Umělá inteligence pro Válku kostek

VUT Fakulta Informačních Technologií v Brně

SUI 2021/2022

Autori: Matej Záhorský `xzahor04`
Matúš Škuta `xskuta04`
Patrik Németh `xnemet04`

1 Princíp hry

Pre kontext sa v krátkosti uvedie princíp hry. Hra Dicewars spočíva v dobíjaní území hracej plochy rozdelenej medzi zúčastnených hráčov. Územia obsahujú hracie kocky, pričom každé územie ich obsahuje minimálne jednu a maximálne osem. Pri dobíjaní územia sa „hodí“ kockami z dvoch území, ktoré sa zúčastňujú bitky. Ak útočníkov súčet hodnôt hodených kociek je vyšší, než obrancov, útočník obsadzuje. Ak je nižší alebo rovný ako obrancov súčet, tak sa neobsadzuje, útočník stráca všetky kocky a obranca stráca kocky dané vzťahom $\lfloor \frac{\text{počet kociek útočníka}}{4} \rfloor$. Vyhráva hráč, ktorý obsadí celú hraciu plochu.

2 Implementácia

Implementácia hráča s umelou inteligenciou (ďalej iba *bot*) je zameraná na prehľadávanie stavového priestoru pomocou Alfa-Beta algoritmu, rozhodovanie o optimálnosti útoku pomocou neurónovej siete a na prehľadávanie priestoru hracej plochy za účelom vhodných podporných presunov kociek. Jedno kolo bota je rozdelené na tri možné fázy: **útok**, **obrana**, **presun**. Poradie týchto fáz sa nemení, avšak je možné preskočiť prvú fázu, prípadne nevykonať ani jednu. Rozhodnutie o prípadnom preskakovaní fáz je vykonané na začiatku každého kola na základe výstupu Alfa-Beta algoritmu.

2.1 Fázy kola

Správanie bota je dané súčasnou fázou kola, v ktorom sa bot nachádza. Bot je inicializovaný do fázy útoku a do tejto fázy sa vracia na konci každého kola pred vydaním príkazu `EndTurnCommand`.

Útok

V prípade *úplného* kola, teda kola, v ktorom sa nepreskakujú fázy, sa najprv volá metóda `AI.get_attack()`. Táto metóda vyhľadáva útoky, ktoré:

- je možné vykonať bez podporných presunov,
- je možné vykonať s podpornými presunmi, ktoré:
  - podporujú útok,
  - podporujú obranu po útoku.

Vyhodnotenie rozhodujúce o optimálnosti útoku poskytuje v metóde `AI.get_attack()` trénovaná neurónová sieť (viac v sekcii 2.3). V prípade, že padne rozhodnutie o vykonaní útoku, pozbierajú sa prípadné podporné presuny, ktoré sa vykonajú v danom poradí (teda presuny podporujúce útok sa vykonajú pred útokom a presuny podporujúce obranu sa vykonajú po útoku). Počet týchto podporných presunov je obmedzený na štyri.

Obrana

Ďalšou fázou je fáza obrany. Vzhľadom na princíp hry je vhodné byť do istej miery agresívny. Avšak napriek tomu, že sa útoky vykonávajú iba v prípade vysokej udržateľnosti prípadného rozšíreného územia, je nutné ochrániť už vlastnené územie v reakcii na zmenu stavu hry zo strany nepriateľov. Metóda `AI.gen_deffense_moves()`[1] generuje obranné presuny s prioritou obrany najslabších území na hraniciach regiónov. Maximálny počet týchto presunov za kolo nie je obmezdený.

Migrácia

V prípade, že po obrannej fáze vystanú presuny pre dané kolo, vykonajú sa „migračné“ presuny. Význam týchto presunov je udržiavať čo najviac kociek v dosahu území, ktorých okolie je najslabšie. Toto sa vyhodnocuje na základe rozdielu súčtov priateľských a nepriateľských kociek v susedstve vyhodnocovaného územia na hranici regiónu. Kocky migrujú buď z území, ktoré sú čo najbližšie k hraniciam (v prípade, že vlastníme menej, než polovicu hracej plochy), prípadne z území, ktoré sú najďalej od hraníc (v prípade, že vlastníme vyše polovicu hracej plochy). Tieto presuny generuje metóda `AI.gen_transfer_moves()`.

2.2 Alfa-Beta algoritmus

Na začiatku každého kola sa spustí algoritmus Max^n s Alfa-Beta orezávaním[2]. Prehľadávanie sa mimo nášho ťahu vykonáva do hĺbky 3, čo činí jedno kolo pre každého hráča. Algoritmus sa spúšťa v metóde `AI.alpha_beta_entry()`, ktorá následne volá rekurzívnu verziu `AI.alpha_beta()`. Každý uzol stromu prehľadávania generuje tri podstromy podľa spôsobu vykonania súčasného kola. Spôsoby vykonania kola sú: úplné kolo, preskočenie fázy útoku a žiadna akcia. Pre každý spôsob vykonania kola sa rekurzívne volá Alfa-Beta algoritmus, ktorý vracia vyhodnotenie hracej plochy na základe počtu obsadených území jednotlivými hráčmi, teda maximalizovanou hodnotou je počet vlastnených území.

Simulácia jednotlivých kôl prebieha v metóde `AI.simulate_turn()`, ktorá vykonáva zjednodušenú simuláciu priebehu kola (vlastne sa jedná o prispôsobený kód z modulu `dicewars.server.game`). Po simulácii každého spôsobu vykonania kola sa rekurzívne volá Alfa-Beta na simulovanú hraciu plochu. Po návrate z rekurzie sa simulovaná hracia plocha vráti do pôvodného stavu metódou `AI.unsimulate_turn()`.

2.3 Neurónová sieť

Model neurónovej siete je založený na neurónovej sieti učiacej sa hrať hru Snake[3], ktorá bola prispôsobená účelom projektu. Jej cieľom je odhadovať, či je nejaké územie udržateľné naprieč jednou rotáciou kôl po ukončení kola bota - teda vracia rozhodnutie, či územie spolu s jeho kockami okamžite nestratíme.

Pred zistením udržateľnosti územia sa najprv útok odsimuluje ako úspešný[4], vykoná sa predikcia a následne je uložený aktuálny stav hry, ktorý sa neskôr použije na trénovanie neurónovej siete. Stav hry je definovaný ako počet kociek územia, ktorého udržateľnosť testujeme, následne počet kociek nanajvyš piatich najsilnejších susedných nepriateľských území, ktoré sú zoradené zostupne a nakoniec vlastníctvo územia, ktoré je síce v čase predikcie vždy rovnaké, ale môže sa zmeniť na základe akcií protivníkov. Okrem stavu hry pred akciou sa na trénovanie využívajú informácie o stave hry po vykonaní akcie, o danej akcii a odmene. Po vykonaní vybranej akcie sa buď pokračuje ďalej, alebo ukončuje ťah. Následne je na začiatku každého kola vykonávané trénovanie neurónovej siete. Pre trénovanie sa získa nový stav hry každej oblasti, pre ktorú sme vykonali predikciu v minulom ťahu a na základe korektnosti predikcie sa neurónovej sieti udelia odmeny (správna predikcia činí

[1] Slovo „defense“ je v zdrojovom kóde na niektorých miestach omylom písané s dvojitým „f“.
[2] KORF, Richard E. Multi-player alpha-beta pruning. *Artificial Intelligence*. Elsevier. 1991, zv. 48, č. 1, s. 99-111.
[3] Odkaz na github repozitár: https://github.com/python-engineer/snake-ai-pytorch.
[4] Stav pred útokom nemá vplyv na rozhodovanie neurónovej siete, čiže týmto sa zvýši počet trénovacích dát.

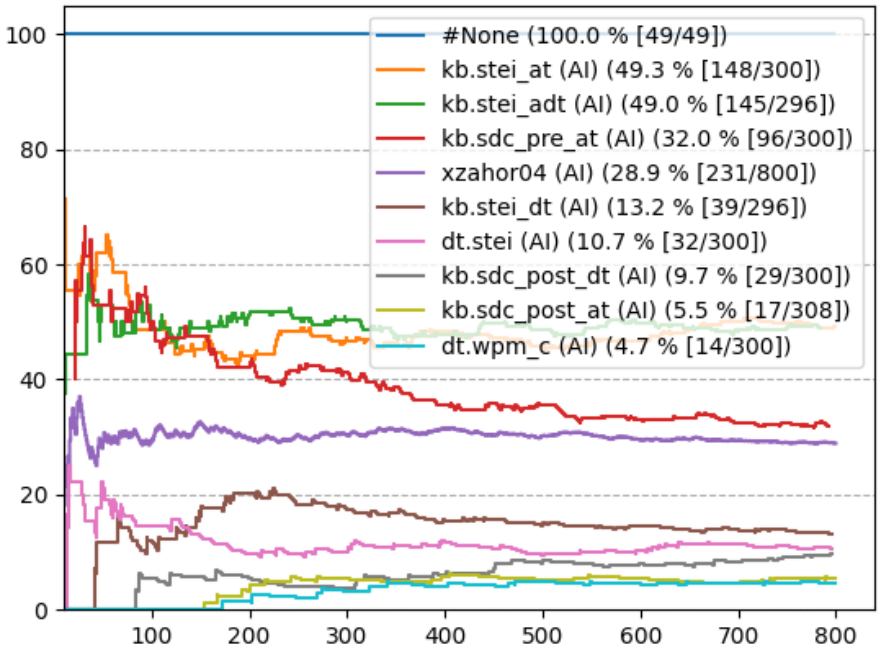
pozitívnu odmenu, nesprávna predikcia činí negatívnu odmenu). Keďže sa týchto akcii za ťah môže vykonať viac, získavame nový stav pre každú predikciu, uďelujeme odmeny a trénujeme sieť až do bodu, keď predikcie začnú byť vo väčšine prípadoch správne.

3 Priebežné vyhodnocovanie implementácie

Vyhodnocovanie prebiehalo so zvyšujúcimi sa nárokmi na úspešnosť voči vždy obtiažnejším súperom. Prvé iterácie umelej inteligencie boli testované proti botom `dt.ste`, `dt.stei` a `kb.sdc_post_dt`. Zpočiatku bolo cieľom poraziť najslabšieho bota z tých, proti ktorým bude vyhodnocovaný (podľa zadania). Zakaždým keď sa botovi podarilo výrazne zlepšiť, tak sa najslabší bot nahradil ďalším v poradí z testovacích botov. Výrazným prínosom bolo posilňovanie útokov a prípadné dodatočné posilnenie území po útoku pomocou presunov. Neurónová sieť prešla niekoľkými iteráciami funkcie generujúcej odmeny. Ako výsledná bola zvolená tá, ktorá robila najmenšie chyby v predikciách.

4 Záver

Bot má v súčasnej verzii vysokú úspešnosť voči `dt.stei` a `kb.stei_dt`. Na základe testovania s fixným poradím hráčov na 1000 hracích plochách je implementovaný bot na približne rovnakej úrovni ako `kb.sdc_pre_at`. Na grafe konvergencie víťazstiev (viď obrázok 1) je však od neho badateľne slabší. Predpoklad je, že pri štatisticky reprezentatívnejšom počte hier sa počty víťazstiev vyrovnajú.



Obr. 1: Graf konvergencie víťazstiev v hrách štyria proti štyrom na 200 rôznych hracích plochách.

4.1 Známe nedostatky a možné vylepšenia

V prvom rade je potrebné reorganizovať, refaktorizovať a zjednodušiť zdrojový kód. Alfa-Beta algoritmus nie je vylepšený, aby efektívnejšie prerezával podstromy. Vyhodnocovacia funkcia využívaná

pri prehľadávaní stavového priestoru by mala zohľadňovať aj defenzívnu silu regiónov jednotlivých hráčov.