Sztuczna inteligencja i inżynieria wiedzy Lista 2 - Raport

Łukasz Wasilewski

30.04.2024

1 Wstęp

Zadaniem które należało wykonać w ramach realizacji listy było zaimplementowanie algorytmu Minimax ora Alfa-Beta Cięć służące do w Halmę oraz wymyślenie 3 strategii do wyliczania heurystki.

2 Strategie

W rozwiązaniu zaimplementowałem następujące strategie:

1. Odległości
2. Liczby możliwych ruchów
3. Przemieszczania w grupie

Strategia odległości jest podstawową i najprostszą zaimplementowaną przeze mnie strategią do gry w Halmę. Polega ona na obliczaniu odległości pionków w kierunku bazy wroga. Im bliżej pionki bazy tym większa wartość funkcji dla gracza pierwszego oraz tym niższa dla gracza drugie. Drugim czynnikiem tej strategii jest dodatkowe nagradzanie za pionki, które dotarły do bazy.

Strategia liczby możliwych ruchów korzysta ze strategii odległości oraz dokłada dodatkowy czynik w postaci wylicznia wszystkich dostępnych ruchów w pozycji. Im większa liczba możliwych ruchów tym lepsza pozycja gracza.

Strategia przemieszczania w grupie również korzysta ze strategii odległości oraz dokłada wylicznie rozrzedzenia pionków gracza. Im większa szerokość oraz wysokość pomiędzy najdalszymi pionkami tym gorsza pozycja gracza. Strategia zakłada, że nie powinno się zostawiać samotnych pionków w tyle lub odskakiwać za mocno do przodu aby odnieść zwycięstwo.

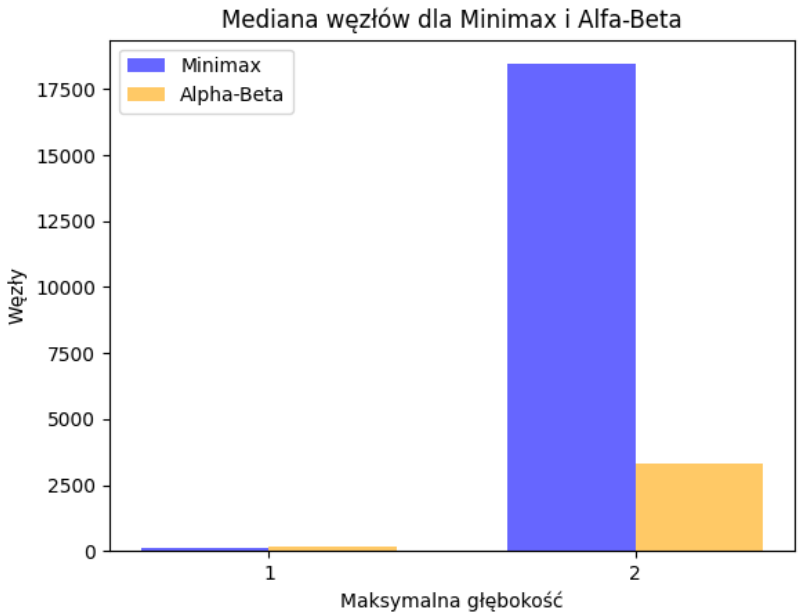
3 Porównanie Minimax i Alfa-Beta Cięć

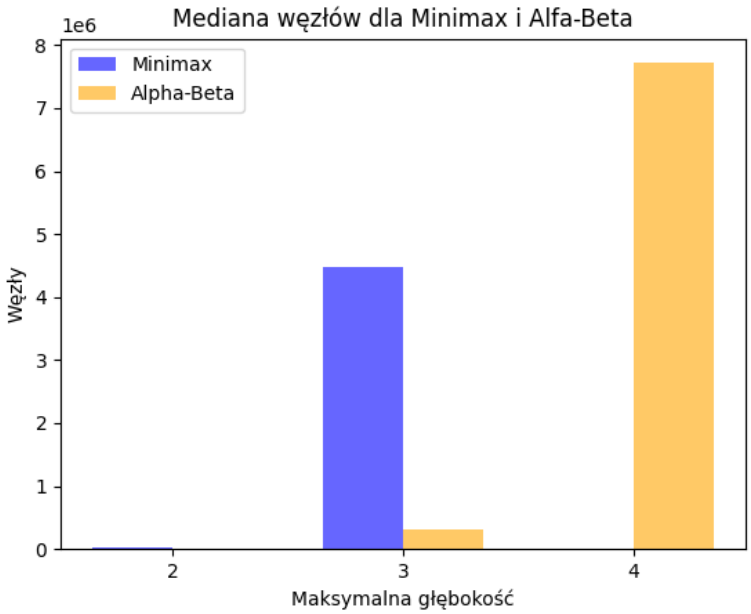
Algorytm Alfa-Beta Cięć jest optymalizacją algorytmu Minimax, która ma zadanie znajdować węzły na pewno nie prowadzące do lepszego rozwiązania i ucinać w ich miejscu dalsze przeszukiwanie. Zysk z tej metody w dużej mierze zależy od tego jak szybko natrafimy na najlepszy węzeł. W najgorszym możliwym przypadku ta metoda nie przyniesie żadnego zysku w czasie obliczeń i liczbie przeszukiwanych węzłów.

Zbadałem medianę liczby przeszukanych węzłów i czasu przeszukiwania jednego ruchu w zależności od maksymalnej głębokości oraz użytego algorytmu. Heurystyka obliczana była strategią odległości. Każda konfiguracja została uruchomiona 5 razy dla mniejszych głębokości co dało około 1000 próbek

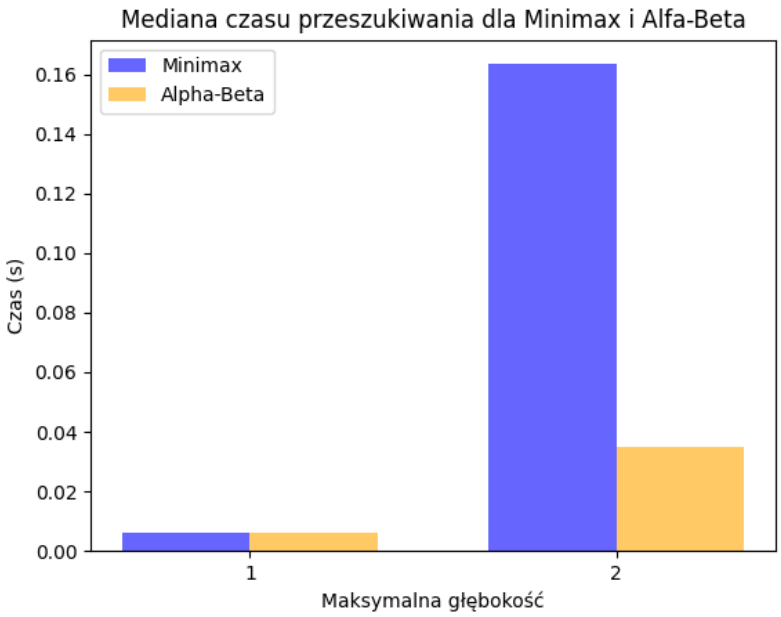
na przypadek oraz po razie przy największych sprawdzanych głębokościach ze względu na zbyt długi czas obliczeń.

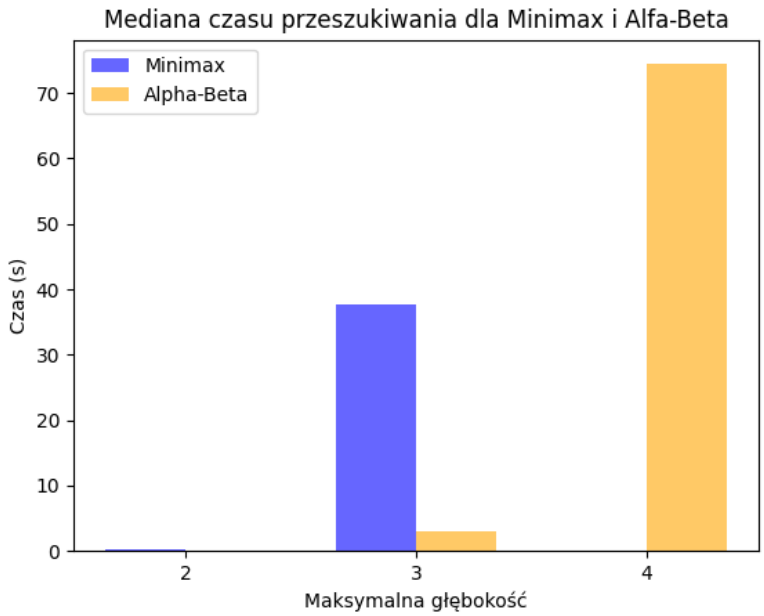
Mediana węzłów:





Mediana czasu:

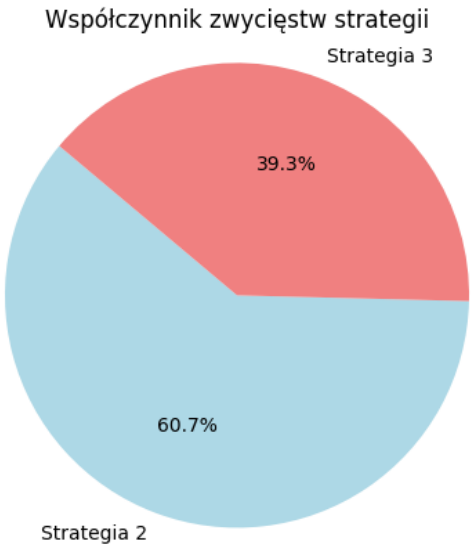




Wnioski: Wraz ze wzrostem maksymalnej głebokości rośnie zysk z użycia Alfa-Beta. Na głębokości 1 jest żaden, na głębokości 2 jest około 4 krotny, a na głębokości 3 jest około 10 krotny. Ponadto widzimy jak szybko rośnie czas i ilość przeszukiwanych węzłów wraz ze zwiększaniem maksymalnej głębokości. W przypadku Minimax moja cierpliwość skończyła się na głębokości 3, a przy Alfa-Beta - 4.

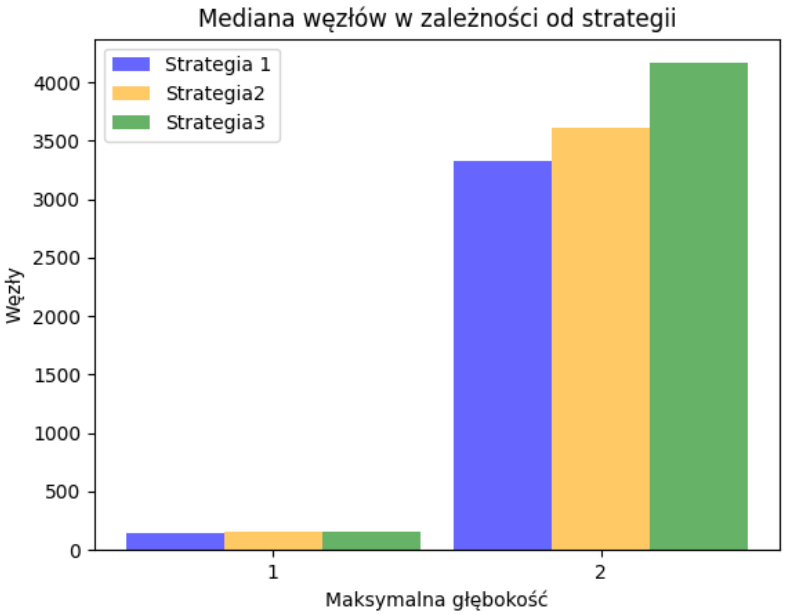
4 Porównanie strategii

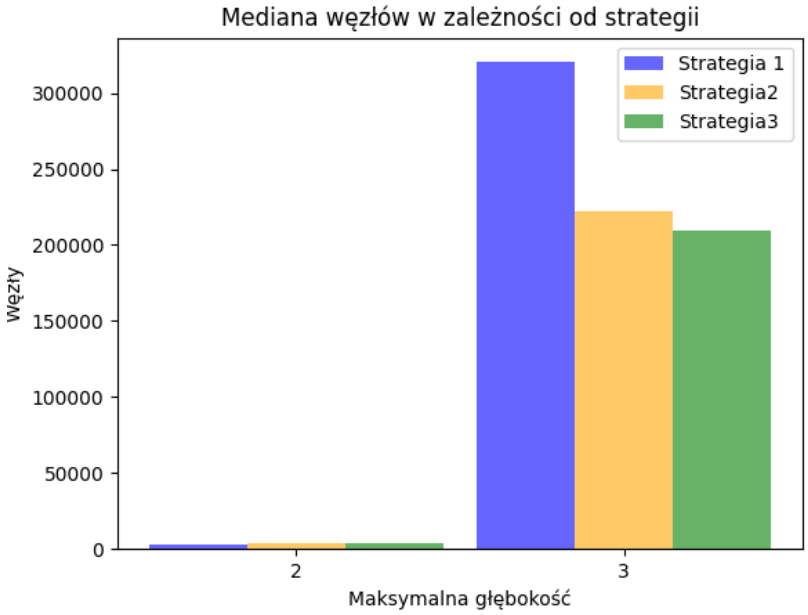
Aby przekanać się, która strategia najlepiej prowadzi do zwycięstwa postanowiłem rozegrać mecze między strategiami każda z każdą. Strategie zawalczyły ze sobą 30 razy, po 10 razy dla głębokości 1, 2 i 3, połowę razy jako gracz 1 i połowę jako gracz 2. Numery strategii odpowiadają kolejności z początku raportu.



Zmierzyłem również medianę czasu przeszukiwania węzłów oraz ich liczby w jednym ruchu w zależności od użytej heurystyki i maksymalnej głębokości.

Mediana węzłów:





Mediana czasu:

Wnioski: Najlepsza przeciwko pozostałym strategiom okazała się strategia liczby możliwych ruchów, natomiast jest ona najbardziej obciążająca obliczeniowo. Biorąc pod uwagę zarówno współczynniki zwycięstw oraz czas obliczeń najlepsza wydaje się strategia przemieszczania w grupie.

5 Podsumowanie i optymalizacje

W swoich rozwiązaniach wykorzystałem podstawowe biblioteki w Pythonie. Przy rozwiązywaniu listy udało mi się wykonać dwie ważne optymalizacje algorytmów. Pierwsza to leniwe tworzenie stanów gry przy przeszukiwania drzewa. Stany znajdujące się na tym samym poziomie są tworzone dopiero wtedy kiedy próbujemy dotrzeć do dzieci albo obliczyć heurystykę. Druga optymalizacja to brak niepotrzebnego tworzenia stanów na ostatniej głębokości, lecz jedynie przekazywanie heurystyki wyliczonej na podstawie stanu gry rodzica. Nie ma potrzeby tworzenia węzłów na samym dole drzewa kiedy interesuje nas jedynie wartość heurystki. Jedynym minusem tego rozwiązania jest dosyć brzydki blok kodu który musi na raz rozważać wszystkie możliwe ruchy i strategie graczy.