Go igrica

Ilija Simić RA15/2014

UVOD

Go je misaona igra na tabli za dva igrača. Na polju koje je uglavnom 19x19(mada table mogu biti i manje 13x13, 9x9), igrači naizmenično stavljaju kamenje. Prvi je crni igrač, te je stoga belom igraču koji se nalazi u lošijem položaju dodaljen bonus od 6.5 poena(*komi*). Cilj igre je osvojiti što veću teritoriju. Svaki postavljeni kamen i svaki zarobljen vrede po 1 poen. Kamen je zarobljen kad on ili grupu u kojoj se on nalazi, nema ni jednu slobodnu poziciju. Grupa predstavlja skup međusobno susednih elemenata. Postoji još pravilo (*ko*) koje onemogućava ponavljanje istih situacija na tabeli. Kraj partije nastaje kada dva igrača uzastopno pasiraju.

Kompjuterski go je oblast veštačke inteligencije koja za cilj ima stvaranje programa koji igra go. Istraživanje je započeto 70ih godina prošloga veka, a kulminiralo je 2016. godine kada je *AlphaGo* uspeo da pobedi profesionalnog igrača Go-a.

SOFTVER

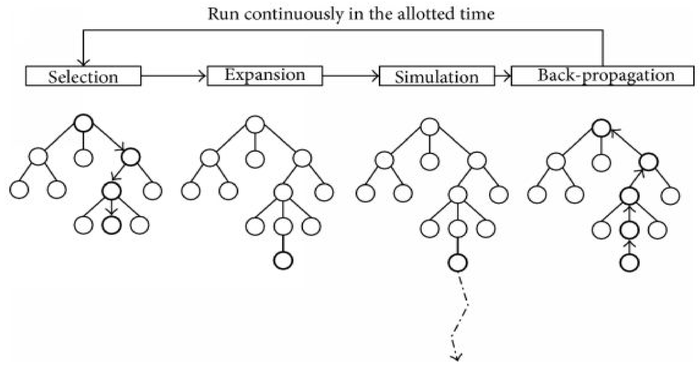
Softver koji je razvijen za potrebe ovog projekta, koristi python biblioteku Flask za server i JavaScript biblioteku WGo.js na klijentskoj strani. Svaki potez korisnika, poziva metodu koja na osnovu *Monte Carlo Tree Search* algoritma, odigrava sledeći potez.

IZBOR ALGORITMA

Svaka *zero-sum* igra kakva je i go, može se predstaviti putem stabla igara(*game tree*), gde su čvorovi moguća stanja, a ivice predstavljaju dozvoljene poteze. *Minimax* algoritam nam omogućava pronalazak najboljeg rešenja, čak i u slučajevima perfektne igre protivnika. Postoje razna varijacije Minimax algoritma(*Alpha-Beta pruning*) koje smanjuju prostor stanja, radi ubrzavanja, ali i oni imaju loše performanse u igrama koje imaju visok faktor grananja(poput Go-a). *Monte Carlo Tree Search* prevazilazi te nedostatke. Prvo, uvek forsira najobećavajuće podstablo, te zbog toga stablo raste asimetrično što je prednost kod igara sa visokim faktorom grananja. Drugo, MCTS ne zahteva evalucionu funkciju(primera radi evaluciona funkcija kod Deep Blue se sastoji od 8000 delova), dovoljno je poznavati pravila igre i imati mogućnost generisanja poteza. Treće, MCTS može biti prekinut u svakom momentu, rezultat će biti najobećavajući u tom momentu.

MONTE CARLO TREE SEARCH

Ovaj paragraf će detaljniji objasniti kako MCTS algoritam funkcioniše. Proces kreiranja stabla pretrage se sastoji iz 4 faze. Svaki čvor pored stanja sadrži i informaciju o broju posetu, broju pobeda koji je doneo, pokazivača na roditelja i pokazivače na decu.
Selekcija: Polazeći od korena **R** prolazi se kroz stablo dok se ne dođe do lista **L**. Sve vreme se bira najbolji mogući čvor.
Ekspanzija: Ukoliko u L nije kraj partije, kreirati novi čvor **C** i izabrati ga.
Simulacija: Odigrati simuliranu igru od **C**, tako što svaki igrač bira nasumične poteze
Propagacija unazad: Kretajući se u nazad kroz stablo, od lista **C** do korena **R** i ažuriraju informacije u čvorovima



Izbor čvora se vrši tako što se maksimizuje vrednost formule $v_i + C * sqrt(ln(N)/n_i)$, *gde je* v_i vrednost količnika broja pobeda i broja poseta za dati čvor, *N* broj poseta roditelja, n_i broj poseta za dati čvor, a *c* konstanta koja se najčešće uzima da je *sqrt(2).* Ova formula naziva se *Upper Confidence Bounds,* a modifikacija MCTS koja je koristi *Upper Confidence Bounds for Trees (UCT).* Koristi se da bi se napravio balans između istraživanja i eksploaticije stabla. Dok čvor još nije posećen, njena vrednost je beskonačno.

ZAKLJUČAK

MCTS, sam po sebi, nije dovoljan za igranje Go-a. Najbliže tome je Pachi, koji još pri izboru pozicije koristi 3x3 šablone i raznorazne taktičke provere, čineći simulaciju polunasumičnom.

Moguća poboljšanja ovog softvera su: keširanje svih grupa na tabeli, povećanje broja simulacija, paralelizacija u toku pretrage(na primer u fazi simulacije), izmena simulacije tako da bude nasumična. Još jedno od mogućih poboljšanja je i ubacivanje dubokog učenje, gde bi neuronske mreže bile obučavane na osnovu partija profesionalaca i na taj način pomogle pri izboru odgovarajućih poteza.

Primera radi, kod *AlphaGo*, MCTS je vođen od strane *value network* i *policy network.* Value network služi da predvidi da li će trenutni igrač pobediti ili izgubiti, dok policy network predviđa kako će protivnik odigrati u datom trenutku.

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.
[2] Petr Baudiš MCTS with information sharing
[3] http://cameronius.com/research/mcts/about/index.html