WSI – ćwiczenie 3 Algorytm minimax z obcinaniem α-β Maciej Łodziński

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

Zmodyfikowałem klasę PickState poprzez dodanie do niej kilku metod odpowiadających za wyliczanie funkcji heurystycznej, a także atrybutów magic_square oraz heurestic_square po to aby nie wyliczać tych macierzy więcej niż raz i zmniejszyć czas wykonywania programu. Funkcja heurystyczna szuka najpierw indeksów elementów macierzy magic_square, które znajdują się w tablicy liczb wybranych przez graczy. Następnie za pomocą owych indeksów pobieram elementy macierzy heurestic_square, której to elementy oznaczają sumę kolumn, wierszy i przykątnych przechodzących przez ten element np.

magic_square 4x4

4	14	15	1
9	7	6	12
5	11	10	8
16	2	3	13

heurestic_square 4x4

3	2	2	***3***
2	***3***	***3***	2
2	***3***	***3***	2
3	2	2	***3***

current_player_numbers = [10, 4, 1]

other_player_numbers = [7, 2, 15]

current_player_indices = magic_square.index(current_player_numbers) # [(2, 2), (0, 0), (0, 3)]

other_player_indices = magic_square.index(other_player_numbers) # [(1, 1), (3, 1), (0, 2)]

current_player_sum = sum(heurestic_square[current_player_indices]) # 3+3+3 = 9

other_player_sum = sum(heuristic_square[other_player_indices]) # 3+2+2 = 7

state_evaluation = current_player_sum - other_player_sum

Element losowości w algorytmie minimax o którym mowa w treści ćwiczenia polega na przetasowaniu listy dostępnych ruchów.

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 depth = [1,1]

Wygrana 1	Wygrana 2	Remis
24	6	0

Dla tej głębokości przeszukiwania gracz pierwszy zdecydowanie dominuje, przez fakt, że zaczynał.

Dla depth = [1, 2]

Wygrana 1	Wygrana 2	Remis
13	17	0

Zwiększenie głębokości przeszukiwania dla gracza nr.2 powoduje częstszą wygraną nad graczem nr.1 Jednak fakt, że ruszał się jako drugi nadal przeważa.

Dla depth = [1, 3]

Wygrana 1	Wygrana 2	Remis
4	26	0

Tu już widać przewagę pierwszeństwa ruchu nad głębokością przeszukiwania.

Dla depth = [2, 1]

Wygrana 1	Wygrana 2	Remis
29	1	0

Z kolei niewielka przewaga w głębokości w połączniu z pierwszeństwem ruchu jest miażdżąca.

Dla depth = [3, 1]

Wygrana 1	Wygrana 2	Remis
29	1	0

Podobnie jak wyżej

Dla depth = [2, 2]

Wygrana 1	Wygrana 2	Remis
24	4	2

Podobnie jak dla [1, 1]

Dla depth = [3, 3]

Wygrana 1	Wygrana 2	Remis
25	5	0

Podobnie jak dla [2, 2]

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 rośnie. Jednakże w omawianej grze znaczącym czynnikiem jest to, który gracz zaczyna grę – nawet przy większej głębokości gracza drugiego, ten pierwszy może często wygrywać. Oczywiście ten efekt słabnie wraz z różnica między głebokościami graczy. Jeśli chodzi o czas wykonywania funkcji minimax z odcinaniem α-β to rośnie on wykładniczo wraz ze wzrostem głębokości.