Fakulta informačních technologií
Vysoké učení technické v Brně

SUI - Umělá inteligence a strojové učení

Umělá inteligence pro Válku kostek

2021

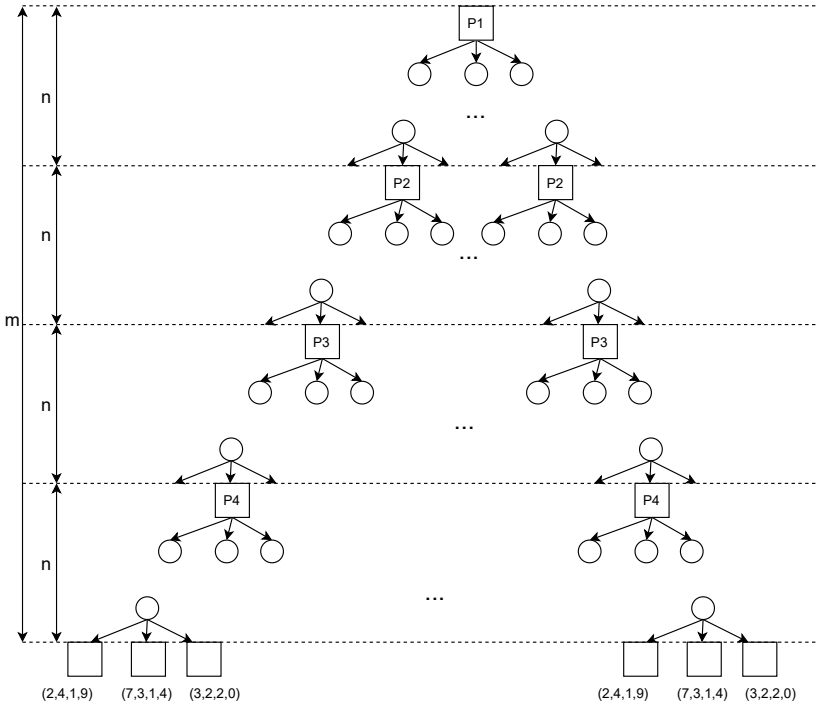
Doležel Marek (xdolez67)

Holub Juraj (xholub40)

Mateáš Branislav (xmatea00)

Rádsetoulal Vlastimil (xradse00)

Obsah



Obr. 1: Prehľadávanie stavového priestoru pomocou *Max-N*.

1 Úvod

2 Prehľadávanie stavového priestoru

Naša implementácia prehľadávania stavového priestoru je založená na algoritme *Max-N*;[1]. Algoritmus je vizualizovaný na obrázku 1. Hranaté uzly označujú prvý ťah každého hráča v aktuálnom kole (hráči P1, P2, P3 a P4). Aktuálny hráč na ťahu určí nasledujúce možné ťahy. Avšak v našej hre počas jedného kola môže hráč podľa pravidiel urobiť viacero ťahov. Preto sa každý hráčov ťah expanduje ďalej až do úrovne n, kde n je konfigurovateľné a obmedzuje rozsah prehľadávaných stavov. Kruhové uzly teda predstavujú jednotlivé ťahy v rámci jedného kola hráča. V úrovni n sa začína kolo pre nasledujúceho hráča.

V úrovni $4n$ (alebo tiež m) odohral každý hráč jedno kolo. V hĺbke m môžeme ohodnotiť listové uzly a ohodnotenie propagovať vyššie. Ak chceme prehľadávať do väčšej hĺbky tak vykonáme ďalším $4n$ expandovaní a prehľadávanie vyhodnocujeme vždy v hĺbke ktorá je násobkom m. Parameter m je teda ďalší konfigurovateľný atribút, ktorým obmedzujeme rozsah prehľadávania.

Uzly na listovej úrovni sú ohodnotené a výsledkom je štvorica (p1, p2, p3, p4), kde p1 až p4 je skóre hráča P1 až P4 v danom stave hry. Uzol o jednu úroveň vyššie zvolí za najlepšieho potomka ten uzol v ktorom má ohodnotenie aktuálne hrajúceho hráča najvyššiu hodnotu (algoritmus *Max-N*).

3 Implementácia prehľadávania

Implementacia bola prevzata z clanku [1].

3.1 Expandovanie stavu

Expandovanie uzlu hráča vykoná funkcia `possible_moves(board, player)`, ktorá pre aktuálny stav hracej dosky (`board`) a hráča (`player`) určí nasledujúce možné ťahy. Možné ťahy sa počítajú podobne ako v botovi

[1] `https://docplayer.net/131108557-Expectimax-n-a-modification-of-expectimax-algorithm-to-solve-multiplayer-stochast html`

STEI. Množina všetkých možných ťahov je ohraničená len na také, ktoré spĺňajú aspoň jedno z nasledujúcich kritérií:

- Pravdepodobnosť úspešného útoku na oblasť a súčasne udržanie tejto oblasti v nasledujúcom ťahu je väčšia alebo rovná $\frac{1}{2}$.
- Hráč má v útočiacom poli plný počet kociek (8).

3.2 Simulácia ťahu hráča

Každý expandovaný ťah je simulovaný funkciou `apply_move_to_board(board, action)`. Simulácia zoberie aktuálny stav hracej dosky (`board`), navrhovaný ťah (`action`) a vytvorí nový stav hracej dosky. Útočný ťah je v tejto hre stochastický avšak simulácia je deterministická a uvažuje, že útočník zvíťazí vždy ak má väčšie množstvo kociek ako obranca.

3.3 Ohodnotenie listového uzlu

Na ohodnotenie listového uzlu používame metriku, ktorá zohľadňuje najväčší región vlastnený každým hráčom. Región je súvislá časť polí na hracej doske, ktorá patrí jedinému hráčovi. V danom listovom uzle sa nájde najväčší región pre každého hráča a počet oblastí v tomto regióne je ohodnotením skóre pre daného hráča.

3.4 Konfigurácia parametrov prehľadávania

Parameter n (hĺbka prehľadávania ťahov v rámci jedného hráča) je nastavená na 1 pretože väčšia hĺbka znižovala pomer výhier nášho AI.

Parameter m (hĺbka prehľadávania v rámci kôl) je nastavená taktiež na 1. Prehľadávame teda stavový priestor v ktorom každý hráč odohrá jednu sériu svojich ťahov.

4 Stratégia

Naše AI funguje podľa stratégie, ktorá je podobná STEI_ADT. Najsilnejší protihráč je práve bot STEI_ADT. Preto je naša stratégia podobná jeho. Avšak naša AI ju vylepšuje o prehľadávanie stavového priestoru. Prehľadávanie by malo byť efektívne práve preto, že poznáme implementáciu súperovho bota. Naša simulácia súperovho útočného ťahu teda vychádza zo stratégie STEI, ktorú súper používa. Preto predpokladáme, že vieme pomerne dobre predvídať súperove ťahy. Na druhej strane, hra je pomerne silne stochastická čo simuláciu značne znepresnuje.

Voľba ťahu v každom kole, pričom v jednom kole má hráč viac ťahov, postupuje podľa nasledujúcej stratégie:

1. Prvé 4 ťahy AI presunie svoje kocky z vnútra poľa na hranice zo súpermi tak aby malo čo najväčšiu útočnú silu (ofenzíva podobná STEI_ADT).
2. AI vykoná prehľadávanie stavového priestoru pomocou *Max-N* a zvolí útočný ťah s najlepším ohodnotením. Tento krok opakuje dokedy mu *Max-N* poskytuje možné útoky. (STEI_ADT v tejto fáze neprehľadáva stavový priestor ale používa heuristickú funkciu)
3. Na záver vykoná AI defenzívny presun. Zvyšné 2 ťahy na presun vlastných kociek využije tak aby svoje kocky presunula od hraníc (defenzíva podobná STEI_ADT).

5 Umělá inteligence

Neuronová síť implementuje heuristickou funkci, která se poté využívá k ohodnocení listových uzlů v rámci procedury MaxN. Vstupem neuronové sítě je vektor reprezentující aktuální stav hry, výstupem je potom vektor čtyř reálných čísel z intervalu $\langle 0, 1 \rangle$, které udává pravděpodobnost výhry daného hráče v aktuální situaci.

5.1 Serializace hry

Pro účely serializace byla upravena serverová část hry, a to konkrétně soubor `dicewars/server/game.py`. Tato drobná úprava umožňuje uložit všechny stavy her(pole objektů `board`), které nastávají v rámci jednoho turnaje do souboru `.pickle`.

Samotná serializace a transformace do HDF[2] souboru je implementována skriptem `process_pickle_to_np_array.py` a probíhá offline (tedy mimo běhové prostředí hry).

Serializovaná hra je reprezentována vektorem čísel o délce V, kde celková délka vektoru se vypočte v závislosti na počtu políček následovně: $\frac{bSize}{2} \cdot (1 + bSize) + 2 \cdot bSize + 1$. Tento vektor se skládá z celkem tří komponent:

- matice sousednosti $\boldsymbol{M}_n$ - určuje, zda políčko i sousedí s políčkem j. Pokud $M_{i,j}$ je rovno 1, pak políčko i sousedí s j. V případě, že $M_{i,j}$ je rovno nule, tak nikoliv. Protože je matice symetrická, ukládá se pouze horní trojúhelník (včetně diagonály). Velikost matice sousednosti je určena vztahem: $\frac{bSize}{2} \cdot (1 + bSize)$.
- vektor vlastnictví $\vec{\boldsymbol{o}}$ - reprezentuje informaci o vlastníkovi políčka, vlastník je reprezentován hodnotou z intervalu $\langle 1; 4 \rangle$. Velikost vektoru $\vec{\boldsymbol{o}}$ je $bSize$.
- vektor kostek $\vec{\boldsymbol{d}}$ - počet kostek na daném políčku (nabývá hodnot z intervalu $\langle 1; 8 \rangle$). Velikost vektoru $\vec{\boldsymbol{d}}$ je $bSize$.
- vítěz v rámci současného turnaje - velikost 1 políčko.

5.2 Neuronová síť

Generování surových dat probíhalo pomocí následujícího příkazu `./scripts/dicewars-tournament.py -g 4 -n 1000 --ai-under-test kb.xdolez67` s upraveným serverem (viz předchozí sekce). Tato data byla následně zpracována pomocí skriptu `process_pickle_to_np_array.py`, který vytvoří soubor HD5 s numpy polem dimenze (N, 664), kde N je celkový počet stavů her a 664 je velikost jednoho vektoru.

Tento soubor je dále zpracován tak, aby zastoupení výher všech hráču bylo uniformní a zároveň se sníží velikost v první dimenzi a provede se operace shuffle podle první dimenze. Toto zpracování provádí skript `extract_subset_fromH5.py`

Takto zpracovaný soubor je poté vstupem pro skript `./scripts/network.py`, který data rozdělí na trénovací, testovací a validační a na těchto datech natrénuje neuronovou síť. Architektura neuronové sítě byla zvolena na základě výsledků experimentů.

6 Záver

Literatúra

[1] Gunawan. Expectimax-n: A modification of expectimax algorithm to solve multiplayer stochastic game. *SIAM Journal on Discrete Mathematics*, 22(1):124–138, 2008.

[2] HDF - Hierarchical Data Format