Aineopintojen harjoitustyö: TiRa
Toteutusdokumentti

Juha Inkinen
Ohjaaja: Kristiina Paloheimo

07.12.2012

Sisällysluettelo

1. Ohjelman yleisrakenne

Shakkiohjelma sisältää 10 luokkaa. Pelin varsinaiset logiikkaa ylläpitävät luokat ovat Shakki (main-luokka), Lauta, Tekoäly ja Siirtolaskuri. Luokat Lista, Puu, Solmu, Ruudut, Ruutu sekä Siirto ovat lähinnä apurakenteita tiedon käsittelyyn ja varastointiin.

Näiden luokkien tarkemman kuvauksen voi löytää java docista.

2. Aika- ja tilavaativuudet

Shakki on vaativa ohjelma, eikä sille ole olemassa ”ratkaisua”. [1] Teoreettiset aika- ja tilavaativuudet kasvaisivat siis täysin mahdottomiksi nykyisille laitteistoille. Siksi tekoälyn täytyy esittää valistuneita arvauksia siitä, että mikä siirto tulisi pelin mittaan kannattavimmaksi.

Tästä syntyy kuitenki uusi ongelma. Koska tekoäly ei pysty laskemaan tilannetta pelin loppuun asti, syntyy horisonttiefekti [2], joka aiheuttaa sen, että tekoäly voi tehdä ihmissilmin erittäin tyhmiä siirtoja. Tämä johtuu siitä, että tekoälylle asetetaan tarkastelusyvyys, jolloin voidaan päätyä tilanteeseen joka vaikuttaa hyvältä n siirron kohdalla, mutta n+1 siirron kohdalla johtaakin strategiseen katastrofiin.

Oma toteutukseni käyttää 6x6-lautaa rajatuilla säännöillä sekä vain viiden siirron syvyistä hakua, mutta silti tarkasteltavien siirtojen kokonaismäärä kasvaa pahimmillaan yli kahteen miljoonaan siirtoa kohden. Käytettävät puurakenteet ovat matalia, mutta erittäin leveitä ja siksi ohjelma hidastuu suuremmilla syvyyksillä.

Esimerkiksi syvyyden kasvattaminen yhdellä viidestä kuuten kasvattaa alkutilanteen perusteella rakennetun puun solmujen määrää noin 200 tuhannesta yli kolmeen miljoonaan. Kokeilin myös syvyyden kasvattamista seitsemään, mutta silloin ohjelman aikavaativuus kasvoi jo niin suureksi, etten jaksanut odottaa tulosten valmistumista edes alkutilanteessa.

Yksittäisen siirron valitseminen vie syvyydellä 5 noin 1,5s. Kun syvyys on 6 kasvoi aikavaativuus yli 25 sekuntiin.

3. Puutteet ja parannusehdotukset

Shakkitekoälyn toteutuksessa ongelmaksi syntyi se, että pelin peruslogiikan toteuttamiseen meni paljon aikaa ennenkuin pääsi käsiksi ratkaistavan ongelman ytimeen. Tekoälyn sain kunnolla toimivaan kuntoon demoa edeltävänä iltana, joten sen virittelyyn jäi huonosti aikaa. Siksi myös suuri osa suunnitteludokumentissa haaveilluista ominaisuuksista jäi teoteuttamatta.

Toistaiseksi pelistä puuttuu kokonaan voitto/häviö-tarkastelu, sekä kuninkaan erikoisaseman tarkkailu. Lisäksi käytössä on mini-shakki variaatio [3], joka vielä karsii joitain ominaisuuksia. Olisi siis hyvä jatkokehittää ohjelma toimimaan myös normaaleilla säännöillä, sekä 8x8 laudalla.

Lisäksi olisi mielenkiintoista vertailla erilaisten heurestiikoiden vaikutusta pelitilanteessa. Tällä nappuloiden heurestiset arvot on tallennettu Lauta-luokan taulukkoon, mutta esim niiden vaihtaminen kesken pelin tai erilaisten arvojen antaminen kahdelle tekoälylle on mahdotonta. Nämä ominaisuudet loisivat mielenkiintoa vertailuun.

Myös AI-vs-Ihminen pelimahdollisuus olisi kiinnostava kokeilu, jonka avulla näkisi onnistuuko luomaan tekoälyn, joka voittaa ohjelmoijansa (mikä olisi luultavasti aika helppoa).

Jatkokehittäminen myös koneoppimisen kannalta voisi osoittautua hyväksi ideaksi. Esimerkiksi loppusiirto-tietokantojen ja vastaavien apuvälineiden käyttö voisi muuttaa tekoälyä paljon tehokkaammaksi ja viisaammaksi.

4. Lähteet

[1] http://en.wikipedia.org/wiki/Solving_chess (4.12.2012)

[2] http://www.chessville.com/misc/whychessengines.htm (6.12.2012)

[3] http://en.wikipedia.org/wiki/Minichess (31.10.2012)