Politechnika Wrocławska

Projektowanie Algorytmów i Metod Sztucznej Inteligencji

Imię i nazwisko prowadzącego	dr inż. Łukasz Jeleń
Termin zajęć	środa, $11^{15} - 13^{00}$
Data oddania sprawozdania	5.06.2019

Temat projektu

Projekt 3 - Gry & AI

Imię i Nazwisko	**Indeks**
Anna Bielecka	241159

1 Wstęp

Głównym zadaniem projektu było opracowanie algorytmu wyznaczającego najlepszy ruch pionka w oparciu o wartość funkcji oceniającej. Do tego celu wykorzystany został standardowy algorytm Mini-Max.

Algorytm MinMax wywodzi się z twierdzenia o grze o sumie stałej. Oznacza to, ze jeżeli dwie osoby grają przeciwko sobie, to poprawa sytuacji jednego z graczy oznacza proporcjonalne pogorszenie się sytuacji gracza drugiego. Dzięki tej informacji można przy użyciu odpowiedniej funkcji heurystycznej przypisać danej sytuacji na planszy konkretną wartość (dodatnia, jeśli jest korzystna dla gracza, dla którego ją rozpatrujemy lub ujemna w przeciwnym wypadku).

2 Opis gry

W warcaby klasyczne gra się na planszy 8 na 8 pól. Każde pole może być: puste, zawierać czerwony pion, czarny pion, czerwoną lub czarną damkę. Do stworzenia planszy wykorzystano tablice typów wyliczeniowych (EMPTY, RED_PION, BLACK_PION, BLACK_PION, BLACK_PION). Klasa 'Plansza' posiada metody umożliwiające wygenerowanie wszystkich możliwych ruchów dla danego pionka lub całego koloru, sprawdzenie czy występuje możliwość bicia dla piona / koloru, wykonanie ruchu, funkcje oceniające sytuacje na planszy oraz funkcje sprawdzające stan gry. Do przechowania ruchów zaimplementowano dwie struktury: Coords, zawierajaca współrzedne piona przed i po wykonaniu ruchu oraz Move, zawierająca wszystkie współrzedne dla danego ruchu (kiedy składa się na przykład z kilku bić).
Graficzna reprezentacja gry została wykonana przy użyciu biblioteki SFML 2.5.1 w wersji x64, a tekstury wykonano przy użyciu programu Inkscape 0.92.4.

Zasady gry:

- gra w warcaby odbywa się jedynie na ciemnych polach, nazywanych aktywnymi polami,
- warcabnica musi być ustawiona między graczami w ten sposób, że każde pierwsze lewe pole dla każdego z graczy jest polem ciemnym,
- bicie jest obowiązkowe i ma pierwszeństwo przed wykonaniem innego ruchu (zasada przymusu bicia),
- jeżeli po biciu można wykonać kolejne, to również jest ono obowiązkowe,
- jeżeli kamień dojdzie do ostatniego rzędu, to jest promowany na damkę,
- jeżeli kamień (zwykła warcaba) przechodzi w czasie bicia przez jedno (z czterech) pól przemiany (w podstawie przeciwnika) i kontynuuje bicie to nie ulega przemianie w damkę i nadal pozostaje kamieniem (zwykłą warcabą),
- koniec gry następuje kiedy jeden z graczy nie będzie miał możliwości wykonania następnego ruchu lub gracz straci wszystkie pionki,
- sytuacja patowa następuje kiedy oboje z graczy wykonają po 15 ruchów damkami bez zmiany ilości figur na planszy.

3 Wykorzystanie AI

3.1 Algorytm MinMax

Algorytm MinMax polega na przeszukiwaniu wszystkich możliwych ruchów w celu wykrycia optymalnego zagrania. Każdej sytuacji na planszy przyporządkowania zostaje liczba zależna od stanu gry. Na podstawie tej gry program wybiera najlepszą ścieżkę i wykonuje związany z nią ruch. Algorytm. Przeszukuje jednocześnie ruchy obydwu graczy. Np. jeśli gracz nr 1 sprawdza dopuszczalne ruchy, to algorytm będzie szukał maksymalnej ścieżki wśród możliwych ruchów gracza i minimalnej wśród ruchów jego przeciwnika. W ten sposób znajdywana jest najlepsza ścieżka z punktu widzenia gracza nr 1.

Analizując drzewo gry w ten sposób, nie da się dojść do liści drzewa, tj. do sytuacji w której gra jest rozegrana. Zmusza to do zastosowania heurystycznych funkcji oceniających stan gry. Algorytm przeszukuje wtedy drzewo gry do pewnego ustalonego poziomu na którym stan gry jest ustalany przez zadaną funkcję. Odpowiednie tej funkcji jest kluczem do sukcesu programu.

3.2 Cięcia Alfa-Beta

Cięcia $Alfa - Beta$ nie są osobnym algorytmem, a raczej techniką optymalizacji algorytmu MinMax. Zakładają one dodanie dwóch zmiennych, które przechowują minimalną – $Beta$ i maksymalna - $Alfa$ wartość, jakie MinMax obecnie może zapewnić na danej głębokości drzewa lub wyżej. Przy starcie algorytmu $Alfa = -\infty$ i $Beta = +\infty$, wartości te są korygowane w dalszych etapach działania algorytmu.

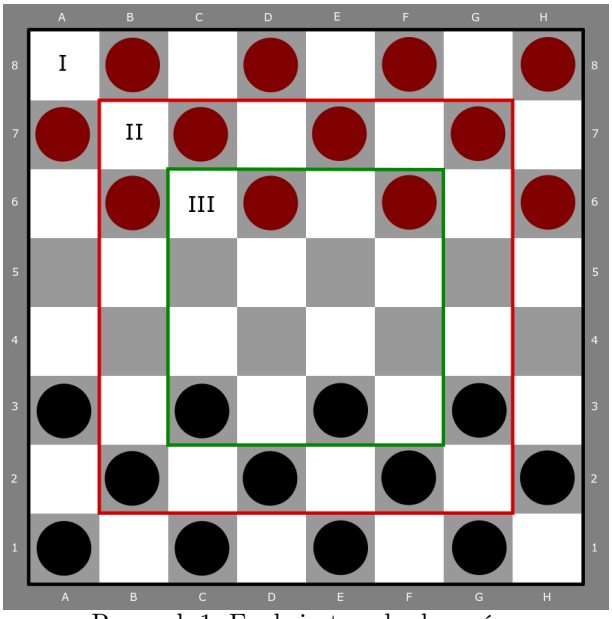
Na ich podstawie, jeśli przy sprawdzaniu kolejnego poddrzewa węzła minimalizującego $Alfa \geqslant Beta$ algorytm może odciąć dane poddrzewo (w tym przypadku cięcie Beta), ponieważ już wie, że maksymalna wartość tego poddrzewa przekroczy minimalna, którą węzeł może obecnie zagwarantować. Bardzo korzystnie wpływa to na złożoność algorytmu, pozwalając szybsze przeszukiwanie a nawet na głębsze poziomy w drzewie gry. Złożoność obliczeniowa algorytmu MinMax z cięciami $Alfa - Beta$ wynosi $O(b^{\frac{m}{2}})$.

3.3 Heurystyczne funkcje oceniające

Na Rysunku 1 przedstawiona jest plansza do gry w warcaby wraz z początkowym ustawieniem pionków, a także zaznaczone są trzy obszary oznaczone odpowiednio I, II, III.

Funkcja oceny polega na rozpoznaniu położenia pionka i przyporządkowaniu mu wartości tego obszaru (punktujemy każdego pionka znajdującego się w obszarze I, II, III odpowiednio 3, 2, 1 punktami). Następnie wyznaczamy sumę wszystkich wartości pionków gracza oraz przeciwnika i odejmujemy sumę wartości pionków gracza od sumy wartości pionków przeciwnika.

```
int Plansza::funkcjaOceniajacaPlansza(Color gracz);
```

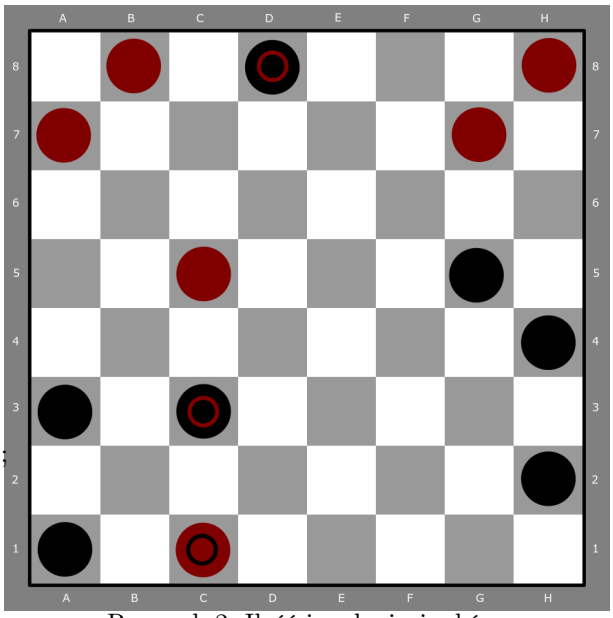


Rysunek 1: Funkcja trzech obszarów.

Podstawowym elementem oceny strategii gracza jest sprawdzenie siły bojowej graczy. Jest to również najprostszy do zaimplementowanie element strategii gracza. Polega na przeszukaniu całej szachownicy w poszukiwaniu pionków gracza i jego przeciwnika. Znalezione pionki/damki sumuje się z odpowiednimi wagami. Po wykonaniu operacji sumowania wyznaczamy ich różnice.

Królowa (czarna, czerwona) – 5 punktów
Pion (czarny, czerwony) – 2 punkty

int Plansza::funkcjaOceniajacaPionkowa(int x, int y);

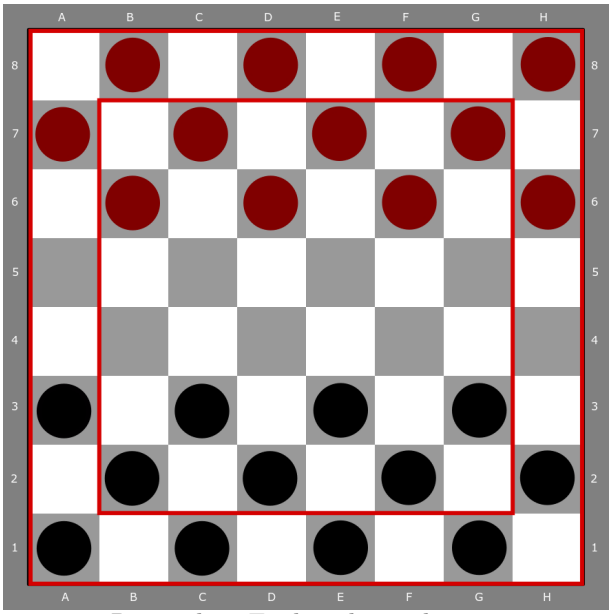


Rysunek 2: Ilość i rodzaj pionków.

W funkcji tej rozróżniamy dwa rodzaje pionków. Pionki krawędziowe znajdujące się na polach przylegających do krawędzi planszy , a także pionki środkowe znajdujący się poza obszarem oznaczonym czerwoną linia. Następnie wykonujemy standardową operację sumowania i odejmowania.

pionki w obszarze czerwonym – 4 punkty
pionki poza obszarem czerwonym – 2 punkty
królowe w obszarze czerwonym – 14 punkty
królowe poza obszarem czerwonym – 8 punkty

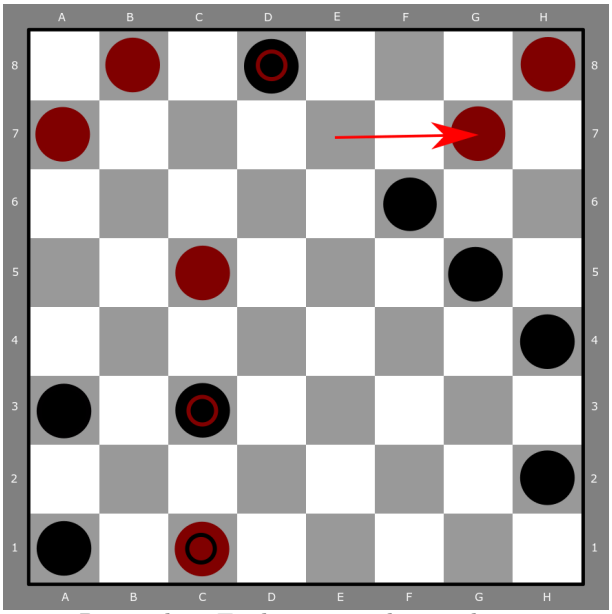
int Plansza::funkcjaOceniajacaKrawedziowa(int x, int y);



Rysunek 3: Funkcja krawędziowa.

Najważniejszym elementem w warcabach jest zbicie pionków przeciwnika. Bez tego nie da się wygrać rozgrywki. Dlatego zaimplementowanie wykrywania możliwości bić było jednym z kluczowych elementów. Wykrywanie bić polega na przeszukiwaniu szachownicy w celu znalezienia pionków gracza i przeciwnika znajdujących się obok siebie, a następnie sprawdzeniu, czy jest możliwość zbicia pionka, tzn. czy za pionkiem znajduje się miejsce puste. Jeśli tak to do końcowej wartości zwracanej przez funkcję jest dodawana jest wartość 10 punktów określająca bicie.

int Plansza::funkcjaOceniajacaCzyBicie(int x, int y);

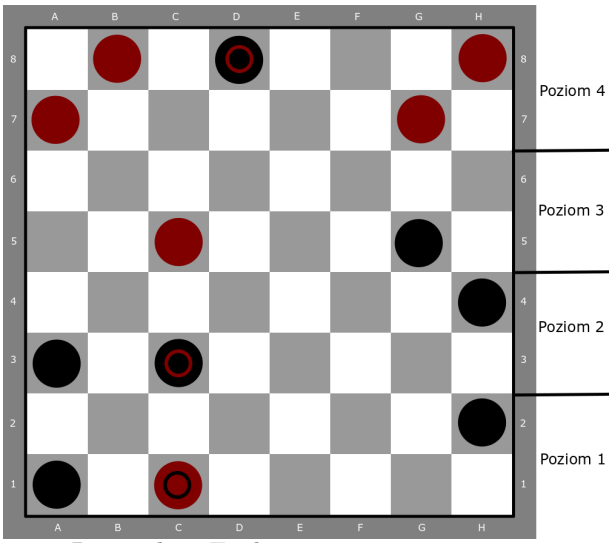


Rysunek 4: Funkcja wyszukująca bicia.

Wykonując każdy ruch pionkiem i tak zbliża się on do linii promocji, jednak ważne aby ruchy nie były losowe, tylko zorganizowane i przemyślane. Można poruszać najbardziej wysuniętym pionkiem, aby jak najszybciej dotrzeć do linii promocji, albo najbardziej oddalonym, aby wszystkie pionki przesuwały się z taką samą prędkością. Funkcja preferuje drugi sposób.

Poziom 2 – 1 punkt
Poziom 3 - 3 punkty
Poziom 4 – 10 punktów

int Plansza::funkcjaOceniajacaPoziomy(int x, int y);



Rysunek 5: Funkcja oceniająca poziomy.

4 Testowanie funkcji oceniających i wnioski

Każda z funkcji podłączona została do drzewa, w którym głębokość przeszukiwania wynosiła 6, przy większych głębokościach przeszukiwania czas oczekiwania na ruch bardzo się wydłużał. Dla każdej funkcji oceniającej przeprowadzono po trzy partie gier kontrolnych które pozwoliły uzyskać podstawowe informacje o efektywności każdej z funkcji oceniających. W tym teście najlepiej wypadły funkcje oceniające „krawędziowa", „oceniająca ilość i jakość pionów" oraz „wyszukująca bicia", gdy były wykorzystywane podane funkcje najtrudniej było wygrać.

Literatura

[1] https://en.wikipedia.org/wiki/Minimax, Minimax algorithm, 18.05.2019.

[2] Uhttps://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/, Geeks4Geeks MinMax algorithm in game theory, 18.05.2019

[3] http://sequoia.ict.pwr.wroc.pl/∼ witold/aiarr/2009_projekty/warcaby/, Funkcja oceniająca do algorytmu gry w warcaby, 18.05.2019

[4] https://docplayer.pl/29466544-Mini-tutorial-warcaby.html, Mini tutorial - warcaby, 19.05.2019

[5] A.Bieliński, *Inteligentne algorytmy*, Helion, 2015