Jakub Kowalczyk Ćwiczenie 3. – Algorytm Minimax

Celem eksperymentu było zbadanie działania algorytmu Minimax z obcinaniem alpha-beta na przykładzie gry Pick34. Różnice w rozgrywce sztucznej inteligencji zachodziły dla różnych głębokości algorytmu dla konkretnych graczy.

Decyzje projektowe

Algorytm został zaimplementowany zgodnie z podanym pseudokodem. Jako obowiązkowy parametr przyjmuje obiekt stanu gry (PickState), który dostarcza informacji o aktualnym przebiegu rozgrywki. Pozostałe parametry są domyśle i ustawione optymalnie, poza głębokością, którą należy manipulować.

Funkcja play_game() przeprowadza symulowaną rozgrywkę, w której gracz Max gra optymalnie przy założeniu wyboru najlepszych ruchów przez gracza Min. Każdy gracz posiada wybraną głębokość algorytmu.

Funkcja heurystyczna nr 1 opiera się na 4 założeniach:

1. Suma, do której dążymy w 4 liczbach to 34/4 = 8.5, zatem liczby najbliżej tej wartości są lepiej punktowane.
2. Jeżeli przeciwnikowi brakuje jednej konkretnej liczby do wygranej, to gracz musi ją wybrać, aby zablokować jego wygraną.
3. Jeżeli graczowi brakuje jednej konkretnej liczby do wygranej, to powinien ją wybrać
4. Jeżeli oceniany stan jest wygraną grą to otrzymuje maksymalną liczbę możliwych punktów.

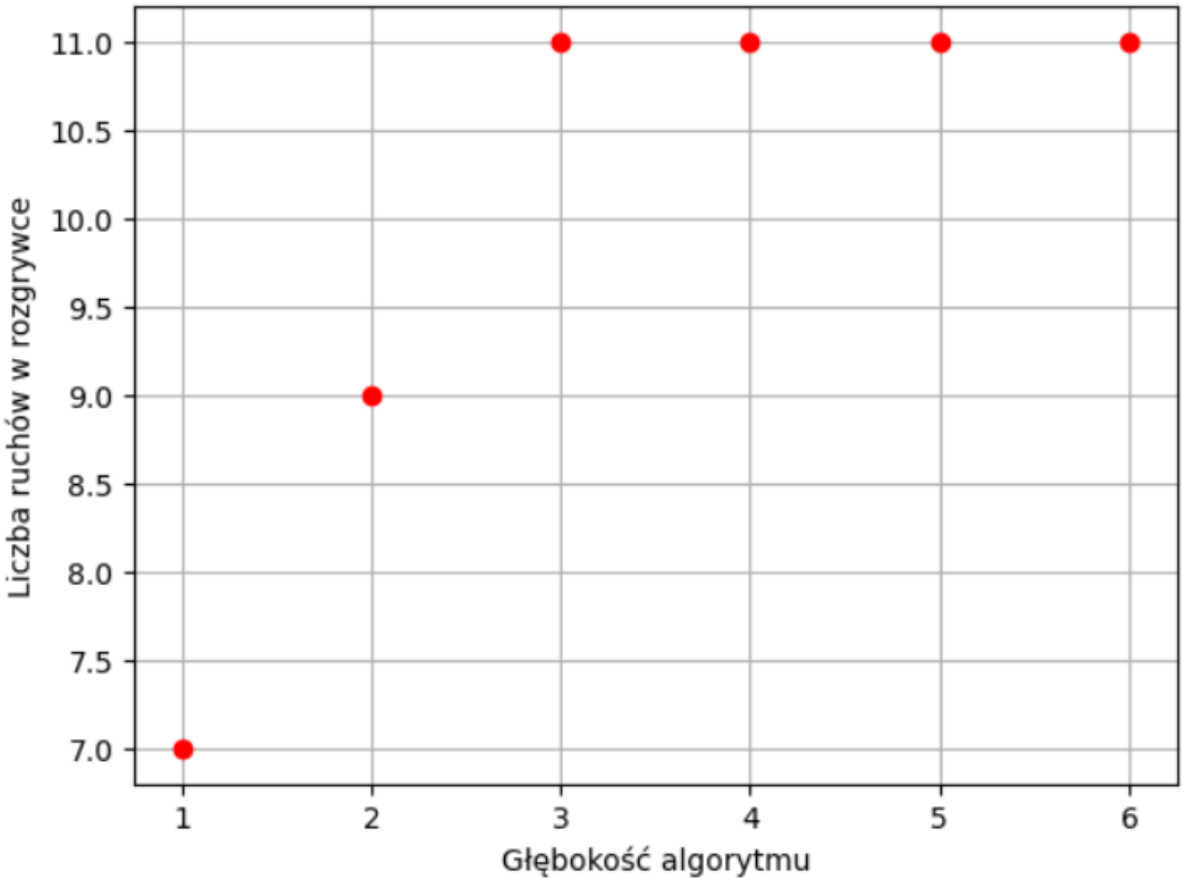
Funkcja heurystyczna nr 2 jest znacznie prostsza i punktuje ruchy gracza względem częstotliwości ich występowania we wszystkich możliwych kombinacjach sumy 34.

Oczekiwane wyniki

Spodziewam się, że wraz ze wzrostem głębokości, algorytm powinien być „mądrzejszy”, a tym samym wygrywać grę.

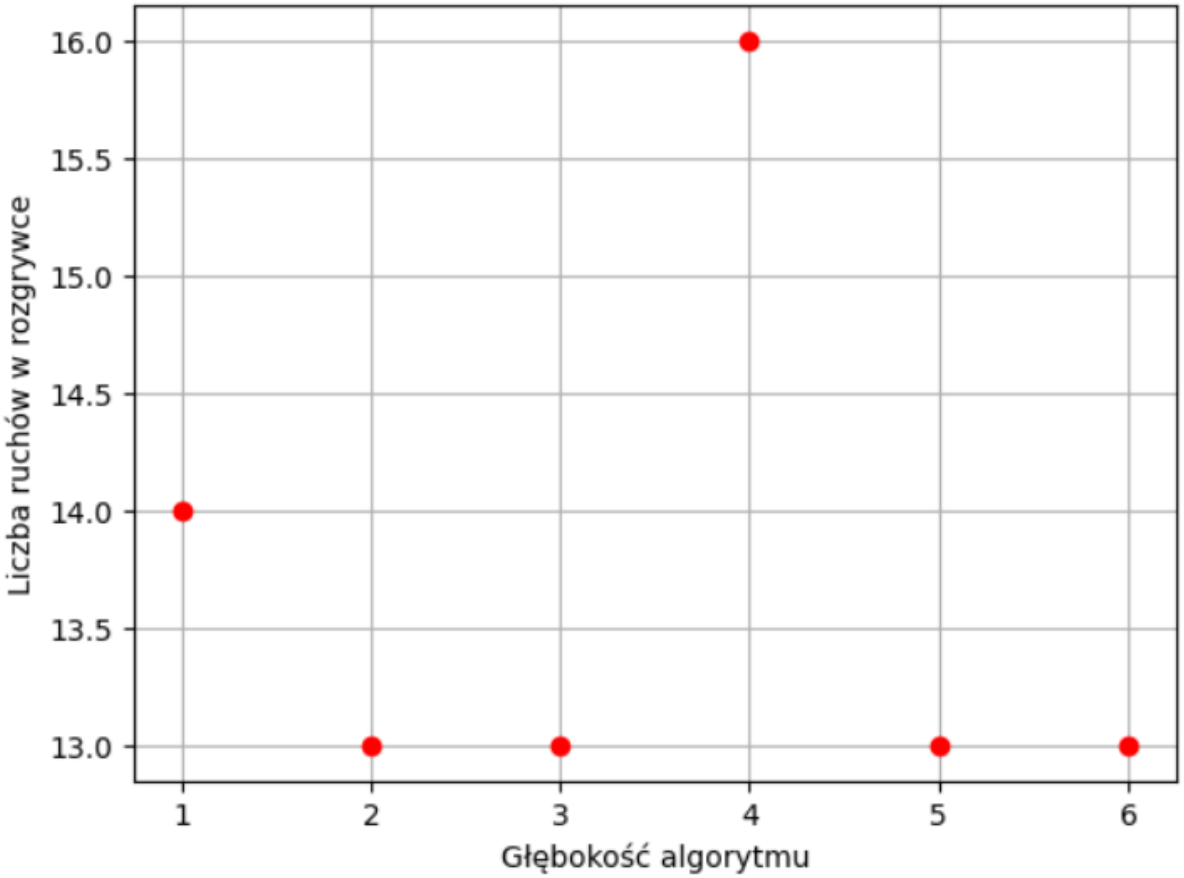
Wyniki i komentarze

Najpierw zbadam przypadek długości rozgrywki dla pierwszej funkcji heurystycznej, która rozpatruje blokady przeciwnika. Jak widać na wykresie nr 1, im większa jest głębokość, tym więcej ruchów trwa rozgrywka. Przy głębokości 1 algorytm źle radzi sobie z przewidywaniem dalekiej przyszłości, dlatego skupia się jedynie na swojej potencjalnej wygranej, zapominając o blokowaniu przeciwnika. Dla głębokości 2 następuje zaledwie 1 trafna blokada, a później gracz Max wygrywa. W pozostałych rozgrywkach obserwujemy 11 ruchów, co świadczy o bardziej zaawansowanym działaniu algorytmu.



Wykres 1: Funkcja - heuristic, gracze grają na tych samych głębokościach

W przypadku prostej funkcji heurystycznej oceniającej sumę punktów gracza, która nie rozpatrza blokowania przeciwnika, gra trwa dosyć długo. Na wykresie obserwujemy minimum 13 ruchów, a zdarzyła się nawet jedna remisowa rozgrywka o maksymalnej liczbie ruchów. W przypadku takiej funkcji heurystycznej trudno wyznaczyć zależność głębokości algorytmu od długości rozgrywki.



Wykres 2: Funkcja – heuristic2, gracze grają na tych samych głębokościach

Tabela wyników (wygrany) dla poszczególnych głębokości gracza Max i Min:

Max \ Min	1	2	3	4	5	6	7
1	Min	Min	Max	Min	Max	Max	Max
2	Max	Max	Max	Max	Max	Max	Max
3	Min	Min	Max	Min	Max	Max	Max
4	Max	Min	Max	Remis	Max	Remis	Max
5	Min	Min	Max	Min	Max	Max	Max
6	Max	Max	Max	Max	Max	Max	Max
7	Max	Max	Max	Max	Max	Max	Max

Wyniki z tabeli powyżej dotyczą funkcji heuristic2, która niestety nie jest najlepsza i z pewnych przyczyn nieco zaburza oczekiwanie wyniki. Dzieje się tak, ponieważ funkcja dla nieparzystej głębokości wybiera liczby nie najkorzystniejsze dla siebie, lecz analizuje najmniej korzystne dla przeciwnika. Jednak problem pojawia się, gdy przeciwnik posiada parzystą głębokość – wówczas on „gra pod siebie" wybierając przede wszystkim najbardziej punktowane dla niego liczby.

Problem ten dobrze dokumentuje szary fragment tabeli – gracz Min ma nieparzystą głębokość, dlatego Max wygrywa zawsze gdy jego głębokość jest <=5. Na zielonym fragmencie tabeli widać przewagę gracza Min (z racji na jego parzystą głębokość).

W niebieskim fragmencie tabeli gracz Max wygrywa prawie zawsze pomimo mniejszej głębokości – dzieje się tak, gdyż skupia się on na doborze dla siebie najkorzystniejszej wartości w bliskiej perspektywie, a Min przewiduje rozgrywkę na zbyt duża liczbę ruchów. Warto dodać, że w przypadku złożonej i dobrej funkcji heurystycznej nie byłby to oczywiście problem.

Pomarańczowy fragment tabeli jest zgodny z oczekiwaniami – dla głębokości powyżej 5 gracz Max zawsze wygrywa.

W niezamalowanym fragmencie tabeli widać omawiane wcześniej zależności funkcji heurystycznej – Dla parzystej głębokości Max i nieparzystej Min – wygrywa Max, zaś dla zbyt dużej nieparzystej głębokości Maxa – wygra Min.

Wnioski

1. Dobór funkcji heurystycznej wpływa na interpretację wyników dla różnych głębokości algorytmu Minimax.
2. Jeżeli funkcja heurystyczna jest złożona względem analizowania poczynań przeciwnika, to algorytm dla większych głębokości dłużej toczy rozgrywkę.
3. Wraz ze wzrostem głębokości, znacząco wzrasta czas wykonywania się algorytmu.
4. Dla głębokości > 5 w przypadku Pick34 algorytm wygrywa jako gracz Max.
5. Zbyt duże zagłębienie może zaburzyć efektywność algorytmu, jeżeli funkcja heurystyczna nie jest dobrze dobrana.