Määrittelydokumentti

Koulutusohjelma: Tietojenkäsittelytieteen kandidaattiohjelma

Käytetty kieli: Python

Hallitsen myös C++ - ja Java-kieliä alenevassa järjestyksessä mutta molempia siinä määrin että pystyn kyllä arvioimaan niillä tuotettua koodia. Haskelia pystyn ehkä myös tulkitsemaan mutta se ei ole kovin tuoreessa muistissa.

Aion luoda Reversiin tekoälyn. Ideoin aihetta niin että mielestäni Reversin asetelma on sellainen, että alkuvaiheessa on pakko nojata strategiaan eli heuristisiin sääntöihin, jotka tässä tapauksessa olisivat, että pyritään ottamaan aina kulmat haltuun ja välttämään siirtoja, jotka antavat vastustajalle mahdollisuuden saada kulma itselleen, tämän jälkeen, jos kulma on hallussa myös kulmaa vasten olevat seinäruudut ovat arvokkaita, jos niillä on vain omaa väriä, tällä tavalla voi rakentaa itselleen varmoja pisteitä. Jossain vaiheessa peliä vaihtoehdot muuttuvat sen verran rajalliseksi, ja tilanne hallittavan deterministiseksi, että on mahdollista vaihtaa puhtaaseen minmax-arviointiin alpha-beta karsinnalla. Kun laudalla on jäljellä 8 ruutua niin mahdollisten siirtojen määrä on maksimissaan 8! (todennäköisesti paljon vähemmän sillä kaikki ruudut ovat hyvin harvoin pelattavissa). Intuitiivisesti ja laskematta arvioin, että noin siinä vaiheessa, kun pöydällä on 7-9 tyhjää ruutua niin tekoälyn toiminta voidaan vaihtaa puhtaaksi minmax-optimoinniksi, mutta sopivan raja tälle selkenee myöhemmin.

Toinen vaihtoehto olisi käyttää pelin alkuvaiheessa monte carlo-simulaatiota mutta ainakin omasta mielestäni pelin alkuvaiheella ei ole juuri väliä ja heuristiset säännöt voivat toimia paremmin koska tärkeintä on vain välttää selkeät virheet. Reversin tilannetta on vaikea arvioida ja lukuun ottamatta varmoja pisteitä tilanne voi kääntyä hyvin äkkiä, siksi on järkevämpää nojata strategiaan ja lopuksi algoritmiseen optimointiin.

Eli rakenne olisi seuraava:
Vaihe 1: siirrot 1-30, pelkkää virheiden välttämistä heuristin säännöin

Vaihe 2: siirrot 30- n. 52 sama heuristiikka mutta alamme painottaa eri muuttujia eri tavoin, lisäksi tässä vaiheessa peliä voi tulla tilanteita, joissa vastustaja voidaan pakottaa pelaamaan niin että saadaan kulma itselle, tähän voisi olla oma minmax (jos toteutuskelpoinen idea, en osaa vielä arvioida sen haastavuutta, voisi toimia esim. niin että jos optioita on vähän ja ollaan lähellä kulmaa, niin lasketaan 3 siirron päähän).

Vaihe 3: alle 10 siirtoa jäljellä, puhdas minmax alpha-beta.

painotuksia olisivat kulmat, varmat ruudut esim. kulmien vieressä ja optioiden pitäminen avoimena, joka on osin myöskin yhteydessä omien nappuloiden määrään, joka on viimeinen painotus.

Syötteinä olisi pelilaudan tilanne, mahdolliset siirrot, vuorossa oleva pelaaja. Tämän voi ehkä toteuttaa tekstipohjaisena matriisina joka kuvaa pelilautaa sekä myöskin logata toteutuneet tapahtumat sekä mahdollisesti algoritmin käyttämät metodit ja niiden syvyys. Ihmispelaaja voi syöttää siirtonsa koordinaatteina matriisiin. Mahdollisesti myös pygame-toteutus, en ole vielä perehtynyt että kumpi olisi nopeampi tehdä.

Minimaxin aikavaativuus on $O(b^d)$ mutta d eli syvyys voidaan puolittaa käyttämällä alpha-beta pruningia (karsintaa) joka tehostaa toimintaa huomattavasti. Soveltaen kyseiseen aiheeseen b on siirtovaihtoehtojen määrä vuorolla mikä on usein rajattu, oman arvion mukaan keskimäärin 5-6 ja vielä laskettavissa oleva syvyys selvinnee työn kehittyessä mutta pelin keskivaiheilla se voisi olla 2-3 siirtoa jos optimoidaan keskipelin taktiikoita ja loppuvaiheessa hieman enemmän koska oletan että silloin b on melko varmasti alemmalla tasolla. Olennaista tässä mielestäni on myös, että vapaiden ruutujen teoreettinen enimmäismäärä laskee jokaisen pelatun nappulan jälkeen, joten b:n arvo ei ole varsinaisesti vakio (sama pätee myös d:n suhteen, kun peli lähenee loppua), ja se juuri mahdollistaa sen, että loppupelissä on mahdollista alkaa käyttämään raakaa voimaa tilanteen selvittämiseksi.

Harjoitustyön ydin on toteuttaa algoritmi, joka nojaa heuristisiin sääntöihin silloin kun tilanne ei ole helposti laskettavissa tai sen laskeminen ei ole järkevää ja vaihtaa puhtaaseen optimointiin silloin kun se on mahdollista tai tuo merkittävää etua. Sama rakenne toteutuu myös ihmispelaajilla monimutkaisissa peleissä joita ei voida ”ratkaista” loppuun asti kuten shakissa strategia (avoimet linjat, vahvat asetelmat, momentum, jne) vs. taktiikka (puhtaasti laskettavissa olevat etua tuovat siirtokuviot, joissa vastustajan vaihtoehdot ovat vähissä).