

Trzecie laboratorium z przedmiotu Wprowadzenie do sztucznej inteligencji



Autor

Łukasz Jaremek
310710

Data

listopad 2021

Praca wykonana samodzielnie.

Użyte technologie

Język:

Python 3.8.8

Biblioteki ponad standardowe:

termcolor 1.1.0

cProfile

Dwuosobowe gry deterministyczne

Problem:

Napisanie programu, który będzie budował drzewo gry Isolation, aby zwiększyć swoje szanse na wygraną.

Działanie algorytmu:

Algorytm sprawdza, jakie ruchy gracz może wykonać stojąc w danym miejscu. Następnie zapisuje statusy gier z wykonanymi ruchami, i dla każdego ponownie sprawdza dostępne opcje, i ponownie zapisuje wszystkie możliwe wykonania ... aż liczba sprawdzanych w przód ruchów będzie zgodna z podaną liczbą ruchów na początku algorytmu.

Następnie, gdy drzewo gry jest już zbudowane, algorytm ocenia które ruchy początkowe są dla gracza najkorzystniejsze (jaka kombinacja ruchów, najbardziej ograniczy pole manewru przeciwnika). Im przeciwnik ma mniej wolnych pól dookoła siebie, tym lepiej, dlatego funkcja oceniająca wartość ruchu, im mniejszą wartość ruchu zwróci, tym korzystniejszy dla gracza jest ruch. Gdy już znajdzie najlepszy ruch początkowy, wykonuje go.

Drzewo vs Ruchy losowe

Sprawdźmy, czy rzeczywiście metoda z drzewem jest lepsza, od pseudolosowych ruchów. Gracz B, będzie dokonywał wyboru ruchu na podstawie wyniku drzewa, natomiast gracz A będzie wykonywał pseudolosowy, możliwy ruch.

Parametry:

- Rozmiar mapy: 4x4
- Wysokość drzewa: 5
- Liczba rozgrywek: 50

Statystyki:

Liczba wygranych pierwszego gracza	15
Liczba wygranych drugiego gracza	35
Średnia liczba ruchów w grze	10.32

Jak widać, gracz używający drzewa ma wyraźną przewagę. Sprawdźmy to jeszcze na większej mapie o rozmiarach 7x7:

Liczba wygranych pierwszego gracza	11
Liczba wygranych drugiego gracza	39
Średnia liczba ruchów w grze	45.88

Drzewo vs Drzewo

Sprawdźmy, czy algorytm jest sprawiedliwy, to znaczy czy każdy z graczy ma takie same szanse na wygraną. Gracze A i B będą podejmowali decyzję na podstawie drzewa.

Parametry:

- Rozmiar mapy: 4x4
- Wysokość drzewa: 5
- Liczba rozgrywek: 50

Statystyki:

Najkrótszy czas ruchu	0.26 s
Średni czas ruchu	1.05 s
Najdłuższy czas ruchu	2.89 s
Liczba wygranych pierwszego gracza	27
Liczba wygranych drugiego gracza	23
Średnia liczba ruchów w grze	10.24

Jak widzimy, szanse na wygraną są sobie równe. Większa liczba rozgrywek by wyrównywała wynik jeszcze bardziej. Lecz dlaczego nie zwiększyć liczby iteracji? Spójrzmy na tabelkę powyżej z czasami, średni czas ruchu to ponad sekunda. Na rundę na mapie 5x5 zostaje wykonanych ~10 ruchów.

Co przy 50 rozgrywkach daje $1 \cdot 50 \cdot 10 = 500$ sekund czyli ponad 8 minut.

Czas jak widzimy jest spory, a mapa mała. Sprawdźmy jakie będą czasy dla wysokości drzewa = 6:

Statystyki:

Najkrótszy czas ruchu	2.23 s
Średni czas ruchu	10.73 s
Najdłuższy czas ruchu	517.75 s
Liczba wygranych pierwszego gracza	31
Liczba wygranych drugiego gracza	19
Średnia liczba ruchów w grze	10.72

Wnioski:

Czas drastycznie się wydłużył oraz widać, że gracz wykonujący pierwszy ruch ma przewagę – zapewne wynika to z wielkości mapy, która jest mała.

Warto sobie zadać pytanie skąd się biorą tak długie czasy, otóż gracz zaczynając grę może się ruszyć w 8 stron, każdy kolejny ruch to 7 bądź mniej możliwości ruchu.

Przy brzegowych sytuacjach i 5 ruchach mamy:

- negatywnie: $8 \cdot 7 \cdot 7 \cdot 7 \cdot 7 = 19\,208$ kombinacji
- pozytywnie: $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 6\,720$ kombinacji
- średnio: 12_964 kombinacji

Są to ilości przypadków, które musi sprawdzić gra przy jednym ruchu dla jednego gracza, liczby robią wrażenie więc średni czas wynoszący sekundę nie wydaje się taki zły.

Aby sprawdzić czy faktycznie gracz rozpoczynający ma przewagę tylko ze względu na małą mapę, przeprowadźmy gry z parametrami:

- Rozmiar mapy: 7x7
- Wysokość drzewa: 5
- Liczba rozgrywek: 50

Statystyki:

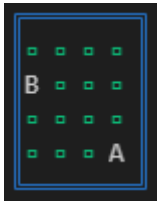
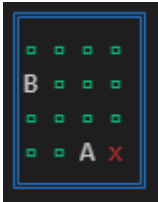
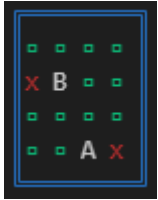


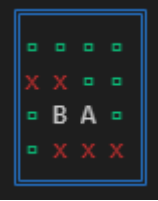
Najkrótszy czas ruchu	0.62s
Średni czas ruchu	3.17s
Najdłuższy czas ruchu	8.92
Liczba wygranych pierwszego gracza	21
Liczba wygranych drugiego gracza	29
Średnia liczba ruchów w grze	26.42

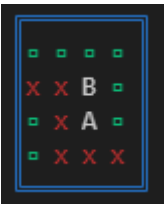

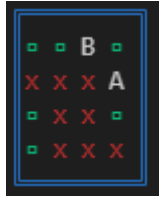
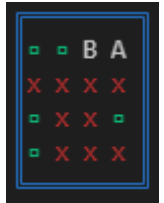
Jak widać, stosunek wygranych delikatnie się zmienił na korzyść drugiego gracza, więc można uznać grę za sprawiedliwą.

Analiza ścieżki wybranej przez drzewo

Prześledźmy ścieżkę w grze o parametrach:

- Rozmiar mapy: 4x4
- Wysokość drzewa: 5
- gracze A i B podejmują decyzję na podstawie drzewa

Który gracz wykonał ruch	Wartość funkcji min-max	Status planszy
Start gry	Start gry	
A	0	
B	0	
A	0	
B	0	
A	0	

B	0	
A	4096	
B	0	
A	4096	

Jak widzimy, przez dużą część pierwszych ruchów funkcja min-max zwraca 0, dlatego, że gracze są tak blisko siebie, że zawsze w kilku ruchach sytuacja przeciwnika się pogorszy.

Jednak w trzech ostatnich ruchach gracz A osiąga bardzo wysokie wyniki (więc niekorzystne) natomiast gracz B cały czas otrzymuje 0. Wynika to z faktu, że gra potoczyła się w taki sposób, że gracz A nie miał szans wygrać, więc nie mógł osiągnąć korzystnych wyników, co za tym idzie wykonywał ruchy, które pozwalały mu już tylko dłużej przetrwać (w przypadku gry z graczem, który poruszałby się w sposób losowy, mógłby liczyć na przypadkowy ruch przeciwnika, który pozbawi go przewagi).

Podsumowanie laboratorium

Metoda z wykorzystaniem drzewa może przynieść korzystne wyniki, lecz im więcej możliwości ma gracz, tym więcej czasu kosztuje znalezienie lepszego ruchu. Więc czy warto go używać? Zależy od gry i mocy obliczeniowej komputera, z pewnością obcinanie niekorzystnych gałęzi by przyspieszyło działanie algorytmu. Jego działanie również dobrze się sprawuje w przypadku, gdy gracz może poruszać się w 4 kierunkach (nie tak jak tutaj w 8), ponieważ wielokrotnie spada liczba ścieżek, które należy sprawdzić.