Uporaba in analiza Monte-Carlo drevesnega preiskovanja na strateški igri

DISPOZICIJA

Jernej Habjan
jh0228@student.uni-lj.si

predvideni MENTOR: doc. dr. Matej Guid
Fakulteta za računalništvo in informatiko Univerze v Ljubljani

14. december 2017

Povzetek

Računalnik, ki igra proti človeškem nasprotniku lahko sprogramiramo na klasičen način pogoj - akcija. Tako so narejene klasične realnočasovno strateške igre, vendar gre veliko časa za implementacijo posameznih sovražnikovih napadov in nabiranju virov. Lahko pa razvijemo algoritem, ki sam ugotovi, katera je optimalna odločitev v določeni situaciji in jo za tem izvede. Na preprostih problemih z malo odločitvami in kratkimi igrami kot je igra križci in krožci, lahko izberemo preprostejše algoritme kot je algoritem Minimax, kjer igralec hoče povečati svojo možnost zmage, nasprotnik pa mu hoče to možnost zmanjšati. Obstajajo tudi algoritmi, ki se naučijo igre iz učne množice, ki predstavlja nekaj iger, pri tem pa računalnik ugotovi zakonitosti igre in način igranja. Taki algoritmi so nevronske mreže ali globoke nevronske mreže, ki pa so zelo računsko potratni, vendar vračajo izjemne rezultate. Lahko pa uporabimo algoritem Monte-Carlo drevesno preiskovanje, ki deluje na principu hevrističnega preiskovanja prostora, kjer imamo prostor stanj (graf, drevo), množico dosegljivih stanj in povezave med stanji. Algoritem bom uporabil na svoji igri narejeni v celostnem pogonu Unreal Engine 4 imenovani Trump Defense 2020.

1 Motivacija za izbrano diplomsko temo

Razvijanje inteligentnega agenta v realno-časovnih igrah je problem, s katerim se mora soočiti večina razvijalcev teh iger, agentove akcije so pa pogosto predvidljive, saj se človeški igralec nauči njihovih načinov delovanja in jih tako lažje premaga. Če pustimo agentu, da sam opravlja akcije nekontrolirano, bo izvajal naključne akcije, ki so pa slabše kot vnaprej definirana taktika. Če pa agentu podamo hevristiko, po kateri se mora ravnati, bo

poskušal izvesti čim boljše akcije, vendar bo to trajalo zelo dolgo, saj bo moral preiskati cel preiskovalni prostor, ki pa pri realno-časovnih strateških igrah zna biti zelo velik. Da bi agent lahko poiskal cel prostor, bi ga morali zelo abstraktirati, vendar še takrat se moramo posluževati algoritmov kot je Monte-Carlo drevesno preiskovanje, ki uporablja naključnost, s katero se pomika skozi preiskovalni prostor. S takim algoritmom lahko dosežemo inteligentnega agenta, ki se prilagaja nasprotnikovim akcijam, vsako igro izbira nov način igranja in deluje dovolj hitro brez daljšega učenja na superračunalnikih, kot to potrebujejo nevronske mreže. Namen diplomske naloge je razviti algoritem na svoji realno-časovni strateški igri narejeni v celostnem pogonu Unreal Engine 4, ki bo dovolj inteligenten da premaga človeškega nasprotnika.

1.1 Pregled področja in sorodnih del

Najbolj popularne tematike na področju igranja iger je sedaj računalniški program AlphaGo, ki uporabja kombinacijo globokih nevronskih mrež, strojnega učenja in drevesnega preiskovanja, in sicer Monte-Carlo drevesno preiskovanje. Algoritem se je učil s pomočjo spodbujevalnega učenja, prav tako so pa algoritem naučili na podlagi 30 milijonov potez. V oktobru 2017 so izdali novejšo verzijo AlphaGo Zero, ki pa ne uporablja znanja ekspertov, vendar se je algoritem naučil potez tako, da je igral sam proti sebi in tako premagal svojega predhodnika 100:0.

V delu Monte-Carlo tree search and rapid action value estimation in computer Go je opisana metodologija pri reševanju igre GO in metoda RAVE (Rapid action value estimation), ki spremlja število zmag v vseh dosedaj raziskanih vozliščih in to število vpliva na izbiro novega vozlišča [1].

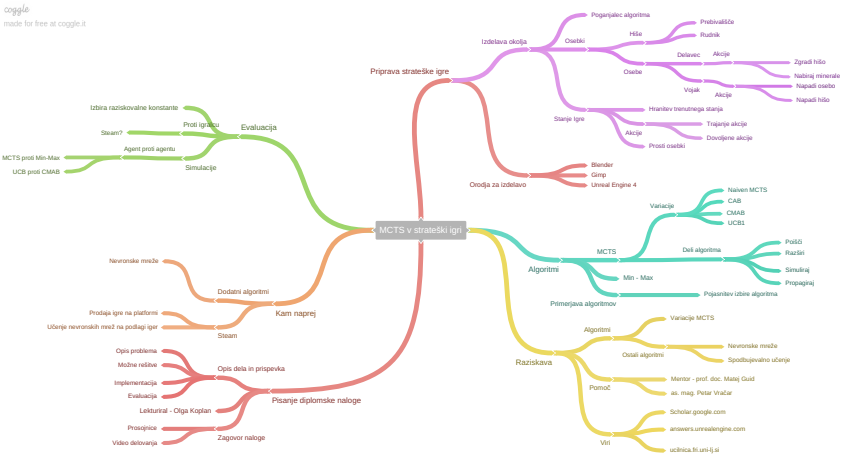
Uriarte z Drexel univerze je za uporabo Monte-Carlo drevesnega preiskovanja v dveh delih opisal abstrakcijo prostora in akcij.
V prvem delu je predstavil stanje, opisal okolico in abstrahiral mapo na posamezne dele, kjer ima vsak definirano svojo velikost [2]. Leto pozneje je v delu Game-Tree Search over High-Level Game States in RTS Games predstavitev stanj in prosotra še izboljšal. Opredelil je še posamezne akcije, ki se zgodijo ob porazu vojske ali ob koncu igre [3].

1.2 Zakaj je predlagani mentor primeren

Izbral sem prof. doc. Mateja Guida, saj ima izkušnje z velikim kombinatoričnim prostorom, ki ga ponujata šah in realno-časovne strateške igre. Prav tako je pri šahu uporablal tehniko MCTS, ki jo bom jaz uporabil pri igri. Realno-časovne strateške igre so predvsem strateške igre kot šah, vendar da se izvede toliko več ukazov, saj lahko namesto ene figure, premaknemo več figur v vsaki potezi. Tako da se bo problem, ki nastane pri igranju

šaha lahko samo razširil na več figur na potezo. Prav tako je predlagani mentor raziskal veliko različnih preiskovalnih algoritmov in mi bo lahko dal napotke, kako razširiti projekt z dodatnim algoritmom oziroma ga pripraviti na delujoče stanje z lažjim.

2 Miselni diagram za izbrano temo diplomske naloge



Slika 1: Miselni vzorec za mojo diplomsko nalogo.

Dopolnjen miselni vzorec (Slika 1).

3 Predvideni prispevki diplomske naloge

Rezultat diplomske naloge bo realno-časovna strateška igra, ki ne vsebuje tradicionalega sprogramiranega inteligentnega agenta, vendar agenta, ki igra na podlagi nasprotnikovih akcij. Rešitve, kot je naprimer število simulacij algoritma MCTS bom primerjal s številom simulacij v drugih igrah kot naprimer šah ali igri Micro-Rts in s tem preverjal hitrost delovanja algoritma. Prav tako bom ocenil igranje računalnika proti računalniku z različnimi algoritmi in vodil statistiko igranja računalnika proti človeškim igralcem.

4 Uporabljena metodologija

Za izdelavo igre bom uporabil celostno okolje Unreal Engine 4, v katerega bom uvažal 3D modele, narejene v programu Blender, ki pa potrebujejo za izgled teksturne slike, predelane v programu Gimp. Algoritem bom razvijal prav tako v okolju Unreal Engine 4, ki pa ima podporo za kodiranje v

programu Microsoft Visual Studio. V tem programu bom lahko pisal C++ kodo, ki je veliko hitrejša kot način programiranja v okolju Unreal Engine 4. Preizkusiti nameravam hevristične algoritme, ki bodo izboljšali agentovo razmišljanje, kot naprimer Monte-Carlo drevesno preiskovanje in njegove variacije, spodbujevalno učenje in druge. Testne podatke bom pa pridobil s člankov, v katerih je avtor uporabil te algoritme pri realno-časovnih strateških igrah. Tak podatek je število simulacij, ki pomeni hitrost algoritma in abstrakcija okolja.

5 Razdelitev potrebnega dela na aktivnosti

6 Potrebne aktivnosti za izdelavo moje diplomske naloge

1. **izdelava strateške igre**
   Igro moram izdelati v celostnem pogonu Unreal Engine 4. Igra bo realno-časovna, kar pomeni, da jo moram dobro optimizirati, če hočem poganjati hevristične preiskovalne algoritme. V tem okolju bom lažje zasnoval grafično prezentacijo algoritma.

   (a) izdelava poganjalca algoritma,

   (b) izdelava osebkov,

   (c) predelava stanja igre.

   Izdelati moram poganjalca algoritma, ki bo izbral določen algoritem in z njim igral proti nasprotniku. Prav tako moram izdelati osebke in njihove akcije (npr. postavi hišo). Te osebke bom pa hranil v stanju igre, ki pa ga algoritem uporablja, ki je abstrakcija za to, kateri osebki in akcije so trenutno na voljo.

2. **raziskava algoritma**
   Ko bom imel izdelano ogrodje strateške igre, se bom lahko posvetil algoritmom. Preiskal bom algoritme:

   (a) Monte-Carlo drevesno preiskovanje,

   (b) variacija Monte-Carla, in sicer Naiven MCTS,

   (c) metode ocenjevanja CAB, UCB1,

   Prav tako bom moral hkrati opisati, zakaj sem se za kateri algoritem odločil.

3. **implementacija algoritmov**
   Ko bom imel algoritem dokončan, ga bom lahko implementiral v igro. Postopek implementacije bo naslednji:

(a) implementacija Min-Max algoritma,
(b) implementacija naključnega MCTS,
(c) sprememba MCTS, da uporablja pravilne ocene,
(d) sprememba preiskovalnega parametra pri ocenjevanju.

4. **ovrednotenje rezultatov**
   Ovrednotenje rezultatov bo pa potekalo po naslednjih korakih:

   (a) simulacija igre računalnika proti računalniku (več sto simulacij),
   (b) igranje računalnika proti človeku (testiranje znancev).

   Pri simulaciji računalnika proti računalniku, se bom osredotočil na rezultate kot so naprimer povprečno število simulacij pri MCTS algoritmu in konsistentnost ukazov. Pri igranju proti človeku bom pa analiziral nekaj odločitvenih dreves ki jih je računalnik zgeneriral in ocenil odločitve na podlagi mojega predznanja igre.

5. **pisanje diplomske naloge**
   Pisanje diplomske naloge bo potekalo med raziskovanjem algoritmov in ovrednotenjem rezultatom. Prav tako bom opisal zakaj sem izbral določen algoritem in kaj mi doprinese k rezultatu. Opisal bom tudi celostni pogon Unreal Engine 4 in kratek opis strateške igre Trump Defense 2020.

6. **zagovor**
   Za zagovor moram pripraviti prosojnice v Microsoft Office, ki bodo glavna podlaga pri prezentaciji. Prav tako bom moral pripraviti video, v katerem bom lahko predstavil igro in končen rezultat.

Čas izdelave strateške igre bo med študijem trajal še kakšen mesec, je pa igra v izdelavi že dva meseca. Raziskava algoritmov bo hitrejša operacija, saj je potencialni mentor doc. Matej Guid v preteklosti že veliko delal s takimi algoritmi, in me bo hitro znal usmeriti. Ta proces zna trajati kakšen teden, saj bom sproti dopolnjeval diplomsko nalogo. Ko bom našel primerne algoritme, jih bom moral implementirati, kar pomeni, da bom moral predelati igro in hkrati testirati če algoritem dela. To lahko traja 2 meseca med študijem. Potem bom moral še ovrednotiti rezultate, kar lahko traja tudi kakšen mesec, ker bom ugotovil, da algoritem ne dela dovolj dobro in ga bom spreminjal. Diplomsko nalogo bom pa pisal en mesec.

Skupen čas celotnega projekta potem znaša približno 6 mesecev prekinjajočega dela. Imam še štiri mesece dela za diplomsko nalogo, kar je spremenljiv čas, glede na to, da je še prvi semester.

Kritična aktivnost je implementacija algoritma, saj moram še dobro raziskati kako bi se lotil rešitve in vpeljati različne vrste algoritmov v realnostrateško igro.

7 Preliminarno kazalo

1. izdelava strateške igre
   (a) izdelava poganjalca algoritma,
   (b) izdelava osebkov,
   (c) predelava stanja igre.
2. raziskava algoritma,
   (a) Monte-Carlo drevesno preiskovanje,
   (b) variacija Monte-Carla, in sicer Naiven MCTS,
   (c) metode ocenjevanja CAB, UCB1,
3. implementacija algoritmov,
   (a) implementacija Min-Max algoritma,
   (b) implementacija naključnega MCTS,
   (c) sprememba MCTS, da uporablja pravilne ocene,
   (d) sprememba preiskovalnega parametra pri ocenjevanju.
4. ovrednotenje rezultatov,
   (a) simulacija igre računalnika proti računalniku (več sto simulacij),
   (b) igranje računalnika proti človeku (testiranje znancev).

8 Seznam literature

Literatura

[1] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[2] Alberto Uriarte and Santiago Ontañón. Game-tree search over high-level game states in rts games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.

[3] Alberto Uriarte and Santiago Ontanón. Automatic learning of combat models for rts games. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.