Iikka Seilonen

Shakin evaluointi tietokoneella ja loppupelitietokannat

Tietotekniikan kandidaatintutkielma

25. toukokuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Iikka Seilonen

Yhteystiedot: iikka.t.seilonen@student.jyu.fi

Ohjaaja: Tuomo Rossi

Työn nimi: Shakin evaluointi tietokoneella ja loppupelitietokannat

Title in English: Computer evaluation of chess and endgame tablebases

Työ: Kandidaatintutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 19+0

Tiivistelmä: Tässä kandidaatintutkielmassa selvitetään miten shakin ratkaiseminen, ja loppupelitietokannat ovat kehittyneet nykytietokoneiden alkuajoista nykypäivään. Tutkielmassa selitetään taustoittavasti pelien ratkaisemisen teoriaa ja selvitetään mitä haasteita ja ratkaisuja suurien tietokantojen luomiseen liittyy.

Avainsanat: shakkitietokannat, loppupelitietokannat, shakin ratkaiseminen, evaluointi

Abstract: This bachelor's thesis researches how solving chess and endgame tablebases have developed from the inception of modern computers until modern day. The thesis explains the theory of solving games and looks into what the challenges and solutions are in creating large endgame tablebases.

Keywords: endgame tablebases, solving chess, evaluation

Kuviot

Sisällys

1 Johdanto

Pelien ratkaiseminen ja tekoälyjen kehittäminen perinteisten strategiapelien pelaamiseen on ollut tutkimuskohteena nykytietokoneiden alkuajoista alkaen. Näistä peleistä yksi keskeisimmistä on shakki. Tässä tutkielmassa selvitetään miten shakkipelin evaluointi ja ratkaiseminen ohjelmallisesti ovat kehittyneet.

Shakki on täydellisen informaation peli eli jokaisella pelaajalla on kaikki peliin liittyvä informaatio saatavilla pelin aikana, ja peliin ei liity satunnaisuutta. Peliteoria selittää, että tällaiset pelit on mahdollista teoriassa ratkaista, eli kehittää täydellinen pelistrategia. Käytännön haasteet tulevat kuitenkin eteen tarpeeksi monimutkaisissa peleissä, kuten shakissa, jonka ratkaiseminen ei ole näköpiirissä lähitulevaisuudessa sen laskennallisesta vaativuudesta johtuen. Tässä tutkielmassa kartoitetaan shakin ratkaisemiseen liittyvää kirjallisuutta, ja selvitetään tähän liittyviä teknisiä ongelmia, historiaa ja tulevaisuuden kehitystä.

Seuraavassa luvussa käsitellään ensin teoreettista taustaa shakin ja yleensäkin pelien ratkaisemisesta. Tämän jälkeen kerrotaan hieman shakkiohjelmien perusperiaatteista. Kolmannessa luvussa keskitytään tarkemmin shakkitietokantoihin, selvittäen näiden kehityksen historiaa, ongelmia ja keinoja ratkaista näitä ongelmia. Viimeisessä, neljännessä luvussa käydään lyhyesti läpi tutkielman pääkohdat ja tehdään näistä päätelmiä.

2 Shakin ratkaiseminen ja evaluointi

Perinteiset pelit, kuten lautapelit ja korttipelit voidaan voidaan jaotella kategorioihin perustuen siihen millaista tietoa ja resursseja pelissä on. Tämän perusteella pelit voidaan jakaa täydellisen - ja epätäydellisen informaation peleihin. Täydellisen informaation pelejä ovat esimerkiksi shakki ja neljän suora, kun taas epätäydellisen informaation pelejä ovat esimerkiksi monet korttipelit kuten pokeri.

2.1 Peliteoria ja pelien ratkaiseminen

Useimmat perinteiset pelit, joita tietoteknikot ovat pyrkineet ratkaisemaan ovat täydellisen informaation pelejä. Ross (2021) selittää tällaisten pelien olevan yksinkertaisimpia loogiselta rakenteeltaan ja näin suoraviivaisimpia ratkaista. Näistä esimerkkejä ovat neljän suora, tammi ja shakki, joista kahden ensimmäisen voidaan katsoa olevan ratkaistu (Schaeffer ym. 2007).

Täydellisen informaation peleissä kaikkien osapuolten siirrot ja resurrsit ovat tiedossa kaikille peliin osallistuville. Kun peli on lisäksi peräkkäinen, eli pelaajat tekevät siirtonsa vuorotellen, ja peli päättyy jossain vaiheessa, teoriassa voidaan selvittää kumpi pelaaja voittaa vai päättyykö peli tasan jatkamalla täydellisellä pelillä loppuun saakka. Tällaisia pelejä kuten shakki, tutkii kombinatorinen peliteoria. Demaine (2001) selittää kombinatoristen pelien olevan yleensä kahden pelaajan pelejä, joissa ei ilmene satunnaisuutta tai piilotettua tietoa.

Nämä pelit voidaan esittää pelipuuna, jossa juuri on pelin alkuasema, haarat ovat siirtojen valintoja ja lehdet ovat pelin lopputilanteita. Lehtien määrä myös määrittää näin pelin pelipuun koon eli erilaisten pelien lukumäärän. Tämä lukumäärä (engl. game-tree complexity) on Allis (1994) mukaan yksi tärkeä määritelmä pelin kompleksisuudelle, toisen ollessa erilaisten laillisten asemien määrä (engl. state-space complexity).

Pelien ratkaisemiseen on määristelty 3 eri tasoa. Allis (1994) määrittelee nämä tasot seuraavalla tavalla:

1. Ultraheikosti ratkaistulle pelille tiedetään pelin lopputulema täydellisellä pelillä, mutta

ei strategiaa jolla tähän päästään.

2. Heikosti ratkaistusta pelistä lopputulema ja täydellinen strategia tiedetään pelin alkuasemasta, mutta ei mistä tahansa asemasta. Schaeffer ym. (2007) saivat tammi-pelin ratkaistuksi tällä tasolla 18 vuoden laskennan jälkeen.
3. Vahvasti ratkaistulle pelille lopputulema tiedetään kaikista mahdollisista asemista.

Shakkia ei ole todettu olevan ratkaistu millään näistä tasoista tieteellisissä julkaisuissa, mutta yleisesti empiirisen todisteen perusteella täydellisen pelin katsotaan päättyvän tasapeliin; pelit parhaiden shakkiohjelmien välillä päättyvät tasan. Voitaisiin siis väittää shakin olevan “empiirisesti ultraheikosti ratkaistu“. Varsinaista todistusta lopputulemasta täydellisellä pelillä ei ole, joten shakki ei ole kuitenkaan virallisesti ratkaistu edes ultraheikosti.

2.2 Shakin ratkaiseminen

Shakin ratkaisemisen vaikeutta arvioi jo Shannon (1950), joka laski mahdollisten shakkipelien lukumäärän olevan 10^{120}. Shannonin mukaan tällöin ensimmäisen siirron laskemiseen tietokoneella, joka laskisi yhden varaation per mikrosekunti menisi 10^{90} vuotta. Shannon esitti myös mahdolliseksi ratkaisutavaksi "hakemiston"(engl. "dictionary") luomista joka sisältäisi kaikki mahdolliset shakkiasemat ja tähän oikean siirron. Tietokoneiden kehityksen myötä tietokoneita alettiin ohjelmoida myös pelaamaan shakkia. Ensimmäiset shakkiohjelmat olivat hyvin yksinkertaisia ja lyhytnäköisiä, ja aloittelijakin pystyi voittamaan tietokonetta vastaan. Shakkiohjelmat kehittyivät nopeasti, mutta vasta 1990-luvulla shakkiohjelma kykeni päihittämään aikansa parhaat ihmispelaajat. Mutta vaikka shakkiohjelmat päihittävät pelaajat nykypäivänä täysin selkeästi, ei tämä tarkoita että ohjelmat pelaisivat täydellistä shakkia tai että ne olisivat “ratkaisseet“ shakin.

2.3 Evaluointifunktiot

Shakkiohjelmat eivät perustukkaan vieläkään siihen että ohjelma laskisi pelin loppuun asti, ja osaisi aina pelata parhaan siirron. Laskettavien variaatioiden määrä on niin valtava useimmissa pelitilanteissa että tämä on mahdotonta. Perinteiset shakkiohjelmat perustuvat ihmisten sille ohjelmoimiin sääntöihin, joiden perusteella ohjelma evaluoi sääntöihin perustuvan

parhaan siirron, joka ei välttämättä ole juuri oikeasti parhain siirto. Näihin sääntöihin, joita jo Shannon (1950) määritteli, kuuluu esimerkiksi materiaalin, eli nappuloiden numeeriset arvot; sotilaat ovat arvoltaan yksi ja ratsut kolme jne. Nappuloiden arvot Shannonin mukaan löytyvät kuviosta 1. Muita sääntöjä voivat olla kuninkaan turvallisuus, nappuloiden aktiivisuus ja ruutujen kontrollointi. Näiden perusteella ohjelma muodostaa evaluointifunktion, jonka tuloksena tuleva plusmerkkinen luku tarkoittaa että asema on parempi valkoiselle, ja miinusmerkkinen merkitsee että etu on mustalla. Jos peli on hyvin tasainen tietokone-evaluointi näyttää nollaa.

Kuvio 1. Nappuloiden arvot Shannonin (1950) mukaan

Hyvin yksinkertainen ohjelma voisi siis evaluoida aseman “-3“, jos valkoiselta on lyöty yksi ratsu mutta mustalta ei, ja muu materiaali on tasan pelaajien kesken. Jos taas ohjelma löytää pakotetun siirtojen kombinaation, joka johtaa shakkimattiin evaluointi voi olla esimerkiksi “M3“, eli pakotettu matti kolmella siirrolla. Täydellinen shakkiohjelma osaisi laskea pelin loppuun asti virheettömällä pelillä, ja evaluoisi siis aseman joko “±MX“ tai “0“ missä X = 0,1,2..6350. Eli ohjelma osaisi sanoa päättyykö peli täydellisillä siirroilla tasan vai voittaako jompikumpi, ja kuinka monella siirrolla, missä tahansa asemassa. Shannonin (1950) mukaan 6350 on suurin siirtomäärä, jonka shakkipelissä voi teoriassa tehdä 50-siirron säännön puitteissa. On kuitenkin epätodennäköistä, että läheskään näin monen siirron pakotettua voittoa löydettäisiin.

Nykyiset shakkiohjelmat arvioivat suuren määrän erilaisia asemia, joihin pelin senhetkisestä asemasta voidaan päätyä. Jotta laskentateho käytettäisiin pelin aikana mahdollisimman tehokkaasti, ohjelmat tekevät niin sanotusti “karsintaa“, eli vähentävät pelipuusta laskettavaksi otettavia haaroja. Karsinnassa evaluoinnista poistetaan pelipuun haarat, jotka epätodennäköisesti sisältävät parhaan pelin jatkumon, ja jäljelle jääneitä asemia analysoidaan yhä syvemmälle, jotta saadaan haettua pelipuusta paras siirto pitkällä tähtäimellä. Tunnetuimpiin shakkiohjelmiin kuuluva Stockfish tukeutuu haussa alfa-beta-karsinnaksi kutsuttuun algoritmiin. Uudemman malliset, koneoppimiseen perustuvat shakkiohjelmat, kuten Alphazero ja tämän pohjalta luotu Leela Chess Zero taas käyttävät Monte Carlo -puuhauksi kutsuttua algoritmia (Maharaj, Polson ja Turk 2022).

Reaaliajassa laskentaa tekevä shakkiohjelma kykenee absoluuttiseen arviointiin (0 tai ±MX) yleensä vain hyvin yksinkertaistetussa loppupelissä. Jotta ohjelma löytäisi absoluuttisesti parhaan siirron monimutkaisessa asemassa nopeasti, pitää kyseinen asema ja paras siirto olla tallennettuna johonkin tietorakenteeseen, josta shakkiohjelma kykenee nämä hakemaan.

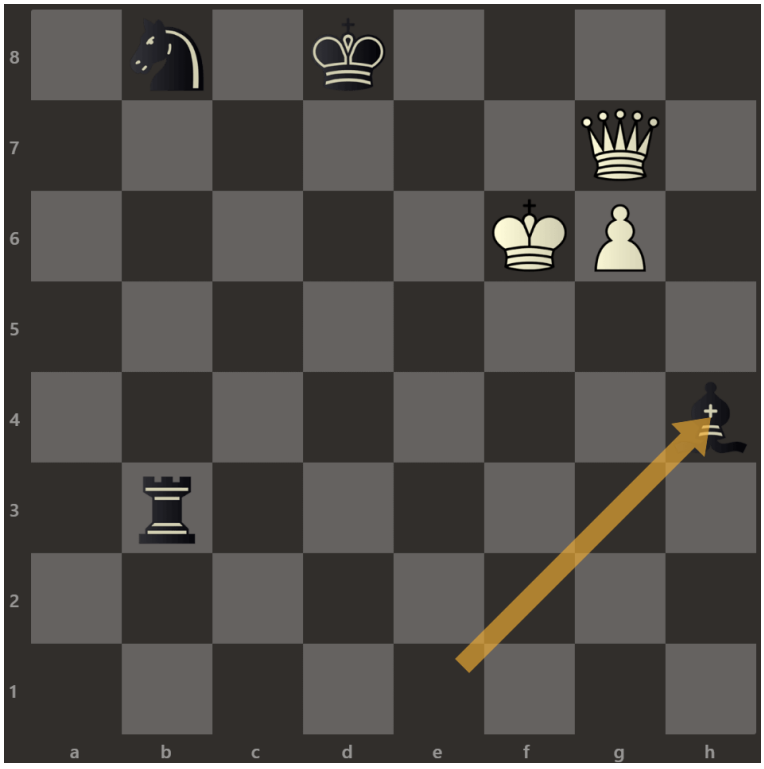
3 Shakkitietokantojen kehitys

Shakkitietokannat ovat ohjelmallisesti generoituja tietokantoja, jotka sisältävät vahvan ratkaisun tietyissä shakin loppupeleissä, eli kun suurin osa materiaalista on poistunut laudalta. Sanalla "shakkitietokanta"tarkoitetaan tässä tekstissä samaa kuin "loppupelitietokanta"ja englanniksi "endgame database"tai "endgame tablebase".

3.1 Historia

Ensimmäiset shakkitietokannat alkoivat kehittyä samaan aikaan kun tietokannat yleensäkin; 1960-luvulla, jolloin tietokoneiden käyttö ja suorakäyttöiset tallennusvälineet yleistyivät yrityksissä ja näiden käytöstä tuli kustannustehokkaampaa. Berg, Seymour, Goel ym. (2013) mukaan termi tietokanta (engl. database) esiintyi ensimmäistä kertaa 1960-luvun puolivälissä, jolloin myös ensimmäiset tietokantojen sovellukset shakkia varten kehittyivät. Bellman (1965) esitti optimaalisen pelitavan ohjelmoimiseksi sotilasloppupeleissä takautuvaa analyysiä, jossa analysoidaan siirtoja shakkimatista tai patista, eli pelin mahdollisesta lopputilanteesta taaksepäin. Parhaiden siirtojen löydyttyä asema ja tätä vastaava paras siirto tallennettaisiin tietokantaan. Tästä kehittyi ensimmäinen “Endgame database“, eli tietokanta jonka avulla tietokone osasi aina pelata parhaan siirron näissä yksinkertaisissa loppupeleissä. Thompson (1986) ym. saivat valmiiksi kaikki neljän ja viiden nappulan tietokannat. 1990-luvulla nimenomaan shakin loppupelitietokantoihin viitatessa alettiin käyttää termiä “tablebase“ sanan “database“ sijaan (Nalimov, Wirth ja Haworth 1999). Kuuden ja seitsemän nappulan tietokannat valmistuivat vuosina 2005 ja 2012 (Zakharov, Mal’kovskii ja Mostyaev 2019). Julkisesti saatavilla olevan tiedon mukaan tähän mennessä (4/2022) ainoastaan pieni osa 8-nappulaisista shakkiasemista on ratkaistu.

Alun perin vuonna 2012 Lomonosov-supertietokoneella luotu 7-nappulan tietokanta vei tallennustilaa 140 teratavua. Vuonna 2018 valmistunut 7-nappulan Syzygy-tietokanta vei sen sijaan vain 18.4 teratavua tallennustilaa. Tähän päästiin sisällyttämättä ratkaisuja joissa “voitto“ saavutettaisiin rikkomalla 50-siirron sääntöä, eli 50 siirtoa menisi ilman lyöntejä tai sotilaan siirtoja, jolloin virallinen peli päättyisi tasapeliin (Haworth 2019). Lomonosov-tietokanta kuitenkin sisälsi nämä “laittomat voitot“, ja löysi uskomattomalta vaikuttavia ratkaisuja loppupeleihin, kuten pisimmän tähän mennessä löydetyn shakkimatin, jonka pituus on 549 siirtoa (Zakharov, Mal’kovskii ja Mostyaev 2019). Tähän johtava asema on kuviossa 2.



Kuvio 2. Shakkimatti valkoiselle 549:llä siirrolla, ainoa voittava siirto on ♔f5

3.2 Haasteet

Shakin ratkaiseminen kokonaan (vahva ratkaisu) tarkoittaisi 32-nappulaisen tietokannan rakentamista. Tällöin tietokanta tietäisi jokaisessa laillisessa asemassa parhaan siirron, ja pelin lopputuleman tästä johdetulla täydellisellä pelillä. Vaikka teoriassa 32-nappulaisen shakkitietokannan luominen olisi mahdollista tunnetuilla algoritmeilla, käytännössä ollaan kaukana tämän toteuttamisesta laskennallisten resurssien ollessa rajallisia. Edes amerikkalaista (8x8) tammi-peliä ei ole ratkaistu vahvasti. “Heikko ratkaisu“ kuitenkin saatiin valmiiksi jo vuonna 2007, jolloin todettiin, että virheettömällä pelillä tammipeli päättyy tasapeliin. Tämän jälkeen tammi on nähty “ratkaistuna pelinä“(Schaeffer ym. 2007). Jo erilaisten sallitujen asemien määrä (engl. state-space complexity) shakissa on luokkaa 10^{44}, kun taas tammessa se on verrattain huomattavasti pienempi 10^{20}, vaikka pelejä pelataan samanlaisella 8x8-laudalla.

Vaikka Claude Shannonin 10^{90}-vuoden laskenta-ajan arviosta tietokoneiden laskentateho on huomattavasti kasvanut, se ei merkittävästi ole muuttanut sitä, että ratkaiseminen veisi silti käsittämättömän pitkän ajan edes parhailla supertietokoneilla. Shannon (1950) laski shakin pelipuun kompleksisuusluvuksi noin 10^{120}, jonka on todettu olevan merkittävästi suurempi kuin havaittavan maailmankaikkeuden atomien lukumäärä ($10^{78} - 10^{82}$). Shakin ratkaiseminen siis vaatinee jonkinlaisen läpimurron tietokonelaskennassa. Kvanttitietokoneiden on spekuloitu teoriassa kykenevän laskemaan shakin ratkaisemisen tapaiset ongelmat huomattavasti pienemmässä ajassa kuin klassisella periaatteella toimivat tietokoneet, mutta tähänastiset toteutukset kvanttitietokoneista eivät vielä ole esittäneet sellaista laskentatehoa, jolla shakki ratkaistaisiin kohtuullisessa ajassa.

3.3 Metodit

Kombinatoristen pelien ratkaisemisessa keskeinen menetelmä on taaksepäinen induktio (engl. backward induction). Shakin ratkaisemiseen viitatessa tästä käytetään usein termiä takautuva analyysi (engl. retrograde analysis). Esimerkiksi Zakharovin ym. (2019) mukaan Lomonosov-loppupelitietokannan (engl. Lomonosov-tablebase) luomiseksi tätä sovellettiin seuraavanlaisella algoritmilla yksinkertaistettuna:

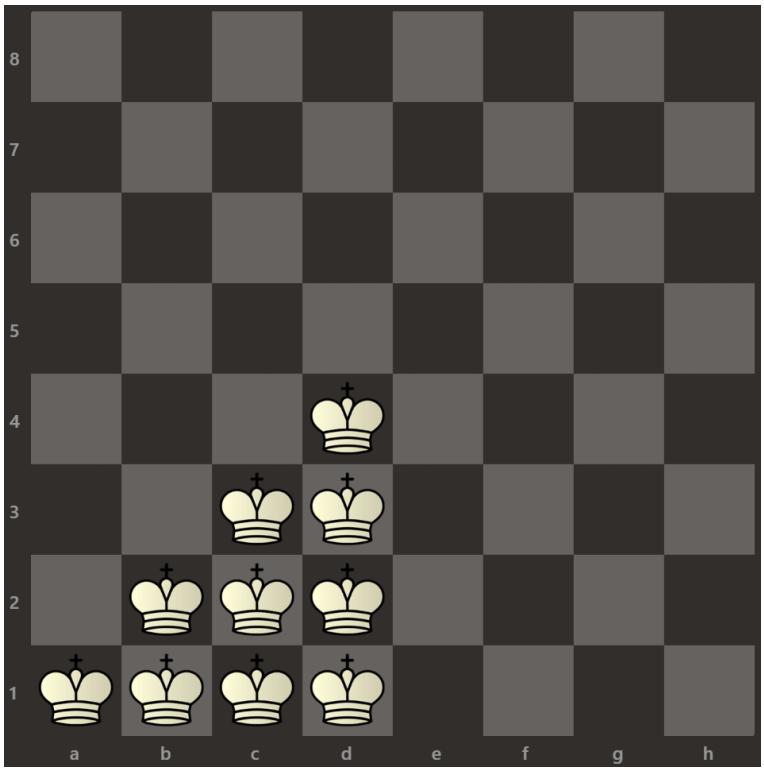
1. Jokaisesta asemasta (jossa 7 tai alle nappulaa) lasketaan ja tallennetaan mahdollisten siirtojen lukumäärä
2. Merkataan asemat jotka ovat shakkimatissa
3. Merkatuista asemista peruutetaan siirtoja edellisiin asemiin. Vastaavissa edellisissa asemissa päivitetään arvio ja vähennetään mahdollisten siirtojen määrä
4. Jos mahdollisten siirtojen määrä menee nollaksi, asema on ratkaistu. Merkitään tämä asema ja otetaan se kohtaan 3.
5. Lopulta prosessi päättyy, ja useimmat asemat on ratkaistu. Asemat joita ei ole ratkaistu katsotaan tasapeleiksi.

Alkuperäinen Lomonosov-tablebase vei tallennustilaa 140 teratavua, mutta tämä saatiin pakattua 96:een teratavuun. Algoritmien kehitys tarvittavan tallennustilan vähentämiseksi tietokannoissa ovatkin keskeinen ongelma. 8-nappulan tietokannan arvioidaan vievän jo yli 5 petatavua tallennustilaa.

Suuret tietokannat pitäisi myös toteuttaa niin, että niistä tietyn tiedon hakeminen olisi mahdollisimman tehokasta ja luotettavaa. Tämä ei Heinz (1999) mukaan toteutunut Thompsonin shakkitietokannoissa, joissa tiedonhaku oli hidasta Huffman-enkoodauksesta johtuen. Huffmanin koodaus on tiedonpakkauksessa käytettävä algoritmi, jossa käytetään puumallia. Heinz (1999) katsoi 1990-luvun alussa tehokkaammaksi 5-nappulan tietokannaksi Edwardsin luomat tietokannat, joita Edwards ei kuitenkaan saattanut loppuun asti. Näissäkin tietokannoissa oli kuitenkin Heinzin (1999) mukaan ongelmansa; Ne veivät huomattavasti enemmän tallennustilaa kuin oli tarpeen, koska niiden indeksöinti oli “kaukana optimaalisesta“.

Loppupelitietokanta, joka käyttäysi hyvin yksinkertaista indeksöintiä sisällyttäisi jokaiselle nappulalle 64 paikkaa, eli yhden jokaiselle shakkilaudan ruudulle. Tallennustilan tarvetta saadaan huomattavasti vähennettyä etenkin niissä tietokantojen osissa, jotka sisältävät asemat ilman sotilaita. Tämä johtuu siitä että muiden nappuloiden kuin sotilaiden siirrot ovat yleensä peruutettavissa. Tällöin monet samankaltaiset asemat voidaan katsoa samaksi asemaksi symmetrisyyden takia, vaikka täsmälliset nappuloiden koordinaatit eroaisivat toisistaan. Symmetrisyyttä havainnollistettu kuvassa alla; 64 kuninkaan eri paikkaa shakkilaudalla voidaan johtaa symmetrisyydellä esimerkiksi näistä kymmenestä kolmion muodossa olevasta ruudusta. Heinz (1999) selittää tämän periaatteen vähentävän sotilaattomien loppupelitie-

tokantojen indeksialuetta kuudennekseen verrattuna siihen, että symmetrisyyttä ei otettaisi huomioon.



Kuvio 3. Kuninkaat paikoissa, joista loput paikat voidaan johtaa symmetrisyydellä

3.4 Tietokantojen soveltaminen käytännössä

Yksi tärkeimpiä virstanpylväitä kilpailullisten shakkiohjelmien kehityksessä on shakkitietokone Deep Blue. Deep Blue on shakin pelaamista varten rakennettu ja ohjelmoitu tietokone, jonka katsotaan olleen ensimmäinen kone, joka on ylittänyt ihmisen kyvykkyyden rajat shakissa. Se päihitti shakin hallitsevan maailmanmestarin Garry Kasparovin kuuteen peliin rajoittuneessa ottelussa vuonna 1997. Campbell, Hoane Jr ja Hsu (2002) mukaan Deep Blue sisälsi kaikki loppupelitietokannat 5 ja alle nappulaisille asemille ja joitakin 6 nappulaisia tietokantoja.

Tekoälyn ja tähän kuuluvan koneoppimisen kehityksen myötä myös koneoppimiseen perustuvat shakkiohjelmat kuten Googlen kehittämä AlphaZero ja avoimen lähdekoodin shakkiohjelma Leela Chess Zero(Lc0) ovat kehittyneet viime vuosinakin vielä entistä paremmiksi, kauas ihmisten ymmärryksen yläpuolelle. McGrath ym. (2021) mukaan nämä shakkiohjelmat yltävät yli-inhimilliseen pelaamisen tasoon evaluoimalla vähemmän asemia perinteisiin shakkiohjelmiin verrattuna, joiden evaluointifunktiot ovat kovakoodattuja. Alphazeron evaluointifunktio perustui Silver ym. (2017) mukaan syvään neuroverkkoon, joka kehittyi Alphazeron peleistä itseään vastaan. Tätä kutsutaan koneoppimisen piiriin kuuluvaksi vahvistusoppimiseksi. Alphazero voitti vuosina 2017 ja 2018 pelatuissa otteluissa Stockfishin, joka voitti 2016 shakkiohjelmien mestaruuden. Alphazero ei tiedettävästi käyttänyt loppupelitietokantoja, kun taas Stockfish käytti Syzygy-tietokantoja.

Haque, Wei ja Müller (2021) huomasivat, että shakkiohjelmat eivät kuitenkaan vielä pelaa täydellistä shakkia edes ratkaistuissa loppupeleissä ilman loppupelitietokantoja, kun shakkiohjelman evaluointi tapahtuu reaaliajassa pelin aikarajojen puitteissa. Tästä johtuen shakkiohjelmat nykyään käyttävätkin tietokantoja hyväkseen, mikä mahdollistaa nopeat ja virheettömät siirrot loppupelissä. Tämän hetken vahvimmat shakkiohjelmat Stockfish ja Lc0 tukevat 7-nappulaisia Syzygy-loppupelitietokantoja.

Useat shakkiverkkosivut, jotka sisältävät analysointia varten shakkiohjelman, mahdollistavat myös shakkitietokantojen käytön loppupelejen analysointiin. Syzygy-tietokannat, jotka sisältävät loppupelitietokannat 7-nappulaiset mukaan lukien ovat vapaasti ladattavissa verkossa. Kaikkien 6-nappulaisten loppupelejen ratkaisujen luomisen mahdollistava ohjelmisto on myös saatavilla avoimena lähdekoodina. Kuvassa alla Syzygy-tietokannan verkkosivujen käyttöliittymä. Kyseisestä asemasta nähdään, että valkoisella on yksi voittava ja yksi tasapeliin johtava siirto, muiden siirtojen johtavan valkoisen häviöön, olettaen täydellisen pelin mustalta.

Kuvio 4. Syzygy-loppupelitietokannan käyttöliittymä
(https://syzygy-tables.info/)

4 Yhteenveto

Shakkiohjelmien kehityksellä ja shakkitietokannoilla on jo pitkä historia alkaen jo 1940-luvulta. Tällä hetkellä kattavimmat, valmiit tietokannat sisältävät ratkaisun kaikkiin loppupeleihin joissa on vähemmän kuin 8 nappulaa. Tietokantojen luomisessa käytetään takautuvaksi analyysiksi kutsuttua algoritmia, joka laskee parhaan siirron jokaiselle ratkaistavalle asemalle. Tietokantojen luomiseen vaadittava laskentateho ja tallennustila kasvavat eksponentiaalisesti lisättäessä nappuloita. Tästä johtuen suuremmat kuin 7-nappulaiset loppupelitietokannat vaativat resursseja jo niin suuren määrän, että niiden laskenta ja tallennus on nykypäivänä epäkäytännöllistä. Vaadittuja resurrseja saadaan vähennettyä esimerkiksi ottamalla huomioon tiettyjen asemien symmetrisyyden.

Shakki on vahvasti ratkaistu ainoastaan kun 7 tai alle nappulaa on pelissä. Ensimmäinen 7-nappulainen loppupelitietokanta luotiin jo lähes 10 vuotta sitten. Viimeaikoina suuria näkyviä edistysaskelia shakin ratkaisemisessa, eli loppupelitietokantojen kehityksessä ei ole tapahtunut. Myöskään tieteellisiä artikkeleita liittyen shakkitietokantoihin ei ole viime vuosina kauheasti julkaistu, suurimman osan näistä tällä hetkellä hakemalla sijoittuvat 1990-luvulle. Viime vuosina tietokoneshakkiin liittyvät tieteelliset julkaisut keskittyvät paljolti koneoppimiseen perustuviin shakkiohjelmiin.

Lähteet

Allis, L.V. 1994. “Searching for solutions in games and artificial intelligence” [kielellä English]. Tohtorinväitöskirja, Maastricht University, tammikuu. https://doi.org/10.26481/dis.19940923la.

Bellman, Richard. 1965. “On the application of dynamic programing to the determination of optimal play in chess and checkers”. *Proceedings of the National Academy of Sciences of the United States of America* 53 (2): 244.

Berg, Kristi L, Tom Seymour, Richa Goel ym. 2013. “History of databases”. *International Journal of Management & Information Systems (IJMIS)* 17 (1): 29–36.

Campbell, Murray, A Joseph Hoane Jr ja Feng-hsiung Hsu. 2002. “Deep blue”. *Artificial intelligence* 134 (1-2): 57–83.

Demaine, Erik D. 2001. “Playing games with algorithms: Algorithmic combinatorial game theory”. Teoksessa *International Symposium on Mathematical Foundations of Computer Science,* 18–33. Springer.

Haque, Rejwana, Ting Han Wei ja Martin Müller. 2021. “On the Road to Perfection? Evaluating Leela Chess Zero Against Endgame Tablebases”.

Haworth, Guy. 2019. “Chess endgame news: 7-man’Syzygy’DTZ50"EGTs”. *ICGA journal* 40 (4): 372–373.

Heinz, Ernst A. 1999. “Endgame databases and efficient index schemes for chess”. *ICGA Journal* 22 (1): 22–32.

Maharaj, Shiva, Nick Polson ja Alex Turk. 2022. “Chess AI: Competing Paradigms for Machine Intelligence”. *Entropy* 24 (4): 550.

McGrath, Thomas, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet ja Vladimir Kramnik. 2021. “Acquisition of Chess Knowledge in AlphaZero”. *arXiv preprint arXiv:2111.09259.*

Nalimov, Eugene V, Christoph Wirth ja Guy Haworth. 1999. “Kqqkqq and the Kasparov-world game”. *ICGA Journal* 22 (4): 195–212.

Ross, Don. 2021. “Game Theory”. Teoksessa *The Stanford Encyclopedia of Philosophy,* Fall 2021, toimittanut Edward N. Zalta. Metaphysics Research Lab, Stanford University.

Schaeffer, Jonathan, Neil Burch, Yngvi Bjornsson, Akihiro Kishimoto, Martin Muller, Robert Lake, Paul Lu ja Steve Sutphen. 2007. “Checkers is solved”. *Science* 317 (5844): 1518–1522.

Shannon, Claude E. 1950. “Programming a Computer for Playing Chess”. *Philosophical Magazine, Ser* 7 (41): 314.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel ym. 2017. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. *arXiv preprint arXiv:1712.01815.*

Thompson, Ken. 1986. “Retrograde analysis of certain endgames.” *J. Int. Comput. Games Assoc.* 9 (3): 131–139.

Zakharov, Victor B, Michael G Mal’kovskii ja AI Mostyaev. 2019. “On Solving the Problem of 7-Piece Chess Endgames”. *Programming and Computer Software* 45 (3): 96–98.