WSI

Zadanie 3

Autor:
Michał Paradowski

19.03.2024 (V1.0)

Contents

Contents I

1 Zadanie 1

2 Algorytm 2

3 Eksperyment 1 4

4 Wnioski 10

Bibliography 11

List of Figures 11

1 Zadanie

Celem zadania jest implementacja algorytmu Min-Max dla gry w kółko i krzyżyk.

Rozwiązanie zadania powinno zawierać następujące elementy:

• Implementacja gry w kółko i krzyżyk na planszy 3x3 (można zaprojektować wersję, w której użytkownik gra z programem, lub program gra sam ze sobą, zostawiam ten wybór Państwu),

• implementacja metody Min-Max, która na podstawie aktualnego stanu gry steruje ruchami programu,

• implementacja algorytmu $\alpha - \beta$,

• analiza działania zaimplementowanego rozwiązania wraz z rejestracją wyników.

Kolejne stany gry powinny być wizualizowane, jednym ze sposobów może być reprezentowanie ich w dwuwymiarowej tablicy. Gra powinna trwać do wyłonienia zwycięzcy lub osiągnięcia remisu (zapełnione wszystkie pola na planszy). Należy przeanalizować ilość zbadanych węzłów i głębokość drzewa dla kilku wybranych stanów na początku i w trakcie trwania gry. Proszę zbadać wpływ algorytmu $\alpha - \beta$ na działanie programu - ilość zbadanych węzłów i głębokość drzewa z i bez implementacji algorytmu $\alpha - \beta$.

2 Algorytm

Wykorzystując algorytm min-max w znajdowaniu strategii wygrywającej w grze bazujemy na założeniu że obydwaj gracze grają zawsze optymalnie, to znaczy że wykonują najbardziej opłacalne ruchy. Z tego założenia wynika na przykład, że w grze w kółko i krzyżyk z perspektywy gracza A ustawienie w którym gracz B może wykonać ruch kończący grę jego zwycięstwem ale może też wykonać ruch który doprowadzi do zwycięstwa gracza A jest równie 'złe' jak takie w którym każdy ruch gracza B doprowadzi do jego zwycięstwa. Jest to założenie w pewnym sensie nieintuicyjne ale konieczne. Analizując grę w kółko i krzyżyk przyjmiemy że zaczyna gracz P_1 zaś gracz P_2 wykonuje ruch jako drugi. Przechodząc do algorytmu, wprowadzimy oznaczenia:

- T - drzewo wszystkich możliwych stanów gry. Korzeń T znajduje się na głębokości 0 i reprezentuje pustą planszę, dzieci każdego wierzchołka W na głębokości k drzewa T to możliwe stany gry po wykonaniu przez odpowiedniego gracza (gdy k parzyste jest to P_1 w.p.p. P_2) swojego ruchu na planszy z wierzchołka W.
- $C(W)$ - zbiór synów wierzchołka W.
- $S(W)$ - ocena stanu w wierzchołku W zdefiniowana rekurencyjnie w następujący sposób:
  1) Jeśli nie da się już wykonać ruchu lub któryś z graczy wygrał:

$$S(W) = \begin{cases} 1 & \text{jeśli } P_1 \text{ wygrywa na planszy w } W, \\ -1 & \text{jeśli } P_2 \text{ wygrywa na planszy w } W, \\ 0 & \text{jeśli stan w } W \text{ oznacza, że gra zakończyła się remisem} \end{cases}$$

  2) w.p.p:

$$S(W) = \begin{cases} \max(\{S(V) : V \in C(W)\}) & \text{jeśli } k \text{ parzyste}, \\ \min(\{S(V) : V \in C(W)\}) & \text{jeśli } k \text{ nieparzyste} \end{cases}$$

  Cały algorytm opiera się na prostej zasadzie, że gracz MAX (W tym przypadku A) podczas swojej tury stara się wykonać ruch prowadzący do zmaksymalizowania oceny stanu gry po tym ruchu, zaś gracz MIN (W tym przypadku B) podczas swojej tury chce tą ocenę zminimalizować. Zasada działania jest prosta i logiczna. Podczas analizy następnych ruchów gracz A analizuje drzewo stanów w głąb. Gdy podczas analizy znajduje się na głębokości na której ruch wykonywałby gracz B to przyjmuje on, że gracz ten wykona ruch z najniższą oceną z dostępnych, Jeśli jesteśmy na głębokości na której ruch wykonuje gracz A, to wybiera on ten z najwyższą. Po dokonaniu analizy całego poddrzewa drzewa T z korzeniem w aktualnym stanie gry wybierany jest następny ruch. W pełni analogicznie

ale minimalizując ocenę następnego stanu zachowuje się gracz MIN.

Ten algorytm o ile bardzo prosty, jest też bardzo niewydajny obliczeniowo gdyż za każdym razem przeszukujemy całe poddrzewo drzewa T z korzeniem w aktualnym stanie co nawet przy analizie tak prostej gry jak kółko i krzyżyk prowadzi sprawdzenia kilkuset tysięcy węzłów (w pierwszym ruchu dokładnie 549946). Złożoność obliczeniową można znacznie poprawić wprowadzając modyfikację do algorytmu zwaną $\alpha - \beta$.

Modyfikacja to opiera się na następującej obserwacji:
Załóżmy że szukamy najlepszego ruchu jako gracz MAX. Załóżmy że sprawdziliśmy już k synów wierzchołka w którym się znajdujemy (wierzchołka V) i do tej pory najlepszy ruch zwracał wycenę α. Jeśli podczas przeszukiwania $k+1$-syna obecnego wierzchołka (niech będzie to U) natrafimy na wnuka V który zwraca wycenę niższą niż α, to wiemy, że gracz MIN podejmujący decyzję w wierzchołku U wybierze ruch o wycenie nie lepszej niż ta znaleziona a co za tym idzie gorszej niż α. Tak więc jako gracz MAX podejmujący decyzję w V nie mamy po co sprawdzać pozostałych synów U bo i tak tamtą ścieżką nie pójdziemy. Analogiczną metodę (gdzie β to najniższa znaleziona wycena) możemy stosować dla gracza MIN. Tą modyfikację wprowadzamy oczywiście na każdym poziomie przeszukiwania, nie tylko na najwyższym. Znacznie przyspiesza to działanie algorytmu co wykażę poniżej.

Przeprowadzę cztery eksperymenty gry człowieka z algorytmem. dwa zakończą się wygraną algorytmu, a dwa remisem. we wszystkich eksperymentach zastosuję obie wersje algorytmu. Zwycięstwo człowieka jest niemożliwe gdyż przy optymalnej grze obu osób gra zawsze kończy się remisem.

3 Eksperyment 1

1.1) Poniżej przedstawię przebieg gry w której zaczyna algorytm. '1' reprezentuje gracza rozpoczynającego a '−1' drugiego. Poza przebiegiem gry przedstawię też liczbę węzłów drzewa odwiedzonych przez algorytm za każdym razem (wartość pod strzałką).

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow[549946]{Algorytm} \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&0&0\\0&-1&0\\0&0&0\end{bmatrix} \xrightarrow[7332]{Algorytm} \begin{bmatrix}1&1&0\\0&-1&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&1&-1\\0&-1&0\\0&0&0\end{bmatrix}$$

$$\xrightarrow[198]{Algorytm} \begin{bmatrix}1&1&-1\\0&-1&0\\1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&1&-1\\-1&-1&0\\1&0&0\end{bmatrix} \xrightarrow[14]{Algorytm} \begin{bmatrix}1&1&-1\\-1&-1&1\\1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&1&-1\\-1&-1&1\\1&0&-1\end{bmatrix}$$

$$\xrightarrow[2]{Algorytm} \begin{bmatrix}1&1&-1\\-1&-1&1\\1&1&-1\end{bmatrix} \rightarrow \text{Remis}$$

1.2) Poniżej identyczna[1] gra z algorytmem korzystającym z dodatku $\alpha - \beta$.

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow[3957]{Algorytm} \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&0&0\\0&-1&0\\0&0&0\end{bmatrix} \xrightarrow[318]{Algorytm} \begin{bmatrix}1&1&0\\0&-1&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&1&-1\\0&-1&0\\0&0&0\end{bmatrix}$$

$$\xrightarrow[21]{Algorytm} \begin{bmatrix}1&1&-1\\0&-1&0\\1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&1&-1\\-1&-1&0\\1&0&0\end{bmatrix} \xrightarrow[4]{Algorytm} \begin{bmatrix}1&1&-1\\-1&-1&1\\1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&1&-1\\-1&-1&1\\1&0&-1\end{bmatrix}$$

$$\xrightarrow[1]{Algorytm} \begin{bmatrix}1&1&-1\\-1&-1&1\\1&1&-1\end{bmatrix} \rightarrow \text{Remis}$$

Widzimy, że zgodnie z przypuszczeniami, algorytm stosujący $\alpha - \beta$ okazał się wielokrotnie szybszy. i odwiedził tylko ułamek węzłów odwiedzonych przez algorytm podstawowy.

[1] W grze kółko i krzyżyk wiele ruchów jest równie opłacalnych. Skutkuje to tym, że to, jaki ruch wykona algorytm wynika z m.in. kolejności rozpatrywania ruchów. Napisałem obie wersje algorytmu w taki sposób, by w tej samej sytuacji wybierany był ten sam ruch.

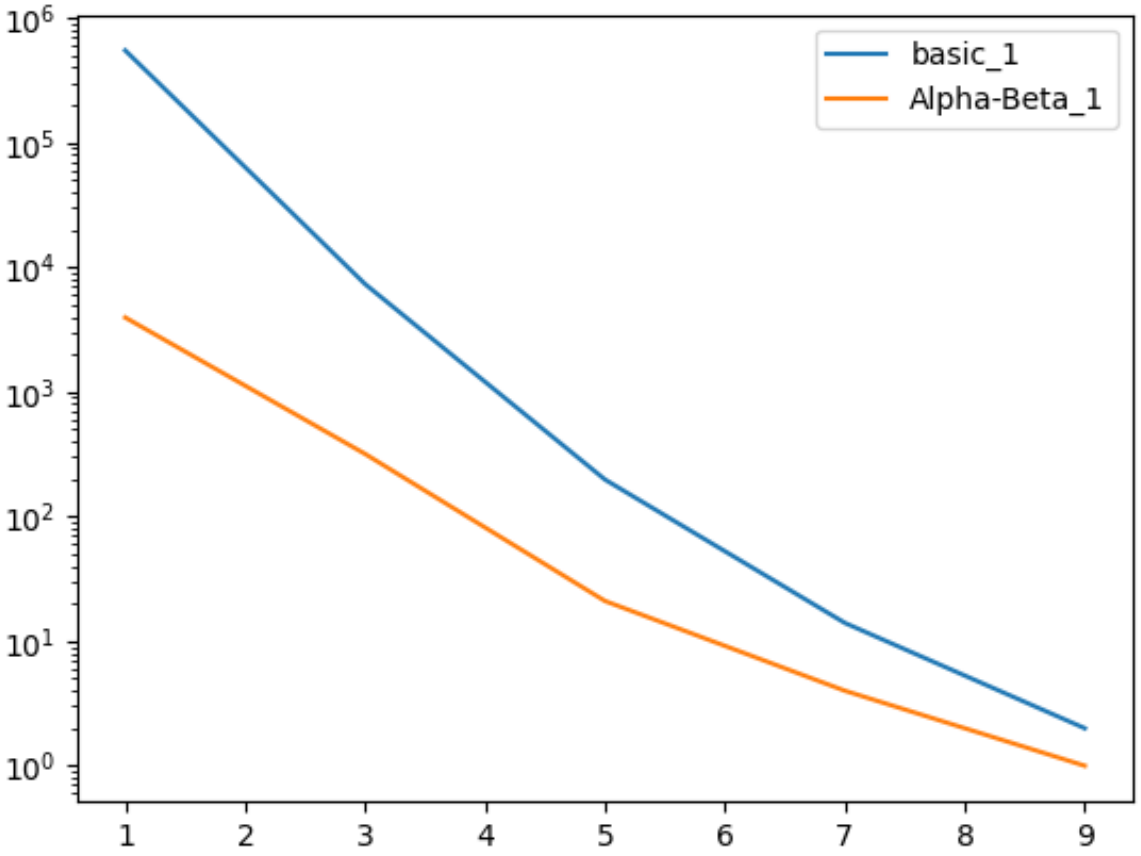

Figure 3.1: Ilość sprawdzonych węzłów w zależności od głębokości drzewa

Jak widać na wykresie ze skalą logarytmiczną, algorytm stosujący $\alpha - \beta$ jest o kilka rzędów wielkości szybszy. Widać też że liczba sprawdzanych węzłów w zależności od numeru ruchu drzewa maleje wykładniczo. Jest to logiczne gdyż rozmiar drzewa rośnie wykładniczo wraz z głębokością. Poniżej zaprezentuję jeszcze 3 gry przeprowadzone na obu algorytmach. W pierwszej zacznie algorytm i doprowadzę do jego wygranej, w drugiej rozpocznę ja i doprowadzę do remisu a w trzeciej zacznę ja i doprowadzę do swojej przegranej.

2.1) Zaczyna algorytm wygrywa algorytm bez $\alpha - \beta$:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow[549946]{Algorytm} \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow[8232]{Algorytm} \begin{bmatrix}1&-1&0\\1&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\1&0&0\\-1&0&0\end{bmatrix}$$

$$\xrightarrow[246]{Algorytm} \begin{bmatrix}1&-1&0\\1&1&0\\-1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\1&1&0\\-1&-1&0\end{bmatrix} \xrightarrow[7]{Algorytm} \begin{bmatrix}1&-1&0\\1&1&1\\-1&-1&0\end{bmatrix} \rightarrow \text{Wygrał Algorytm}$$

2.2) Zaczyna algorytm wygrywa algorytm z $\alpha - \beta$:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow[3957]{Algorytm} \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow[241]{Algorytm} \begin{bmatrix}1&-1&0\\1&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\1&0&0\\-1&0&0\end{bmatrix}$$

$$\xrightarrow[27]{Algorytm} \begin{bmatrix}1&-1&0\\1&1&0\\-1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\1&1&0\\-1&-1&0\end{bmatrix} \xrightarrow[1]{Algorytm} \begin{bmatrix}1&-1&0\\1&1&1\\-1&-1&0\end{bmatrix} \rightarrow \text{Wygrał Algorytm}$$

3.1) Zaczyna gracz, remis, bez bez $\alpha - \beta$

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} \xrightarrow[55505]{Algorytm} \begin{bmatrix}-1&0&0\\0&1&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}-1&0&1\\0&1&0\\0&0&0\end{bmatrix} \xrightarrow[933]{Algorytm} \begin{bmatrix}-1&0&1\\0&1&0\\-1&0&0\end{bmatrix}$$

$$\xrightarrow{Gracz} \begin{bmatrix}-1&0&1\\1&1&0\\-1&0&0\end{bmatrix} \xrightarrow[51]{Algorytm} \begin{bmatrix}-1&0&1\\1&1&-1\\-1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}-1&1&1\\1&1&-1\\-1&0&0\end{bmatrix} \xrightarrow[5]{Algorytm} \begin{bmatrix}-1&1&1\\1&1&-1\\1&-1&0\end{bmatrix}$$

$$\xrightarrow{Gracz} \begin{bmatrix}-1&1&1\\1&1&-1\\1&-1&1\end{bmatrix} \rightarrow \text{Remis}$$

3.2) Zaczyna gracz, remis, z $\alpha - \beta$

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} \xrightarrow[1560]{Algorytm} \begin{bmatrix}-1&0&0\\0&1&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}-1&0&1\\0&1&0\\0&0&0\end{bmatrix} \xrightarrow[57]{Algorytm} \begin{bmatrix}-1&0&1\\0&1&0\\-1&0&0\end{bmatrix}$$

$$\xrightarrow{Gracz} \begin{bmatrix}-1&0&1\\1&1&0\\-1&0&0\end{bmatrix} \xrightarrow[9]{Algorytm} \begin{bmatrix}-1&0&1\\1&1&-1\\-1&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}-1&1&1\\1&1&-1\\-1&0&0\end{bmatrix} \xrightarrow[2]{Algorytm} \begin{bmatrix}-1&1&1\\1&1&-1\\1&-1&0\end{bmatrix}$$

$$\xrightarrow{Gracz} \begin{bmatrix}-1&1&1\\1&1&-1\\1&-1&1\end{bmatrix} \rightarrow \text{Remis}$$

4.1) Zaczyna gracz, wygrywa Algorytm, bez $\alpha - \beta$

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} \xrightarrow[59705]{Algorytm} \begin{bmatrix}0&0&0\\0&-1&0\\0&0&1\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix} \xrightarrow[1053]{Algorytm} \begin{bmatrix}1&-1&0\\0&-1&0\\0&0&1\end{bmatrix}$$

$$\xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\0&-1&0\\1&0&1\end{bmatrix} \xrightarrow[24]{Algorytm} \begin{bmatrix}1&-1&0\\0&-1&0\\1&-1&1\end{bmatrix} \rightarrow \text{Wygrywa Algorytm}$$

4.2) Zaczyna gracz, wygrywa Algorytm, z $\alpha - \beta$

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} \xrightarrow[573]{Algorytm} \begin{bmatrix}0&0&0\\0&-1&0\\0&0&1\end{bmatrix} \xrightarrow{Gracz} \begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix} \xrightarrow[30]{Algorytm} \begin{bmatrix}1&-1&0\\0&-1&0\\0&0&1\end{bmatrix}$$

$$\xrightarrow{Gracz} \begin{bmatrix}1&-1&0\\0&-1&0\\1&0&1\end{bmatrix} \xrightarrow[1]{Algorytm} \begin{bmatrix}1&-1&0\\0&-1&0\\1&-1&1\end{bmatrix} \rightarrow \text{Wygrywa Algorytm}$$

Poniżej przedstawię wykresy ilości przeszukanych węzłów w zależności od numeru ruchu dla wszystkich 3 nowych par gier, oraz jeden wykres zestawiający wszystkie rozegrane gry:

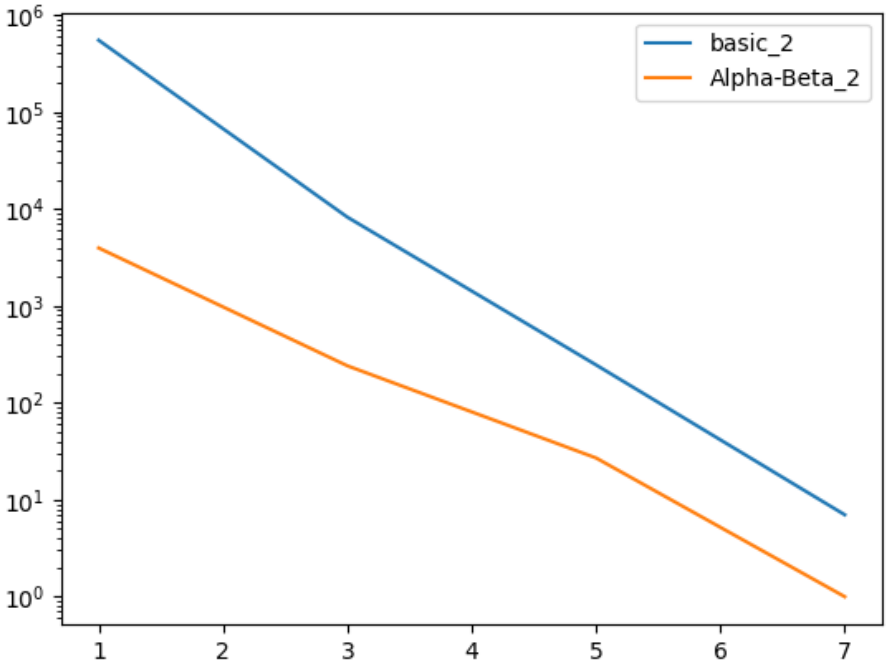

Figure 3.2: Ilość sprawdzonych węzłów w zależności od głębokości drzewa dla obu algorytmów w grze 2

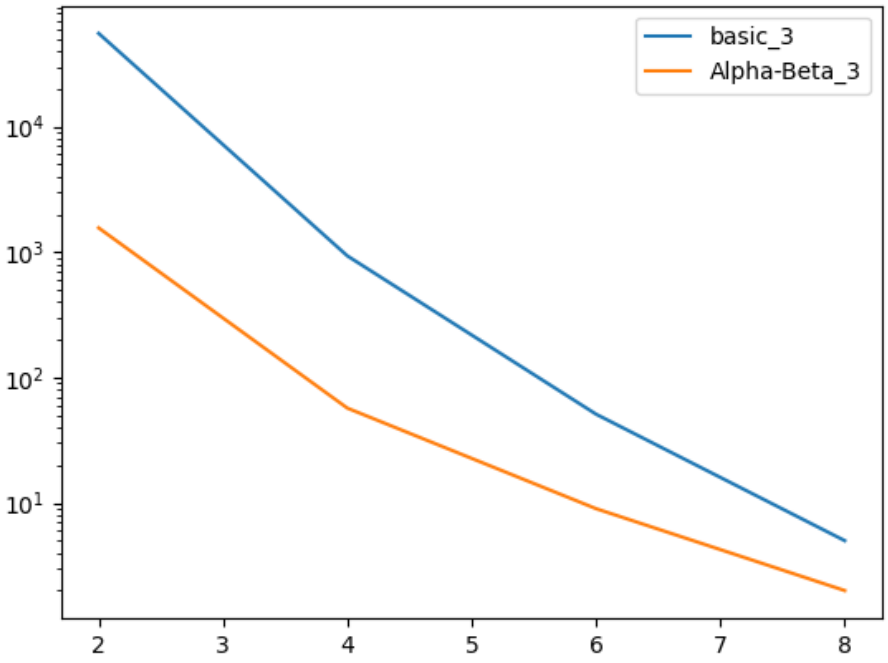

Figure 3.3: Ilość sprawdzonych węzłów w zależności od głębokości drzewa dla obu algorytmów w grze 3

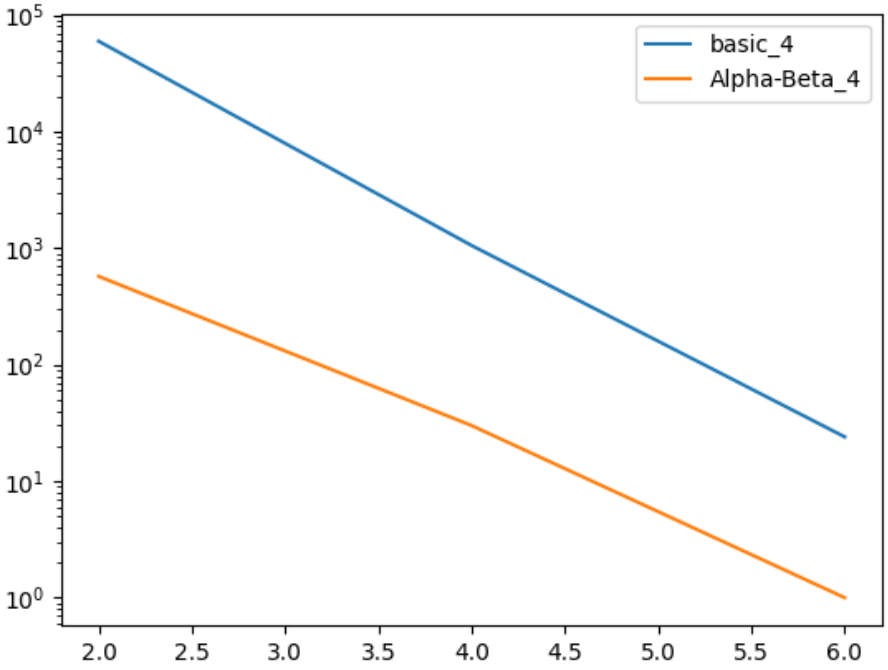

Figure 3.4: Ilość sprawdzonych węzłów w zależności od głębokości drzewa dla obu algorytmów w grze 4

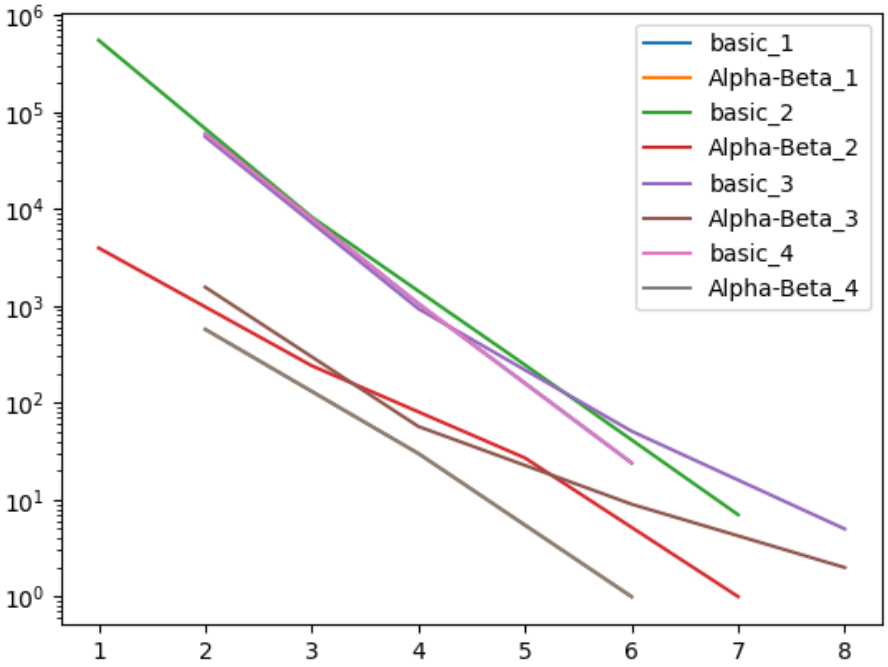

Figure 3.5: Zestawienie wszystkich wyników

4 Wnioski

Na wszystkich wykresach widoczna jest jasno przewaga algorytmu korzystającego z $\alpha - \beta$. jest on konsekwentnie szybszy o około 2 rzędy wielkości. Można z tego wywnioskować, że odrzucana jest zdecydowana większość ścieżek w drzewie. Na zestawieniu wszystkich gier, widać, że to kto zaczynał albo jak rozwijała się rozgrywka miało niewielki wpływ na szybkość działania algorytmu, w szczególności tego bez $\alpha - \beta$. Fluktuacje widoczne w sprawności algorytmu korzystającego z $\alpha - \beta$ mogą wynikać z tego, że w zależności od sytuacji na planszy większość ruchów może być niekorzystna co znacznie zmniejsza liczbę przeszukiwanych ścieżek. Zjawisko to nie występuję w algorytmie podstawowym gdyż niezależnie od czegokolwiek przeszukuje on całe poddrzewo.

List of Figures

3.1 Ilość sprawdzonych węzłów w zależności od głębokości drzewa 5
3.2 Ilość sprawdzonych węzłów w zależności od głębokości drzewa dla obu algorytmów w grze 2 8
3.3 Ilość sprawdzonych węzłów w zależności od głębokości drzewa dla obu algorytmów w grze 3 8
3.4 Ilość sprawdzonych węzłów w zależności od głębokości drzewa dla obu algorytmów w grze 4 9
3.5 Zestawienie wszystkich wyników 9