WSI - ćwiczenie 3.

Dwuosobowe gry deterministyczne

grupa 101

Wykonał: Adrian Pruszyński

1. Treść Zadania

W ramach trzeciego ćwiczenia należy zaimplementować algorytm minimax (bez obcinania alfa-beta). Powinien on być napisany ogólnie, tj. jedna implementacja powinna działać dla dowolnej podanej gry. W przypadku ruchów równie dobrych powinien być wybierany losowy z nich. Następnie należy, wykorzystując zaimplementowany algorytm, napisać program ”podpowiadający” użytkownikowi, który ruch jest najlepszy w podanej sytuacji.

2. Założenia

Analizowany przykład opiera się o grę w kółko i krzyżyk

Implementacja algorytmu zakłada, że gracz posługujący się „O” maksymalizuje natomiast „X” minimalizuje.

Implementacja zawiera klasę bazową dającą możliwość zdefiniowania innych gier oraz klasę przygotowaną dla gry kółko i krzyżyk.

W celu odpowiedniej wizualizacji wyników użyte zostały dodatkowe biblioteki.

Dla zachowania różnowartościowości nazw wierzchołków w do każdej z nich dołączony został identyfikator obiektu planszy.

Sugerowany ruch wskazuje pokolorowana krawędź wychodząca z wierzchołka reprezentującego zadany stan planszy. Kolejne krawędzie wyznaczają dalszy przebieg rozgrywki dla optymalnych wyborów względem heurystyki.

Heurystyka wykorzystana w implementacji zaczerpnięta została z wykładu:

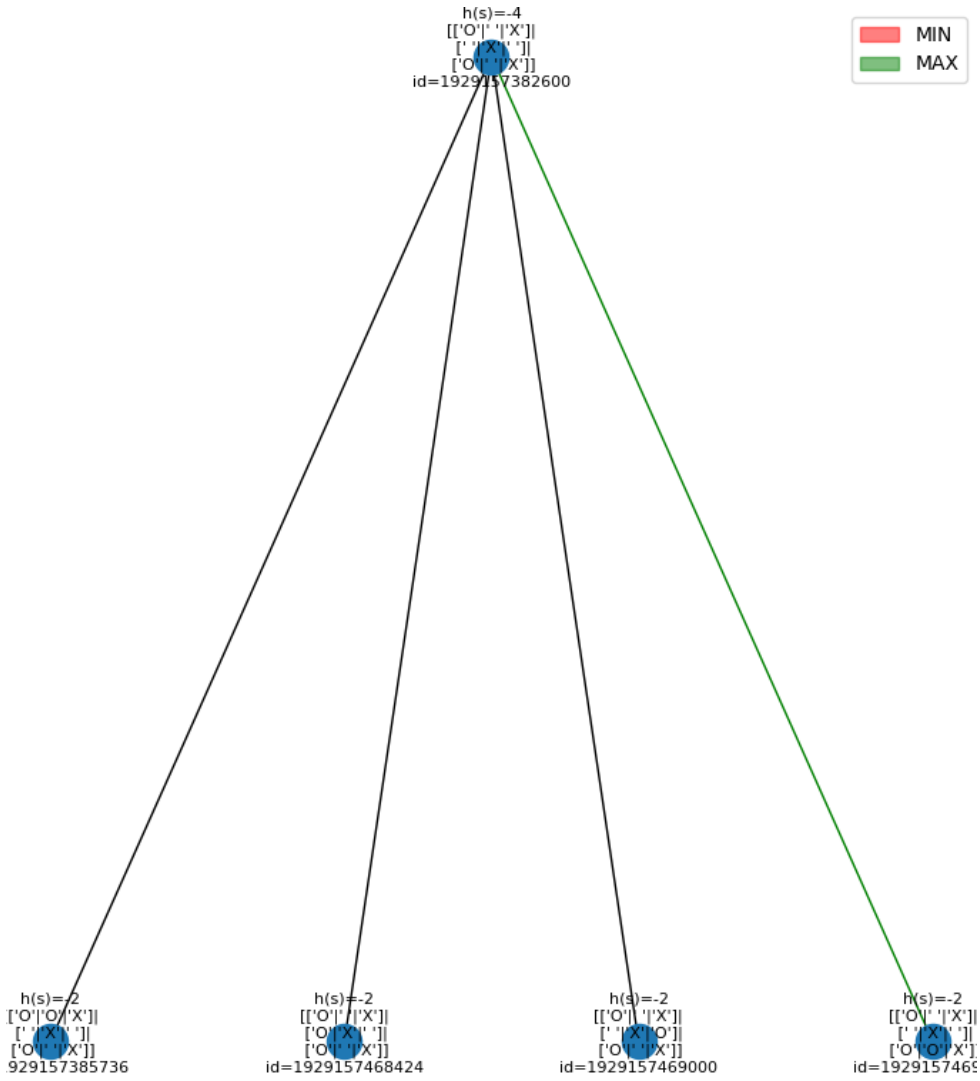
Heurystyczna funkcja oceny sytuacji w grze

Przykład heurystyki dla gry kółko i krzyżyk to suma liczba punktów za pola zgodnie z macierzą:

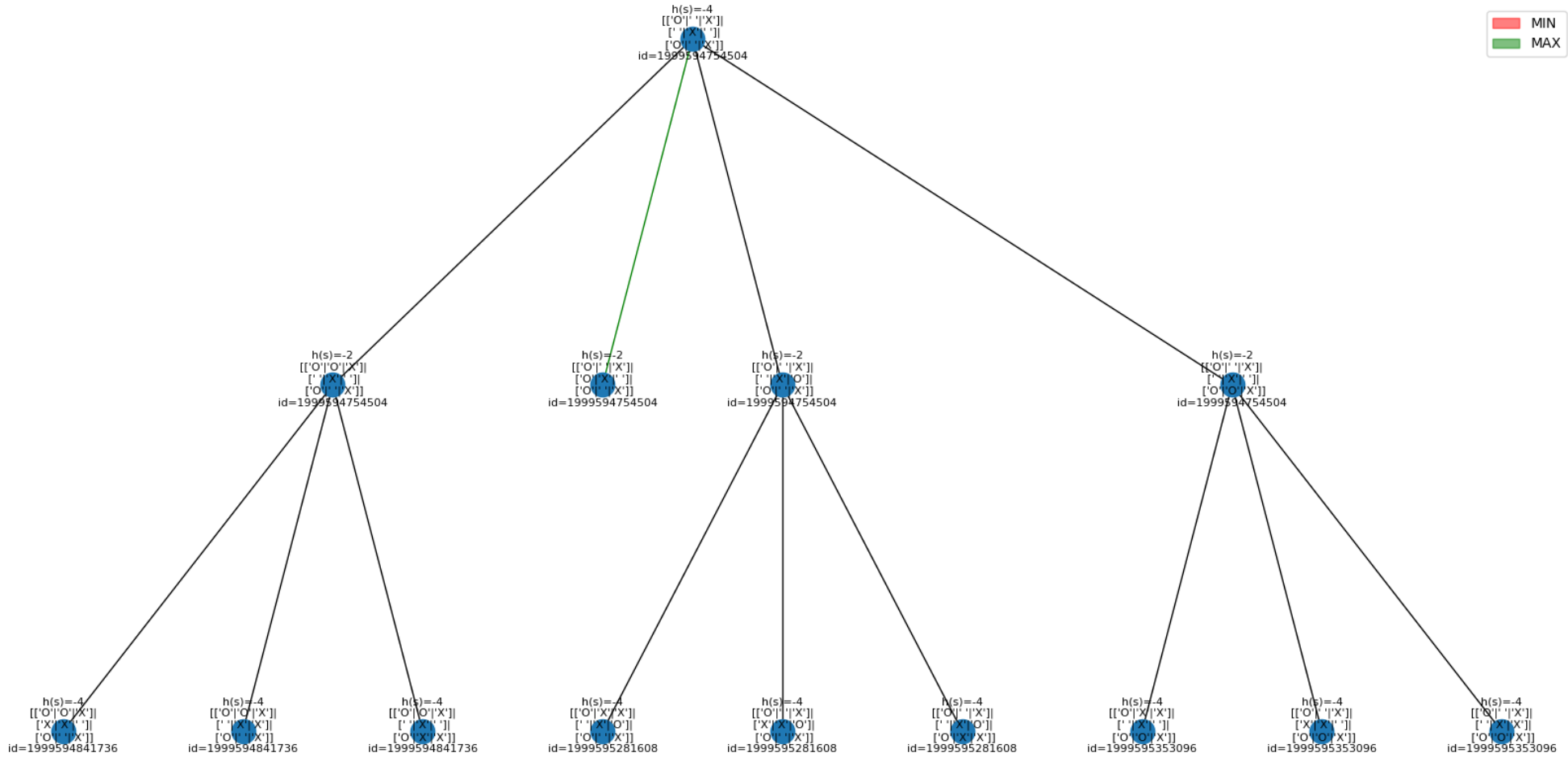
3	2	3
2	4	2
3	2	3

3. Analiza działania algorytmu

Rysunek 1

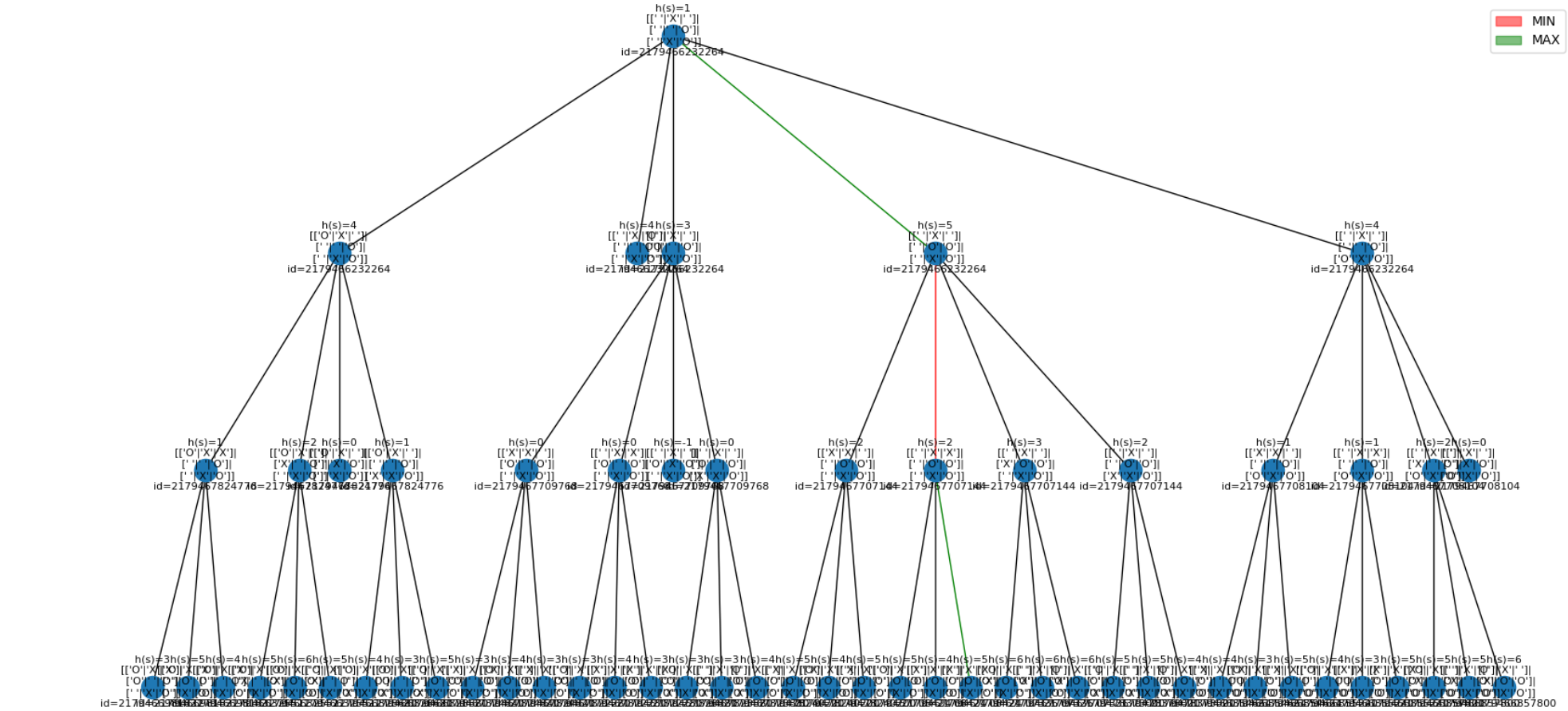


Rysunek 2



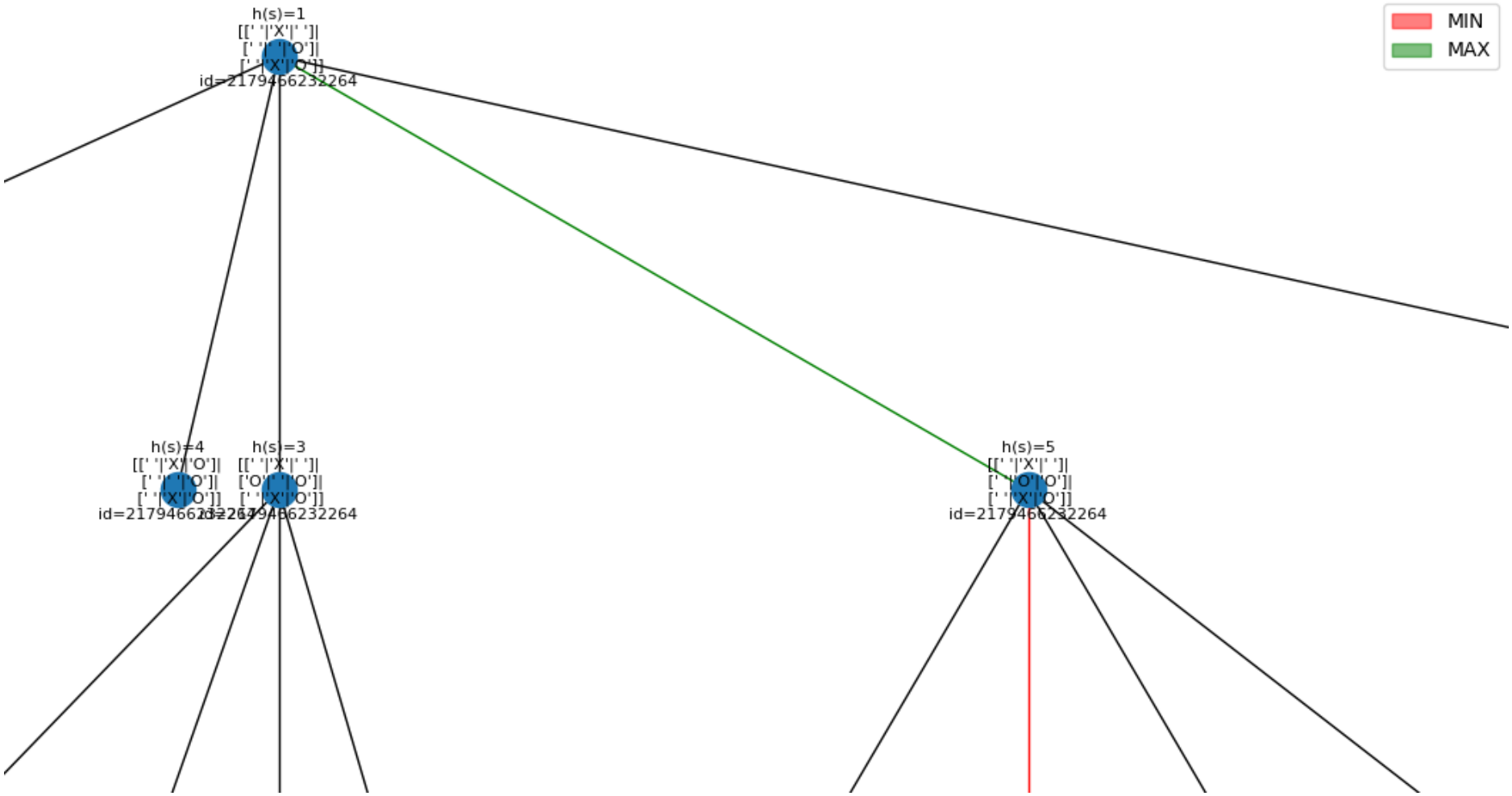
Rysunek 1 oraz 2 przedstawiają wynik uruchomienia programu dla tej samej planszy jednak różnych głębokości. Jak widać sugestie algorytmu co do kolejnego ruchu różnią się. Jak widać w przypadku rysunku 1 dla głębokości 1 wszystkie posunięcia są tak samo dobre, natomiast przy głębokości 2 (Rysunek 2) widzimy, że wybór wierzchołka terminalnego jest lepszy. Ujawnia się również słabość wybranej heurystyki, która nie premiuje dążenia do wierzchołków terminalnych wygrywających w rozumieniu gracza dokonującego wyboru.

Rysunek 3



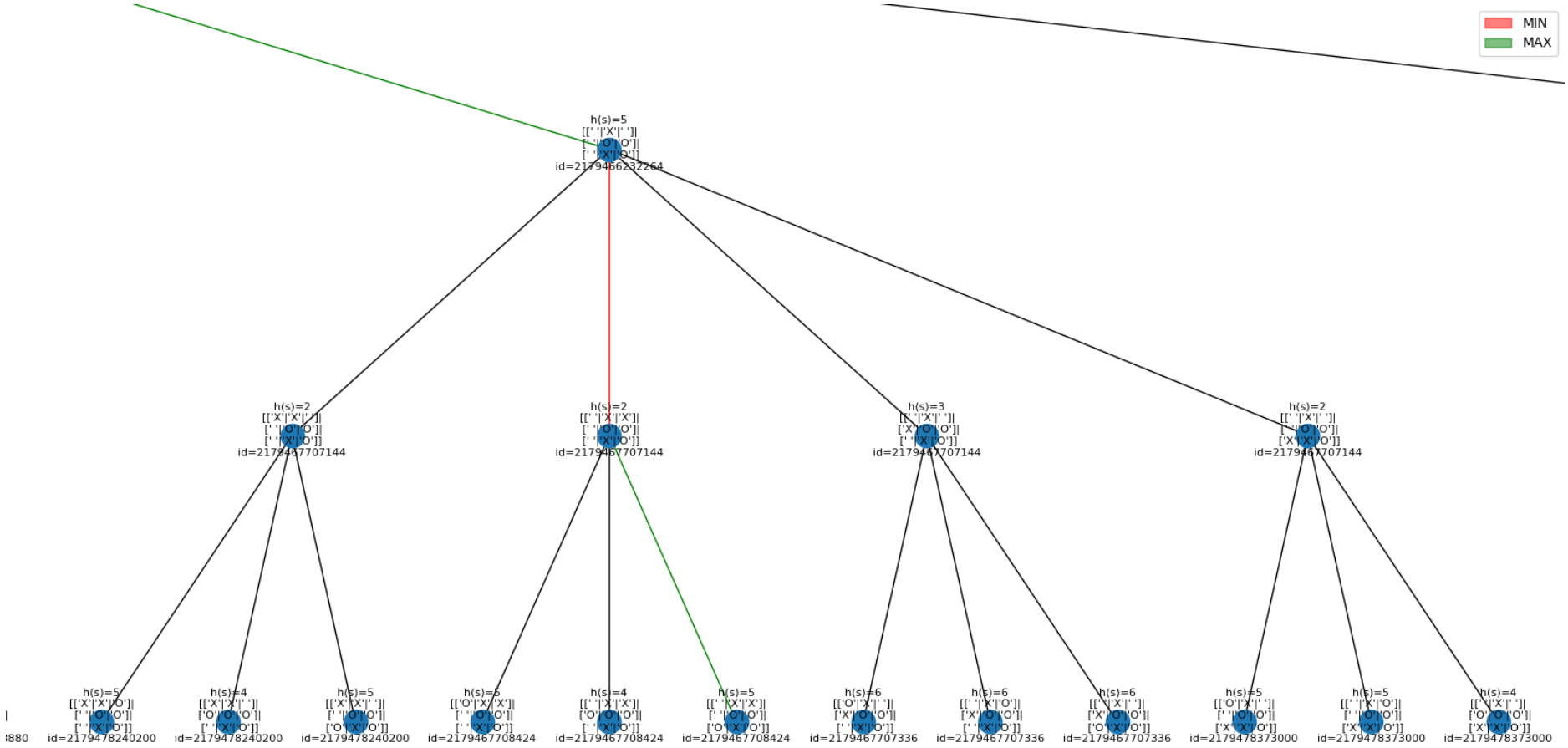
Na rysunku 3 widzimy kolejny przykład wskazania przez program nieoptymalnej decyzji względem zasad gry, jednak jest ona optymalne względem wybranej heurystyki.

Rysunek 4



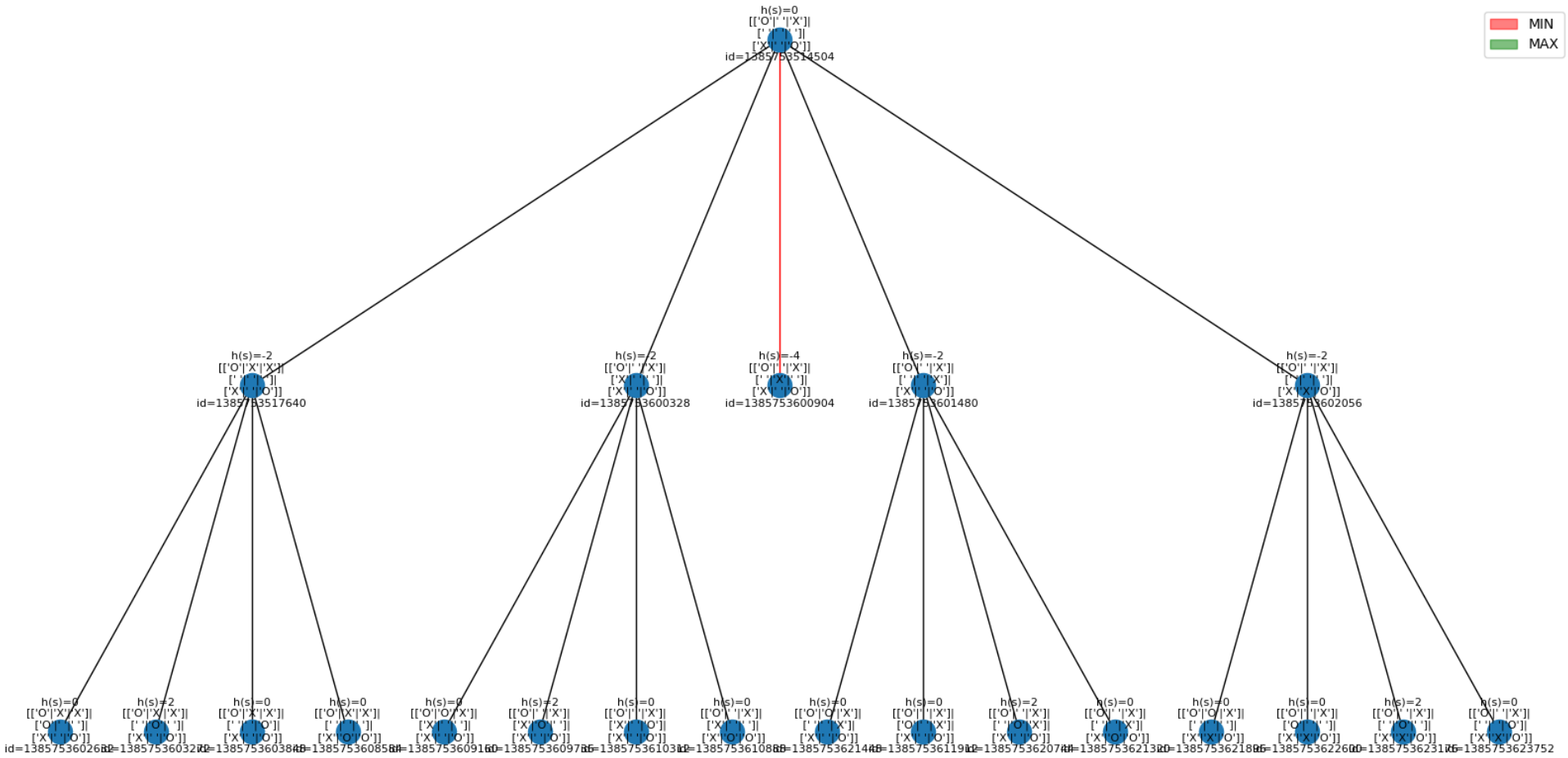
Rysunek 4 przedstawia ten sam graf co w przypadku rysunku 3 skupia się ona obszarze umożliwiającym zaobserwowanie sugerowanego nieoptymalnego wyboru. Jak widać wybór optymalny, względem zasad gry, ma wartość heurystyki równą 4.

Rysunek 5



Rysunek 5 przedstawia ten sam graf co w przypadku rysunku 3 oraz 4 skupia się ona obszarze umożliwiającym zaobserwowanie powodu zasugerowania nieoptymalnego wyboru. Jak wiemy z rysunku 4 wartość heurystyki dla optymalnego wyboru, względem zasad gry, była równa 4. Widzimy, że sugerowany wybór dla zadanej głębokości jest wynikiem możliwości osiągnięcia wyższej wartości heurystyki dla nieoptymalnego wyboru, względem zasad gry.

Rysunek 6



Rysunek 6 pokazuje przypadek, gdy program wskazuje optymalne rozwiązanie zarówno pod względem heurystyki jak i zasad gry.

4. Wnioski i podsumowanie

Analiza algorytmu uwidoczniła jak ważne jest wybranie odpowiedniej heurystyki oraz dopasowanie głębokości.

Wybrana funkcja oceny nie jest najlepsza. Dodanie do niej znaczącej premii dla stanów wygranych korzystnych gracza podejmującego decyzję znacząco poprawiłoby jej skuteczność.

Głębokość ma ogromny wpływ na wydajność programu. Dlatego nie powinien być zbyt duży. Dla dostatecznie dużej wartości głębokości algorytm sprowadza się do przeglądu wyczerpującego.