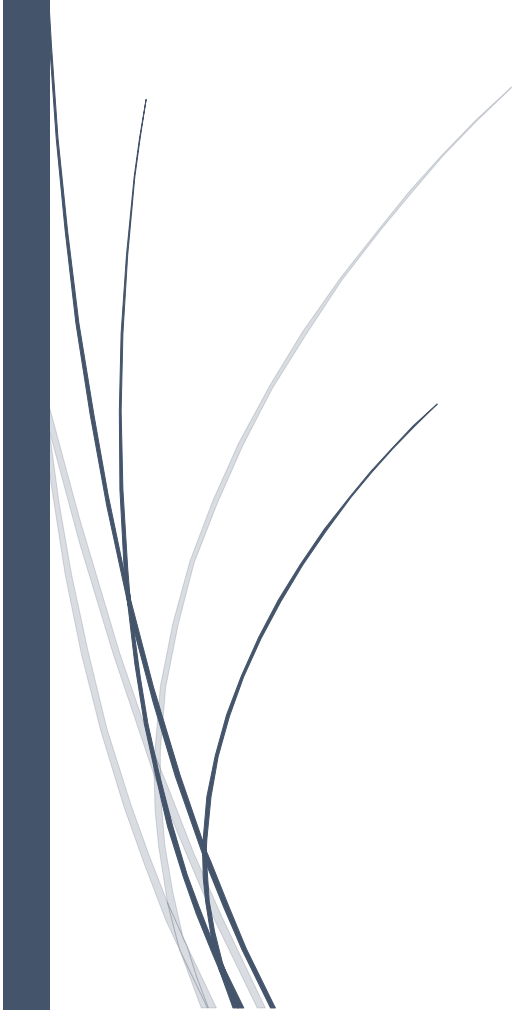
Algorytmy rozwiązywania gier o sumie zerowej

Sprawozdanie z ćwiczenia 3.

Sztuczna Inteligencja i Inżynieria Wiedzy - laboratorium



Spis treści

Connect4 – założenia implementacyjne

W celu badania zaimplementowano dwa algorytmy rozwiązywania gier o sumie zerowej. Rozważaną grą jest Connect4. Zaimplementowano algorytm *min-max* oraz *alfa-beta* z maksymalną głębokością przeszukiwania drzewa rozwiązań. Porównano także trzy funkcje oceny planszy.

Implementacja heurystyk oceny planszy

1. *(SimpleEvaluator)* Funkcja oceny przyznająca punkty tylko za planszę z wygraną (4 żetony jednego koloru w linii pionowej, poziomej lub ukośnej). Liczba punktów przyznawanych za taką planszę (ruch) była równa maksymalnej liczbie punktów za wygraną (przyjęłam za nią podwojoną maksymalną głębokość przeszukiwania) pomniejszonej o głębokość – liczbę ruchów wymaganą do wygranej:

$$val = 2 * maxDepth - currentDepth$$

   W przypadku wygranej jednego z graczy plansza miała wartość *val*, w przypadku drugiego gracza - wartość przeciwną.

2. *(ThreeEvaluator)* Drugi sposób oceny przyznawał punkty za wygraną w sposób opisany w punkcie 1., jednak dodatkowo używał heurystyki do oceny planszy, jeżeli osiągnął maksymalną głębokość, a nie było wygranej. Przy ocenie planszy brał pod uwagę wystąpienia w linii długości 4 trzech żetonów jednego gracza oraz pustego pola (na dowolnej pozycji z tych czterech). Jeżeli znalazł w dowolnym miejscu taki ciąg, plansza otrzymywała 40% maksymalnej liczby punktów za wygraną, a więc

$$val = 0{,}8 * maxDepth$$

3. *(ThreeEvaluator v2)* Trzeci sposób oceny bazował na dwóch poprzednich i je ulepszał – w przypadku planszy z wygraną punkty były przyznawane jak w punkcie 1. Jednak jeżeli osiągnął maksymalną głębokość, a nie było wygranej, nie tylko szukał wystąpienia w linii długości 4 trzech żetonów jednego gracza oraz pustego pola, ale także sprawdzał liczbę pustych pól pod pustym polem z ciągu – potrzebnych do zapełnienia przed położeniem czwartego żetonu w ciągu. Plansza z takim ciągiem dostawała liczbę punktów o 1 mniejszą niż połowa maksymalnej liczby punktów (połowę punktów dostałaby plansza wygrana na maksymalnej głębokości) pomniejszoną o liczbę pustych pól pod brakującym żetonem. Gdyby liczba punktów spadła w ten sposób poniżej 0, przypisywano 1. W przypadku znalezienia na planszy większej liczby takich ciągów, plansza dostawała maksymalną znalezioną w ten sposób liczbę punktów.

$$val = \max\left(depth - 1 - emptyCells_i\right)$$

   Wymogiem badań był także losowy wybór pierwszego ruchu gracza rozpoczynającego.

Porównanie średniego czasu przetwarzania

Porównanie średniego czasu przetwarzania dla *min-max* i *alfa-beta*

Badanie 1: Porównanie średniego czasu przetwarzania dla min-max i alfa-beta przy prostej funkcji oceny w zależności od głębokości przeszukiwania

Cel badania: porównanie wzrostu czasu przetwarzania w zależności od głębokości przeszukiwania oraz wyboru algorytmu

Stałe w badaniu:

funkcja oceny: SimpleEval

Zmienne w badaniu:

algorytmy: min-max i alfa-beta

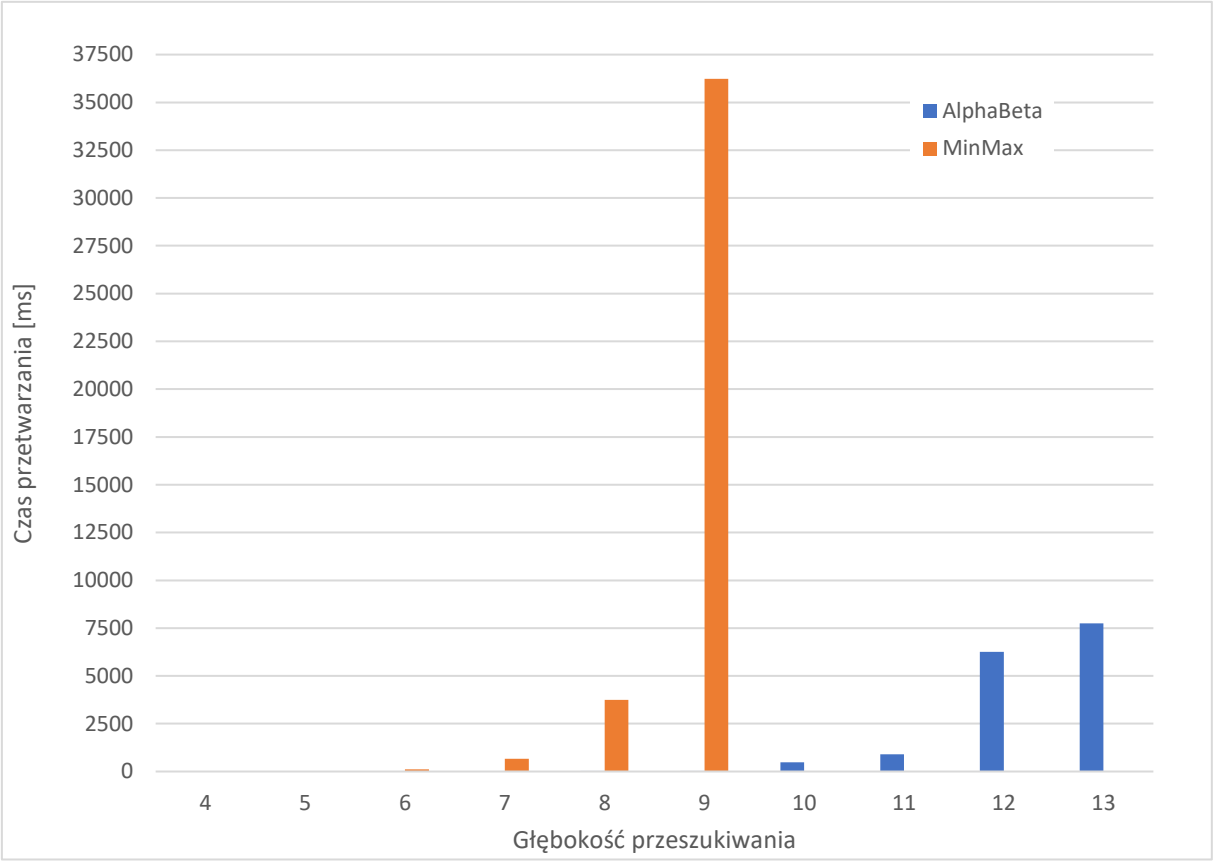
głębokość przeszukiwania: od 3 do 9 dla min-maxa i od 3 do 13 dla alfa-bety

Przebieg badania: Uruchomienie rozgrywki w trybie AI vs AI dla min-maxa z każdą wartości głębokości z zakresu wybranego dla min-maxa vs alfa-beta z każdą z wartości głębokości z zakresu dla alfa-bety. Powtórzyć dwa razy z zamianą gracza rozpoczynającego. Zapis danych o pierwszym ruchu, parametrach graczy, liczbie ruchów oraz średnim czasie przetwarzania gracza wygrywającego.

Wyniki:

Tabela 1 Średnie czasy przetwarzania jednego ruchu w ms - wyniki badania 1

Głębokość przeszukiwania	AlphaBeta	MinMax
4	0.67	2.29
5	0.00	20.38
6	1.57	117.50
7	1.86	659.19
8	33.00	3744.53
9	21.50	36225.50
10	482.79	
11	908.00	
12	6253.27	
13	7746.31	



Wykres 1 Średnie czasy przetwarzania w zależności od głębokości i algorytmu dla prostej ewaluacji planszy

Wnioski:

Użycie alfa-beta cięć znacząco zmniejsza czas przetwarzania algorytmu.

Badanie 2: Porównanie średniego czasu przetwarzania dla min-max i alfa-beta z użyciem heurystyk w zależności od głębokości przeszukiwania

Cel badania: porównanie wzrostu czasu przetwarzania w zależności od głębokości przeszukiwania oraz wyboru algorytmu

Zmienne w badaniu:

algorytmy: min-max i alfa-beta

głębokość przeszukiwania: od 3 do 8 dla min-maxa i od 3 do 12 dla alfa-bety

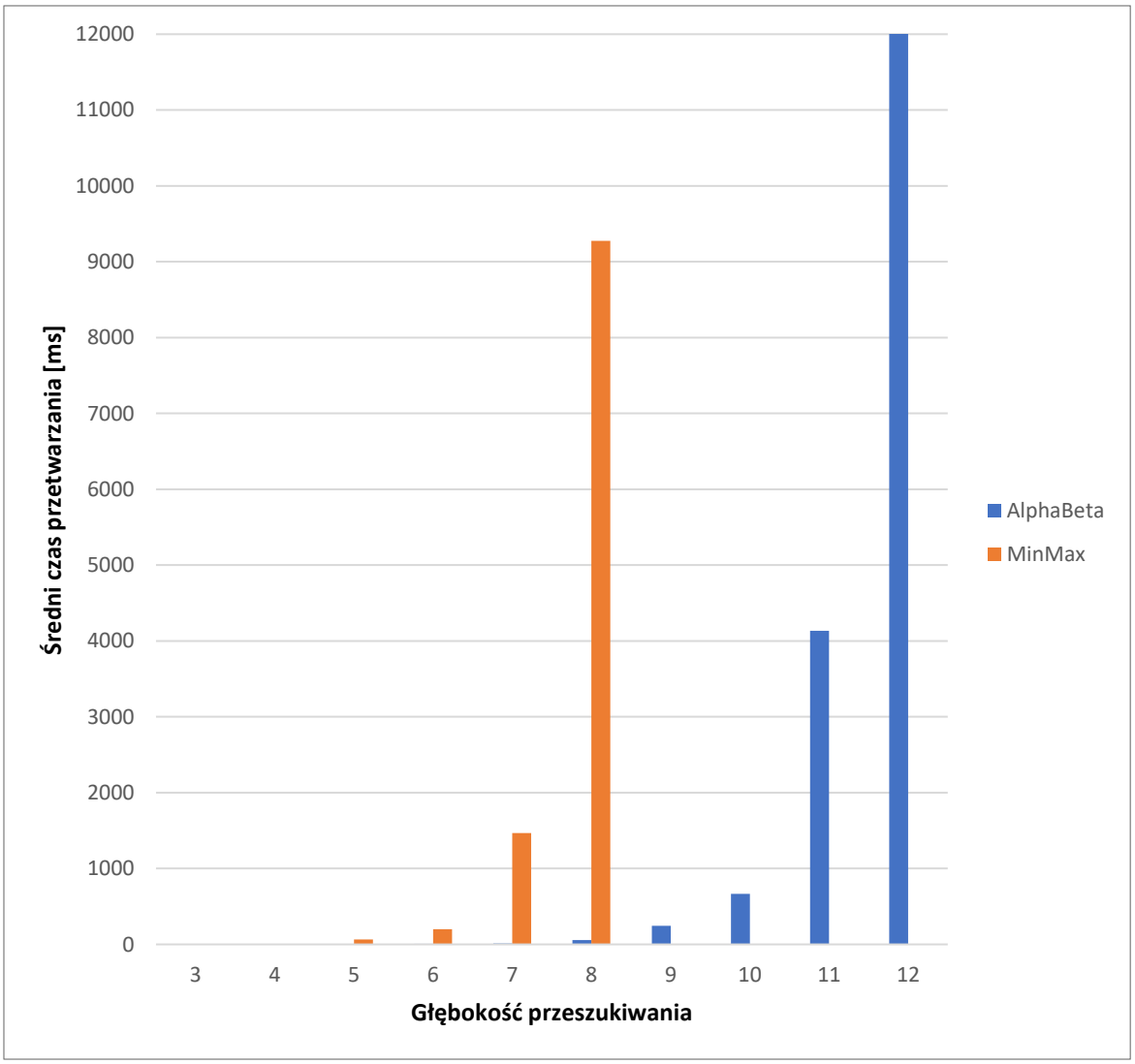
funkcje oceny: ThreeEvaluator i ThreeEvaluator v2

Przebieg badania: Uruchomienie rozgrywki w trybie AI vs AI dla min-maxa z każdą wartości głębokości z zakresu wybranego dla min-maxa vs alfa-beta z każdą z wartości głębokości z zakresu dla alfa-bety. Powtórzyć dwa razy dla każdej heurystyki i z zamianą gracza rozpoczynającego. Zapis danych o pierwszym ruchu, parametrach graczy, liczbie ruchów oraz średnim czasie przetwarzania gracza wygrywającego.

Wyniki:

Tabela 2 Średni czas przetwarzania w ms dla heurystyki ThreeEvaluator

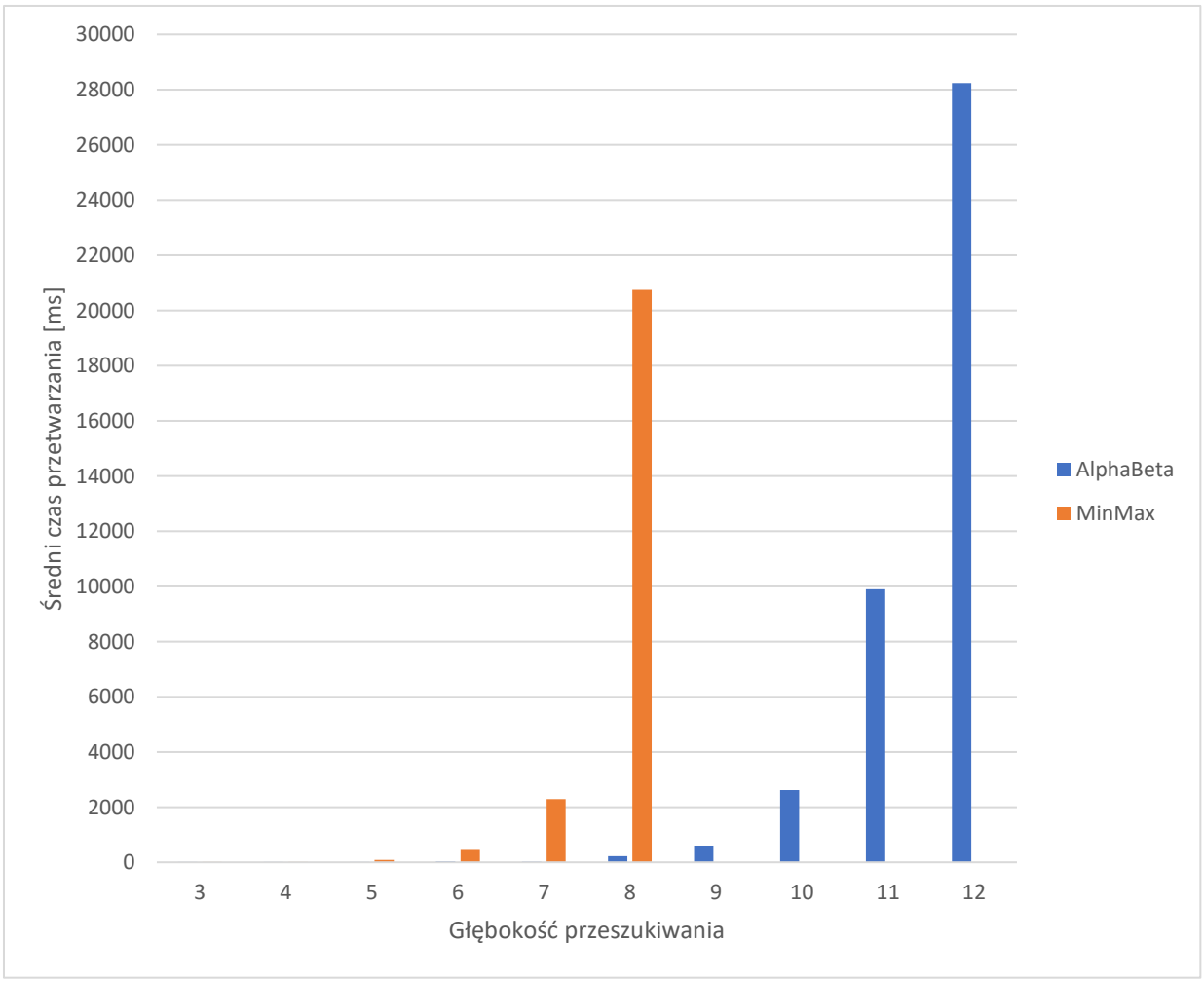
Głębokość przeszukiwania	AlphaBeta	MinMax
3	0.50	0.00
4	0.29	5.50
5	1.43	65.00
6	6.07	199.76
7	12.96	1466.58
8	55.06	9273.67
9	243.85	
10	666.98	
11	4132.72	
12	12058.33	



Wykres 2 Średnie czasy przetwarzania z użyciem ThreeEvaluator

Tabela 3 Średnie czasy przetwarzania w ms z użyciem ThreeEvaluator v2

Głębokość przeszukiwania	AlphaBeta	MinMax
3	0.60	1.00
4	1.00	8.13
5	2.33	92.80
6	27.50	447.62
7	25.41	2286.42
8	227.33	20741.39
9	602.18	
10	2614.99	
11	9895.62	
12	28238.43	



Wykres 3 Średnie czasy przetwarzania z użyciem ThreeEvaluator v2

Wnioski:

Użycie alfa-beta cięć znacząco zmniejsza czas przetwarzania algorytmu. Wraz ze zwiększaniem się głębokości przeszukiwania czas przetwarzania rośnie wykładniczo.

Porównanie średniego czasu przetwarzania w zależności od heurystyki

Badanie 3: Porównanie średniego czasu przetwarzania dla różnych funkcji oceny dla min-max i alfa-beta w zależności od głębokości przeszukiwania

Cel badania: porównanie czasu przetwarzania w zależności od funkcji oceny, głębokości i algorytmu

Zmienne w badaniu:

algorytmy: min-max i alfa-beta

głębokość przeszukiwania: od 3 do 8 dla min-maxa i od 3 do 12 dla alfa-bety

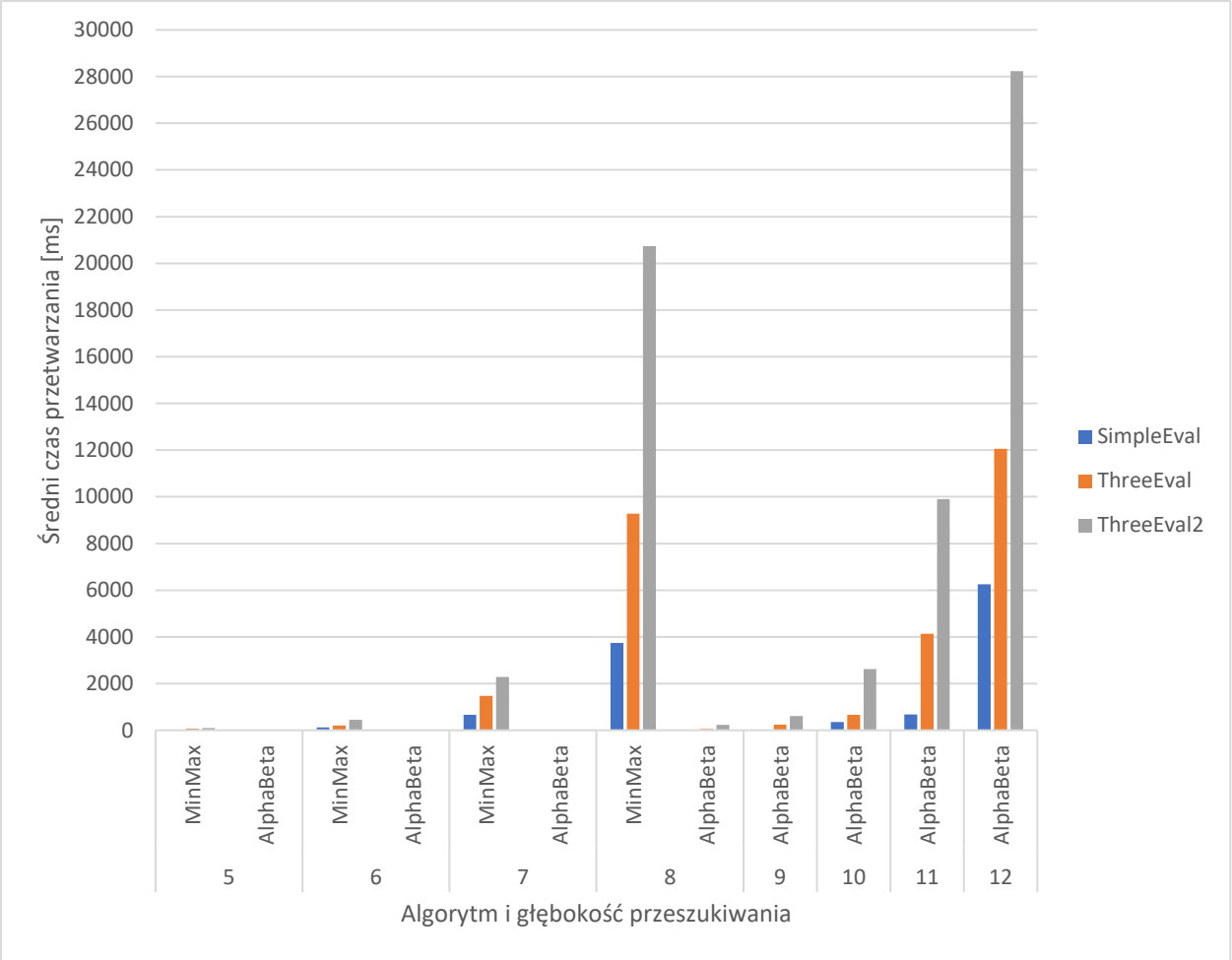
funkcje oceny: SimpleEvaluator, ThreeEvaluator i ThreeEvaluator v2

Przebieg badania: Uruchomienie rozgrywki w trybie AI vs AI dla min-maxa z każdą wartości głębokości z zakresu wybranego dla min-maxa vs alfa-beta z każdą z wartości głębokości z zakresu dla alfa-bety. Powtórzyć dwa razy dla każdej heurystyki i z zamianą gracza rozpoczynającego. Zapis danych o pierwszym ruchu, parametrach graczy, liczbie ruchów oraz średnim czasie przetwarzania gracza wygrywającego.

Wyniki:

Tabela 4 Średnie czasy przetwarzania w ms w zależności od funkcji oceniającej, głębokości i algorytmu

Głębokość przeszukiwania i algorytm	SimpleEval	ThreeEval	ThreeEval2	Średnia końcowa
4	**2.00**	**3.07**	**5.75**	**3.69**
AlphaBeta	1.00	0.29	1.00	0.62
MinMax	2.29	5.50	8.13	5.43
5	**15.18**	**30.77**	**49.95**	**31.17**
AlphaBeta	1.33	1.43	2.33	1.77
MinMax	20.38	65.00	92.80	51.38
6	**81.31**	**108.97**	**287.57**	**142.68**
AlphaBeta	1.70	6.07	27.50	9.94
MinMax	117.50	199.76	447.62	226.92
7	**476.75**	**460.23**	**493.21**	**479.08**
AlphaBeta	2.40	12.96	25.41	18.59
MinMax	659.19	1466.58	2286.42	1243.50
8	**1885.90**	**2513.36**	**4961.35**	**3638.32**
AlphaBeta	27.27	55.06	227.33	146.91
MinMax	3744.53	9273.67	20741.39	12017.71
9	**23.94**	**243.85**	**602.18**	**396.45**
AlphaBeta	23.94	243.85	602.18	396.45
10	**358.54**	**666.98**	**2614.99**	**1541.80**
AlphaBeta	358.54	666.98	2614.99	1541.80
11	**671.35**	**4132.72**	**9895.62**	**6656.44**
AlphaBeta	671.35	4132.72	9895.62	6656.44
12	**6253.27**	**12058.33**	**28238.43**	**14410.95**
AlphaBeta	6253.27	12058.33	28238.43	14410.95
Średnia końcowa	**823.88**	**1430.78**	**4002.91**	**2451.19**



Wykres 4 Średnie czasy przetwarzania w zależności od funkcji oceniającej, głębokości i algorytmu

Wnioski:

Najprostsza funkcja oceniająca przyznająca punkty tylko za wygraną ma najkrótszy czas przetwarzania, użycie heurystyk dodatkowo wydłuża ten czas. *ThreeEvaluator v2* jest najwolniejszy, dość znacząco potrafi wydłużać czas przetwarzania ruchu.

Badanie skuteczności algorytmów

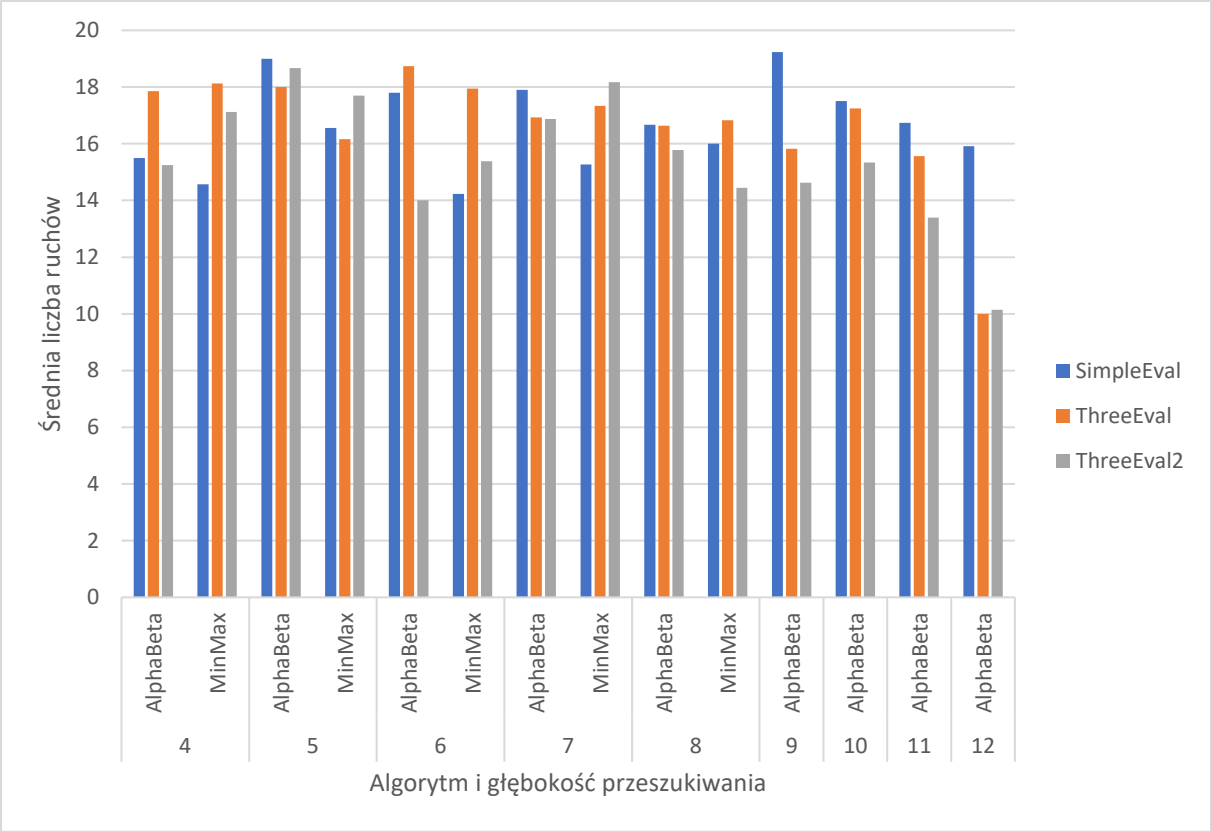
Porównanie liczby ruchów gracza wygrywającego w zależności od jego parametrów

Badanie 4: Zbadanie liczby ruchów gracza wygrywającego w zależności od jego parametrów

Cel badania: wybór najskuteczniejszych parametrów gracza
Zmienne w badaniu:
algorytmy: min-max i alfa-beta
głębokość przeszukiwania: od 3 do 8 dla min-maxa i od 3 do 12 dla alfa-bety
funkcje oceny: SimpleEvaluator, ThreeEvaluator i ThreeEvaluator v2
Przebieg badania: Uruchomienie rozgrywki w trybie AI vs AI dla min-maxa z każdą wartości głębokości z zakresu wybranego dla min-maxa vs alfa-beta z każdą z wartości głębokości z zakresu dla alfa-bety. Powtórzyć dwa razy dla każdej heurystyki i z zamianą gracza rozpoczynającego. Zapis danych o pierwszym ruchu, parametrach graczy, liczbie ruchów oraz średnim czasie przetwarzania gracza wygrywającego.
Wyniki:

Tabela 5 Zestawienie średniej liczby ruchów gracza wygrywającego

Głębokość i algorytm	SimpleEval	ThreeEval	ThreeEval2	Średnia końcowa
4	**14.78**	**18.00**	**16.50**	**16.69**
AlphaBeta	15.50	17.86	15.25	16.69
MinMax	14.57	18.13	17.13	16.70
5	**17.23**	**17.15**	**18.16**	**17.54**
AlphaBeta	19.00	18.00	18.67	18.55
MinMax	16.56	16.17	17.70	16.84
6	**15.34**	**18.31**	**14.86**	**16.34**
AlphaBeta	17.80	18.73	14.00	17.30
MinMax	14.23	17.94	15.38	15.73
7	**16.00**	**17.05**	**17.14**	**16.80**
AlphaBeta	17.90	16.93	16.87	17.01
MinMax	15.27	17.33	18.17	16.46
8	**16.33**	**16.69**	**15.47**	**16.00**
AlphaBeta	16.67	16.64	15.78	16.17
MinMax	16.00	16.83	14.44	15.60
9	**19.24**	**15.82**	**14.63**	**15.74**
AlphaBeta	19.24	15.82	14.63	15.74
10	**17.50**	**17.25**	**15.34**	**16.38**
AlphaBeta	17.50	17.25	15.34	16.38
11	**16.74**	**15.56**	**13.39**	**14.59**
AlphaBeta	16.74	15.56	13.39	14.59
12	**15.91**	**10.00**	**10.14**	**13.14**
AlphaBeta	15.91	10.00	10.14	13.14
Średnia końcowa	**16.57**	**16.76**	**15.21**	**16.01**



Wykres 5 Zestawienie średniej liczby ruchów gracza wygrywającego

Wnioski:

Na podstawie zebranych danych ciężko jest wyciągnąć jednoznaczne wnioski. Możemy zobaczyć, że powyżej głębokości 10 alfa-beta staje się bardziej skuteczny. Bardzo dużo zależy prawdopodobnie także od przeciwnika, nie jesteśmy także w stanie zawsze przewidzieć zwycięzcy, aby mieć jak najbardziej Do kolejnego badania jednak zawężę AI biorące w nim udział.

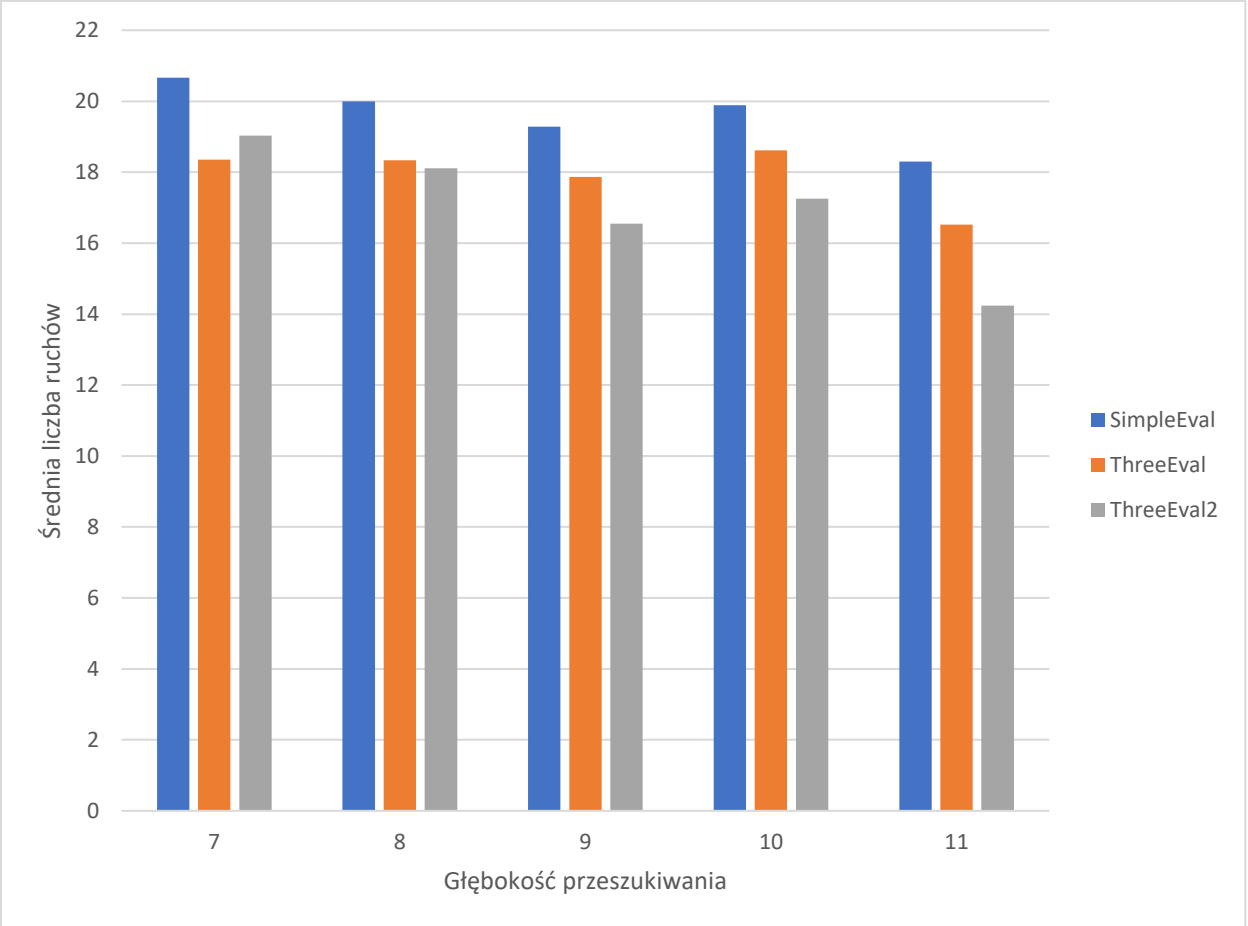
Badanie 5: Zbadanie liczby ruchów gracza wygrywającego w zależności od funkcji oceniającej

Cel badania: znalezienie najskuteczniejszej funkcji oceniającej
Stałe w badaniu:
algorytm: alfa-beta
Zmienne w badaniu:
głębokość przeszukiwania: od 7 do 11
funkcje oceny: SimpleEvaluator, ThreeEvaluator i ThreeEvaluator v2
Przebieg badania: Uruchomienie rozgrywki w trybie AI vs AI dla alfa-beta z każdą z wartości głębokości z zakresu „każdy z każdym”. Powtórzyć dwa razy dla każdej heurystyki i z zamianą gracza rozpoczynającego. Zapis danych o pierwszym ruchu, parametrach graczy, liczbie ruchów oraz średnim czasie przetwarzania gracza wygrywającego.

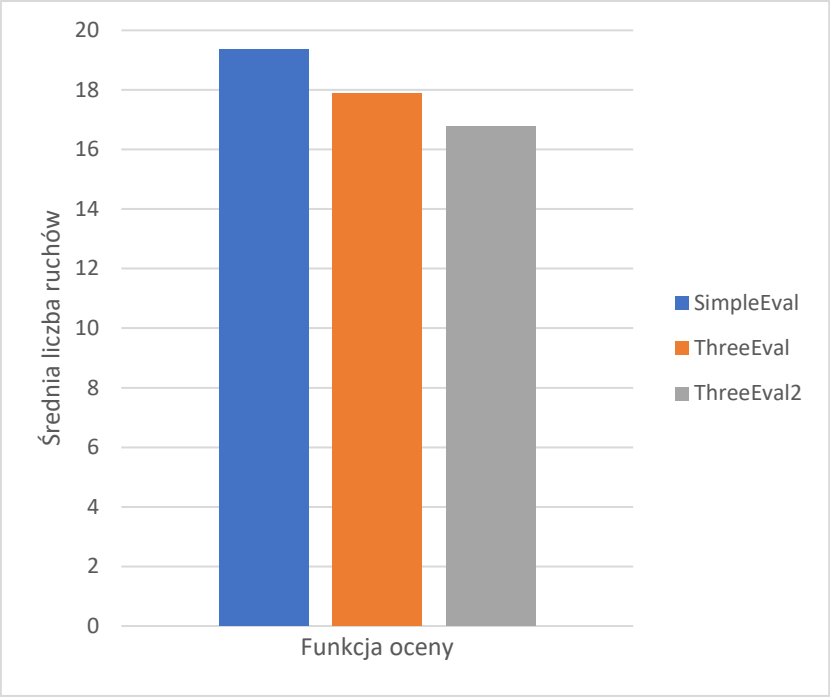
Wyniki:

Tabela 6 Średnia liczba ruchów wykonana przez gracza wygrywającego alfa-beta w zależności od jego parametrów

Głębokość	SimpleEval	ThreeEval	ThreeEval2
7	20.67	18.36	19.03
8	20.00	18.33	18.11
9	19.29	17.87	16.55
10	19.89	18.62	17.25
11	18.30	16.52	14.24
Średnia końcowa	**19.34**	**17.88**	**16.79**



Wykres 6 Średnia liczba ruchów wykonana przez gracza wygrywającego alfa-beta w zależności od jego parametrów



Wykres 7 Porównanie liczby ruchów gracza wygrywającego w zależności od wyboru heurystyki

Wnioski:

Na podstawie danych Tabeli 6, a szczególnie ich wizualizacji na Wykresie 7, można stwierdzić, że SimpleEval jest najprostszy, ma najkrótszy czas przetwarzania, ale jest słabszy i potrzebuje więcej ruchów na pokonanie przeciwnika. ThreeEvaluator v2 ma najdłuższy czas przetwarzania, ale jest najbardziej skuteczny.

Porównanie liczby ruchów gracza wygrywającego w zależności od przeciwnika

Badanie 6: Zbadanie liczby ruchów gracza wygrywającego w zależności od parametrów przeciwnika

Cel badania: porównanie, jak długo gracz jest w stanie odpierać ataki, w zależności od parametrów

Zmienne w badaniu:

algorytmy: random, min-max i alfa-beta

głębokość przeszukiwania: od 3 do 9 dla min-maxa i od 3 do 13 dla alfa-bety

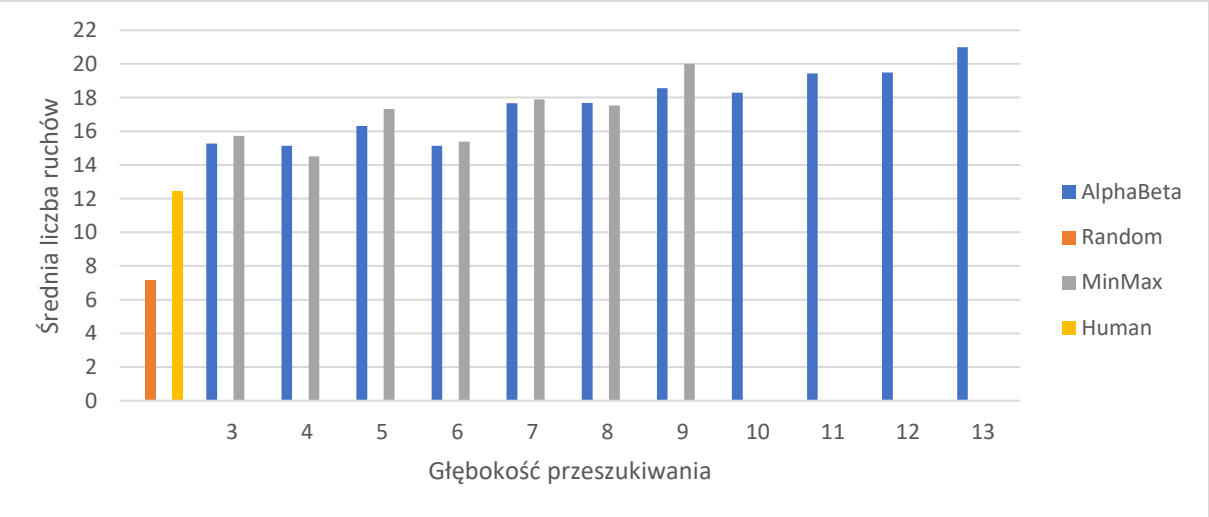
funkcje oceny: SimpleEvaluator, ThreeEvaluator i ThreeEvaluator v2

Przebieg badania: Uruchomienie rozgrywki w trybie AI vs AI dla min-maxa z każdą wartości głębokości z zakresu wybranego dla min-maxa vs alfa-beta z każdą z wartości głębokości z zakresu dla alfa-bety. Rozegrać także rozgrywki z graczem losowym. Powtórzyć dwa razy dla każdej heurystyki i z zamianą gracza rozpoczynającego. Zapis danych o pierwszym ruchu, parametrach graczy, liczbie ruchów oraz średnim czasie przetwarzania gracza wygrywającego.

Wyniki:

Tabela 7 Liczba ruchów gracza wygrywającego w zależności od algorytmu i głębokości przeszukiwania przeciwnika

Głębokość przeszukiwania	AlphaBeta	MinMax	Random
			7.13
3	15.26	15.72	
4	15.14	14.52	
5	16.31	17.31	
6	15.13	15.39	
7	17.68	17.90	
8	17.68	17.53	
9	18.56	20.00	
10	18.30		
11	19.43		
12	19.50		
13	21.00		



Wykres 8 Średnia liczba ruchów gracza wygrywającego w zależności od przeciwnika

Wnioski:

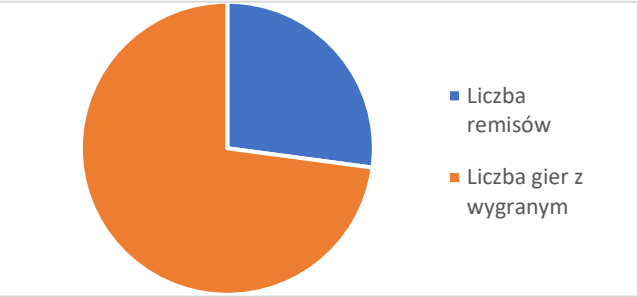
Na podstawie danych z Tabeli 7 można zauważyć, że zazwyczaj im większa głębokość przeszukiwania, tym algorytm „mądrzejszy" i dłużej będzie w stanie się bronić. Na wykresie 8 jako ciekawostkę zestawiono wyniki z przeciwnikiem wybierającym kolumny w sposób losowy oraz kilkoma testami przeciwko grze z człowiekiem.

Inne wnioski

W ramach badań 1-6 oraz dodatkowego losowego generowania rozgrywek zebrano dane o 1258 rozgrywkach.

Tabela 8 Stosunek liczby remisów do wszystkich gier

Liczba remisów	341
Liczba gier z wygranym	917
Liczba wszystkich gier	1258

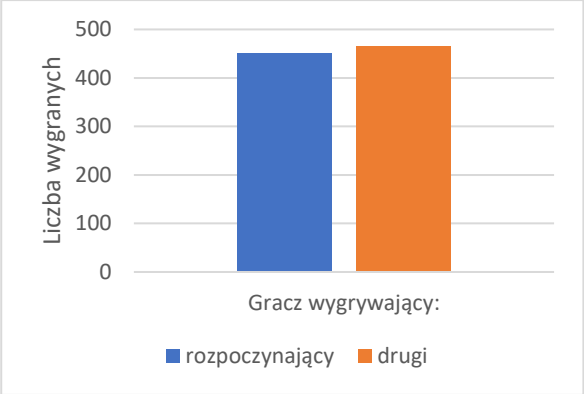


Wykres 9 Stosunek liczby remisów do wszystkich gier

Co ciekawe, to który gracz jest graczem rozpoczynającym, zdaje się nie mieć wpływu na jego szanse wygranej.

Tabela 9 Liczba wygranych gracza rozpoczynającego oraz drugiego w kolejce

Gracz wygrywający:	
Rozpoczynający	451
Drugi	466



Wykres 10 Liczba wygranych gracza rozpoczynającego oraz drugiego w kolejce

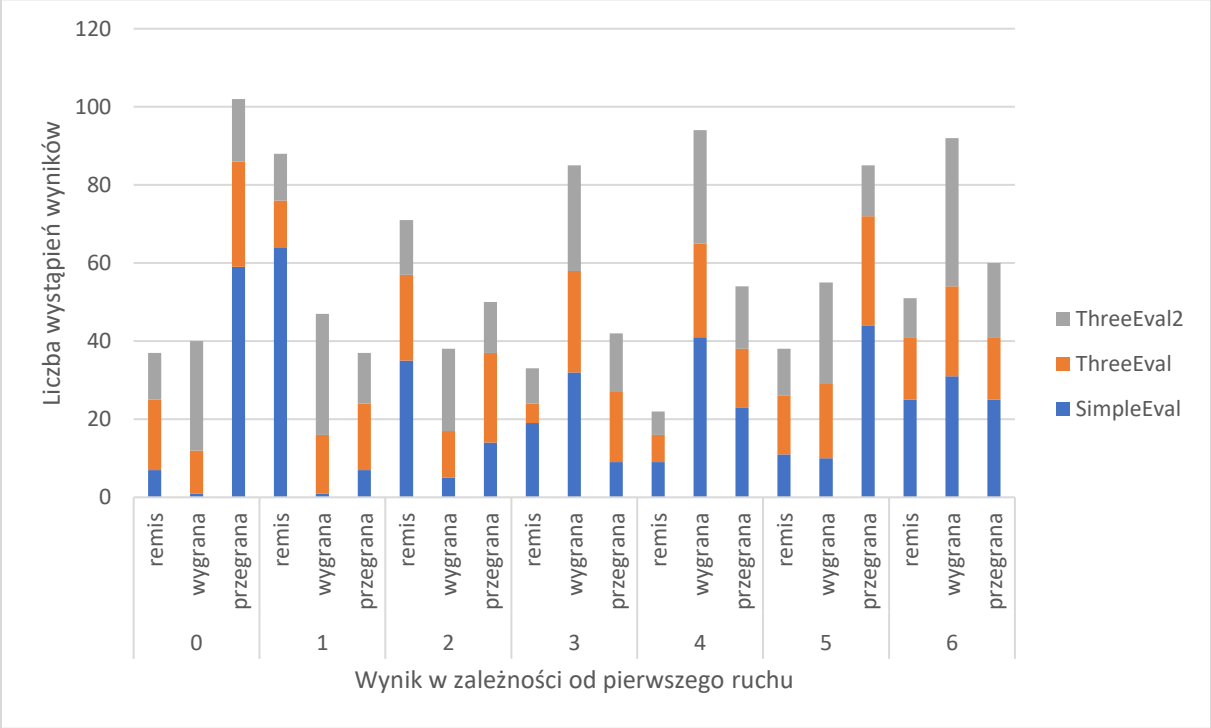
Wylosowany pierwszy ruch

Postanowiłam także sprawdzić, jak kończyły się gry dla gracza rozpoczynającego, w zależności od losowo przydzielonego pierwszego ruchu.

Tabela 10 Wyniki rozgrywki w zależności of pierwszego ruchu oraz funkcji oceny

Wynik gry w zależności od pierwszego ruchu	**SimpleEval**	**ThreeEval**	**ThreeEval2**	**Suma końcowa**
0	**67**	**56**	**56**	**179**
remis	7	18	12	37
wygrana	1	11	28	40
przegrana	59	27	16	102
1	**72**	**44**	**56**	**172**
remis	64	12	12	88
wygrana	1	15	31	47
przegrana	7	17	13	37
2	**54**	**57**	**48**	**159**
remis	35	22	14	71
wygrana	5	12	21	38
przegrana	14	23	13	50
3	**60**	**49**	**51**	**160**
remis	19	5	9	33

wygrana	32	26	27	85
przegrana	9	18	15	42
4	**73**	**46**	**51**	**170**
remis	9	7	6	22
wygrana	41	24	29	94
przegrana	23	15	16	54
5	**65**	**62**	**51**	**178**
remis	11	15	12	38
wygrana	10	19	26	55
przegrana	44	28	13	85
6	**81**	**55**	**67**	**203**
remis	25	16	10	51
wygrana	31	23	38	92
przegrana	25	16	19	60



Wykres 11 Wyniki rozgrywki w zależności of pierwszego ruchu oraz funkcji oceny

Jak widać na Wykresie 11, największe szanse na przegraną są w przypadku startu od kolumny 0, szczególnie w przypadku SimpleEval (ta praktycznie nie jest w stanie wygrać startując od tego pola). Największe szanse na wygraną są w przypadku rozpoczęcia od kolumny 3, 4 oraz 6.

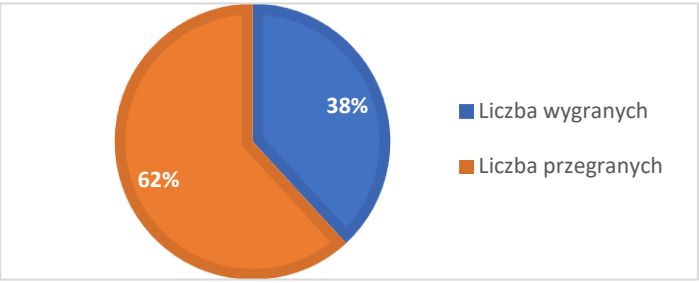
Pewne anomalie od matematycznych reguł w tym przypadku wynikają z tego, że ograniczając głębokość przeszukiwania drzewa nie mamy do czynienia z graczami grającymi optymalnie. Co więcej, w momencie, gdy algorytm nie znajdzie dobrego ruchu lub znajdzie kilka ruchów o takiej samej wartości, wybiera pierwszą znalezioną wartość.

Skuteczność funkcji oceny

Porównano także stosunek liczby wygranych gier do liczby przegranych gier dla każdego z trzech sposobów oceny planszy:

Tabela 11 Liczby wygranych i przegranych dla SimpleEval

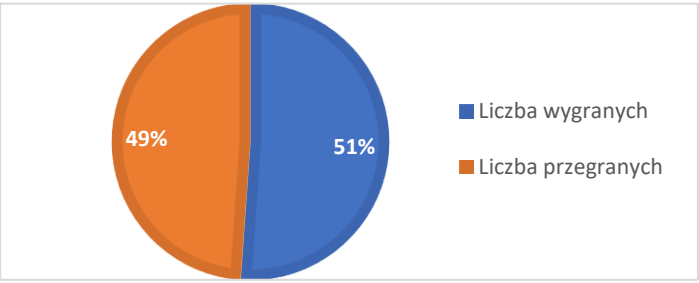
SimpleEval	
Liczba wygranych	229
Liczba przegranych	371
Liczba gier bez remisu:	600



Wykres 12 Stosunek wygranych do przegranych dla SimpleEval

Tabela 12 Liczby wygranych i przegranych dla ThreeEval

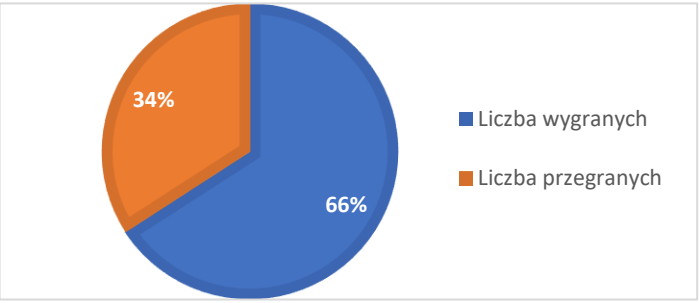
ThreeEval	
Liczba wygranych	278
Liczba przegranych	266
Liczba gier bez remisu:	544



Wykres 13 Stosunek wygranych i przegranych dla ThreeEval

Tabela 13 Liczby wygranych i przegranych dla ThreeEval v2

ThreeEval v2	
Liczba wygranych	405
Liczba przegranych	210
Liczba gier bez remisu:	615



Wykres 14 Stosunek wygranych i przegranych dla ThreeEval v2

Podsumowanie

Na różne sposoby można implementować algorytmy rozwiązywania gier o sumie zerowej. Znacznym ulepszeniem algorytmu min-max jest metoda alfa-beta cięć, która znacząca skraca czas przetwarzania i pozwala na głębszą analizę drzewa rozwiązań. Bardzo istotny jest także dobór heurystyki do oceny planszy w razie osiągnięcia maksymalnej głębokości oraz sposób przyznawania punktów.