Algorytmy rozwiązywania gier o sumie zerowej

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z głównymi pojęciami gier – gra i rozgrywka, stan i przestrzeń stanów gry, strategia oraz drzewo i węzły drzewa gry, wypłata, przeszukiwanie drzewa gry, racjonalne działanie graczy jako podmiotów decyzyjnych za pomocą własnej implementacji dwuosobowej gry logiczno-strategicznej Connect 4 i zbadanie właściwości wykonanej implementacji tej gry.

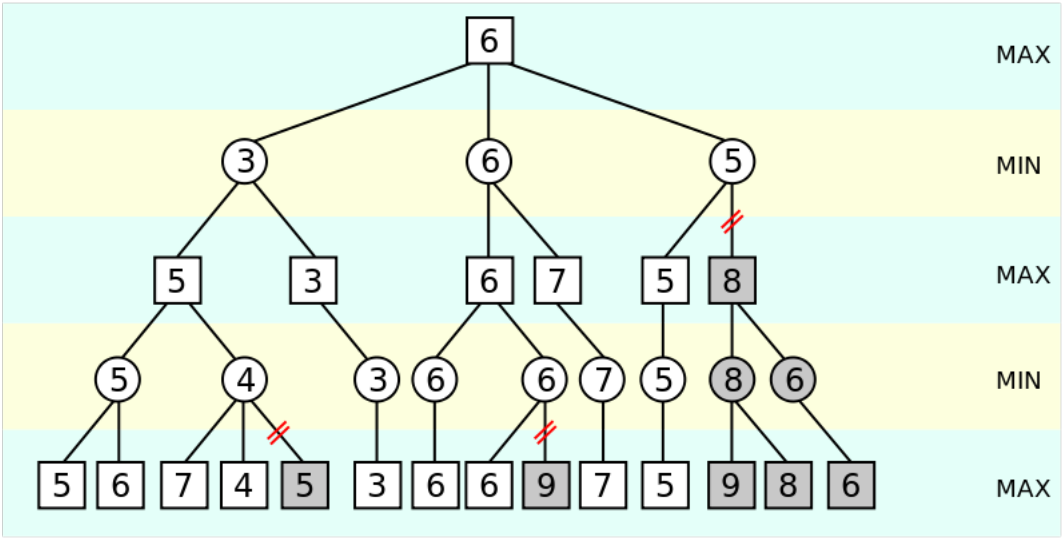
2. Wstęp

Gra o sumie stałej – gra, w której zysk jednego gracza oznacza stratę drugiego.

Gra o sumie zerowej – jest to szczególny jej przypadek gry o sumie stałej. Gra o sumie zerowej jest matematyczną reprezentacją sytuacji, w której zysk lub strata każdego uczestnika jest dokładnie taka sama jak strata lub zysk innych uczestników. Jeśli łączne zyski uczestników są sumowane, a łączne straty są odejmowane to wynik będzie równy zeru.

Metoda min-max - jest to metoda minimalizacji maksymalnych możliwych strat. Opiera się ona na teorii gry o sumie zerowej, obejmującej oba przypadki, ten, w którym gracze wykonują ruchy naprzemiennie i ten, w którym wykonują ruchy jednocześnie. W przypadku gry Connect4 gracze wykonują ruchy naprzemiennie. Algorytm tworzy drzewo rozgrywki i ocenia opłacalność poszczególnych ruchów. Dzięki przeanalizowaniu możliwych przebiegów rozgrywki jest w stanie określić, które ruchy są bardziej opłacalne od innych.

Metoda Alfa-Beta – jest to algorytm przeszukujący, redukujący liczbę węzłów, które muszą być rozwiązywane w drzewach przeszukujących przez algorytm min-max. Znajduje on zawsze te same rozwiązania co algorytm min-max, jednak robi to szybciej. W związku z tym algorytm Alfa-Beta jest niegorszy od algorytmu min-max. Korzyść płynąca z algorytmu alfa-beta leży w fakcie, że niektóre gałęzie drzewa przeszukiwania mogą zostać odcięte. Czas przeszukiwania ograniczony zostaje do przeszukania najbardziej obiecujących poddrzew, w związku z czym możemy zejść głębiej w tym samym czasie.

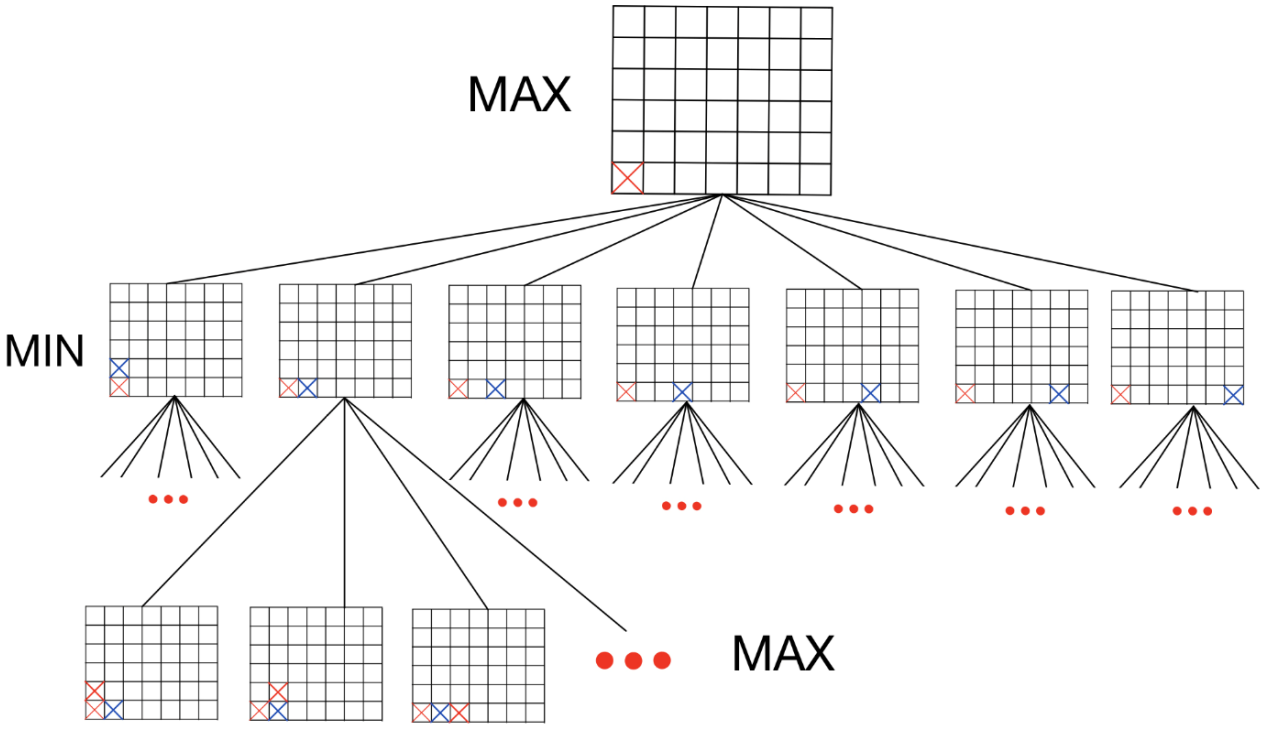


Rys. 1. Porównanie drzewa przeszukiwań rozwiązania dla algorytmu min-max i alfa-beta. Wyszarzone elementy są gałęziami odrzuconymi przez algorytm alfa-beta.

Zaznaczone na szaro poddrzewa (rys. 1) nie muszą być przeszukiwane, ponieważ wiemy, że nie wpłyną na poprawę wartości węzła leżącego powyżej odcięcia, toteż ich odrzucenie nie wpłynie na ostateczny ich wynik.

Cechy wspólne obu algorytmów dla gry Connect 4 :

- gracz wykonuje najbardziej korzystny dla niego ruch
- przeciwnik wykonuje zawsze najbardziej niekorzystny dla gracza ruch



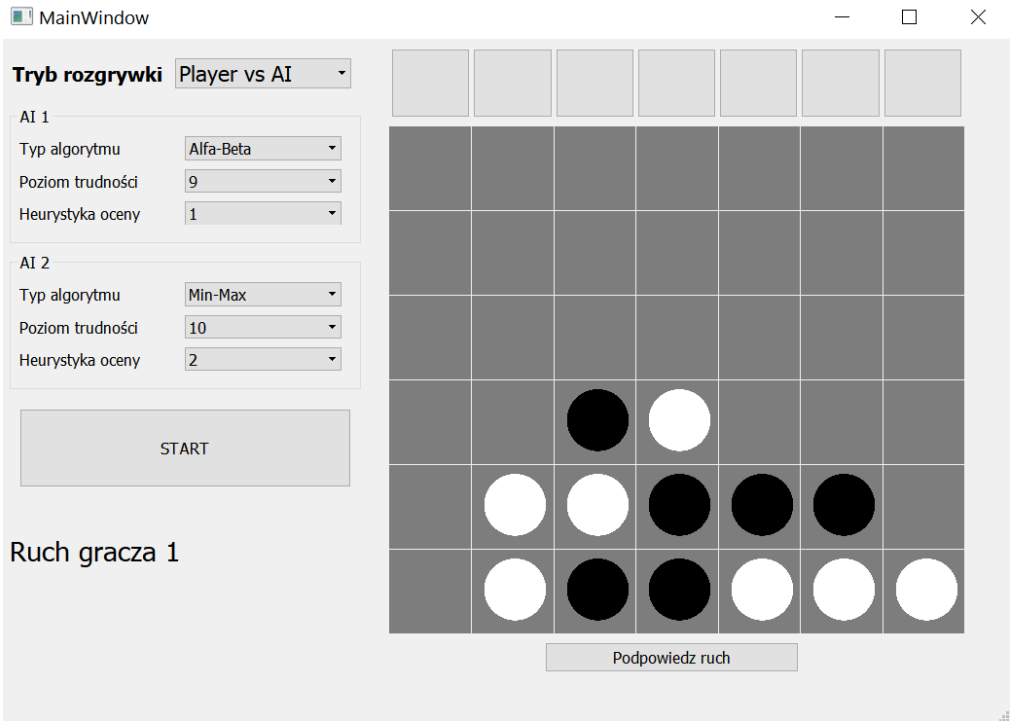
Rys. 2. Przykład drzewa rozgrywki w algorytmie min-max dla gry Connect 4

Na rysunku 2 przedstawiono schemat drzewa rozgrywki dla gry Connect 4. W przykładzie gracz ma wykonać pierwszy ruch. Przed jego ruchem plansza jest pusta. Algorytm wykonuje pierwszy ruch i jego celem jest maksymalizacja wyniku gry. Następnie uruchomione zostaje drzewo gry i sprawdzane są możliwe odpowiedzi przeciwnika. Ruch przeciwnika ma być jak najbardziej niekorzystny dla gracza – stąd też zastosowano tutaj funkcje minimalizującą. Po ruchu gracza następuje ponowne sprawdzenie wszystkich możliwości w taki sposób aby ruch był jak najbardziej korzystny dla gracza. Algorytm stopniowo schodzi w dół tak długo aż któryś z graczy nie wygra lub nie zostanie osiągnięta maksymalna głębokość drzewa.

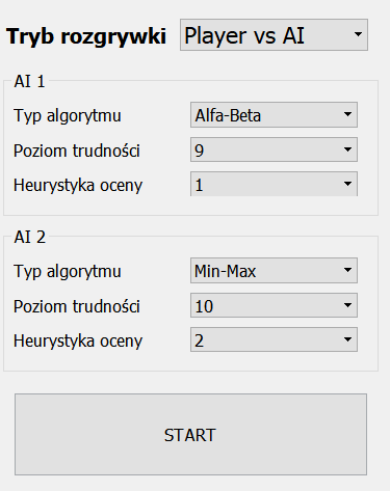
3. Opis projektu i analiza wyników

W ramach projektu zrealizowano następujące zadania:

1. Trening w rozgrywkach w Connect 4
    - zbudowano planszę gry oraz przestrzeń stanów rozgrywki,
    - zaimplementowano logikę wykonywania ruchów oraz funkcje sprawdzania stanu planszy,
    - umożliwiono prowadzenie rozgrywki między dwoma użytkownikami.
2. Dodano do silnika gry tryb pracy z algorytmem min-max oraz umożliwiono rozgrywkę w trybie użytkownik-AI.
3. Dodano do silnika gry tryb pracy z algorytmem alfa-beta cięć.
4. Uzupełniono implementacje kontrolera gry do celu rozgrywek w trybie AI vs AI z odpowiednią wizualizacją stanów planszy oraz ogłaszania wyników rozgrywki i losowaniem pierwszego ruchu.
5. Stworzono interfejs graficzny GUI wraz z funkcjonalnością gry człowiek-AI.
6. Sformułowano dla gry 3 heurystyk dla funkcji oceny stanów planszy w grze.
7. Opracowano dokumentacje porównawczą wraz z badaniami wydajności.



Rys. 3. Główne okno programu

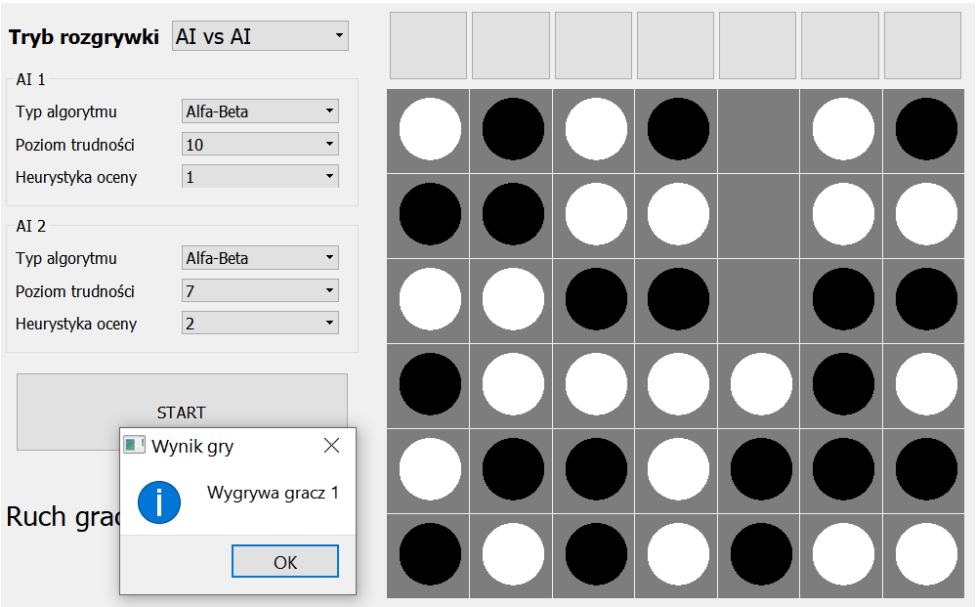


Rys. 4. Panel kontroli rozgrywki

W ramach projektu zaimplementowano trzy typy rozgrywki:

- użytkownik kontra użytkownik
- użytkownik kontra AI
- AI kontra AI

Wszystkie wymienione funkcjonalności przedstawione zostały na rysunkach 3 i 4. Użytkownik może wybrać jaki typ algorytmu ma zostać użyty przez AI oraz jaki ma być poziom gry przeciwnika. Poziom gry przeciwnika jest ustalany na zasadzie przydziału odpowiedniej ilości czasu i doboru maksymalnej głębokości drzewa. W programie istnieje także możliwość wyboru jednej z trzech heurystyk oceny stanu planszy. Po każdym ruchu użytkownik może wprowadzić zmiany w trybie rozgrywki i parametrach AI. Dzięki temu rozgrywka może być na bieżąco modyfikowana i kontrolowana. Stan rozgrywki jest także na bieżąco wizualizowany. Na rysunku 5 pokazano wynik przykładowej rozgrywki w trybie AI kontra AI dla algorytmu alfa-beta i heurystyki 1 oraz 2 (patrz opis poniżej).



Rys. 5. Przykład rozegranej rozgrywki w trybie AI kontra AI. Wygrał gracz 1, któremu przydzielono więcej czasu obliczeniowego

W projekcie wykorzystano 3 różne typy działania algorytmów:

1. Wersja podstawowa drzewo budowane jest do pewnego określonego poziomu. Po osiągnięciu maksymalnego głębokości następuje ocena stanu planszy i wnioskowanie, który ruch jest najbardziej opłacalny. Ocena stanu planszy polegała na wyróżnieniu 3 stanów.
    - jeśli wygrywa AI, to funkcja zwraca wartość 1
    - jeśli wygrywa przeciwnik, to funkcja zwraca wartość -1
    - jeśli nikt nie wygrywa, to funkcja zwraca wartość 0

2. Wersja zmodyfikowana, w której po osiągnięciu maksymalnego poziomu głębokości następuje ocena stanu planszy poprzez wyróżnienie stanów:
    - jeśli wygrywa AI, to funkcja zwraca wartość wyliczoną na podstawie ilości ruchów w których następuje zwycięstwo. Im zwycięstwo następuje szybciej, tym funkcja ma większą wartość.
    - jeśli wygrywa przeciwnik, to funkcja zwraca wartość wyliczoną na podstawie ilości ruchów w których następuje porażka. Im porażka następuje szybciej, tym wartość funkcji jest mniejsza.
    - jeśli nie następuje ani zwycięstwo ani porażka to analizowany jest stan planszy z uwzględnieniem podwójnych niezabezpieczonych pól. Pola które nie sąsiadują z podwójnymi żetonami mają wartość 0. Pola, które stykają się z podwójnymi żetonami mają odpowiednio większe lub mniejsze wartości w zależności czyje są te żetony. Miejsca w których występują dwa niezabezpieczone żetony obok siebie są potencjalnymi punktami niebezpiecznymi.

3. Wersja ze zmniejszonym obszarem przeszukiwań. Algorytm zapamiętuje ostatni ruch przeciwnika i zawęża obszar planszy do określonej ilości pól wokół. Dzięki temu algorytmy mogą zejść znacznie głębiej w poszukiwaniach i dotrzeć praktycznie do każdego liścia drzewa. Tutaj ocena następuje podobnie jak w wersji pierwszej.

Wyniki działania poszczególnych algorytmów oraz ich wersji zestawiono w tabeli 1. Jak wynika z uzyskanych danych, algorytm alfa-beta zapewnia o wiele lepsze rezultaty niż algorytm min-max. Warto pamiętać, że stosowanie algorytmu alfa-beta zapewnia znalezienie takich samych rozwiązań w krótszym czasie. Nigdy nie jest on mniej opłacalny.

Maksymalna głębokość drzewa	Średnia liczba wywołań dla 5 pierwszych ruchów
5	16851
7	782610
9	brak rozwiązania
11	brak rozwiązania

Tabela 1. Wyniki testowania średniej ilości odwiedzonych węzłów drzewa dla algorytmu min-max dla pierwszych pięciu ruchów

Maksymalna głębokość drzewa	Średnia liczba wywołań dla 5 pierwszych ruchów
5	5651
7	21273
9	86319
11	953027

Tabela 2. Wyniki testowania średniej ilości odwiedzonych węzłów drzewa dla algorytmu alfa-beta dla pierwszych pięciu ruchów

Heurystyka	% wygranych	Średni czas obliczeń dla całej rozgrywki [s]
1	23	12.1
2	66	11.9
3	37	12.4

Tabela 3. Porównanie skuteczności heurystyk oceny stanu planszy względem algorytmu testowego**.**

W celu porównanie heurystyk oceny stanu planszy stworzono program testowy którego zadaniem było wyłonienie najlepszej heurystyki. Wyniki działania programów są całkowicie uzależnione od czasu obliczeń w związku z tym próbowano dobrać parametry procesu tak, aby średnia długość rozgrywki była taka sama dla wszystkich trzech metod. W tabeli 3 zestawiono średnią ilość zwycięstw dla 20 prób. Z testów wynika, że dla czasu rozgrywki około 12 s najlepiej sprawdza się heurystyka 2. Wyniki te jednak nie powinny być ostatecznym kryterium oceny skuteczności algorytmów. Wystarczy bowiem delikatnie wydłużyć czas obliczeń lub dobrać bardziej zaawansowanego przeciwnika, a wartości testu ulegną znacznym zmianom. Z przeprowadzonych symulacji i doświadczenia testującego wynika, że heurystyka numer 1 ma najmniejszą skuteczność. Jeśli chodzi o heurystyki 2 i 3, to są one komplementarne i w zależności od przebiegu rozgrywki mogą przynosić lepsze lub gorsze rezultaty. Najlepszym rozwiązaniem byłoby zastosowanie heurystyki hybrydowej, która w początkowej fazie gry zawężałaby pole przeszukiwań, a wraz z wypełnianiem się planszy rozszerzała obszar testowy.

4. Podsumowanie i wnioski

Dla wszystkich testowanych przypadków algorytm alfa-beta zachowywał się znacznie lepiej niż algorytm min-max. Najlepszą heurystyką okazał się model drugi który uwzględniał głębokość zejścia oraz obecność par żetonów. Najmniej wydajną heurystyką był model pierwszy. Ocena działania algorytmu jest bardzo ciężka i zależy od wielu czynników. Przykładowo, jeśli plansza jest pusta, to czas przeszukiwania jest o wiele większy niż w przypadku końcowej fazy gry. Czas przeszukiwania zmienia się także w znacznym stopniu w zależności od ułożenia żetonów na planszy. W początkowej fazie gry najlepiej jest analizować mały obszar schodząc bardziej w głąb. Gdy plansza jest pusta nie ma sensu sprawdzać wszystkich pól, ponieważ i tak nie ma możliwości dotarcia do żadnego liścia drzewa. Wraz z rozwojem gry coraz ważniejsze jest kontrolowanie dużego obszaru planszy. Przeszukiwania nie muszą być już bardzo wgłębne, ponieważ algorytm coraz częściej dochodzi do liści drzewa. Dodatkowo coraz więcej pól jest już zajęta i znacząco zmniejsza się ilość możliwych kombinacji ruchów. Podczas testów rozgrywki w trybie użytkownik kontra AI, największą rolę odgrywał czas obliczeń. Jeśli maksymalna głębokość drzewa ustawiona została na kilka ruchów to użytkownik miał

duże szanse na wygraną. W przypadki głębokości drzewa około 10, szanse na zwycięstwo z AI były niewielkie.