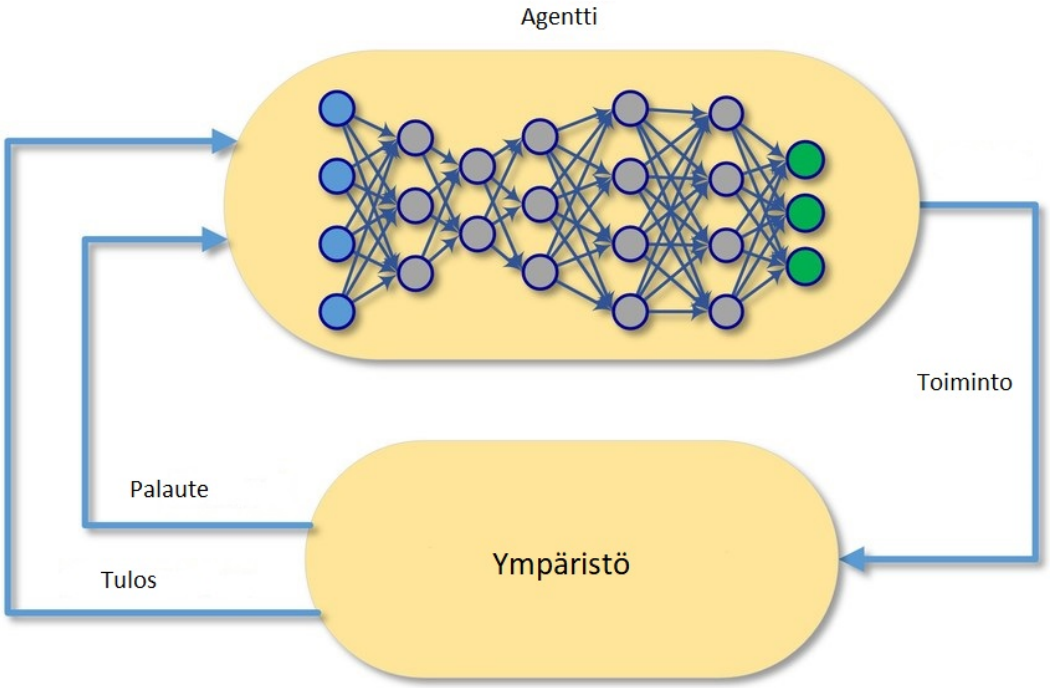
Neuroverkon ei heti alkuun voida olettaa olevan kykenevä tunnistamaan kuvia tai käsinpiirrettyjä numeroita; sen neuronien välisien synapsien painoarvot määritetään alkuun yleensä satunnaisesti (Alpaydin 2016). Neuroverkkoa opetetaan luvussa 2.1 esitellyn mallin mukaisesti. Syötteen käsittelyn jälkeen määritellään virhefunktio (engl. error function) oikean vastauksen ja neuroverkon antaman vastauksen erotuksesta (LeCun, Bengio ja Hinton 2015). Virheen määrän ollessa tiedossa käydään neuroverkko läpi vastavirta-algoritmilla (engl. backpropagation), joka muokkaa synapsien painoarvoja oikean vastauksen suuntaan (Nielsen 2015, s. 37-43). Jos neuroverkolle näytettäisiin tämän jälkeen sama kuva, olisi sen vastaus päivitettyjen painoarvojen ansiosta lähempänä totuutta. Neuroverkkoa opettaessa tämä prosessi toistetaan jokaisella opetuserän alkiolla.

Kahden tai useamman piilokerroksen neuroverkkoa kutsutaan syväksi neuroverkoksi (engl. deep neural network) (Nielsen 2015, s. 37). Syvät neuroverkot kykenevät suorittamaan monimutkaisempia ongelmia helpommin, sillä eri piilokerrosten neuronit voivat keskittyä syötteen eri tekijöihin (Lai 2015). Esimerkiksi käsinpiirrettyjä numeroita tunnistava syvä neuroverkko voi ensimmäisessä piilokerroksessa tunnistaa piirroksessa erilaisia muotoja, kuten kaaria tai suoria viivoja, jonka jälkeen syvemmän kerroksen neuronit tarkastelevat minkälaiseksi kuvioksi aikaisemman kerroksen löytämät muodot liittyvät. Syvien neuroverkkojen oppimista kutsutaan syväoppimiseksi (engl. Deep Learning) (Alpaydin 2016). Myös AlphaZeron neuroverkko on syvä neuroverkko (Silver ym. 2017).

2.3 Vahvistettu oppiminen

Vahvistettu oppiminen on koneoppimisen alaluokka, jolle ominaista on oppivan agentin oman ympäristönsä antaman palautteen avulla oppiminen (Sutton ja Barto 2018). Toisin kuin ohjatussa oppimisessa, missä agenttia opetetaan syöte-vastaus -parien avulla, vahvistetun oppimisen agentti tekee päätöksentekofunktionsa (esim. neuroverkon) pohjalta toimintoja, joista agentti saa ympäristöltään palautetta (Wiering ja Van Otterlo 2012, s. 5). Agentin saama palaute riippuu sen tekemän toiminnon aikaansaamasta lopputuloksesta ollen sitä parempi, mitä lähemmäksi toivottua tulosta ollaan päästy (Sutton ja Barto 2018). Tämän jälkeen agentin tekemä toiminto ja siitä seurannut palaute tallennetaan agentin muistiin palautefunktiolla (engl. reward function), jolloin agentti on oppinut kyseisen toiminnon seurauksen (Moham-

med, Khan ja Bashier 2017). Prosessi on havainnollistettu kuviossa 2.



Kuvio 2: Hahmotelma vahvistetun oppimisen toimintaperiaatteesta (Mohammadi ja Al-Fuqaha 2018). Tässä oppiva agentti on neuroverkko.

Sutton ja Barto (2018) esittelevät kirjassaan kaksi vahvistetun oppimisen vahvuutta. Ensimmäinen vahvistetun oppimisen hyöty ohjattuun oppimiseen verrattuna on se, että monessa interaktiivisessa tilanteessa voi neuroverkon syötteelle olla erittäin hankalaa määritellä tiettyä täysin oikeaa vastausta, jolloin alaluvussa 2.1 kuvailtua ohjattua oppimista ei voi tapahtua. Ei ole esimerkiksi olemassa vain yhtä oikeaa tapaa navigoida satunnaista maastoa, tai tehdä voileipää. Sen sijaan vahvistetussa oppimisessa keskitytään tiettyjen vastausten sijaan parhaaseen mahdolliseen lopputulokseen, jonka saavuttamiseen agentti voi oppia usean eri strategian. Toinen esitelty vahvuus ilmenee tilanteissa, jossa opetettavasta asiasta ei ole ennestään olemassa tarkkaa tietoa, jolloin oppisen on tapahduttava kokemuksen kautta.

Vahvistetulle oppimiselle ainutlaatuinen ongelma on kysymys opitun hyödyntämisen (engl. exploitation) ja uusien vaihtoehtojen etsimisen välillä (engl. exploration) (Sutton ja Barto 2018). Esimerkiksi shakkia pelaava agentti voi tietyssä tilanteessa todeta saamansa palauteen perusteella liikkeen a olevan hyvä, joten agentin kannattaa hyödyntää tietoaan ja pelata

liikettä *a*. Liikkeet *b*, *c* ja *d* ovat kuitenkin agentille vielä tuntemattomia, ja täten voivat potentiaalisesti olla liikettä *a* parempia. Saadakseen parasta mahdollista palautetta on agentin käytettävä hyödykseen oppimiaan siirtoja, mutta löytääkseen totisesti parhaat mahdolliset siirrot ja saavuttaakseen parhaan mahdollisen palautteen täytyy agentin tutkia myös uusia tuntemattomia vaihtoehtoja (Wiering ja Van Otterlo 2012). Sutton ja Barto (2018) toteavat kirjassaan, ettei opitun tiedon ja uuden etsimisen tasapainottamisen ongelmaa ole täydellisesti ratkaistu, mutta Chaslot ym. (2008) mainitsevat ongelman ratkaisuun kuitenkin olevan useita tehokkaita vaihtoehtoja. Tästä voi päätellä, että tasapainon löytämiselle olisi valittava parhaiten tehtävään soveltuva ratkaisu projektikohtaisesti.

Toinen vahvistetun oppimisen varjopuoli on sen riippuvuus simulaatioiden kautta oppimisesta (Bratko 2018). Simuloidessa agentti voi oppia huomattavasti nopeammin, eikä se kärsi mahdollisista epäonnistumisen haittavaikutuksista (esim. törmäilyn aiheuttama vahinko), joita simuloimaton oppiminen voisi tuottaa (Wiering ja Van Otterlo 2012). Bratko (2018) käytti tästä esimerkkinä Wileyn ym. tutkimusta (2015), jossa epätasaisessa maastossa liikkumista oppivalle robotille ei ollut saatavilla tarkkaa mallia simulaatioita varten, joten opetuksen piti tapahtua käyttäen robottia itseään. Tämä vähensi mahdollisten iteraatioiden määrää opetuksessa murto-osaan siitä, mitä simulaatiolla olisi ollut mahdollista toteuttaa.

Shakki on monimutkainen peli, eikä suurimpaan osaan tilanteista ole vain yhtä oikeaa lähestymistapaa. Tästä johtuen voidaan päätellä shakin opettamisen ohjatun oppimisen tapaan syöte-vastaus -pareilla olevan vähintäänkin todella hankalaa. Vahvistetun oppimisen agentti voi sen sijaan saada oppiessaan positiivista palautetta kaikista tekemistään hyvistä siirroista, ja täten oppia shakkia monipuolisemmin. Myös valmiin tiedon puute soveltuu AlphaZeron tapaukseen; DeepMindin tavoitteena oli luoda AlphaZerosta alkuun ”tabula rasa” (suom. tyhjä taulu) (Silver ym. 2017), eli AlphaZero pyrittiin toteuttamaan minimaalisella määrällä synnynnäistä tietoa ja ilman synnynnäisiä rakenteita (Marcus 2018). Näin ollen ihmiskunnan tietämys shakista ei ole saatavilla AlphaZerolle, joten peli olisi opittava kokonaan omasta kokemuksesta. Shakkipelejä on myös äärimmäisen helppo simuloida, joten vahvistetun oppimisen riippuvuus simulaatioista ei ole AlphaZeron tapauksessa ongelma. Vahvistetun oppimisen voidaan täten todeta soveltuvan hyvin AlphaZeron shakin opetukseen.

3 AlphaZeron kehittäminen

Tässä luvussa tutkitaan miten AlphaZero on kehitetty, ja miten luvussa 2 käsiteltyjä menetelmiä on sovellettu AlphaZeron kehityksessä ja opetuksessa. Shakki on täydellisen informaation peli, mikä tarkoittaa, että kaikki pelissä oleva tieto on jatkuvasti pelaajien nähtävissä (esimerkiksi pokeri tai Starcraft 2 eivät ole täydellisen informaation pelejä, sillä pelaajat eivät näe toistensa kortteja/joukkoja jatkuvasti). Kaikki mitä pelin tilanteesta tarvitsee tietää on siis shakissa aina mahdollistä nähdä laudalta. Tästä, sekä pelin iästä johtuen on lukuista ammattilaiset ja asiantuntijat vuosien mittaan kehitelleet shakista valtaisan määrän teoriaa erilaisista optimoiduista aloituksista lukuisiin loppupelin tilanteisiin, ja kaikkeen siltä väliltä. Suuren laskentatehon lisäksi perinteinen shakkikone pyrkii parhaansa mukaan hyödyntämään tätä saatavilla olevaa shakin teoriaa (Marcus 2018). Voidaan siis sanoa, että perinteiset shakkikoneet osaavat pelata ihmisten optimoimaa shakkia erinomaisella tasolla, mutta ihmiset eivät tunnetusti ole aina täysin oikeassa.

Kuten luvussa 2.3 mainittiin, AlphaZero ei käytä hyväkseen shakin teoriaa. Alkutietona shakista AlphaZerolle annettiin ainoastaan pelin säännöt sekä tieto tyypillisestä määrästä laillisia siirtoja pelin tilanteissa (Silver ym. 2017).

Tämän kappaleen alaluvuissa tutkitaan tarkemmin AlphaZeron syvän neuroverkon toiminnallisuutta. Myös AlphaZeron päätöksenteossa käytettyä Monte-Carlo -puuhakua käsitellään omassa alaluvussaan. Lopuksi tarkastellaan AlphaZeron opetusprosessia.

3.1 Rakenne

AlphaZeron syväoppivan neuroverkon rakenne on esitetty Silverin ym. artikkelissa (2017). AlphaZeron neuroverkko on muotoa

$$(p, v) = f_\theta(s). \tag{3.1}$$

Neuroverkko ottaa siis syötteeksi laudan senhetkisen tilanteen s, ja tuottaa arvoparin (ρ, v), jossa ρ on kaavassa 3.2 avattu vektori, joka kuvastaa jokaisen tilanteessa s tehtävän mahdol-

lisen liikkeen a todennäköisyyttä,

$$\rho = Pr(a|s). \tag{3.2}$$

Arvoparin (ρ, v) arvo v on muuttuja, joka arvioi oletetun arvon funktiolla E oletettua lopputulosta z tilanteesta s,

$$v \approx E[z|s]. \tag{3.3}$$

AlphaZero oppii tarkentamaan edellämainittuja liikkeiden todennäköisyyksiä sekä pelitilanteen arviointia neuroverkon painoarvoja θ muokkaamalla. Pelatessaan AlphaZero käyttää liikkeidensä valitsemiseen Monte-Carlo -puuhakua (engl Monte-Carlo Tree Search), jonka toimintaa tutkitaan tarkemmin luvussa 3.2. Opettaessa AlphaZero aloittaa satunnaisilla painoarvoilla θ ja jokaisen harjoituspelin jälkeen pelin tuloksesta muodostetaan lopputulos z. Neuroverkko saa palautteen l palautefunktiolla

$$l = (z-v)^2 - \pi^\top log\rho + c||\theta||^2, \tag{3.4}$$

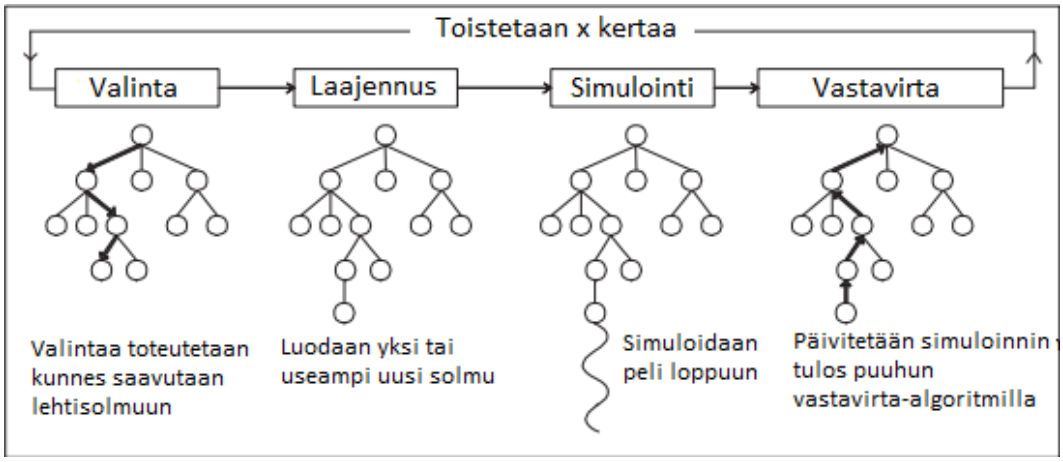
johon sisällytetään edellä mainittujen arvojen z, v, ρ ja θ lisäksi Monte-Carlo -puuhaun kehittelemät omat liikkeiden todennäköisyydet π, sekä vakio c hallitsemaan parametrien θ painoarvojen muokkaamisen tehokkuutta.

Parametreja päivittämällä pyritään minimoimaan pelin oletetun lopputuloksen v ja pelin todellisen lopputuloksen z eroavaisuus, sekä saamaan neuroverkon liikkeiden todennäköisyydet ρ ja Monte-Carlo -puuhaun tulokset π mahdollisimman samoiksi. Huomataan, ettei AlphaZero siis varsinaisesti pyri suoraan maksimoimaan voittoprosenttiaan, vaan sen sijaan pyrkii arvioimaan mistä tahansa tilanteesta seuraavaa lopputulosta mahdollisimman tarkasti. AlphaZeron neuroverkko vaikuttaa siis toimivan pelitilanteen arviointifunktiona.

3.2 Monte-Carlo -puuhaku

Monte-Carlo -puuhakua (lyhyesti MCTS) kuvataan Chaslotin ym. artikkelissa (2008) pelien tekoälyjen toimintaa tehostavaksi hakualgoritmiksi, jota luvussa 3.1 todettiin myös AlphaZeron käyttävän liikkeidensä valitsemiseen. Puurakenteen juurisolmu kuvastaa nykyistä

pelitilannetta, ja kaikki siitä erkanevat solmut edustavat tutkittuja liikkeitä, joihin tallennetaan kyseisen liikkeen lisäksi solmun käyntikerrat ja tieto solmun tilanteesta jatkuneen pelin lopputuloksesta (Sutton ja Barto 2018). MCTS:n toiminnan voi jakaa neljään osaan: valinta (engl. selection), laajennus (engl. expansion), simulointi (engl. simulation) ja vastavirta-algoritmi (engl. backpropagation) (Sutton ja Barto 2018). Puuhaun vaiheet on visualisoitu kuviossa 3.



Kuvio 3: Monte-Carlo -puuhaun neljä vaihetta (Chaslot ym. 2008).

Valintavaiheessa valitaan puusta siirtoja edeten solmuissa alaspäin. Liikkeitä pyritään valitsemaan luvussa 2.3 esiteltyyn tapaan priorisoiden tehokkaita siirtoja (exploitation), ja aika-ajoin tutkien tuntemattomampia variaatioita (exploration) (Chaslot ym. 2008). Miten valinta tarkalleen ottaen tapahtuu, riippuu ohjelman etsintämenetelmästä (engl. policy) (Sutton ja Barto 2018), joka luvussa 2.3 mainittua päätelmää mukaillen määritellään projektikohtaisesti parhaiten vastaamaan puuhakua hyödyntävän ohjelman tarpeita. AlphaZeron MCTS:n etsintämenetelmään vaikuttaa tuntemattomien liikkeiden tutkimiseen kannustava Dirichlet-melufunktio (engl. Dirichlet noise), jonka vahvuus riippuu go:n, shakin ja shogin tyypillisestä sallittujen liikkeiden määrästä eri peleissä (Silver ym. 2017).

Laajennusvaihe alkaa, kun valinta saapuu lehtisolmuun, jolloin seuraavan liikkeen valinta menee puun ulkopuolelle. Tämä valinta lisätään puuhun, joka varmistaa puun laajenemisen jokaisella toistokerralla (Chaslot ym. 2008). Joissain tapauksissa on myös mahdollista, että puuhun lisätään useampi uusi solmu (Sutton ja Barto 2018). Tämän jälkeen peli pelataan loppuun ilman puuhun kirjattuja liikkeitä. Tässä simulaatiovaiheessa voidaan valita täysin sa-

tunnaisia liikkeitä pelin loppuun saakka, mutta kuten Chaslotin ym. artikkelissa mainitaan, ei tämä ole optimaalinen ratkaisu, sillä satunnaisten liikkeiden tuottama satunnainen lopputulos ei heijasta puuhaun tekemiä valintoja, jolloin oppimista ei tapahdu (Chaslot ym. 2008). Sen sijaan päätöksentekoprosessia voi jatkaa joku toinen tekijä tai algoritmi, jolloin tuloksista saadaan parempia (Sutton ja Barto 2018). AlphaZeron tapauksessa loput pelistä simuloidaankin sen syväoppivalla neuroverkolla (Silver ym. 2017).

Kun etsintäkerran lopputulos on saatu simuloitua, se päivitetään uuteen/uusiin lehtisolmuihin sekä kaikkiin kyseisellä toistokerralla käsitellyihin puun solmuihin käyttäen vastavirta-algoritmia (Chaslot ym. 2008). Nyt puussa on yksi tai useampi uusi solmu, ja käsillä olevan hakukerran simulaation perusteella päätelty lopputulos on kirjattu kaikkiin kyseisellä toistokerroilla valittuihin puun solmuihin.

Yllä kuvattu prosessi toistetaan agentin toimiessa jokaisessa tilanteessa mahdollisimman monta kertaa kunnes yksittäiselle tilanteelle myönnetty aika loppuu (Sutton ja Barto 2018). Ajan loputtua agentti suorittaa eniten tutkitun toiminnon, sillä sen toiminnon arvio on käyntikertojen johdosta muita tarkempi (Chaslot ym. 2008). Tämän jälkeen puuhaun juurisolmu siirtyy valittuun liikkeeseen ja prosessia voidaan jatkaa seuraavalle siirrolle jatkaen tästä solmusta (Sutton ja Barto 2018). AlphaZeron ja Stockfishin välisissä peleissä osapuolille annettiin minuutti mietintäaikaa vuoroa kohti (Silver ym. 2017), eli AlphaZeron MCTS oli rajoitettu minuuttiin per siirto. MCTS:n tehokkuuden riippuvuus ajan määrästä ehdottaa, ettei MCTS välttämättä sovellu hyvin reaaliaikaisiin peleihin, joissa MCTS:llä olisi maksimissaan muutamia sekunteja aikaa toimia. Tätä ajatusta tukee Total War: ROME 2 -videopelin tekoälyn toteutus, jossa MCTS:ää on käytetty nimenomaan sen vuoropohjaisen kampanjan tekoälyssä, muttei sen reaaliaikaisten taisteluiden tekoälyssä (Champandard 2014).

AlphaZeron opetuksen aikana luvussa 3.1 mainitut MCTS:n neuroverkolle palauttamat liikkeiden todennäköisyydet π vastaavat opetuksen aikana tapahtuneen puuhaun eniten tutkittuja solmuja ja niistä oppivalla neuroverkolla simuloituja lopputuloksia (Silver ym. 2017).

3.3 Shakin opettaminen

Tähän mennessä on käyty läpi AlphaZeron käyttämät teknologiat, ja tässä alaluvussa tutkimme, miten AlphaZero hyödyntää niitä oppiessaan shakkia. Opetusprosessi on kuvattu Silverin ym. (2017) artikkelissa.

Opetuksen aikana AlphaZero ylläpitää yksittäistä syväoppivaa neuroverkkoa, jonka parametrit päivittyvät vahvistetun oppimisen kautta AlphaZeron pelatessaan itseään vastaan. AlphaZeron opetusdata koostui 700,000 dataerästä (engl. mini-batch), joista jokainen sisälsi 4,096 syötettä harjoitusdataa. Neuroverkon syöte on pelin tilannetta sekä tilanteesta tehtäviä laillisia liikkeitä kartoittava pino kuvia (engl. image stack). Pelien prosessointi toteutettiin Googlen tensoriprosessoreilla (engl. Tensor Processing Unit, lyhyesti TPU); 5000 ensimmäisen sukupolven TPU:ta käytettiin neuroverkon itseään vastaan pelaamisen simulointiin, ja 64 toisen sukupolven TPU:ta käytettiin neuroverkon opettamiseen. Googlen TPU:t ovat tehokkaita erityisesti neuroverkkojen opettamiseen erikoistuneita prosessoreja (Jouppi ym. 2017), joiden käyttö mahdollistaa AlphaZerolle huomattavan määrän luvussa 2.3 painotettua mahdollisuutta toteuttaa opetus käyttäen simulaatioita.

Opetus kesti kokonaisuudessaan noin yhdeksän tuntia, jonka aikana AlphaZeron Elo-luvun kasvua arvioitiin (shakissa pelaajan taitotasoa mitataan Elo-luvulla). Silverin ym. havaintojen mukaan AlphaZero saavutti Stockfishin Elon neljässä tunnissa. Lisäksi opetuksen aikana havainnointiin AlphaZeron pelityyliä ja todettiin AlphaZeron pelanneen useasti ihmisten kahtatoista suosituimpaa shakin aloitusta. Tämä voidaan tulkita ensimmäiseksi todisteeksi AlphaZeron mahdollisesti ihmisläheisemmästä pelityylistä, jota käsitellään enemmän luvussa 4.1.

3.4 Eroavaisuudet perinteisestä shakkikoneesta

Kaikki shakkikoneet ovat luonnollisesti jollain tapaa toisistaan poikkeavia, mutta perinteisen shakkikoneen malli vaikuttaa kuitenkin olevan pääpiirteittäin yhtäläinen (Oshri ja Khandwala 2016). Oshrin ja Khandwalan (2016) mukaan perineinen shakkikone koostuu pelitilannetta arvioivasta funktiosta, sekä mahdollisia liikkeitä etsivästä funktiosta, joka hyödyntää arviointifunktiota liikkeiden toteuttamiskelpoisuuden tutkimiseen. Täten perinteisen shak-

kikoneen rakennetta ja toimintaa voidaan verrata AlphaZeroon ottamalla esimerkiksi jokin edellämainittuun tapaan kehitetty shakkikone. Tässä tapauksessa AlphaZeron toteutusta verrataan Stockfishiin.

Stockfishilla on jokaiseen pelitilanteeseen s käytössään laaja kirjo erilaisia pelin kulkuun vaikuttavia arvoja $\phi(s)$, kuten nappuloiden materiaaliarvot eri kohdassa peliä, kuninkaan turvallisuus, sotilaiden muodostelma ja lähettiparit (Lai 2015). Jokaiselle arvolle ϕ_i määritellään myös painoarvo w_i vastaamaan kyseisen arvon vaikutusta pelin kulkuun (Silver ym. 2017). Painoarvojen määrittämistä käsin auttaa se, että jotkut arvot ovat shakin kirjoitetun teorian mukaan lähes, ellei täysin yksimielisesti toisia tärkeämpiä (esimerkiksi kuninkaan turvallisuus on lähes poikkeuksetta tärkempää kuin lähettiparin voittaminen). Lisäksi pelin alussa hyödynnetään aloituksien liikesarjojen teoriaa (Silver ym. 2017).

Näiden arvojen s, w ja ϕ avulla Stockfish arvioi pelin tilanteita lineaarikombinaatiolla

$$v(s,w) = \phi(s)^\top w. \tag{3.5}$$

Tämä funktio ei kuitenkaan pysty arvioimaan tarkasti tilanteita, joissa on mahdollista syödä toisen pelaajan nappuloita, joten tällaiset tilanteet Stockfish ratkaisee ensin käyttäen ”uinuvaa hakua” (engl. quiescence search). Uinuvasta hausta saadut ”hiljaiset” tilanteet voidaan tämän jälkeen arvioida yllä esitellyllä arviointifunktiolla (Silver ym. 2017).

Huomataan, että Stockfishin arviointifunktio vastaa tarkoitukseltaan AlphaZeron syvää neuroverkkoa, mutta ero näkyy funktioiden painoarvojen määrittelyssä. Stockfishille arvot määritellään käsin, kun taas AlphaZero muokkaa neuroverkonsa painoarvoja opetuksen aikana saamansa palautteen perusteella.

Päätös Stockfishin pelaamasta liikkeestä tehdään sen hakualgoritmia käyttäen, jolla pyritään karsimaan päätösprosessista pois todistettavasti huonoja vaihtoehtoja (Silver ym. 2017) käyttäen useita shakin heuristiikkoja. Esimerkiksi odottava siirto todetaan olevan aina heikko vaihtoehto, ja täten kaikki odottavat siirrot voidaan karsia pois. Vastaavasti hakufunktio voi pyrkiä priorisoimaan liikkeitä, jotka edellämainittujen heuristiikkojen mukaan vaikuttavat hyviltä siirroilta (Lai 2015). Esimerkiksi vastustajan vahvan pelinappulan syöminen omalla

heikolla pelinappulalla vaikuttaa hyvältä, joten tämän variaation etsintäsyvyyttä voidaan kasvattaa. Kuten Stockfishin arviointifunkton arvot ϕ, myös sen karsinta- ja hakualgoritmit ovat hyvin pitkälle hiottuja erikoistuneita shakkialgoritmeja, joiden soveltaminen muihin peleihin olisi työlästä (Silver ym. 2017). Tämäkin on täydellinen vastakohta AlphaZerolle, jonka algoritmi ja puuhaku soveltuvat samanaikaisesti shakkiin, shogiin ja go:hon.

Vaikka Stockfishin hakufunktio sisältää huomattavan määrän yksinomaan shakille hyödyllistä teoriaa, on jälleen sen ja AlphaZeron Monte-Carlo -puuhaun perimmäinen tarkoitus identtinen; molemmat pyrkivät löytämään tilanteeseen parhaan siirron. Huomataan siis, että sekä AlphaZeron syvä neuroverkko että sen MCTS vastaavat toiminnallisuuksiltaan perinteisen shakkikoneen kaksiosaista mallia, mutta osien toteutukset eroavat ratkaisevasti. Kummankin osan kohdalla selkeimpänä erona nähdään vaadittavan tiedon määrä; AlphaZero ei vaadi ollenkaan shakille spesifiä tietoa, mikä mahdollistaa sen soveltamisen muihinkin peleihin, kun taas Stockfishin toteutuksessa molemman funktion ulostulo riippuu täysin shakin teoriasta. AlphaZero ei kuitenkaan ole täysin ”tabula rasa”, kuten Silverin ym. (2017) artikkelissa väitetään. Marcus (2018) huomauttaa ettei Monte-Carlo puuhakua ole opittu datan perusteella, vaan se rakennettiin AlphaZerolle synnynnäisesti. Täten luvussa 2.3 esitelty kuvaus ”tabula rasa” -toteutuksesta ei täyty. Lisäksi luvussa 3.2 todettiin, että AlphaZeron MCTS:n etsintämenetelmään vaikuttava Diriclet-melufunktion painoarvo oli shakkia varten käsin määritelty.

AlphaZeron MCTS:n tekemä huonojen liikkeiden poissuljenta vaikuttaa olevan Stockfishiä tehokkaampi, mikä näkyy kummankin shakkikoneen tutkittujen tilanteiden määrästä. Stockfish ja sen hakufunktio tutkivat peräti 70 miljoonaa tilannetta sekunnissa, kun AlphaZero ja MCTS tutkivat vain 80 tuhatta eri tilannetta sekunnissa (Silver ym. 2017). Tästä huolimatta AlphaZero on onnistunut voittamaan Stockfishin sadan pelin mittaisessa ottelussa, jota käsitellään tarkemmin luvussa 4.2. Kuten entinen shakin maailmanmestari ja myös peleistään Deep Blue -shakkikonetta vastaan tuttu Garry Kasparov (2018) kirjoitti AlphaZerosta, AlphaZero ei työskentele kovemmin, vaan se työskentelee älykkäämmin.

On myös syytä uskoa, että AlphaZeron tilanteita arvioiva syvä neuroverkko on täysin opetettuna tarkempi kuin Stockfishin käsintehtyjä arvoja käyttävä arviointifunktio. Kasparov mainitsi artikkelissaan (2018), miten käsin määritellyt arvot riippuvat aina määrittelijän tulkinnasta, kun taas AlphaZeron neuroverkko oppii ilman minkäänlaista puolueellisuutta. Tätä

ajatusta tukee myös DeepMindin aikaisempi tutkimus Atari-tekoälystä, jossa ehdotetaan syväoppivan neuroverkon olevan kykenevä ylittämään tarkkuudellaan käsintehdyt määritelmät (Mnih ym. 2013).

AlphaZero ei vaikuta kuitenkaan olevan kaikin puolin Stockfishiä tehokkaampi. Luvussa 3.2 esitelty päätelmä MCTS:n heikkoudesta nopeissa peleissä vahvistuu myös Silverin ym. (2017) artikkelissa, jossa AlphaZeron Elo arvioitiin olevan Stockfishiä heikompi peleissä, joissa osapuolille vuoroa kohti myönnetty aika on noin $10^{-0.5}$ sekuntia tai vähemmän. Silverin ym. artikkelissa myös todetaan Stockfishin kykenevän tekemään täydellisiä siirtoja enintään 6-7 nappulaa sisältävissä loppupelin tilanteissa, joten loppupelissä Stockfish ja AlphaZero ovat tehokkuudeltaan enintään tasoissa. Lisäksi Wan ja Kaneko (2018) toteavat, että tuhansien Googlen tehokkaiden TPU:iden käyttäminen opetuksessa antaa AlphaZerolle huomattavan edun moniin muihin shakkitekoälyprojekteihin verrattuna, eikä AlphaZero tästä syystä yksinään takaa toteutuksensa tehokkuutta pienemmillä määrillä prosessointitehoa. Tästä huolimatta AlphaZeron voi todeta saavan aikaan erinomaisia tuloksia, joita tarkastellaan luvussa 4.

4 AlphaZero shakin pelaajana

Tässä luvussa tarkastellaan täysin oppineen AlphaZeron shakkitaitoja käsitellen sen pelityyliä ja tehokkuutta. Viimeisessä alaluvussa esitellään eräs AlphaZeron ja Stockfishin välinen shakkipeli esimerkkinä AlphaZeron pelityylistä.

4.1 Pelityyli ja tehokkuus

Kuten luvun 3 alussa todettiin, perinteisten shakkikoneiden shakin pelaaminen perustuu ihmisten kehittämään shakin teoriaan, sekä raakaan prosessointitehoon. Kasparov (2018) kuvasi monen perinteisen shakkikoneen pelityylin olevan hidas ja tasapeleihin kannustava.

Sen sijaan AlphaZero oppi pelaamaan shakkia kokonaan omalla tyylillään ilman minkäänlaista suuntaviivaa ihmisten pelaamista peleistä tai ihmisten laatimasta teoriasta, täten kehittäen kokonaan oman tyylinsä pelata shakkia. Kasparov kuvasi (2018) AlphaZeron pelityylin olevan dynaaminen ja avoin; se priorisoi aktiivisuutta materiallisen johdon sijaan, mikä johtaa agressiivisiin tilanteisiin, jotka näyttävät ihmisen silmään riskialttiilta. Bratko kertoo artikkelissaan (2018), miten ihmiset kompensoivat kyvyttömyyttään arvioida pelin tilannetta muutamaa liikettä pidemmälle pyrkimällä intuitiivisesti asemallista etua parantaviin siirtoihin. AlphaZeron asemallinen pelityyli ei sen sijaan johdu sen kyvyttömyydestä arvioida peliä pidemmälle, vaan juuri päinvastoin sen kyvystä tunnistaa asemallisten liikkeiden tuoma vaikutus lukuisia vuoroja eteenpäin (Bratko 2018). Vaikuttaa siis siltä, että vaikka ihmisten ja AlphaZeron syyt asemalliseen pelityyliin ovat täysin päinvastaiset toisiinsa nähden, muistuttaa AlphaZeron pelityyli enemmän ihmistä kuin shakkikonetta.

Luvussa 4.2 esitellään eräs AlphaZeron Stockfishiä vastaan pelaama peli, jossa AlphaZeron omalaatuinen pelityyli käy selkeästi ilmi.

4.2 AlphaZeron Immortal Zugzwang

AlphaZero pelasi vuoden 2017 lopussa Stockfishia vastaan sata shakkipeliä virallisilla turnaussäännöillä, poislukien lisättyä minuutin aikarajaa per siirto. AlphaZero pelasi saman-

aikaisesti samalla koneella Shakkia Stockfishia vastaan, shogia maailman johtavaa shogitekoälyä Elmoa vastaan, sekä go:ta pikaisesti koulutettua AlphaGo Zeroa vastaan (Silver ym. 2017). AlphaZero onnistui voittamaan vastustajansa jokaisessa pelissä, eikä hävinnyt shakissa Stockfishille kertaakaan. Sadasta pelaamastaan shakkipelistä AlphaZero voitti 28 ja pelasi tasan loput 72 peliä. Tässä alaluvussa esitellään yksi näistä peleistä, jossa AlphaZeron luvussa 4.1 mainittuja vahvuuksia käy ilmi selkeästi.

Kyseinen peli on nimetty AlphaZeron kuolemattomaksi Zugzwang -peliksi (Immortal Zugzwang). Shakkipeliä kutsutaan kuolemattomaksi, mikäli se on jollain tapaa erittäin mieleenpainuva peli, joka ei todennäköisesti tule unohtumaan shakinpelaajien keskuudessa. Zugzwang tarkoittaa saksan kielellä liikkumisen pakkoa, ja shakissa sitä käytetään kuvaamaan tilannetta, jossa ei ole olemassa hyvää siirtoa. Pelin nimen voi siis tulkita tarkoittavan ”ikimuistoista nurkkaanajoa.”

AlphaZero (valkoinen) vastaan Stockfish (musta):

1. Nf3 Nf6 2. c4 b6 3. d4 e6 4. g3 Ba6 5. Qc2 c5 6. d5 exd5 7. cxd5 Bb7 8. Bg2 Nxd5 9. O-O Nc6 10. Rd1 Be7 11. Qf5 Nf6 12. e4 g6 13. Qf4 O-O 14. e5 Nh5 15. Qg4

Pelin tilanne vuorolla 15 on esitetty kuvassa 4a. AlphaZeron omanlaatuinen ihmisistä riippumaton pelityyli näkyy tilanteessa hyvin. Teorian mukaan on shakin aloituksessa lähes poikkeuksetta suositeltavaa pyrkiä siirtämään lähetit, hevoset ja tornit parempiin asemiin niiden aloituspaikoista, mielummin kuin siirrellä yhtä tai muutamaa nappulaa edestakaisin. AlphaZero on kuitenkin vuoroon 16 mennessä liikuttanut kuningatartaan jo yhteensä neljä kertaa, ja sen kuningattaren puolen nappulat ovat vielä täysin lähtökuopissaan.

15. ... Re8 16. Nc3 Qb8 17. Nd5 Bf8 18. Bf4 Qc8 19. h3 Ne7 20. Ne3 Bc6 21. Rd6 Ng7 22. Rf6

Viimeisellä kahdella siirrolla AlphaZero on käyttänyt torniaan estämään Stockfishin laajentumisen rajoittaen vastustajansa pelitilaa (kuva 4b). Tämä on malliesimerkki AlphaZeron asemallisesta pelityylistä, sekä Kasparovin (2018) mainitsemasta AlphaZeron tavasta pelata todella riskialttiilta näyttäviä siirtoja. AlphaZeron torni näyttää ihmisen silmään olevan epämukavan ahtaalla, mutta AlphaZeron luvussa 4.1 mainitun pitkänäköisyyden ansiosta

AlphaZero kokee tornin olevan hyvässä asemassa. Seuraavaksi AlphaZero jatkaa siirroillaan Stockfishin pelitilan rajoittamista.

22. ... Qb7 23. Bh6 Nd5 24. Nxd5 Bxd5 25. Rd1 Ne6 26. Bxf8 Rxf8 27. Qh4 Bc6 28. Qh6 Rae8 29. Rd6 Bxf3 30. Bxf3 Qa6 31. h4 Qa5 32. Rd1 c4 33. Rd5 Qe1+ 34. Kg2 c3 35. bxc3 Qxc3 36. h5 Re7 37. Bd1 Qe1 38. Bb3 Rd8 39. Rf3 Qe4 40. Qd2 Qg4 41. Bd1 Qe4 42. h6 Nc7 43. Rd6 Ne6 44. Bb3

44. Bb3 on erinomainen siirto, jonka myötä AlphaZero uhraa keskuksen sotilaansa saadakseen lähettinsä hyvään asemaan ja ajaakseen Stockfishin kuningattaren pois aktiivisesta ruudusta (kuva 4c).

44. ... Qxe5 45. Rd5 Qh8 46. Qb4 Nc5

Tässä tilanteessa (kuva 4d) AlphaZero pelaa toisen asemallisen uhrauksen 47. Rxc5 (kuva 4e). AlphaZero uhraa torninsa saadakseen Stockfishin ainoan aktiivisen nappulan pois pöydältä, minkä jälkeen AlphaZero voi viimeistellä kuristusotteensa.

47. Rxc5 bxc5 48. Qh4 Rde8 49. Rf6 Rf8 50. Qf4 a5 51. g4

Tilanne 4f on Zugzwang. Stockfishillä ei ole tässä tilanteessa hyvää siirtoa.

51. ... d5 52. Bxd5 Rd7 53. Bc4 a4 54. g5 a3 55. Qf3 Rc7 56. Qxa3 Qxf6

Stockfish saa purettua Zugzwangin vain uhraamalla kuningattarensa (kuva 4g), minkä jälkeen AlphaZerolla on ratkaisevan suuri materiaalietu.

57. gxf6 Rfc8 58. Qd3 Rf8 59. Qd6 Rfc8 60. a4 1-0

Zugzwangin jälkeen pelattiin vielä muutama siirto, mutta Stockfishin materiaalitappiot saivat sen lopulta luovuttamaan. Pelin lopputilanne näkyy kuvassa 4h. AlphaZeron omintakeiset liikkeet ja asemallinen nerokkuus saavutti voiton Stockfishista ja ikuisti sen kuolemattomaksi Zugzwang -peliksi.

(a) 15. Qg4

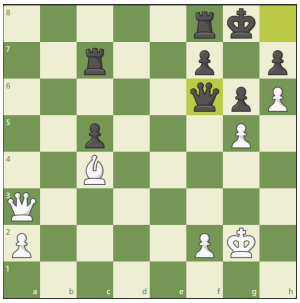
(b) 22. Rf6

(c) 44. Bb3

(d) 46. ... Nc5

(e) 47. Rxc5

(f) 51. g4, Zugzwang



(g) 56. ... Qxf6

(h) 60. a4, Stockfish luovuttaa

Kuvio 4: Kahdeksan vaihetta AlphaZeron Immortal Zugzwang -pelistä

5 Yhteenveto

Tutkielmassa pyrittiin selvittämään, miten AlphaZero on kehitetty, miten sen käyttämät teknologiat toimivat sekä millä tapaa AlphaZero eroaa perinteisestä shakkikoneesta.

Todettiin, että AlphaZero on syvää neuroverkkoa ja Monte-Carlo -puuhakua käyttävä tekoäly, jolle opetettiin shakkia, shogia ja go:ta käyttäen vahvistettua oppimista. AlphaZero oppi shakkia pelamaalla itseään vastaan yhdeksän tunnin ajan saaden pelatessaan neuroverkon sekä MCTS:n tulosten perusteella palautetta, jonka avulla neuroverkko oppi tarkemmaksi. Täysin oppinut AlphaZero onnistui voittamaan tehoikkaimman shakkikoneen titteliä kantaneen Stockfishin. Vahvistetun oppimisen arvioitiin soveltuvan hyvin shakin kaltaiseen peliin, jota on helppoa simuloida, ja jossa suurimmalle osalle tilanteista ei voida määritellä yhtä parasta mahdollista ratkaisua. AlphaZeron pidättäytyminen saatavilla olevasta shakin teoriasta mahdollisti AlphaZeron ihmisistä riippumattoman oppimisen.

Monte-Carlo -puuhaun huomattiin olevan tehokas vähempiin mutta parempiin liikkeisiin suuntautunut hakualgoritmi AlphaZerolle, mikä ilmeni AlphaZeron ja Stockfishin tutkittujen liikkeiden määrästä. MCTS:n tarkkuuteen vaikuttaa kuitenkin sille myönnetty ajan määrää toimia, mikä voi tuottaa ongelmia ajan pienentyessä.

Havaittiin, että AlphaZero pelaa tietokoneshakissa harvinaista avointa ja aggressiivista tyyliä, ja pyrkii hyödyntämään asemallista etua enemmän kuin perinteiset shakkikoneet. AlphaZeron omalaatuinen pelityyli johtuu sen riippumattomuudesta shakin teoriaan, jonka perusteella perinteiset shakkikoneet toimivat.

Huomattiin myös, että vaikka AlphaZeron taitotaso ja ennen kaikkea pelityyli vaikuttavat olevan muihin shakkikoneisiin verrattuna omaa luokkaansa, vastaa AlphaZeron rakenne yllättävän paljon perinteisten shakkikoneiden kaksiosaista mallia. AlphaZeron neuroverkko vastaa tilanteiden arvioinnista, ja pelattavat liikkeet valitaan Monte-Carlo -puuhaun avulla. Ratkaisevat eroavaisuudet ilmenevät kuitenkin kummankin osan toteutuksessa. Vaikuttaa siltä, että neuroverkon puolueeton painoarvojen säätely, sekä Monte-Carlo -puuhaun ja neuroverkon tekemä yhteistyö saavat aikaan erinomaisia tuloksia, mikä näkyy AlphaZeron peleissä Stockfishiä vastaan.

Lähteet

Wan, Shanchuan, ja Tomoyuki Kaneko. 2018. “Building evaluation functions for chess and shogi with uniformity regularization networks”. Teoksessa *2018 IEEE Conference on Computational Intelligence and Games (CIG),* 1–8. IEEE.

Wiering, Marco, ja Martijn Van Otterlo. 2012. “Reinforcement learning”. *Adaptation, learning, and optimization* 12:3.

Wiley, Timothy, Claude Sammut ja Bernhard Hengst. 2015. “A multi-strategy architecture for on-line learning of robotic behaviours using qualitative reasoning”. Teoksessa *Proceedings of the Third Annual Conference on Advances in Cognitive Systems ACS,* 16.

Alpaydin, Ethem. 2016. *Machine Learning : The New AI* [kielellä English]. ID: 4714219. Cambridge: MIT Press. ISBN: 9780262337595.

Bratko, Ivan. 2018. “AlphaZero - Whats Missing?” *Informatica* 42, No 1 (2018).

Caterini, Anthony L, ja Dong Eui Chang. 2018. *Deep Neural Networks in a Mathematical Framework.* Springer.

Champandard, Alex J. 2014. “Monte-Carlo tree search in TOTAL WAR: ROME II’s campaign AI”. *AIGameDev. com: http://aigamedev. com/open/coverage/mcts-rome-ii.*

Chaslot, Guillaume, Sander Bakkes, Istvan Szita ja Pieter Spronck. 2008. “Monte-Carlo Tree Search: A New Framework for Game AI.” Teoksessa *AIIDE.*

Jouppi, Norman P, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers ym. 2017. “In-datacenter performance analysis of a tensor processing unit”. Teoksessa *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA),* 1–12. IEEE.

Kasparov, Garry. 2018. “Chess, a Drosophila of reasoning”. *Science* 362 (6419): 1087–1087. ISSN: 0036-8075. doi:10.1126/science.aaw2221. eprint: http://science.sciencemag.org/content/362/6419/1087.full.pdf. http://science.sciencemag.org/content/362/6419/1087.

Lai, Matthew. 2015. “Giraffe: Using Deep Reinforcement Learning to Play Chess”. *CoRR* abs/1509.01549. arXiv: 1509.01549. http://arxiv.org/abs/1509.01549.

LeCun, Yann, Yoshua Bengio ja Geoffrey Hinton. 2015. “Deep learning”. *nature* 521 (7553): 436.

Marcus, Gary. 2018. “Innateness, AlphaZero, and Artificial Intelligence”. *CoRR* abs/1801.05667. arXiv: 1801.05667. http://arxiv.org/abs/1801.05667.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra ja Martin Riedmiller. 2013. *Playing Atari with Deep Reinforcement Learning.* arXiv: 1312.5602 [cs.LG].

Mohammadi, Mehdi, ja Ala Al-Fuqaha. 2018. “Enabling cognitive smart cities using big data and machine learning: Approaches and challenges”. *IEEE Communications Magazine* 56 (2): 94–101.

Mohammed, Mohssen, Muhammad Badruddin Khan ja Eihab Bashier Mohammed Bashier. 2017. *Machine Learning : Algorithms and Applications.* CRC Press. ISBN: 9781498705387. http://search.ebscohost.com.ezproxy.jyu.fi/login.aspx?direct=true&db=nlebk&AN=1293656&site=ehost-live.

Nielsen, Michael A. 2015. *Neural networks and deep learning.* Nide 25. Determination press USA.

Oshri, Barak, ja Nishith Khandwala. 2016. *Predicting moves in chess using convolutional neural networks.*

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot ym. 2017. “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm”. *CoRR* abs/1712.01815. arXiv: 1712.01815. http://arxiv.org/abs/1712.01815.

Sutton, Richard S, ja Andrew G Barto. 2018. *Reinforcement learning: An introduction.*