Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych

Wstęp do Sztucznej Inteligencji

ćwiczenie 3 - Dwuosobowe gry deterministyczne

Autor:

Jarosław Jaworski 342189

Data oddania ćwiczenia:

26.04.2025

SPIS TREŚCI

1 Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z problemem gier dwuosobowych deterministycznych o sumie całkowitej zero oraz zaprogramowanie algorytmu MiniMax z obcinaniem $\alpha - \beta$. W ramach ćwiczenia należało przeprowadzić odpowiednie testy funkcjonalności kodu oraz przeanalizować wyniki rozgrywek między graczami komputerowymi.

2 Implementacja algorytmu, porównanie rozwiązań

Ćwiczenie rozpoczęto od implementacji podstawowego algorytmu MiniMax, następnie ulepszenia poprzez dodanie obcinania $\alpha - \beta$. Obie wersje porównano aby ustalić różnice pomiędzy wydajnością rozwiązań. Przy pomocy biblioteki *pytest-benchmark*, zastosowano w tym celu funkcję *benchmark.pedantic* aby sprawdzić czas działania funkcji minimax dla stanu początkowego gry (initial) i stanu w środku gry (midgame). Dla średnich czasów egzekucji, wskazanych przez parametr końcowy *Mean* obliczono średnią arytmetyczną. Wyniki zamieszczono w Tabeli 1.

Tabela 1. Wyniki pomiarów dla MiniMax bez oraz z obcinaniem $\alpha - \beta$

Obcinanie $\alpha - \beta$	**Stan gry**	**Średni czas egzekucji [ms]**
Tak	Initial	0,325
Tak	Midgame	0,33
Nie	Initial	7,193
Nie	Midgame	5,53

3 Porównanie jakości dla gry z głębokością N przeciwko M

Następnym elementem zadania było porównanie jak radzi sobie gracz grający z głębokością N przeciwko graczowi grającemu z głębokością M. W celu zmierzenia jakości działania algorytmu i wpływu na efekt końcowy (zwycięstwo/porażka) wprowadzono miarę *win_ratio* dla gracza N (procentowy wskaźnik wygranych do rozegranych gier). Sposób obliczenia przedstawia wzór 1.

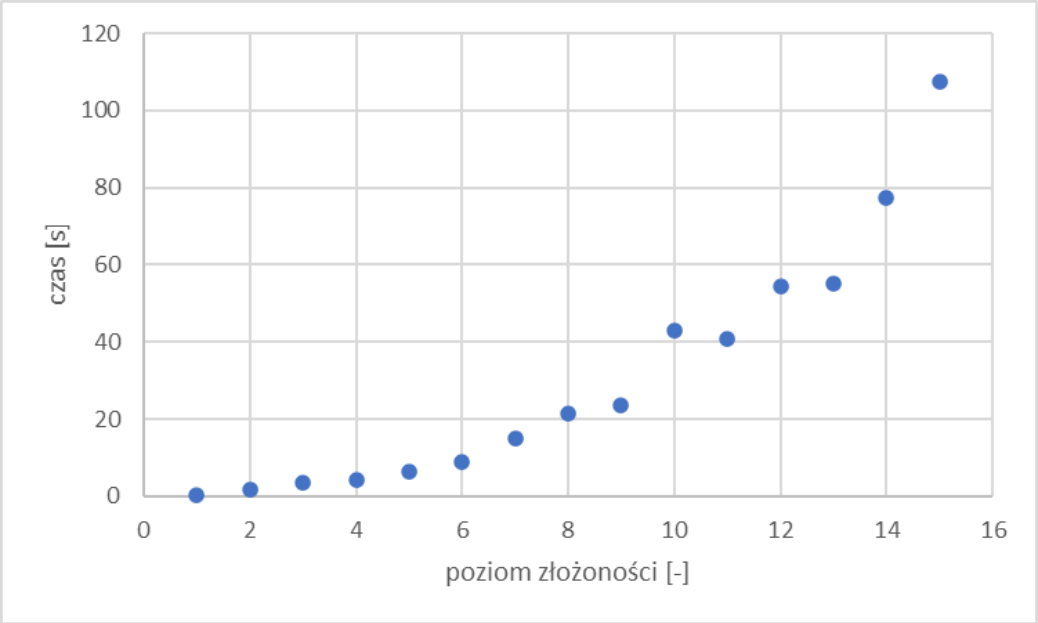
$$win\_ratio = \frac{gry\_wygrane}{gry\_rozegrane} \tag{1}$$

Gry rozgrywano dla głębokości analizy N oraz M o wartościach między 1 a 5. Analizowano próbkę stu rozegranych gier, mierząc przy tym czas egzekucji programu z wykorzystaniem biblioteki *time*. Wyniki pomiarów przedstawiono w Tabeli 2. W tej części ćwiczenia korzystano wyłącznie z końcowej wersji MiniMax z obcinaniem $\alpha - \beta$.

Tabela 2. Wyniki porównania dla graczy, grających z głębokościami N oraz M

N	M	win_ratio [%]	czas symulacji gier [s]
1	1	57	0.355
2	1	89	1.542
2	2	55	3.617
3	1	92	4.233
3	2	66	6.477
3	3	60	8.704
4	1	95	14.795
4	2	76	21.555
4	3	79	23.468
4	4	49	42.979
5	1	98	40.674
5	2	80	54.472
5	3	77	55.234
5	4	45	77.516
5	5	61	107.490

Na podstawie wyników utworzono graficzną wizulazację czasu symulacji w zależności od umownej jednostki poziomu złożoności obliczeniowej. Przedstawia to Rys. 1.



Rys. 1. Wykres zależności czasu wykonywania programu (symulacji stu gier) w zależności od rosnącego poziomu złożoności obliczeniowej

4 Wnioski

Przy stosowaniu algorytmu minimax dla gier dwuosobowych istotnym jest korzystanie jedynie z wersji z obcinaniem $\alpha - \beta$. Wybór ten poparty jest znaczącym zmniejszeniem czasu wykonania (tu: aż około 10-krotnym!). Obcinanie to jest kluczowe, ponieważ pozwala nie sprawdzać głebiej ruchów, które na pewno nie są optymalne. Działa to podobnie jak filtr pasmowoprzepustowy, który nie wpuszcza ruchów gorszych lub lepszych niż pewien poziom odpowiednio dla gracza maksymalizującego lub minimalizującego.

Niestety, z punktu widzenia złożoności obliczeniowej algorytm minimax nie jest rozwiązaniem dobrze radzącym sobie ze skomplikowaną analizą. Na podstawie przeprowadzoncyh testów widać, że dla głębokości 5 algorytm MiniMax działa lekko poniżej milisekundy. Obiektywnie nie jest to zły czas, jednakże istnieją inne algorytmy realizujące podobne zadania w szybszym czasie lub z mniejszą złożonością obliczeniową (np. MCTS lub Reinforcement Learning). Przyjmując model szachowy, w którym ocena ruchu jest wyznaczana przez agregat (sumę) wszystkich figur jakie pozostaną na planszy mamy do czynienia z ogromną ilością węzłów drzewa. Gra ConnectFour jest znacząco prostsza i szybsza do oceny ze względu na proste zasady i brak konieczności przeprowadzania takowych obliczeń.

W kwestii porównania gry z głębokością N kontra M, zauważalnym jest fakt, iż czym większa różnica N-M tym wyższe win_ratio. Oznacza to, że z czym wyższą głębokością analizuje się rozgrywkę, tym łatwiej zwyciężyć z przeciwnikiem. Win_ratio nie rośnie liniowo wraz ze wzrostem głebokości wobec stałego M. Oznacza to, że korzyść z większej głębokości nie jest proporcjonalna do różnicy i ciężka do oszacowania.