WSI Ćwiczenie 3
Algorytm Min Max

Autor:
Jakub Kowieski

Przedmiot:
WSI, semestr: 21Z

Numer Albumu:
310765

Celem ćwiczenia jest implementacja algorytmu MiniMax z obcinaniem α − β. Następnie należy zbadać jego działanie dla różnych rozmiarów planszy oraz stanów gry „Szewc”.

Stałe:

RANGE = 10 - liczba przeprowadzonych gier dla każdego parametru
MAX_RATING = 1000 - wartość oceny ruchu jeśli gracz nim wygrywa
params - lista parametrów zdefiniowanych (znajduję się w ./utils/params.py): (głębokość alg. dla gracza 1, głębokość alg. dla gracza 2, wielkość planszy)

Założenia:

1. **Player 1** zaczyna pierwszy
2. Jeżeli gracz wygrywa, ocena tego ruchu wynosi zawsze 1000 lub -1000 (zależne czy wygrywa 1 gracz czy 2)
3. Jeżeli są co najmniej dwa ruchy z taką samą oceną wtedy jest wybierany jeden losowy z nich
4. Jeżeli depth = 0, to znaczy że gracz wykonuje losowe ruchy
5. Gdy gracz wygrywa zyskuje 1 punkt, gdy remis 0.5 punkta dla obu
6. Winrate jest to łączna liczba punktów zebrana podzielona przez liczbę gier

Posiadanie modułów:

- numpy
- https://github.com/lychanl/two-player-games

Testowanie:

python3 table.py - wykonuje eksperymenty i tworzy plik csv w folderze ./csv z wynikami eksperymentów dla zadanych parametrów
python3 simulation.py - symulacja eksperymentu dla jednego parametru z wyświetlaniem informacji o grze
python3 test_first_player.py - służy do debugowania czy algorytm dobrze wylicza i obcina gałęzie dla gracza max (pierwszego)
python3 test_second_player.py - służy do debugowania czy algorytm dobrze wylicza i obcina gałęzie dla gracza min (drugiego)
python3 play.py - przykładowa gra użytkownika kontra algorytm

Legenda:

(głębokość alg. dla gracza 1, głębokość alg. dla gracza 2, wielkość planszy) - taki zapis to odwołanie się do eksperymentu dla zadanych parametrów

Eksperymenty:

Player 1: Depth	Wins	Winrate	Player 2: Depth	Wins	Winrate	Board size	Average Time
0	0	0.0	3	10	100.0	5	4.17
3	10	100.0	0	0	0.0	5	4.02
1	0	0.0	5	10	100.0	3	3.25
6	9	90.0	2	1	10.0	3	13.82
2	10	100.0	1	0	0.0	10	10.79
3	4.5	45.0	3	5.5	55.0	4	1.49
0	0	0.0	1	10	100.0	4	0.01
1	10	100.0	0	0	0.0	10	1.07
100	10	100.0	1	0	0.0	2	9.11
4	7.5	75.0	8	2.5	25.0	2	1.89
100	0	0.0	0	10	100.0	1	0.0

4	4	40.0	4	6	60.0	3	0.64	

Wnioski:

Algorytm Min Max działa bezbłędnie z grą przeciwko graczu który wykonuje losowe ruchy. Nie ważne jest jak mała czy duża jest plansza np.(0, 3, 5),(1, 0, 10) algorytm wygrywa zawsze gdy board_size > 1.

Przy podobnych i małych głębokościach (3, 3, 4) powstają remisy jest to spowodowane, że gracze są tak samo dobrzy więc gra kończy się remisem. Wielkość board size wpływa na wynik. Gdy plansza jest bardzo duża (2, 1, 10) to niby mała różnica głębokości pozwala na wykonywanie lepszych ruchów w czasie rozgrywki co skutkuje wygranymi.

Jeżeli board_size = 1 to nie liczy się które krawędzie gracze wybiorą tylko kto pierwszy zacznie grę, dlatego przy parametrach (100, 0, 1) zawsze wygrywa gracz drugi, który wybiera losowe krawędzie.

Dla mocno różniących się głębokości sytuacja jest różna. Jeśli różnica jest znacznie większa (1, 5, 3) wtedy znacznie częściej wygrywa gracz z większą głębokością. Jednakże może się zdarzyć tak jak (4, 8, 2) że różnica jest duża, ale głębokość nie ma aż takiego znaczenia, gdyż obie są wystarczające duże by zobaczyć optymalny ruch, jednakże w tym przypadku pierwszy ruch ma większe znaczenie (mały rozmiar planszy board_size = 2) przez co zazwyczaj wygrywa pierwszy gracz.

Jeżeli głębokość danego gracza jest większa niż zbiór wszystkich możliwych ruchów to gracz od razu wie czy wygra, przegra lub zremisuje i będzie robił najlepsze możliwe ruchy, jak w eksperymencie (100, 1, 2) gdzie pierwszy gracz zawsze wygrywa i od początku rating wynosi 1000 (można sprawdzić za pomocą komendy *python3 simulation.py*).

Największa różnica wyników jest w skoku z robienia losowych ruchów do depth = 1, możliwość zauważenie krawędzi zamykającą kwadrat daje ogromną przewagę dlatego gracz wybierający losowo ruchy ma znikome szanse na wygraną.

Podsumowanie:

Algorytm Min Max działa bardzo dobrze i daje nam o wiele większą szansę na wygraną niż robienie losowych ruchów. Na wynik wpływa jednak dużo czynników głębokości algorytmu, kto pierwszy zaczyna i rozmiar planszy. Jednak na wybór jak najlepszych ruchów znaczenie ma tylko głębokość. Niestety im większa głębokość tym czas zwiększa się wykładniczo co ogranicza jej wartość. Jednakże dla mniejszych głębokości algorytm działa sprawnie nawet dla dużych rozmiarów planszy np.(2, 0, 15). Kto pierwszy zaczyna ma znaczenie szczególnie na małych planszach (dla np. board_size = 1 wiadomo że drugi gracz zawsze wygra), a dla dużych daje dużą przewagę większym głębokościom (2, 1, 10).