Aplicarea cautarii in arbore de tip Monte Carlo pe un joc de masa

Rezultate si concluzii

Rezultate

Am rulat proiectul pe telefon si am cerut mai multor persoane sa joace acest joc pentru a avea un rezultat cat mai bun.

Rezultatul este 15% win rate (7 jocuri castigate dintr-un total de 46)

Ce am observat este ca AI-ul pierde in momentul in care pe tabla mai exista mutari in asa fel incat se mai creeaza o mutare disponibila sau se reduce o mutare. In aceste conditii, un jucator poate invinge AI-ul deoarece poate sa isi mai creeze o mutare disponibila in ultimele runde ale jocului. Aceasta problema poate fi rezolvata in viitor printr-o euristica, deoarece, in prezent, AI-ul adopta o strategie de joc agresiva, acesta mutand, concentrandu-se direct pe victorie.

Aceasta euristica ar fi una defensiva, blocand adversarul in drumul sau spre a castiga jocul.

Configurari experimentale

Datorita faptului ca suntem limitati de memorie si de timp, AI-ul face miscari prostesti cat timp pe tabla sunt mai mult de 30 de patratele libere (cum am discutat si in analiza problemei, primele mutari sunt aproape inutile => este inutil sa folosim putere de procesare pe ceva ce nu aduce niciun rezultat). AI-ul se concentreaza mai mult pe ultimele mutari. Inca o limitare o reprezinta numarul de simulari. Datorita faptului ca nu putem tine pe loc jocul pentru a simula enorm de multe jocuri, numarul maxim de frunze in arbore la care poate sa ajunga algoritmul in timpul unei runde este de 5000.

Pentru ca Monte Carlo este un tip de cautare care nu necesita procesarea intregului spatiu de cautare, executia poate fi oprita oricand, in cazul nostru, cand pragul de 5000 de frunze a fost atins, iar urmatoarea miscare se alege pe baza rezultatelor curente.

Analiza matematica a parametrilor:

Daca modificam parametrul ca sa declansam mai repede AI-ul, acest lucru nu o sa mareasca sansele de succes. Datorita faptului ca pentru acest algoritm conteaza doar ultimele mutari, este inutil sa pornim algoritmul mai din timp.

In sens invers, daca pornim algoritmul prea tarziu, este posibil ca acesta sa nu mai aiba niciun folos, deoarece domeniul de cautare este foarte mic si nefavorabil.

Daca modificam numarul maxim de simulari, algoritmul o sa genereze din ce in ce mai multe simulari, numarul acestora fiind imens la inceput, dar din nefericire, inceputul jocului nu este foarte important pentru AI.

Daca restrangem numarul maxim de simulari, este posibil sa pierdem anumite solutii din domeniul de cautare.

Memorie – creste direct proportional cu numarul de simulari si nu este afectata in mod direct de momentul in timp al inceperii executiei AI-ului

Timp – creste direct proportional cu numarul de simulari si este afectat direct de momentul in timp al inceperii executiei AI-ului

Posibile imbunatatiri ale AI-ului

Alternarea stilului de joc: agresiv/defensiv pentru anumite situatii cum ar fi:

Uneori, din analiza arborelui pentru un stil de joc agresiv, gasim o mutare cu fitness cel mai mare, dar relativ mic, in comparatie cu analiza arborelui pentru un stil de joc defensiv. Implementarea unor euristici cu cazuri speciale in care cautarea in arbore nu ofera un rezultat foarte bun.