WSI – ćwiczenie 3

Algorytm minimax z obcinaniem α-β

Adam Wróblewski

Wstęp:

Ćwiczenie polegało na implementacji algorytmu minimax z obcinaniem α-β, a następnie sprawdzeniu jego działa przy użyciu gry *Pick34*.

Cel eksperymentów:

porównanie jakości działania algorytmu dla różnych głębokości przeszukiwania dla gry *Pick34*.

Założenia, decyzje początkowe:

Zgodnie z sugestią skorzystałem i zmodyfikowałem implementację gry *Pick* dostępną w repozytorium: *https://github.com/lychanl/two-player-games*

Modyfikacji uległa klasa *PickState* w pliku *Pick.py*, gdzie dodałem funkcję heurystyczną *heuristic()* oceniającą dany stan gry. Funkcja ta posługuje się pewnym kwadratem magicznym do zmapowania liczb na odpowiadające im wartości oznaczające liczbę kombinacji w których dana liczba bierze udział. Następnie obliczana jest suma tych wartości dla wszystkich liczb które zostały przez jednego jak i drugiego gracza i jako wartość heurystyki zwracana jest ich różnica. Idea jest analogiczna do tej przedstawionej na wykładzie dla gry kółko-krzyżyk.

Implementacja algorytmu minimax z obcinaniem α-β znajduje się w pliku *alpha_beta_algorithm.py*.

Funkcja służąca do symulacji przebiegu gry z 2 komputerami, o zadanych głębokościach przeszukiwania znajduje się w pliku *simulate_game.py*.

Element losowości w algorytmie minimax o którym mowa w treści ćwiczenia polega na zebraniu wszystkich ruchów z taką samą oceną a następnie losowym wyborze jednego z nich.

Eksperymenty i porównanie jakości działania algorytmu dla różnych głębokości przeszukiwania:

Przeprowadziłem szereg eksperymentów porównując różne głębokości przeszukiwania. Dla każdej głębokości wykonałem 30 testów i zliczałem w ilu przypadkach nastąpiła wygrana 1 gracza, ile 2 gracza i ile razy wystąpił remis.

Głębokości przeszukiwania będę oznaczał następująco: (2,3) – oznacza głębokość przeszukiwania równą 2 dla gracza 1 i głębokość przeszukiwania równą 3 dla gracza 2

Dla głębokości przeszukiwania (1,1):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
18	12	0

Dla głębokości przeszukiwania (1,1) wyniki są dość zbliżone, gracz 1 wygrywa nieco częściej, co może być spowodowane faktem że pierwszy rozpoczyna rozgrywkę.

Zadane głębokości przeszukiwania są zbyt małe aby miały wpływ na wynik gry – algorytm przewidując o krok nie jest w stanie wyznaczyć optymalnego ruchu zapewniającego przewagę nad drugim graczem.

Dla głębokości przeszukiwania (1,2):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
2	22	6

Zwiększenie głębokości przeszukiwania dla gracza nr.2 powoduje częstszą wygraną nad graczem nr.1. Jest to zgodne z oczekiwaniami, gdyż algorytm minima dla gracza nr.2 analizuje więcej ruchów „do przodu”.

Dla głębokości przeszukiwania (1,3):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
1	23	6

Zwiększając różnice w głębokości przeszukiwania między graczami powodujemy liczba wygranych dla gracza o większej głębokości nieznacznie się zwiększa, jednak może być to także wpływ losowości.

Dla głębokości przeszukiwania (1,4):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
1	25	4

Zwiększając różnice w głębokości przeszukiwania między graczami jeszcze bardziej nie zauważam znaczących różnic w stosunku wygranych.

Dla głębokości przeszukiwania (2,1):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
30	0	0

Gdy głębokość przeszukiwania dla gracza nr.1 będzie większa niż dla gracza nr.2 obserwujemy pełną dominację w liczbie wygranych gracza nr.1 nad graczem nr.2. Nie otrzymujemy także żadnych remisów.

Przyczyną takiego zjawiska jest fakt że gracz pierwszy ma większą głębokość przeszukiwania, przez co analizuje stany gry o jeden ruch do przodu więcej w porównaniu do gracza nr.2, a ponadto gracz nr.1 pierwszy rozpoczyna rozgrywkę, dlatego już na starcie ma większe szanse na wygraną.

Dla głębokości przeszukiwania (3,1):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
30	0	0

Dla głębokości przeszukiwania (4,1):

Wygrana 1 gracza	Wygrana 2 gracza	Remis
30	0	0

Dla głębokości (3,1) oraz (4,1) obserwujemy analogiczną sytuację jak w przypadku głębokości (2,1).

Wnioski ogólne:

Działanie algorytmu minimax z odcinaniem alfa-beta w dużej mierze zależy od funkcji heurystycznej – to ona determinuje jak oceniamy dany stan gry. Jeśli funkcja heurystyczna będzie niepoprawnie zaimplementowana, algorytm nie ma szans na dobre oszacowanie stanu gry, a także na wybór najlepszego możliwego ruchu spośród wszystkich analizowanych stanów gry.

Wpływ na działanie algorytmu ma także głębokość przeszukiwania, im jest ona większa tym analizuje on więcej ruchów „do przodu” i przy założeniu że oboje gracze wykonują optymalne ruchy wybiera lepszy ruch - taki który bardziej opłaci się w przyszłości, a tym samym doprowadzi do zwycięstwa. W rezultacie, jeśli mamy do czynienia z graczami o różnych wartościach głębokości przeszukiwania, szansa na wygranie tego większej wartości drastycznie rośnie. Na wynik gry ma także niewielki wpływ to który gracz rozpoczyna rozgrywkę – ma on wówczas nieco podniesione szanse na wygraną, co pokazują przeprowadzone eksperymenty.

Jeszcze jednym aspektem zwiększenia głębokości przeszukiwania jest zwiększenie czasu potrzebnego na wykonanie algorytmu, wynika to z faktu, że wraz ze wzrostem głębokości zwiększa się całkowita liczba stanów gry która zostaje poddana analizie. Dobrym aspektem algorytmu minimax jest element odcinania gałęzi, sprawia on że część gałęzi nie przeglądamy, a tym samym czas wykonywania algorytmu przy dużych głębokościach nie jest bardzo długi.