WSI - Minimax

Jan Szymczak

Marzec 2024

1 Cel i opis eksperymentów

1.1 Opis problemu

Celem było stworzenie implementacji algorytmu Minimax wykorzystującym odcinanie alfa-beta do gry Szewc (ang. Dots and boxes). Algorytm działa dla dowolnej wielkości planszy oraz wskazanej, nieujemnej głębokości przeszukiwania. Uwzględnia ponadto mechaniki gry, takie jak możliwość wykonania więcej niż jednego ruchu z rzędu przez danego gracza. Jako implementacja gry została wykorzystana implementacja znajdująca się w tym repozytorium: `https://github.com/lychanl/two-player-games`. Wprowadzone zostały niewielkie zmiany w implementacji, ale żadne w funkcjonalności, jedynie w sposobie wyświetlania niektórych danych. Zmieniony plik znajduje się w repozytorium razem z algorytmem. Jako funkcję heurystyczną przyjęta została różnica w zdobytych punktach pomiędzy graczem Max a Min. W przypadku, gdy więcej niż jeden ruch dawał optymalną wartość, wybierany został losowy ruch z optymalnych z równym prawdopodobieństwem. Zatem, podczas testowania algorytmu z zerową głębokością przeszukiwania, gracz wykonywał losowe ruchy.

1.2 Wpływ parametrów

Wyniki problemu są zależne od dwóch parametrów:

- **Głębokość przeszukiwania:** czym większa, tym lepszy ruch może zostać znaleziony. Drastycznie wydłuża jednak czas wykonywania programu.
- **Rozmiar planszy:** ma wpływ jedynie na długość wykonywania programu, a nie na poprawność wyników. Naturalnie - większa plansza oznacza dłuższy czas działania.

2 Wyniki

W celu przetestowania algorytmu przeprowadziłem 50 gier pomiędzy dwoma graczami o różnych głębokościach na planszy o wielkośći 3 (po 4 kropki w rzędach). Tak prezentuje się heatmapa przedstawiająca wyniki:

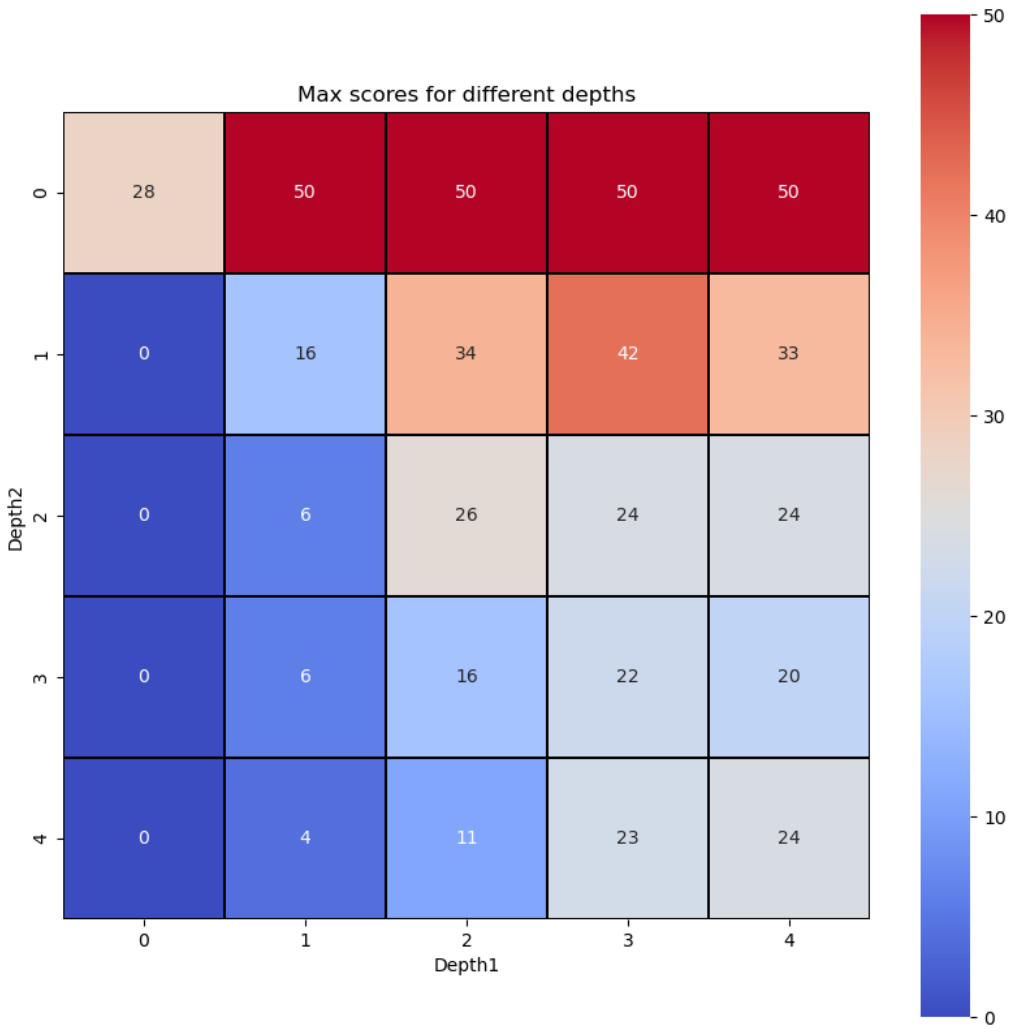

Figure 1: Heatmapa przedstawiająca wyniki testów dla wielkości 3

Oś X to głębokość przeszukiwania pierwszego gracza, oś Y drugiego. Wartość pól to ilość zwycięstw gracza max. Na takiej planszy remis jest niemożliwy, więc zawsze ilość zwycięstw gracza min to 50 - ilość zwycięstw gracza max.
W przypadku, gdy którykolwiek gracz ma głębokość 0, a drugi dodatnią, za każdym razem wygrywa przeciwnik. To dobrze, nawet z tak płytkim przeszukiwaniem gracz „myślący" powinien wygrywać z grazem wykonującym całkowicie losowe ruchy. W reszcie przypadków widać minimalną przewagę gracza min, co także jest sensowne w przypadku tej gry, drugi gracz ma przewagę. W ogólności, min zawsze wygrywa więcej niż 50% gier w przypadku, gdy działa z większą głębokością. W przypadku równych głębokości lub bliskich sobie (np. Depth1 = 4, Depth2=3) widać także niewielką przewagę gracza min. Ponadto, wyniki i tak są wystawione na niewielki wpływ losowości - funkcja heurystyczna jedynie bierze pod uwagę wyniki, a ruchy są wykonywane losowo, jeśli więcej niż jeden okazuje się optymalny. To powoduje, że na początku rozgrywki, gracze losowo wypełniają plansze: widzą tylko ruchy, w których dalej wychodzą „na zero" i te, po których stracą punkty, jeśli wypełnią trzeci blok kwadratu. To ma całkowicie sens, w rzeczywistości właśnie na tym polega omawiana gra: składa się z etapu układania planszy, a potem serią zajmowania jak największej ilości kwadratów.

Można jeszcze przyjrzeć się jak pewnie wygrywali poszczególni gracze. Tak przedstawia się wykres średniej ilości zdobytych punktów przez gracza max w zależności od użytych głębokości:

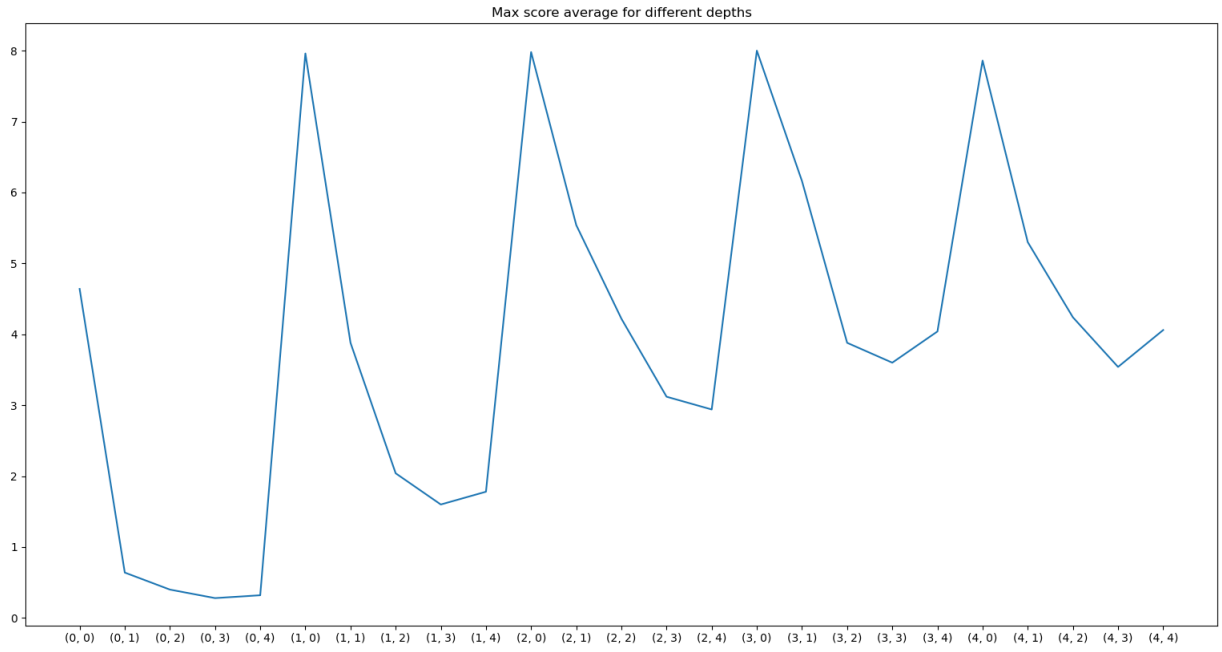

Figure 2: Wykres przedstawiający wyniki testów

W przypadkach, gdy któryś gracz wygrywał ze 100% skutecznością robił to prawie zawsze do zera. Widać też jak wykres stopniowo rośnie wraz ze wzrostem głębokości przeszukiwania gracza max (pomijając nagłe skoki wartości, gdy gracz min miał zerową głębokość). Gracz max grał coraz lepiej wraz ze wzrostem głębokości przeszukiwania, więc to także podkreśla poprawność algorytmu.

2.1 Inne wielkości planszy

Dla innych wielkości planszy wyniki były bardzo podobne, nie ma to wpływu na wyniki programu, z tym wyjątkiem, że w innych konfiguracjach remis może występować i maksymalna ilość punktów do zdobycia się zmienia. Plik *results4.csv* z wynikami testów dla planszy o wielkości 4 także udostępniam w repozytorium. Przez parzystość rozmiaru planszy tym razem to pierwszy gracz, max, ma przewagę. Pojawiły się także remisy, głównie w sytuacjach, gdy głębokości przeszukiwań były podobne. Tak prezentuje się analogiczna heatmapa:

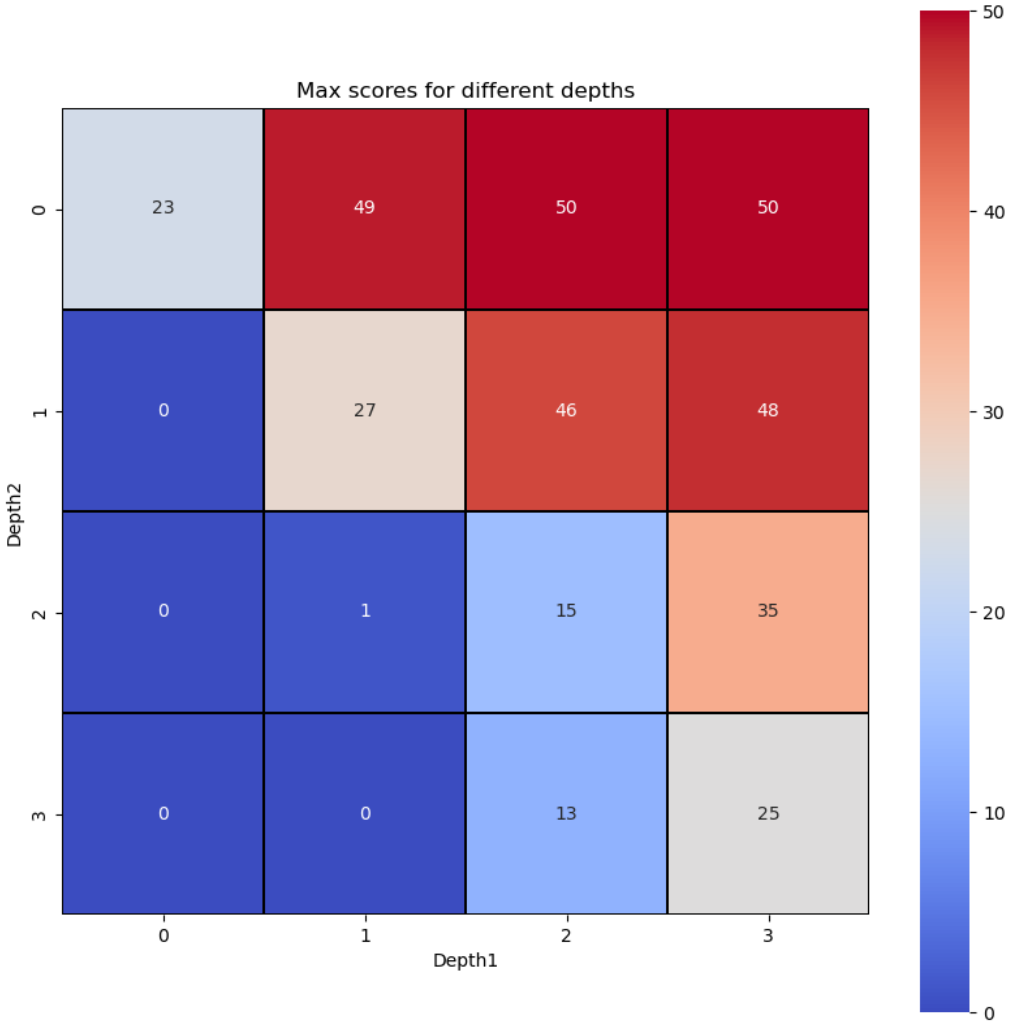

Figure 3: Heatmapa przedstawiająca wyniki testów dla wielkości 4

Dla innych wielkości również przeprowadziłem testy, ale wyniki były takie same - z tego powodu nie będę udostępniać kolejnych plików. Ogólnie na planszach o parzystej wielkości przewagę ma pierwszy gracz, o nieparzystej drugi. Czym mniejsza plansza tym losowość też zaczyna mieć większe znaczenie.