Ihmismäisen pelityylin tavoittelu shakkiohjelmissa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Joulukuu 2024
Roni Nousiainen

TURUN YLIOPISTO
Tietotekniikan laitos

Roni Nousiainen: Ihmismäisen pelityylin tavoittelu shakkiohjelmissa

TkK-tutkielma, 32 s.
Tietotekniikka
Joulukuu 2024

Shakki on kiinnostanut ihmisiä jo vuosisatoja, ja shakin koneellistaminen on pitkään ollut yksi tekoälyn parhaista soveltamiskohteista. Tämä kandidaatintutkielma on kirjallisuuskatsaus ihmismäisten shakkiohjelmien kehityksestä. Tässä tutkielmassa selvitetään, miten shakkiohjelmat toimivat ja minkälaisia tekoälyn menetelmiä niissä käytetään. Tutkielmassa tutkitaan erityisesti sitä, miten shakkiohjelmissa on tavoiteltu ihmismäistä pelityyliä. Lopuksi tutkielmassa pohditaan ihmismäisten shakkiohjelmien potentiaalisia sovelluskohteita muun muassa pelaajien pelityylien tunnistamisessa ja huijauksenestossa. Tuloksissa todetaan, että ihmismäiset shakkiohjelmat ovat edistyneet huomattavasti viime vuosina ja niiden kehitys jatkunee vielä hyvin tehokkaasti.

Asiasanat: shakki, shakkiohjelma, ihmismäinen, Maia, imitaatio-oppiminen, koneoppiminen, NNUE

Sisällys

1 Johdanto

Shakki on peli, joka on jossain muodossa kiinnostanut ihmisiä jo ainakin 600-luvulta alkaen. Shakissa kaksi pelaajaa kilpailee toisiaan vastaan 64 ruudun laudalla täyden informaation symmetrisessä strategiapelissä. Myös koneellinen shakki on ollut ihmisten kiinnostuksen kohteena jo pitkään. Tietokoneiden kehittyessä shakkiohjelmien tavoitteena oli yksinomaan saavuttaa pelitasoltaan vahvempia ohjelmia, mutta viime vuosikymmeninä ohjelmat ovat ohittaneet parhaatkin ihmiset pelitasoltaan [1]. Tämän myötä ihmiset ovat kiinnostuneet myös ihmismäisesti pelaavista shakkiohjelmista. Ihmismäisen pelityylin tavoittelu shakkiohjelmissa on edelleen avoin ongelma, johon ei ole keksitty täydellistä ratkaisua, sillä ohjelmien tapa valita pelattavat siirrot on hyvin erilainen kuin ihmisillä. Ongelmaa on kuitenkin viime vuosina lähestytty monilla eri tavoilla varsin hyvällä menestyksellä [2][3].

Tämän tutkielman tarkoitus on selvittää, missä vaiheessa ihmismäisten shakkiohjelmien kehitys on. Tutkielmassa selvitetään myös, mitä shakkiohjelmia on olemassa ja miten ne käytännössä toimivat. Tutkielmassa pyritään vastaamaan kolmeen tutkimuskysymykseen:

TK1: Mitä shakkiohjelmia on olemassa?

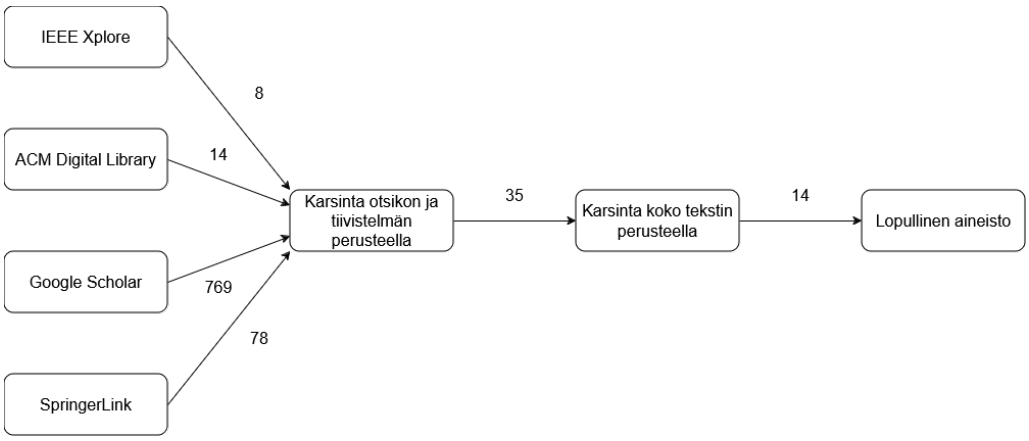
TK2: Minkälaisia tekoälyn menetelmiä ohjelmat hyödyntävät?

TK3: Miten shakkiohjelmien pelityylistä saisi tehtyä ihmismäisemmän?

Tämä tutkielma on toteutettu kirjallisuuskatsauksena. Tietoa on haettu neljästä eri

tietokannasta hakulauseella *(”chess program*” OR ”chess engine*”) AND (”human-like” OR ”human like”).*

Tiedonhaussa käytetyt tietokannat ovat IEEE Xplore, ACM Digital Library, SpringerLink ja Google Scholar. Hakulauseen muodostuksessa parhaita tuloksia tuotti hakulause, jossa haettiin termejä ”chess program” ja ”chess engine”, sillä näitä käytetään englanninkielisissä teksteissä synonyymeinä. Toisena kriteerinä hakulauseessa oli keskittyminen erityisesti ihmismäisiin ohjelmiin, jossa eniten tuloksia tuli termin ”human-like” eri kirjoitusasuista. Tämän lisäksi tutkielmassa on käytetty shakkiohjelmien toiminnan ja historian yleiseen esittelyyn lähteenä eri verkkosivuja ja artikkeleja, jotka eivät liity suoraan ihmismäisiin ohjelmiin. Nämä eivät sisälly kuvassa 1.1 esiteltyihin hakutuloksiin.



Kuva 1.1: Hakutulosten rajausprosessi

Hakutuloksia on rajattu ensisijaisesti julkaisukielen perusteella. Tämän jälkeen julkaisuja on karsittu otsikon ja tiivistelmän perusteella, minkä jälkeen osa tuloksista on karsittu koko tekstin perusteella. Lopulliseen aineistoon jäi neljätoista tulosta.

Tutkielman lukijalta odotetaan tietotekniikan kandidaattitason pohjatiedot. Näihin pohjatietoihin sisältyviä termejä ja algoritmeja ei ole tutkielmassa erikseen selitetty.

Tutkielman toisessa luvussa esitellään shakki pelinä ja kerrotaan lyhyesti shakin historiasta sekä ihmisten että koneiden pelaamana. Kolmannessa luvussa tutustutaan tarkemmin shakkiohjelmien toimintaan ja eritellään ne kahteen eri luokkaan, perinteistä tekoälyä käyttäviin ohjelmiin ja koneoppimisohjelmiin, sekä kerrotaan näiden välisistä eroista. Neljännessä luvussa tarkastellaan ihmismäisiä shakkiohjelmia ja selvitetään, miten niiden kehitystä on lähestytty viime vuosina. Viidennessä luvussa vastataan tutkimuskysymyksiin ja pohditaan ihmismäisten shakkiohjelmien hyötyjä ja mahdollisia ongelmia sekä niiden tulevaisuutta.

2 Tausta

Tässä luvussa tarkastellaan shakin historiaa sekä ihmisten että tietokoneiden pelaamana. Ensimmäisessä alaluvussa käsitellään shakkia pelinä. Tämän alaluvun tiedot perustuvat kansainvälisen shakkiliiton julkaisemiin virallisiin shakin sääntöihin [4]. Toisessa alaluvussa tutkitaan shakin kehittymistä sen nykyiseen muotoonsa, ja kolmannessa alaluvussa tarkastellaan shakkitietokoneiden historiaa lyhyesti.

2.1 Shakki pelinä

Shakki on pelinä hyvin yksinkertainen kahden pelaajan täyden informaation lautapeli. Pelissä molemmat pelaajat hallitsevat kuudentoista nappulan armeijaansa, joka koostuu kahdeksasta sotilaasta, kahdesta tornista, kahdesta ratsusta, kahdesta lähetistä, yhdestä kuningattaresta ja yhdestä kuninkaasta. Pelin idea on uhata vastustajan kuningasta siten, että vastustaja ei voi millään pelin sääntöjen puitteissa sallitulla (”laillisella”) siirrolla tätä uhkaa torjua.

Pelin kulku

Pelin alussa valkoisilla nappuloilla pelaava pelaaja (”valkea”) tekee ensimmäisen siirron. Tämän jälkeen mustilla nappuloilla pelaava pelaaja (”musta”) tekee seuraavan siirron ja pelaajat jatkavat vuorotellen siirtojen tekemistä, kunnes peli päättyy. Useimmiten pelaajan siirtoihin käyttämä aika on rajoitettu. Tavallisesti yhden pelaajan peliin käytettävissä oleva aika vaihtelee kolmesta minuutista kahteen tuntiin,

mutta varsinkin tietokoneella pelattavissa peleissä aika voi olla vielä lyhyempikin, jopa vain viisitoista sekuntia. Pidemmätkin aikarajat ovat mahdollisia erityisesti kirjeshakkia pelattaessa. Yhden siirron tekemiseen voi olla aikaa monia päiviä ja yksi peli voi kestää jopa vuosia.

Shakkipeli voi päättyä kolmella eri tavalla: valkean voittoon, mustan voittoon tai tasapeliin. Pelin voittaa pelaaja, joka uhkaa vastustajan kuningasta siten, että vastustaja ei voi tätä uhkaa millään tavalla estää. Tällaista pelitilannetta kutsutaan shakkimatiksi. Peli voi päättyä pelaajan voittoon myös, mikäli vastustaja ylittää peliin varatun aikansa tai luovuttaa. Tasapeliin peli voi päättyä monella eri tavalla, joista yleisin lienee sovittu tasapeli. Tällöin toinen pelaaja ehdottaa tasapeliä, ja peli päättyy, mikäli vastustaja tämän hyväksyy. Aloittelijoiden välisissä peleissä peli useimmiten päättyy jommankumman pelaajan voittoon, mutta huippupelaajien, ja varsinkin tietokoneohjelmien, välisissä peleissä yleisin tulos on tasapeli.

Tähän päivään mennessä shakkia pelinä ei ole ratkaistu, eli ei tiedetä, päättyykö peli optimaalisesti pelattuna valkean voittoon, tasapeliin vai mustan voittoon. On epätodennäköistä, että tätä ratkaisua peliin ikinä löydetäänkään, sillä shakin pelipuu on valtava. Jo vuonna 1950 on osattu arvioida erilaisia shakkipelejä olevan jopa 10^{120} kappaletta, ja saavutettavissa olevia peliasemiakin noin 10^{43} [5]. Moderni analyysi saavutettavien peliasemien määrästä on tarkentanut tätä arvioita, ja tänä päivänä paras arvio tälle on $(4{,}59 \pm 0{,}38) \cdot 10^{44}$ [6].

Siirtojen luokittelu

Shakkipelissä tehtäviä siirtoja pyritään usein luokittelemaan niiden hyvyyden perusteella. Ennen modernien shakkiohjelmien kehitystä tämä luokittelu saattoi olla hyvinkin subjektiivista, sillä se perustui pelkästään ihmisten mielipiteisiin kyseisestä siirrosta, mutta nykyään ihmisiä paljon vahvemmat ohjelmat osaavat arvioida asemia hyvinkin luotettavasti. Shakkia pelaavat ohjelmat osaavat antaa tietystä pe-

litilanteesta jonkinlaisen arvion, ja siirron laatua voidaan arvioida tämän arvion muutoksen perusteella.

Yleinen käytäntö on, että ohjelmat antavat arvion desimaalilukuna, jossa positiivinen arvo tarkoittaa valkean pelaajan etua ja negatiivinen arvo mustan pelaajan etua. Mitä lähempänä arvo on nollaa, sitä tasaisempana kyseinen ohjelma pitää asemaa. [5]

2.2 Shakin kehitys nykyiseen muotoonsa

Shakkipelin historia on pitkä, mutta täyttä varmuutta pelin alkuperästä ei vieläkään ole. Pelin tiedetään saaneen alkunsa viimeistään 600-luvulla, mutta aiempiakin alkuperiä pidetään mahdollisena. Aikaisimmat vahvistettavat tiedot shakin kaltaisen pelin olemassaolosta löytyvät intialaisesta kirjallisuudesta, josta on löydetty mainintoja *chaturanga*-nimisestä pelistä juuri 600-luvun teksteistä. On mahdollista, että peli on ollut olemassa aiemminkin, sillä 64 ruudun (kahdeksan kertaa kahdeksan) pelilauta, *ashtapada*, on mainittu jo 100-luvun teksteissä, mutta ilman tietoa siitä, minkälaista peliä tällä pelataan. [7] [8]

Vuosisatojen kuluessa peli levisi ympäri maailmaa ja sitä pelattiin monilla eri säännöillä. Nykyiset säännöt levisivät Euroopassa 1400-luvun loppupuolella, mutta tarkkaa vuotta tälle ei tiedetä [9]. Vanhin shakkipeli, jonka siirrot tunnetaan nykyisten sääntöjen mukaisina, on peräisin runosta *Scachs d'amor*. Tämä runo sisältää dokumentoidut siirrot kyseisestä pelistä runomuodossa. [10]

2.3 Tietokoneshakin historia

Koneellinen shakki on kiinnostanut ihmisiä jo pitkään. Shakkia pelaavista koneista haaveiltiin jo pitkään ennen tietokoneiden keksimistä; 1700-luvun lopulla unkarilainen Wolfgang von Kempelen väitti kehittäneensä shakkia pelaavan koneen, joka

tunnettiin Turkkilaisena (engl. *The Mechanical Turk*). Tämä kone kiersi ympäri maailmaa yli kahdeksan vuosikymmenen ajan, ennen kuin se tuhoutui tulipalossa vuonna 1854. Vaikka konetta epäiltiin huijaukseksi jo aikanaan, varmistuivat epäilyt todeksi vasta koneen tuhoutumisen jälkeen, kun koneen viimeisen omistajan poika paljasti koneen toiminnan: todellisuudessa pöydän sisällä oli vahva shakkimestari, joka liikutteli Turkkilaisen käsiä ja teki siirrot tämän puolesta. [11]

Jopa ensimmäinen todellinen shakkiohjelma tehtiin aikana, jolloin sen ajamiseen kykeneviä tietokoneita ei vielä ollut olemassa. Tämä oli Alan Turingin kehittämä *Turochamp*, joka perustui hyvin yksinkertaiseen algoritmiin, jonka arvo määritteli seuraavan siirron. Turingin aikana tätä ohjelmaa käytettiin laskemalla algoritmin tulos käsin. Historian ensimmäiseksi todelliseksi shakkikoneeksi *Turochamp* oli itse asiassa varsin hyvä; shakin kolmastoista maailmanmestari Garri Kasparov on todennut, että hyvin nopeassa pelissä, jossa pelaajilla on vain viisi sekuntia aikaa tehdä seuraava siirto, ohjelma voisi voittaa suurimman osan harrastelijashakinpelaajista. [12]

Tietokoneiden kehittyessä myös shakkia pelaavat ohjelmat kehittyivät. Vuonna 1950 shakin koneellistaminen alkoi todella, kun Claude Shannon julkaisi artikkelinsa koneellisesta shakista, jossa hän määritteli kaksi eri luokkaa shakkia pelaaville koneille. Shannonin luokan A ohjelma on ohjelma, joka laskee puuhaussa kaikki siirrot minimax-algoritmin tyyliin. Luokan B ohjelma taas laskee vain osan mahdollisista siirroista. [5] Shannonin luokat määrittelivät useamman vuosikymmenen ajan shakkiohjelmien kehitystä. Suurin osa kehittäjistä, mukaan lukien Shannon itse, suosi luokan B ohjelmia, sillä hakupuuta läpikäytäessä kaikkien siirtojen tutkimista pidettiin liian vaativana. Shakkiohjelmat kehittyivät hyvin nopeasti vastaamaan tasoltaan amatööripelaajia. Jo vuonna 1967 Shannonin luokan B ohjelma nimeltään *Mac Hack* voitti amatööripelaaja ja tekoälykriitikko Hubert Dreyfusin pelissä. Tietokoneiden kehittyessä Shannonin luokan B ohjelmat kuitenkin jäivät taka-alalle, sillä tietyssä asemassa mahdollisten siirtojen nopea karsiminen ilmeni haastavak-

si. Reilua vuosikymmentä myöhemmin, vuonna 1978, Shannonin luokan A ohjelma *Chess 4.0* hävisi viiden pelin ottelussa kansainväliselle mestarille (engl. *International Master*) pistein 1,5–3,5. [13] Vaikka ohjelma hävisi ottelun, se voitti yhden peleistä ja sai tasapelin toisesta todistaen, että ohjelmat voivat pärjätä jopa varsin vahvoille pelaajille.

Meni kuitenkin vielä kaksi vuosikymmentä, ennen kuin parhaat ohjelmat alkoivat todella kilpailla parhaiden ihmisten kanssa. Vuonna 1997 IBM:n kehittämä *Deep Blue* voitti hallitsevan maailmanmestarin Garri Kasparovin ottelussa. [14] Tämäkin otteluvoitto oli melko vaikea, sillä *Deep Bluen* shakkiohjelma ei ollut pelivahvuudeltaan huomattavasti muita ohjelmia vahvempi, vaan sen suurin etu muihin shakkikoneisiin verrattuna oli sen laitteisto. *Deep Blue* -järjestelmää ajettiin nimittäin aikaansa nähden erittäin tehokkaalla tietokoneella, joka oli suunniteltu arkkitehtuuria myöten vain ja ainoastaan shakin pelaamiseen. [15] *Deep Bluen* voiton myötä huipputason ihmiset pelasivat ohjelmia vastaan vain harvoin, ja tällöinkin ohjelmia ajettiin usein aikansa supertietokoneilla. Siksi on hieman epäselvää, milloin shakkiohjelmat todella ohittivat parhaat ihmispelaajat ajettaessa ohjelmaa ns. tavallisella tietokoneella.

Ohjelmien ja huipputason ihmispelaajien väliset ottelut jatkuivat *Deep Bluen* ja Kasparovin välisen ottelun jälkeen vielä vuosikymmenen ajan. Vaikka ohjelmien etumatka ihmisiin kasvoi koko tämän ajan, kaikissa näissä otteluissa ihmispelaajat pystyivät voittamaan yksittäisiä pelejä ja saamaan moniakin tasapelejä. Lopullisesti ohjelmien todettiin ohittaneen parhaatkin ihmiset vuonna 2006, kun ohjelma *Deep Fritz* voitti hallitsevan maailmanmestarin Vladimir Kramnikin kuuden pelin ottelussa pistein 4–2, häviämättä kertaakaan. Tämän jälkeen kaikissa merkittävissä otteluissa ihmispelaajien ja tietokoneiden välillä ohjelmat ovat antaneet jonkinlaista tasoitusta ihmisille. [16]

3 Shakkiohjelmat

Tässä luvussa esitellään shakkiohjelmien toimintaperiaatteita ja pohditaan niiden ongelmia. Ensimmäisessä alaluvussa keskitytään perinteisen tekoälyn menetelmiin perustuviin ohjelmiin. Toisessa alaluvussa esitellään koneoppimiseen perustuvia ohjelmia ja niiden eroavaisuuksia perinteisiin ohjelmiin.

3.1 Perinteistä tekoälyä käyttävät ohjelmat

Varhaisimmat shakkiohjelmat perustuivat hyvin yksinkertaisiin menetelmiin. Esimerkiksi *Turochamp* laski jokaiselle siirrolle tietyn arvon yksinkertaisen algoritmin avulla ja näiden perusteella valitsi niistä parhaan [12]. Vasta kun tietokoneiden laskentateho parani, todellisia shakkiohjelmia voitiin alkaa toteuttaa. Tunnetuin esimerkki varhaisesta shakkiohjelmasta lienee ensimmäisenä tietokoneohjelmana shakin hallitsevan maailmanmestarin voittanut IBM:n *Deep Blue*. Varhaisista shakkiohjelmista puhuttaessa on kuitenkin hyvä tiedostaa, että niiden menestys perustui ohjelmiston lisäksi myös juuri shakin pelaamista varten tarkoin suunniteltuun laitteistoon. Tässä tutkielmassa keskitytään kuitenkin vain ohjelmistoon.

Perinteiset shakkiohjelmat, aina varhaisimmista ohjelmista 2010-luvun lopulle asti, toimivat hyvin pitkälti samoja periaatteita noudattaen. Ohjelma koostuu kahdesta eri osasta: käsintehdystä evaluaatiofunktiosta ja alfa–beeta-karsitusta puuhausta. Ohjelman toimintaperiaate on yksinkertainen: puuhaku käy läpi pelipuuta, ja jokaisessa asemassa ajetaan evaluaatiofunktio antamaan tästä kyseisestä asemasta

jonkinlainen arvio. [15] Yksi keskeinen kysymys perinteisiä shakkiohjelmia tehtäessä on: millainen on mahdollisimman hyvä evaluaatiofunktio, jonka ajamalla saa mistä tahansa laudan asemasta hyvin nopeasti numeerisen arvion, joka kuvastaa hyvin eri pelaajien voiton mahdollisuuksia?

Evaluaatiofunktio

Hyvällä evaluaatiofunktiolla on siis kaksi vaatimusta, joista ensimmäinen määrää, että sen ajaminen ei saa kestää kauan, sillä mitä nopeammin yksi asema voidaan arvioida, sitä useampi asema ehditään tietyn ajanjakson aikana tutkia. Toinen taas vaatii evaluaatiofunktiolta tarkkuutta; on tärkeää, että ohjelma osaa arvioida asemia tarkasti. Jos ohjelma arvioi tietynlaisen aseman systemaattisesti väärin, tällöin ohjelma joko tavoittelee häviävään asemaan pääsyä tai välttää voittavaan asemaan pääsyä. Tässä vaiheessa on hyvä tiedostaa, että evaluaatiofunktion tulos tietylle asemalle ei tarkoita samaa kuin koko ohjelman antama arvio asemalle. Evaluaatiofunktio antaa arvion perustuen pelkästään kyseisen aseman piirteisiin eikä se huomioi tulevia siirtoja millään tavalla. On siis hyvin tyypillistä, että evaluaatiofunktio voi esimerkiksi antaa voittavan arvion asemalle, joka todellisuudessa on häviävä. Tämä väärä arvio korjataan puuhaun myötä, kun ohjelma tutkii pelipuuta syvemmälle ja pääsee ns. ”hiljaiseen asemaan” (engl. *quiescent position*), jossa evaluaatiofunktion antama tulos on todennäköisesti varsin luotettava [15]. Puuhakua ja sen karsimista käsitellään lisää myöhemmin.

Perinteisissä shakkiohjelmissa evaluaatiofunktio on useimmiten erittäin vahvojen shakinpelaajien käsin muodostama funktio (engl. *Hand-Crafted Evaluation*, HCE), joka ottaa huomioon monia eri piirteitä asemassa[1] ja laskee niiden perusteella yksinkertaisen algebrallisen summan [3]. Jotkin ohjelmat ovat käyttäneet evaluaatio-

[1]Tässä tutkielmassa ei tarkastella tämän tarkemmin, mitä piirteitä evaluaatiofunktiot ottavat huomioon asemaa arvioidessa. Mikäli lukija on tästä kiinnostunut, parhaiten tietoa löytynee eri avoimen lähdekoodin ohjelmien lähdekoodista ja/tai dokumentaatioista.

funktion parametrien hienosäätöön myös koneoppimista antamalla ohjelman pelata itseään vastaan eri parametrien arvoilla [17].

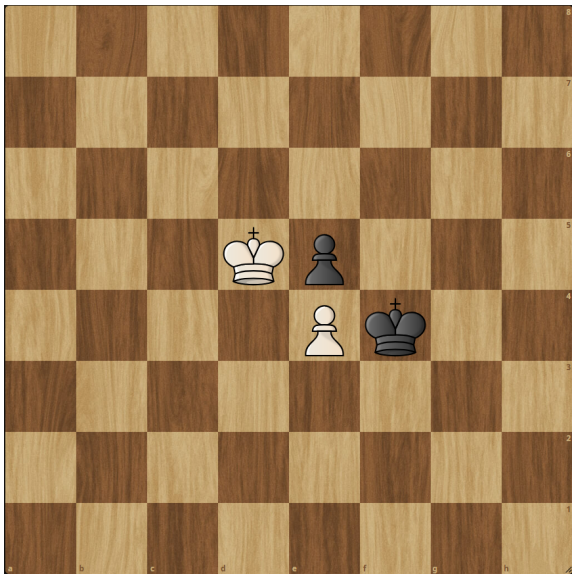
Hyvä evaluaatiofunktio on kuitenkin vain osa hyvää shakkiohjelmaa. Jotta ohjelma todella osaisi pelata peliä, tulee siinä olla myös puuhakualgoritmi, joka käy läpi pelipuuta. Yksinkertainen puuhakualgoritmi shakin kaltaisille kahden pelaajan peleille on minimax-algoritmi, mutta on ilmeistä, että shakin pelipuu on aivan liian suuri pelkällä minimax-haulla läpikäytäväksi. Hakua täytyy siis jotenkin karsia, mutta miten?

Alfa–beeta-haku

Puuhaku perinteisissä shakkiohjelmissa perustuu lähtökohtaisesti minimax-algoritmiin ja alfa–beeta-karsintaan. Näitä pidetäänkin yleisesti kahden henkilön pelien algoritmien ytiminä [18]. Shakissa nämä eivät kuitenkaan riitä, sillä shakin pelipuu on valtava, ja ohjelman täytyy päättää pelattava siirto järkevässä ajassa. Käytännössä haku pitää siis lopettaa jossain pisteessä ja luottaa siihen, että tässä pisteessä evaluaatiofunktio antaa hyvän arvion asemasta, joka voidaan välittää puun juureen kyseisen haaran minimax-arvona. Hyvä ratkaisu tähän ongelmaan on käydä läpi hakupuuta lähtökohtaisesti tiettyyn syvyyteen asti, mutta lopettaa haku tähän syvyyteen vain, jos kyseessä on hiljainen asema. Mikäli näin ei ole, hakua jatketaan pidemmälle, kunnes saavutetaan hiljainen asema. [15]

Toinen yleisesti käytössä oleva karsintatekniikka shakin pelipuussa on **tyhjän siirron karsinta** (engl. *null-move pruning*). Tämä karsinta perustuu siirtovuoron vaihtoon tekemättä mitään siirtoa. Vaikka tällainen teko ei ole shakin sääntöjen puitteissa sallittu, se voi antaa hyödyllistä tietoa asemasta ennen tätä. Jos esimerkiksi valkea pelaaja tekee siirron ja musta vastaa tähän tyhjällä siirrolla, voidaan suorittaa hyvin pienen syvyyden alfa–beeta-haku saatavalle asemalle. Mikäli valkea ei saavuta huomattavaa etua asemaan, vaikka sai siirtää kaksi kertaa peräkkäin,

siitä voi päätellä valkean ensimmäisen siirron olleen heikko. Jos taas valkea toisen peräkkäisen siirron jälkeen saavuttaa voittavan aseman, kannattaa tämän ensimmäistä siirtoa tutkia tarkemmin ja laskea mustan pelaajan todellisia vastauksia siihen. [15] Käytännössä siis karsintamenetelmä perustuu olettamukseen, että siirron väliin jättäminen ei ole paras siirto vaan aina tätä huonompi vaihtoehto [19]. Tämä karsintamenetelmä ei kuitenkaan ole ongelmaton, sillä shakissa tulee joskus vastaan asemia, joissa siirtopakko (saks. *Zugzwang*) aiheuttaa häviävän aseman. Esimerkki tällaisesta tilanteesta on esitetty kuvassa 3.1.



Kuva 3.1: *Zugzwang.* Siirtovuorossa oleva pelaaja joutuu tekemään huonon siirron.

Kuvassa siirtovuorossa oleva pelaaja, valkea tai musta, on siirtopakossa. Pelaaja joutuu siirtämään kuninkaansa pois suojaamasta sotilasta, minkä jälkeen vastustaja voi lyödä tämän sotilaan kuninkaallaan. Pelaaja ei voi estää vastustajaa korottamasta sotilastaan ja häviää pelin. Mikäli pelaaja voisi jättää siirron välistä antaen siirtovuoron vastustajalle, pelaaja voittaisi pelin.

Shakin pelipuussa on tyypillistä, että samaan asemaan voidaan päätyä usean eri siirtojärjestyksen kautta. Samaa asemaa ei kuitenkaan haluta joutua analysoimaan useasti, joten tämän välttäminen on tärkeää ohjelman tehokkuuden kannalta.

Tähän ratkaisuksi on kehitetty **transponointitaulukot** (engl. *transposition table*). Kyseessä on karsintamenetelmä, jossa läpikäydyt asemat tallennetaan suureen hajautustauluun, josta näiden arvio voidaan hakea hyvin nopeasti. Käytännössä tämä tarkoittaa sitä, että mikäli tietty asema on analysoitu jo puun aiemmassa haarassa, sitä ei tarvitse analysoida uudelleen, jos siihen päädytään jonkin toisen haaran kautta. [18]

Monte Carlo -puuhaku

Toinen erityisesti moderneissa shakkiohjelmissa käytetty puuhakualgoritmi on Monte Carlo -puuhaku. Toisin kuin alfa–beeta-haussa, Monte Carlo -puuhaussa jokaisessa asemassa ei ajeta evaluaatiofunktiota antamaan tästä asemasta arviota. Monte Carlo -haku perustuu satunnaisten simulaatioiden ajamiseen tietystä asemasta. Yksinkertaisimmillaan Monte Carlo -haussa tietyssä asemassa ajetaan tuhansia simulaatioita, joissa siirrot valitaan satunnaisesti mahdollisten siirtojen joukosta. Kun yksi simulaatio päättyy, tämän lopputulos välitetään alkuasemaan. Kun simulaatioiden määrä on suuri, voidaan odottaa alkuasemassa parhaan siirron johtavan suurimpaan voittoprosenttiin satunnaisten siirtojen simulaatioista. [15]

Shakissa Monte Carlo -puuhaun simulaatioita ei usein voida kuitenkaan suorittaa loppuun asti, vaan simulaatio pitää lopettaa jossain vaiheessa. Tämän takia evaluaatiofunktion käytöltä ei voida kokonaan välttyä, vaan loppuasema pitää analysoida tämän avulla. Evaluaatiofunktion kutsumäärä kuitenkin laskee murto-osaan alfa–beeta-hakuun verrattuna, sillä sen arvoa ei tarvitse laskea jokaiselle puun asemalle, vaan pelkästään loppuasemalle. Tämä mahdollistaa hitaammin laskettavien funktioiden käytön. [15]

3.2 Koneoppimista käyttävät ohjelmat

Vaikka perinteistä tekoälyä käyttämällä päästiin erittäin pitkälle ja saavutettiin huomattavasti yli-inhimillisellä tasolla pelaavia ohjelmia, viime vuosina nämä ovat jääneet pitkälti uudempien, neuroverkkoihin perustuvien ohjelmien jalkoihin. Näistä ensimmäinen huipputason ohjelma, joka aloitti uuden ajanjakson shakkitietokoneiden osalta, oli Googlen DeepMindin kehittämä *AlphaZero*. Se voitti vuonna 2018 tietokoneshakin maailmanmestaruuskilpailun (engl. *Top Chess Engine Championship*, TCEC) voittajan *Stockfishin* sadan pelin ottelussa 28 voitolla ja 72 tasapelillä, häviämättä kertaakaan [19]. Tätä ennen kaikki huipputason shakkiohjelmat olivat perustuneet HCE:hen ja pelipuun läpikäymiseen alfa–beeta-haun avulla.

AlphaZero ei kuitenkaan ollut ensimmäinen koneoppimiseen perustuva shakkiohjelma. Todellisuudessa jo vuonna 1994 julkaistu *NeuroChess* oli ensimmäinen shakkiohjelma, jonka evaluaatiofunktion muodostamiseen käytettiin neuroverkkoja. Evaluaatiofunktiota ei kuitenkaan muodostettu alusta alkaen neuroverkoilla, vaan evaluaatioverkolle annettiin lista piirteistä, joille verkko asetti painoarvot. Tämä evaluaatiofunktio yhdistettiin toisesta shakkiohjelmasta, *GNU-Chessistä*, lainattuun puuhakumekanismiin. *NeuroChess* voitti kokeissa noin 13 % kahden siirron kiinteällä hakusyvyydellä pelatuista peleistä *GNU-Chessiä* vastaan[2]. [20]

Toinen mainitsemisen arvoinen *AlphaZeroa* edeltänyt koneoppimisohjelma oli *DeepChess*, joka julkaistiin vuonna 2016. Tämä ohjelma oli ensimmäinen suurmestaritason[3] ohjelma, jonka evaluaatiofunktio muodostettiin alusta asti neuroverkoilla. Kuten kaikki muutkin shakkiohjelmat tätä ennen, myös *DeepChess* kävi läpi pelipuuta alfa–beeta-haulla [21]. Vaikka koneellisessa *Go*-pelissä saavutettiin samoihin

[2]Eri lopputulosten tarkka määrä ole tiedossa. Tasapelien määrä heikkojen ohjelmien välillä lienee kuitenkin pieni, minkä takia tätä ei koettu tarpeelliseksi julkaista.

[3]Suurmestari (engl. *grandmaster*) on korkein virallinen titteli, jonka ihmispelaaja voi saada. Suurmestaritittelin saavuttamiseen tarvitaan Elo-luku 2500 mutta parhaat ihmispelaajat ovat saavuttaneet kuitenkin lähes 2900 Elo-luvun. Parhaiden tietokoneohjelmien Elo-luvun on arvioitu olevan noin 3600 [1]. Suurmestaritason ohjelmalla tarkoitetaan siis ohjelmaa, joka vastaa pelitasoltaan parhaita ihmisiä, mutta ei ole välttämättä kilpailukykyinen parhaiden tietokoneohjelmien kanssa.

aikoihin suurta edistystä Monte Carlo -puuhakualgoritmia käyttäen [22], tätä algoritmia pidettiin sopimattomana shakkiin pelin terävyyden takia. Terävyydellä tarkoitetaan sitä, että usein vain yksi siirto vastustajalta voi kumota hyväksi kuvitellun siirron, vaikka mikä tahansa muu vastustajan siirto johtaisi hyvään asemaan. [21]

Mikä sitten teki *AlphaZerosta* niin paljon muita koneoppimisohjelmia paremman? Syynä oli erityisesti täysin erilainen puuhakumalli. Vaikka vain paria vuotta aikaisemmin Monte Carlo -puuhakualgoritmi oli todettu shakkiin sopimattomaksi [21], *AlphaZero* sai tämän hakumallin toimimaan [19]. Yleisenä ongelmana neuroverkkoevaluaatiofunktiossa oli niiden hitaus; alfa–beeta-haussa evaluaatiofunktiota pitää kutsua kerran jokaiselle pelipuun asemalle, joten kaikki evaluaatiofunktion arvon laskemiseen kuluva aika vähentää analysoitavien asemien määrää tietyn ajanjakson aikana [15]. Koska Monte Carlo -puuhaussa evaluaatiofunktiokutsujen määrä on vain murto-osa alfa–beeta-haussa tarvittavasta määrästä, hitaiden neuroverkkoevaluaatiofunktioiden käyttö oli mahdollista [19].

AlphaZeron menestyksen myötä myös muut shakkiohjelmat ovat pitkälti siirtyneet käyttämään neuroverkkoja. Jopa perinteisistä shakkiohjelmista pelitasoltaan vahvin, *Stockfish*, siirtyi vuonna 2020 käyttämään neuroverkkoihin perustuvaa evaluaatiofunktiota [23], ja vuonna 2024 se jätti HCE:n kokonaan pois ohjelmastaan [24]. *Stockfishin* käyttöön ottama konsepti on alun perin shogi-peliä pelaavista ohjelmista lähtöisin oleva tehokkaasti päivitettävien neuroverkkojen (engl. *Efficiently Updatable Neural Networks*, NNUE) teknologia. Tämä eroaa *AlphaZerosta* ratkaisemalla yleisen neuroverkkojen hitausongelman Monte Carlo -puuhaun sijaan evaluaatiofunktiotasolla. NNUE on yksinkertainen eteenpäin kytketty neuroverkko, jonka yksinkertaisuus nopeuttaa evaluaatioarvon laskemista huomattavasti. Tämä mahdollistaa sen parittamisen perinteisen alfa–beeta-haun kanssa [25].

NNUE-teknologian rinnalla myös Monte Carlo -puuhaku on edelleen laajalti käytössä shakkiohjelmissa. Alkuvuodesta 2018 julkaistiin *Leela Chess Zero* -niminen

projekti, jonka tarkoituksena oli tehdä Monte Carlo -puuhakuun ja syväoppiviin neuroverkkoihin perustuva avoimen lähdekoodin shakkiohjelma. Kun loppuvuodesta DeepMind julkaisi lisätietoja *AlphaZeron* koulutuksesta [19], nämä muutokset otettiin käyttöön myös *Leela Chess Zeroon*. Tällä hetkellä *Leela Chess Zero* on yksi maailman parhaista shakkiohjelmista, ja tuli TCEC:n kaudella 26 toiseksi häviten *Stockfishille* finaalissa [26].

4 Ihmismäisen pelityylin tavoittelu

Tähän mennessä tässä tutkielmassa esitellyt shakkiohjelmat ovat keskittyneet yksinomaan shakin pelaamiseen mahdollisimman hyvin. Toinen näkökulma shakkiohjelmiin on kuitenkin se, miten hyvin niiden pelaamat pelit vastaavat ihmisten pelaamia pelejä. Tässä luvussa käsitellään shakkiohjelmia erityisesti ihmisten näkökulmasta. Ensimmäisessä alaluvussa keskitytään tähän asti käsiteltyjen shakkiohjelmien ongelmiin ihmisten työkaluina. Toisessa alaluvussa esitellään potentiaalisia tekniikoita, joilla shakkiohjelmista saisi tehtyä ihmismäisempiä ja kolmannessa alaluvussa keskitytään yksittäisten pelaajien mallintamiseen ohjelmissa.

4.1 Shakkiohjelmien ongelmat ihmisten työkaluina

Shakkiohjelmien kehityksen myötä niitä on myös alettu käyttää ihmispelaajien apuvälineinä. Viime vuosituhannella ammattilaispelaajien harjoittelu oli vaikeaa: ei ollut olemassa mitään ohjelmaa tai muutakaan tahoa, joka voisi osoittaa pelaajien tekemät virheet hyvällä tarkkuudella. Nykyään asiat ovat toisin ja kuka tahansa, olipa kyseessä sitten aloittelija tai mestaritason pelaaja, voi hyvin nopeasti katsoa mistä tahansa peliasemasta numeerisen evaluaation, joka kertoo, kumpi pelaaja on voitolla. Voisi siis kuvitella, että shakkiohjelmat olisivat kaikentasoisille pelaajille mittaamattoman arvokas apuväline.

Nykyisissä shakkiohjelmissa on kuitenkin ihmisten näkökulmasta paljon ongelmia. Nykyajan tietokoneet pystyvät laskemaan asemia niin paljon nopeammin ja

syvemmälle kuin ihmiset, että tietokoneiden antamat arviot eivät välttämättä ole ihmispelaajille realistisia. Ohjelma voi esimerkiksi muutamassa minuutissa antaa jostain asemasta numeerisen evaluaation, joka viittaa aseman olevan pelaajalle voittava, mutta todellisuudessa voittoon johtaa vain yksi kymmeniä siirtoja syvälle menevä pelipuun haara, jonka löytäminen veisi parhailtakin ihmisiltä tuntikausia. Tällainen tilanne tapahtui esimerkiksi vuoden 2018 maailmanmestaruusottelun kuudennessa pelissä, jossa 68. siirrolla tietokoneohjelmat löysivät mustalle 58 siirtoa pitkän shakkimattiin johtavan siirtojen sarjan.[1] Mustilla nappuloilla pelannut Fabiano Caruana ei löytänyt tätä, ja peli päättyi lopulta tasan. [27]

Toinen mahdollinen keino käyttää shakkiohjelmia ihmisten apuvälineenä on näitä vastaan pelaaminen. Shakkiohjelmien taso riittää antamaan vähintäänkin riittävän vastuksen parhaillekin ihmispelaajille, eikä ohjelma väsy koskaan. Tässäkin on kuitenkin ongelmansa. Ihmisen on hyvin vaikea oppia mitään vastustajalta, jota vastaan hän voi pelata tuhat kertaa peräkkäin häviten joka ikisen pelin. Ihminen ei välttämättä edes ymmärrä, miksi hän hävisi ohjelmalle. Usein huippupelaajat kokevat pelikokemuksen tietokonetta vastaan epämieluisana ja suosivat ihmisiä vastaan harjoittelua [28]. Ohjelman pelitaso pitäisi siis jotenkin saada rajoitettua tietylle, pitkälti pelaajan omaa pelitasoa vastaavalle tasolle. Useimmat tunnetut shakkiohjelmat tarjoavatkin mahdollisuuden rajoittaa ohjelman pelitaso vastaamaan tiettyä Elo-lukua, mutta tämä jättää vielä paljon toivottavaa [28].

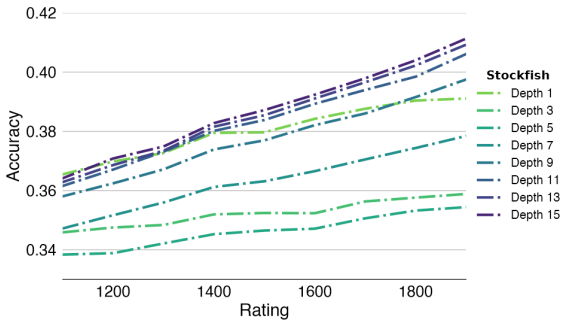
Pelitason rajoittamisen ongelmat

Suurin osa shakkiohjelmista on suunniteltu pelaamaan mahdollisimman hyvin. Tämä on johtanut siihen, että rajoittamattomien shakkiohjelmien pelissä tekemät päätökset ovat usein ihmisille hyvin epäintuitiivisia. [18] Ohjelmien pelitasoa voi kuiten-

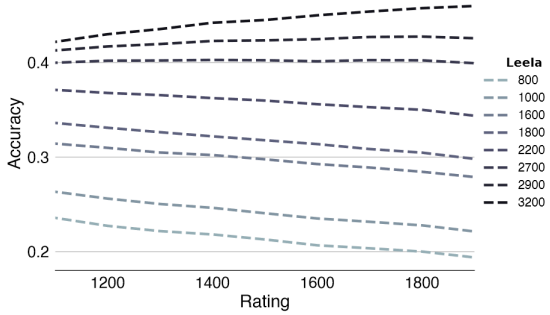
[1] Molemmat pelaajat tekevät 57 siirtoa, ja mustan 58. siirto tekee shakkimatin. Tämä linja olettaa molempien pelaajien pelaavan täydellisesti: musta yrittää voittaa pelin mahdollisimman nopeasti ja valkea yrittää pitkittää tätä mahdollisimman pitkään.

kin helposti rajoittaa, ja monet suositut shakkiohjelmat antavatkin käyttäjän valita ohjelman pelitason vapaasti [28]. Pelitason rajoittamiseen on käytetty monia tapoja, mutta lähes kaikissa ohjelmissa rajoitus perustuu ainakin osittain joko käytettävissä olevan laskenta-ajan tai puuhaun syvyyden rajoittamiseen. [3] Tämä mekaniikka on periaatteessa toimiva; kun halutulle tasolle rajoitettu ohjelma laitetaan pelaamaan muita ohjelmia tai ihmisiä vastaan, sen pitkän aikavälin tulos vastaa pitkälti halutun tason pelaajaa.

Kun rajoitusta tarkastelee tarkemmin, siitä löytää kuitenkin huomattavia heikkouksia. Yksi helppo tapa tarkastella rajoituksen mielekkyyttä on verrata ohjelman tekemiä siirtoja vastaavan tasoisten pelaajien tekemiin siirtoihin. Mikäli ohjelma rajoitetaan esimerkiksi vastaamaan Elo-lukua 1500, voisi olettaa, että ohjelman tekemät siirrot vastaavat suurimmalla todennäköisyydellä juuri tämän tasoisen ihmispelaajan tekemiä siirtoja ja päin vastoin; kaikista ohjelman rajoitustasoista tämä taso vastaa kaikkein parhaiten halutun pelaajan siirtoja [3]. Näin ei kuitenkaan ole. Verrattaessa eritasoisten ihmispelaajien tekemiä siirtoja *Stockfishin* tai *Leela Chess Zeron* tekemiin siirtoihin huomataan, että pelaajan tasosta riippumatta suurimmalla todennäköisyydellä samoja siirtoja tekevä versio on ohjelman rajoittamaton versio [3]. Siirtojen vastaavuus on esitelty kuvissa 4.1 ja 4.2. On siis selvää, että virheet, joilla rajoitetut ohjelmat saavuttavat halutun pelitason, eivät ole samoja virheitä, joita tämän tasoiset ihmispelaajat tekevät.



Kuva 4.1: *Stockfishin* eri rajoitustasojen tekemien siirtojen vastaavuus eri tason ihmispelaajien siirtoihin. [3]



Kuva 4.2: *Leela Chess Zeron* eri rajoitustasojen tekemien siirtojen vastaavuus eri tason ihmispelaajien siirtoihin. [3]

Virheiden ennustaminen

Yksi suurimmista ongelmista, joita ihmismäisen shakkiohjelman pitää ratkaista, on inhimillisten virheiden tekeminen. Toisin kuin useimmat huippuohjelmien väliset pelit, suurin osa erityisesti heikkojen ihmispelaajien välisistä peleistä ei pääty virheisiin, jotka realisoituvat vasta huomattavasti syvemmällä hakupuussa. Sen sijaan suuri osa näistä päättyy yksinkertaisiin virheisiin, jotka johtuvat pelaajan tekemästä inhimillisestä virheestä: pelaaja ei huomaa jotain vastustajan mahdollista siirtoa, joka johtaa tälle voittavaan asemaan, usein hyvin nopeasti, jopa heti seuraavalla vuorolla. [15] Tällaisen virheen ennustaminen on shakkiohjelmalle hyvin vaikeaa,

sillä jopa hyvin pienisyvyyksinen puuhaku huomaisi uhan heti eikä tekisi tätä uhkaa sallivaa siirtoa [3].

4.2 Ihmismäisen pelityylin eri lähestymistavat

Ihmismäistä pelityyliä tavoittelevien shakkiohjelmien kehitys on alkanut vasta lähivuosina, mutta jo tähän mennessä on yritetty monia eri menetelmiä tavoitteen saavuttamiseksi. Tässä alaluvussa esitellään eri metodeja ja tarkastellaan niiden suorituskykyä.

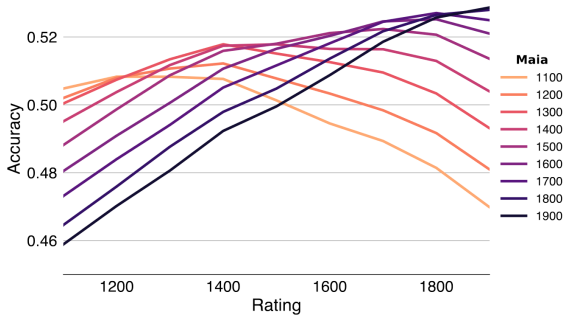
4.2.1 Muunnettu Monte Carlo -puuhaku

Ensimmäinen varteenotettava yritys tehdä ihmismäinen shakkiohjelma julkaistiin vuonna 2019, kun tutkijat Bar Ilanin yliopistosta kehittivät tähän tarkoitukseen ohjelman, *säädettävän shakkiohjelman* (engl. *Adjustable Chess Engine*, *ACE*). *ACE* perustui *AlphaZeron* kaltaiseen Monte Carlo -puuhakuun, jota muokattiin säätämällä eri hyperparametrejä ihmismäisyyden saavuttamiseksi. Monissa muissa peleissä vastaavaan tarkoitukseen oli käytetty imitaatio-oppimista (engl. *imitation learning*). Imitaatio-oppiminen on ohjatun oppimisen muoto, jossa ohjelma koulutetaan matkimaan ihmisten pelejä käyttämällä olemassa olevia ihmisten välisiä pelejä oppimiseen. *ACE:n* kehittäjät totesivat tämän kuitenkin olevan epäsopiva shakin pelaamiseen. Ohjelmaa testattiin eräänlaisella Turingin testillä, jossa vahvalle ihmispelaajalle annettiin huippuohjelman, ihmisten ja *ACE:n* pelaamista peleistä koostuva aineisto, ja ihmisen tehtävänä oli päätellä, mitkä peleistä olivat koneellisia ja mitkä ihmisten pelaamia. Kaikki huippuohjelman peleistä tuomittiin koneellisiksi, mutta ihmisten ja *ACE:n* pelit tuomittiin pitkälti samalla tavalla: puolet peleistä luokiteltiin ihmisten pelaamiksi ja puolet koneellisiksi. [28] Ohjelma siis onnistui tavoitteessaan. On kuitenkin huomattava, että lopputulokset ovat tulkinnanvaraisia. Turingin testi ei

välttämättä ole erityisen hyvä mittapuu ihmismäisyydelle, sillä myös ihmisten pelaamista peleistä puolet luokiteltiin koneellisiksi. Turingin testi on siis subjektiivinen testi, jolloin lopputuloksiin vaikuttaa ihmisen ennakko-oletus pelien koneellisuudesta.

4.2.2 Neuroverkkojen kouluttaminen eri aineistoilla

Yksi viime vuosina tutkimuksen kohteena olleista keinoista saavuttaa ihmismäisemmin pelaavia shakkiohjelmia perustuu neuroverkkojen kouluttamistavan muuttamiseen. Kuten luvussa 3.1 todettiin, nykyisten neuroverkkoihin perustuvien huipputason shakkiohjelmien neuroverkot koulutetaan vahvistusoppimisella, eli antamalla ohjelman pelata itseään vastaan lukuisia kertoja. Yksi tämän hetken lupaavimmista ihmismäisistä shakkiohjelmista, vuonna 2020 julkaistu *Maia*, perustuu neuroverkon kouluttamiseen ihmisten peleillä [3], vaikka vain vuotta aiemmin imitaatiooppiminen oli todettu sopimattomaksi shakkiin. Ohjelman kehittäjät ovat ottaneet koulutusaineistoksi eritasoisten ihmisten `Lichess.org`-verkkosivulla pelaamat pelit ja niiden pohjalta kouluttaneet yhdeksän eri tasoista ohjelmaa, joiden tavoitteena on ennustaa tietyn tason ihmisten tekemiä siirtoja tietyssä asemassa [3]. Suurin ero *Maian* ja muiden neuroverkko-ohjelmien välillä on siis se, että *Maia* ei tavoittele asemassa parhaan siirron tekemistä, vaan sen tavoite on pelata sama siirto, jonka halutun tason ihminen tekisi välittämättä siitä, onko tämä siirto hyvä vai huono. Koska eri tason ihmispelaajat tekisivät usein samassa asemassa eri siirtoja, yksi *Maian* instanssi pystyy siis ennustamaan vain tietyn tason ihmispelaajien siirtoja, ja *Maian* siirtojen vastaavuus ihmispelaajien siirtoihin laskee, kun ihmispelaajien tasoero ohjelman koulutusaineistona toimineisiin ihmisiin kasvaa [3]. Tässä *Maia* onnistuu varsin hyvin, mikä on ilmeistä verrattaessa kuvaa 4.3 aiemmin esiteltyihin vastaaviin kuviin 4.1 ja 4.2 *Stockfishin* ja *Leela Chess Zeron* rajoitettujen versioiden käyttäytymisestä.



Kuva 4.3: *Maian* eri versioiden tekemien siirtojen vastaavuus eri tason ihmispelaajien siirtoihin. Kunkin tason ihmispelaajien siirtoja vastaa parhaiten *Maian* saman tason instanssi, ja jokainen *Maian* instanssi vastaa lähes aina parhaiten juuri tämän tason ihmispelaajia: siirtojen vastaavuus laskee molempiin suuntiin mentäessä. [3]

Suurin toiminnallinen ero *Maian* ja tavallisten shakkiohjelmien välillä on se, että *Maia* ei käytä apunaan ollenkaan puuhakufunktiota. Kun ohjelman toimintaa hienosäädettiin, huomattiin, että puuhaku laskee siirtojen vastaavuutta ihmispelaajien kanssa inhimillisten virheiden ongelman vuoksi. Vaikka puuhaku nostaisi ohjelman pelitasoa huomattavasti, tason nousu johtuisi ihmismäisten virheiden välttämisestä ja näin ollen laskisi ihmismäisyyden mittapuuna käytettyä siirtojen vastaavuutta ihmisiin. Huipputason ohjelmiin verrattuna *Maia* onnistuu erityisen hyvin juuri ihmismäisten virheiden ennustamisessa. [3] Vaikka *Maian* lähestymistapa ongelman ratkaisuun on hyvä ja saavutti huomattavia parannuksia huipputason ohjelmiin verrattuna, ei sekään kuitenkaan ole ongelmaton. Jopa *Maian* vahvimman version pelitaso on melko heikko, eikä se kykene mallintamaan mestaritason pelaajien pelejä [2]. Puuhaun puuttumisesta johtuen *Maia* on erityisen huono myös shakkitehtävien[2] ratkaisemisessa, jossa se säännöllisesti epäonnistuu yksinkertaisissakin tehtävissä, jotka suurin osa vastaavan tason ihmisistä pystyisi ratkaisemaan [2]. Näiden ongelmien vuoksi puuhaun käyttöä on alettu tutkia uudelleen.

[2]Tietty – usein käsin laadittu – asema, jossa siirtovuorossa olevalla pelaajalla on vain yksi hyvä siirto tai siirtojen sarja, joka on huomattavasti parempi kuin muut. Tämä asema näytetään pelaajalle, jonka täytyy keksiä tämä oikea ratkaisu.

Vuonna 2024 julkaistussa tutkimuksessa Barrish ja muut [2] ovat tutkineet *Maiaa* tarkemmin ja selvittäneet mahdollisuuksia sen ongelmien ratkaisuun. Toisin kuin McIlroy-Young ja muut [3], [29], he totesivat puuhaun sopivan myös *Maiaan*, kun se rajoitetaan oikein. Tutkimuksessaan [2] Barrish ja muut kouluttivat *Maian* avoimen lähdekoodin pohjalta kaksi uutta mallia käyttäen huipputason pelaajien välillä, verkossa ja laudan ääressä, pelattuja pelejä. Pienemmästä koulutusaineistosta huolimatta nämä mallit saavuttivat yhtä hyviä tuloksia kuin alkuperäinen *Maia*. Tutkimuksen perusteella siirtojen vastaavuutta saa korotettua entisestään käyttämällä alkuperäisen ohjelman rinnalla kiinteän syvyyden *Monte Carlo* -puuhakua [2]. Testatut versiot on selitetty taulukossa 4.1 ja syvyydet, joilla eri versioille saadaan parhaat tulokset taulukossa 4.2.

Taulukko 4.1: Käytetyt *Maian* versiot ja niiden koulutusaineistot [2]

Version nimi	Pelien tyyppi	Vahvuuslukualue
Maia1900	*Lichess*-verkkopalvelin	1901-2000
Lichess2400	*Lichess*-verkkopalvelin	2400+
OTB2400	Pöydän ääressä pelatut pelit	2400+

Taulukko 4.2: MCTS-haun parhaat syvyydet [2]

Koulutettu versio	Parhaan vastaavuuden syvyys
Maia1900	4
Lichess2400	6
OTB2400	7

Erityisesti mestaritason pelaajia mallinnettaessa toinen huomattava parannus *Maiaan* saatiin muuttamalla tehtävän siirron valinta-algoritmia. *Maia*-mallien heikkous tehtävien ratkaisussa on ratkaistu antamalla ohjelmalle parametrinä huipputason shakkiohjelman, *Stockfishin*, evaluaatio kyseisestä asemasta. Tämän evaluaatio-

arvon perusteella ohjelma suodattaa pois kaikki siirrot, jotka eroavat tästä arviosta liikaa. Tämä evaluaatioikkunamalli parantaa siirtovastaavuutta tavalliseen *Maiaan* verrattuna erityisesti huipputason ihmispelaajia mallinnettaessa, sillä he tekevät suhteessa paljon vähemmän vakavia virheitä kuin heikommat pelaajat. Muissa tapauksissa se kuitenkin jää siirtovastaavuudessa hieman parhaan syvyyden puuhakumallia huonommaksi. [2]

Tutkimuksen perusteella on kuitenkin selvää, että ihmismäisyyden tavoittelussa on olemassa eräänlainen kompromissi ohjelman vahvuuden ja pelityylin välillä. Ohjelman vahvuutta – ja samalla vahvoja pelaajia mallinnettaessa siirtojen vastaavuutta – saa helposti parannettua antamalla ohjelman käyttää puuhakua, mutta tämän tekeminen heikentää ohjelman kykyä imitoida pelaajan tyyliä ja saa ohjelman pelaamaan tavalla, joka on lähempänä tavallisia huipputason ohjelmia. [2]

4.3 Yksittäisten ihmispelaajien mallintaminen

Vaikka yleisellä tasolla ihmismäisesti pelaava ohjelma onkin jo suuri harppaus ihmismäisyyden tavoittelussa, se ei vastaa kuitenkaan täysin ihmistä. Todellisuudessa ihmiset pelaavat hyvinkin eri tavoilla, ja eri ihmispelaajat tekevät samoissa tilanteissa eri päätöksiä. Ihmismäisesti pelaavien shakkiohjelmien ns. ultimaattinen tavoite olisikin pystyä viemään mallinnus niin pitkälle, että ne voisivat imitoida jopa *yksittäisiä* ihmispelaajia. On kuitenkin hyvin vaikeaa tehdä yleistä ohjelmaa, jonka voisi suhteellisen helposti saada koulutettua tekemään samoja päätöksiä kuin eri ihmiset, sillä eri ihmiset voivat lähestyä shakkiasemaa monella eri tavalla ja jokaisella heistä on oma tapansa tehdä päätös pelattavasta siirrosta. Tämä päätös perustuu ainakin osittain myös tiedostamattomiin tekijöihin, joten ihmisen on mahdotonta selittää päätöksentekoprosessia niin tarkasti, että sen perusteella voitaisiin tehdä prosessia imitoiva tietokoneohjelma. [29]

Vuonna 2022 *Maian* kehittäjät alkoivat kuitenkin tutkia myös yksittäisten ihmispelaajien mallinnusta. Koska ihmispelaajien mallintaminen algoritmitasolla on käytännössä mahdotonta, potentiaalisimmaksi vaihtoehdoksi ilmenivät tässäkin tapauksessa neuroverkot. Ilmeinen neuroverkkojen käyttöön liittyvä ongelma oli koulutusaineiston määrä: jotta yksittäistä pelaajaa voidaan mallintaa, tarvitaan suuri määrä juuri hänen pelaamiaan pelejä. *Lichess*-verkkopalvelimen peleistä löytyy kuitenkin paljon pelaajia, jotka ovat pelanneet riittävän määrän pelejä neuroverkon koulutusta varten. Käyttämällä näitä pelejä koulutusaineistona kehittäjät kouluttivat siirto-oppimista käyttäen olemassa olevan *Maian* version mallintamaan yksittäisten pelaajien pelejä. Alkuperäisenä ohjelmana käytetyn *Maian* versiolla ei huomattu olevan huomattavaa merkitystä tuloksiin, joten kehittäjät käyttivät *Maia1900*-versiota kaikkien pelaajien kohdalla. [29].

Siirto-Maia (engl. *transfer-Maia*) onnistui mallintamaan yksittäisiä pelaajia varsin hyvin. Parhaaseen yleisen *Maian* tasoon verrattuna *Siirto-Maia* paransi siirtojen vastaavuutta halutun pelaajan tekemiin siirtoihin noin neljällä prosenttiyksiköllä siirron hyvyydestä riippumatta, ja usein ero oli paljon suurempikin. Erityisen hyvin *siirto-Maia* onnistui mallintamaan pelin alussa tehtyjä siirtoja, joissa se saavutti jopa 40 prosenttiyksikön eron parhaaseen yleisen *Maian* tasoon. [29] Tämä oli odotettua, sillä pelin alussa tehtävät siirrot ovat usein ns. teoriaa, eli ulkoa opeteltuja siirtoja. Vaikka myös pelin alussa on lähes kaikissa tilanteissa monia eri siirtoja, jotka johtavat hyvin erilaisiin peleihin, monet pelaajat pelaavat samassa asemassa lähes aina saman siirron saavuttaakseen saman tyyppisen pelin kuin mihin he ovat tottuneet. Ero *siirto-Maian* ja yleisen *Maian* välillä pysyi huomattavana kuitenkin myös pelin loppuvaiheilla, jolloin suurin osa saavutettavista asemista on täysin uusia eivätkä ne löydy koulutusaineistosta [29]. Tämä viittaa siihen, että *siirto-Maia* oppi myös tiettyjä piirteitä pelaajan päätöksentekoprosessista, eikä vain oppinut tekemään tunnetuissa asemissa samoja siirtoja [29].

Stylometria

Yksittäisten pelaajien mallintamisen myötä on tutkittu myös mallien käytettävyyttä stylometriassa (engl. *stylometry*). Shakin tapauksessa stylometrialla tarkoitetaan ihmispelaajien pelityylin tutkimusta. Shakissa stylometriaa voidaan käyttää esimerkiksi tunnistamaan, kuka on pelannut tietyn pelin tai klusteroimaan eri pelaajia pelityyliensä perusteella. Shakissa stylometriaa on tutkittu käyttämällä muiden tieteenalojen, kuten puheentunnistuksen ja käsialantunnistuksen, stylometriassa käytettyjä tekniikoita. [30] Kun McIlroy ja muut tutkivat yksittäisten pelaajien mallinnusta *siirto-Maiaa* käyttäen, he testasivat sen toimivuutta myös stylometriseen analyysiin pelaajien peleistä. Ohjelman huomattiin toimivan erittäin hyvin myös stylometriaan, vaikka sitä ei ollut kehitetty tätä varten. Kun testiaineisto oli riittävän suuri, noin sata saman pelaajan pelaamaa peliä, malli onnistui tunnistamaan oikean pelaajan neljänsadan pelaajan joukosta 98 %:ssa tapauksista. Malli onnistui tunnistamaan pelaajan myös pelkästään tämän tekemien virheiden perusteella. [29] Tämä tekee ohjelmasta lupaavan opetuskäytössä, sillä jos yksittäisen pelaajan tekemissä virheissä on jotain kaavamaista, mihin tämä tulos viittaa, pelaajan voi olla helpompi kehittyä välttämään näiden virheiden tekoa jatkossa.

5 Yhteenveto

Shakkiohjelmat ovat kehittyneet valtavasti kolmen viime vuosikymmenen aikana. Shakkiohjelmat voittivat hallitsevan maailmanmestarin ensimmäisen kerran vasta vuosituhannen vaihteessa, mutta nyt ohjelmat ovat jo huomattavasti ihmisiä vahvempia ja kiinnostus on alkanut keskittyä entistä enemmän shakkiohjelmien pelityylin muuttamiseen ihmismäisempään suuntaan.

5.1 Vastaukset tutkimuskysymyksiin

Tässä tutkielmassa pyrittiin vastaamaan kolmeen tutkimuskysymykseen:

TK1: Mitä shakkiohjelmia on olemassa?

Koneellisen shakin kehityksen kannalta erityisen merkityksellisiä shakkiohjelmia ovat olleet:

Turochamp, joka oli ensimmäinen yritys pelata shakkia algoritmillisesti,

Deep Blue, joka oli ensimmäinen hallitsevan maailmanmestarin voittanut ohjelma,

Stockfish, joka on maailman vahvin shakkiohjelma ja oli pitkään myös esimerkki huipputason perinteiseen tekoälyyn perustuvasta shakkiohjelmasta,

AlphaZero, joka aloitti ns. uuden aikakauden shakkiohjelmissa voittamalla *Stockfishin* ensimmäisenä koneoppimiseen perustuvana ohjelmana ja

Maia, joka on tämän hetken lupaavin kehitteillä oleva ihmismäisesti pelaava shakkiohjelma.

TK2: Mitä tekoälyn menetelmiä shakkiohjelmat käyttävät?
Shakkiohjelmat jakautuvat kahteen luokkaan, perinteistä tekoälyä käyttäviin ohjelmiin ja koneoppimiseen perustuviin ohjelmiin. Perinteistä tekoälyä käyttävät ohjelmat koostuvat käsintehdystä evaluaatiofunktiosta, joka laskee asemalle numeerisen arvon moniin eri piirteisiin perustuen ja minimax-hausta, joka käy läpi pelipuuta niin syvälle kuin ajankäytöllisesti mahdollista. Minimax-hakua karsitaan aggressiivisesti monilla eri karsinta-algoritmeilla kuten alfa–beeta-karsinnalla. Koneoppimista käyttävät ohjelmat perustuvat vahvistusoppimisella koulutetuilla neuroverkoilla muodostettuun evaluaatiofunktioon, joka paritetaan sopivan puuhakualgoritmin kanssa. Monet neuroverkko-ohjelmat käyttävät puuhakualgoritmina Monte Carlo -puuhakua välttääkseen evaluaatiofunktion ajamisen jokaisessa asemassa, mutta myös alfa–beeta-hakua käytetään.

TK3: Miten shakkiohjelmien pelityylistä saisi tehtyä ihmismäisemmän?
Ihmismäisen pelityylin tavoittelua on lähestytty monin eri tavoin. Ensimmäinen varteenotettava tutkimus keskittyi tavallisen neuroverkko-ohjelman parametrien muuttamiseen siten, että ohjelman pelitapa olisi lähempänä ihmisiä. Tällä hetkellä lupaavin ihmisen kaltainen ohjelma on kuitenkin ottanut eri lähestymistavan, jossa sen neuroverkko koulutetaan vahvistusoppimisen sijaan ihmisten peleillä. Ohjelman pelitasoa ja -tyyliä voi säätää kouluttamalla verkko eri aineistoilla, ja sen voi kouluttaa jopa matkimaan yksittäisten ihmisten pelityyliä ottamalla koulutusaineistoon vain kyseisen ihmisen pelejä. Ohjelmassa on silti heikkouksia, mutta monet tutkijat ovat pyrkineet ratkomaan näitä tekemällä ohjelmaan muutoksia.

5.2 Tulosten arviointi ja pohdinta

Shakki on jo pitkään ollut yksi kiinnostavimmista tekoälyn tutkimuskohteista. Shakkia pelaavia ohjelmia on kehitetty käyttämällä monia eri tekoälyn menetelmiä, eikä

yksikään niistä ole paljastunut huomattavasti muita paremmaksi. Shakki on helppo ala tekoälyn soveltamiselle, sillä shakkia pelataan diskreetissä ympäristössä, jossa mahdollisia siirtoja on suuri mutta rajattu määrä. Shakissa, ja peleissä yleisesti, myös eettiset haasteet ovat varsin vähäisiä, mikä tekee siitä turvallisen kohteen tekoälytutkimukselle ja sen soveltamiselle.

5.2.1 Ihmismäisten shakkiohjelmien sovelluskohteet

Viime vuosina tutkimus erityisesti ihmismäisen pelityylin tavoittelusta shakkiohjelmissa on edennyt huomattavasti. Vain viisi vuotta sitten ihmismäisiä shakkiohjelmia ei vielä ollut olemassakaan, mutta nyt ohjelmat osaavat mallintaa jopa yksittäisiä pelaajia. Ihmismäisten shakkiohjelmien kehityksessä on paljon potentiaalia, ja niistä voi olla hyötyä erityisesti opetuksessa. Ihmismäisesti pelaavia shakkiohjelmia voidaan käyttää antamaan pelaajille sopivan tasoinen vastus harjoittelua varten ilman, että ohjelmaa tarvitsee rajoittaa riittävän heikoksi.

Toinen mahdollinen hyötykäytön kohde ihmismäisille shakkiohjelmille on huijaamisen tunnistaminen. Shakkiohjelmien kehittyminen yli-inhimilliselle tasolle on myös mahdollistanut niiden käyttämisen huijaamiseen ihmisten välisissä peleissä. Kuten monissa muissakin peleissä, erityisesti videopeleissä, myös shakissa huijauksen tunnistaminen on avoin ongelma, jossa tunnistusmekanismit kilpailevat jatkuvasti huijauskeinojen kanssa. Paikan päällä pelatuissa huippupeleissä huijauksen vastaiset toimet koostuvat nykyään pääasiassa keinoista, jotka pyrkivät estämään huijaamisen esiintymisen. Tällaisia keinoja ovat esimerkiksi matkapuhelinkiellot pelialueella, metallinpaljastimien käyttö tunnistamaan mahdollisia piilotettuja elektronisia laitteita ja katsojien vuorovaikutuksen pelaajien kanssa estäminen pelien aikana. [31]

Ihmismäisten shakkiohjelmien avulla huijauksen vastaisiin mekanismeihin voitaisiin kuitenkin lisätä myös pelien analysoiminen niiden päätyttyä. Erityisesti stylometriaa käyttäen voitaisiin mahdollisesti tunnistaa pelaajien joukosta sellaiset pe-

laajat, jotka pelaavat itselleen epätyypillisellä tavalla tietokoneella huijaamisen johdosta. Tämä olisi erittäin hyödyllistä huijauksen estämiseksi, sillä toistaiseksi pelien analysoiminen vie aikaa, eikä jokaista peliä ehditä eikä jakseta analysoida syvällisesti vain huijauksenestomielessä. Mikäli stylometria paljastuu tehokkaaksi keinoksi tässä, voitaisiin helposti kehittää ohjelma, johon syötetään peli ja sen pelaaja, minkä jälkeen ohjelma vertaa peliä tyylillisesti muihin kyseisen pelaajan peleihin. Tähän mennessä tällaista järjestelmää ei ole vielä kehitetty, mutta tämä olisi hyvä potentiaalinen jatkotutkimuksen kohde.

5.2.2 Ihmismäisten shakkiohjelmien riskit

Vaikka shakki on tekoälyn tutkimuskohteena ollut aina varsin turvallinen, ihmismäisten shakkiohjelmien kehityksen myötä myös eettisiä ongelmia on pohdittava. Vaikka ohjelmat voisivat auttaa huijauksen estämisessä, niitä voi olla mahdollista myös hyödyntää huijauksessa. Erityisesti yksittäisiä ihmispelaajia mallintavissa ohjelmissa on olemassa riski, että mallin voisi kouluttaa pelaamaan tyylillisesti kuten kyseinen pelaaja, mutta välttämään suuret virheet. Tällainen ohjelma voisi olla immuuni stylometrisille huijauksenestojärjestelmille ja mahdollistaa huijaamisen kaiken tasoisissa peleissä ilman riskiä jäädä kiinni siirtoihin perustuvassa analyysissa. Tällä hetkellä riski on pieni stylometristen analyysityökalujen kehityksen ollessa vielä kesken, joten kysyntä ihmismäisille huijauskeinoille lienee vähäinen, mutta tämä riski voi kasvaa työkalujen kehittyessä.

Toinen tärkeä eettinen kysymys erityisesti stylometrian kohdalla on yksityisyydensuoja. Erityisesti eri verkkopalvelimilla on tällä hetkellä mahdollista pelata pelejä täysin anonyymisti ilman, että kukaan tietää pelaajan todellista henkilöllisyyttä. Tehokkaat stylometrian keinot voisivat kuitenkin rikkoa tämän yksityisyyden ja mahdollistaa pelaajan identiteetin selvittämisen vertaamalla tämän pelaamia pelejä tunnettujen pelaajien peleihin. Erityisesti huippupelaajille tämä voi olla suurikin

ongelma, sillä heidän pelinsä ovat julkisesti saatavilla kaikille, jolloin vertailu tunnettuihin esimerkkeihin on helppoa.

5.2.3 Ihmismäisten shakkiohjelmien tulevaisuus

Ihmismäisten shakkiohjelmien kehitys on vasta aluillaan ja se tulee jatkumaan vielä pitkään. Ihmismäisillä shakkiohjelmilla on potentiaalia mullistaa pelin opettelua ja pelaajien kehittymistä. Vielä ei kuitenkaan tiedetä, onko ihmismäisyydellä olemassa jokin raja, jota tietokoneet eivät voi ainakaan lähitulevaisuudessa ylittää. Ihmisten ja tietokoneiden tapa ratkaista ongelmia on niin erilainen, että kaikentasoisia pelaajia täydellisesti mallintavaa ihmismäistä ohjelmaa ei ehkä voida koskaan saavuttaa. Tällaista rajaa ei kuitenkaan vielä ole saavutettu, eikä kukaan tiedä, miten pitkälle kehitys voi jatkua.

Ihmismäisten shakkiohjelmien potentiaali on kuitenkin rajallinen. Ne voivat toimia merkittävänä apuna shakkimaailmassa, mutta niiden kehitystä voi olla vaikeaa hyödyntää muilla tieteenaloilla. Myös juuri ihmismäisiin ohjelmiin ja niiden sovelluksiin liittyvät eettiset haasteet voivat vaikeuttaa tekoälyn kehitystä nimenomaan shakin osalta.

Lähdeluettelo

[1] Computer Chess Rating Lists. ”CCRL 40/15”. (2024), url: http://computerchess.org.uk/ccrl/4040/ (viitattu 29. 10. 2024).

[2] D. Barrish, S. Kroon ja B. van der Merwe, ”Making Superhuman AI More Human in Chess”, teoksessa *Advances in Computer Games*, Springer Nature Switzerland, 2024, s. 3–14, ISBN: 9783031549687. DOI: 10.1007/978-3-031-54968-7_1.

[3] R. McIlroy-Young, S. Sen, J. Kleinberg ja A. Anderson, ”Aligning Superhuman AI with Human Behavior: Chess as a Model System”, teoksessa *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, sarja KDD ’20, ACM, elokuu 2020, s. 1677–1687. DOI: 10.1145/3394486.3403219.

[4] Kansainvälinen Shakkiliitto (FIDE). ”FIDE Handbook”. (2023), url: https://handbook.fide.com/chapter/E012023 (viitattu 29. 10. 2024).

[5] C. E. Shannon, ”XXII. Programming a computer for playing chess”, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, nro 314, s. 256–275, maaliskuu 1950, ISSN: 1941-5990. DOI: 10.1080/14786445008521796.

[6] J. Tromp. ”ChessPositionRanking”. (2021), url: https://github.com/tromp/ChessPositionRanking (viitattu 07. 10. 2024).

[7] A. A. Macdonell, "Art. XIII.—The Origin and Early History of Chess", *Journal of the Royal Asiatic Society of Great Britain & Ireland*, vol. 30, nro 1, s. 117–141, tammikuu 1898, ISSN: 2051-2066. DOI: 10.1017/s0035869x00146246.

[8] M. Mark, "The Beginnings of Chess", teoksessa *Ancient Board Games in Perspective*, I. Finkel, toim., British Museum Press, 8. elokuuta 2007, luku 18, s. 144–162, ISBN: 978-0714111537.

[9] R. Calvo. "Valencia Spain: the cradle of European chess". (1998), url: http://history.chess.free.fr/papers/Calvo%201998.pdf (viitattu 21. 10. 2024).

[10] G. Westerveld, *The Poem Scachs d'amor (1475). First Text of Modern Chess.* Lulu Press, Inc., 2015, ISBN: 9781326374914.

[11] T. Standage, *The Turk: The life and times of the famous eighteenth-century chess-playing machine*. New York, NY: Berkley Trade, 2003.

[12] G. Kasparov ja F. Friedel, "Reconstructing Turing's "paper machine"1", *ICGA Journal*, vol. 40, nro 2, s. 105–112, helmikuu 2019, ISSN: 1389-6911. DOI: 10.3233/icg-180044.

[13] Chessprogramming Wiki. "History". (2021), url: https://www.chessprogramming.org/History (viitattu 27. 11. 2024).

[14] M. Campbell, A. Hoane ja F.-h. Hsu, "Deep Blue", *Artificial Intelligence*, vol. 134, nro 1–2, s. 57–83, tammikuu 2002, ISSN: 0004-3702. DOI: 10.1016/s0004-3702(01)00129-1.

[15] D. Klein, *Neural Networks for Chess*, 2022. DOI: 10.48550/ARXIV.2209.01506. (viitattu 05. 10. 2024).

[16] Chessprogramming Wiki. "Kramnik versus Deep Fritz 2006". (2022), url: https://www.chessprogramming.org/Kramnik_versus_Deep_Fritz_2006 (viitattu 27. 11. 2024).

[17] O. David-Tabibi, H. J. van den Herik, M. Koppel ja N. S. Netanyahu, ”Simulating human grandmasters: evolution and coevolution of evaluation functions”, teoksessa *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, sarja GECCO09, ACM, heinäkuu 2009, s. 1483–1490. DOI: 10.1145/1569901.1570100.

[18] V. Chole ja V. Gadicha, ”Hybrid Optimization for Developing Human Like Chess Playing System”, teoksessa *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, IEEE, lokakuu 2022, s. 1–5. DOI: 10.1109/gcat55367.2022.9971825.

[19] D. Silver, T. Hubert, J. Schrittwieser et al., ”A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”, *Science*, vol. 362, nro 6419, s. 1140–1144, joulukuu 2018, ISSN: 1095-9203. DOI: 10.1126/science.aar6404.

[20] S. Thrun, ”Learning to Play the Game of Chess”, teoksessa *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky ja T. Leen, toim., vol. 7, MIT Press, 1994, s. 1069–1076. url: https://proceedings.neurips.cc/paper_files/paper/1994/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf.

[21] O. E. David, N. S. Netanyahu ja L. Wolf, ”DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess”, teoksessa *Artificial Neural Networks and Machine Learning – ICANN 2016*, Springer International Publishing, 2016, s. 88–96, ISBN: 9783319447810. DOI: 10.1007/978-3-319-44781-0_11.

[22] D. Silver, A. Huang, C. J. Maddison et al., ”Mastering the game of Go with deep neural networks and tree search”, *Nature*, vol. 529, nro 7587, s. 484–489, tammikuu 2016, ISSN: 1476-4687. DOI: 10.1038/nature16961.

[23] Stockfish. ”Introducing NNUE Evaluation”. (2020), url: https://stockfishchess.org/blog/2020/introducing-nnue-evaluation/ (viitattu 28. 10. 2024).

[24] Stockfish. ”Stockfish 16.1”. (2024), url: https://stockfishchess.org/blog/2024/stockfish-16-1/ (viitattu 29. 10. 2024).

[25] Q. A. Sadmine, A. Husna ja M. Müller, ”Stockfish or Leela Chess Zero? A Comparison Against Endgame Tablebases”, teoksessa *Advances in Computer Games*. Springer Nature Switzerland, 2024, s. 26–35, ISBN: 9783031549687. DOI: 10.1007/978-3-031-54968-7_3.

[26] Top Chess Engine Championship. ”Top Chess Engine Championship”. (2024), url: https://tcec-chess.com/#x=archive (viitattu 09. 12. 2024).

[27] P. Wong. ”Carlsen-Caruana WCC Game 6 – The actual forced-mate sequence that was missed”. (26. heinäkuuta 2022), url: https://www.chess.com/blog/Rocky64/carlsen-caruana-wcc-game-6-the-actual-forced-mate-sequence-that-was-missed (viitattu 06. 12. 2024).

[28] H. Rosemarin ja A. Rosenfeld, ”Playing Chess at a Human Desired Level and Style”, teoksessa *Proceedings of the 7th International Conference on Human-Agent Interaction*, sarja HAI ’19, ACM, syyskuu 2019, s. 76–80. DOI: 10.1145/3349537.3351904.

[29] R. McIlroy-Young, R. Wang, S. Sen, J. Kleinberg ja A. Anderson, ”Learning Models of Individual Behavior in Chess”, teoksessa *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, sarja KDD ’22, ACM, elokuu 2022, s. 1253–1263. DOI: 10.1145/3534678.3539367.

[30] R. McIlroy-Young, Y. Wang, S. Sen, J. Kleinberg ja A. Anderson, ”Detecting Individual Decision-Making Style: Exploring Behavioral Stylometry in Chess”,

teoksessa *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang ja J. W. Vaughan, toim., vol. 34, Curran Associates, Inc., 2021, s. 24 482–24 497. url: https://proceedings.neurips.cc/paper_files/paper/2021/file/ccf8111910291ba472b385e9c5f59099-Paper.pdf.

[31] S. Zaksaitė, "Anti-cheating protection measures in chess: current state of play", *Crime Prevention and Community Safety*, vol. 24, nro 3, s. 255–265, toukokuu 2022, ISSN: 1743-4629. DOI: 10.1057/s41300-022-00149-x.