Warszawa, 30.11.2015 r. (aktualizacja: 1.12.2015 r.)

Podstawy Sztucznej Inteligencji

Projekt LŚ.GD.3 – 15Z – Warcaby

Dokumentacja wstępna

Prowadzący projekt: mgr inż. Leszek Śliwa

Skład grupy:

- Jędrzej Blak, 261412
- Michał Lipiński, 261463
- Grzegorz Majchrzak, 261468

Wersja 1.1

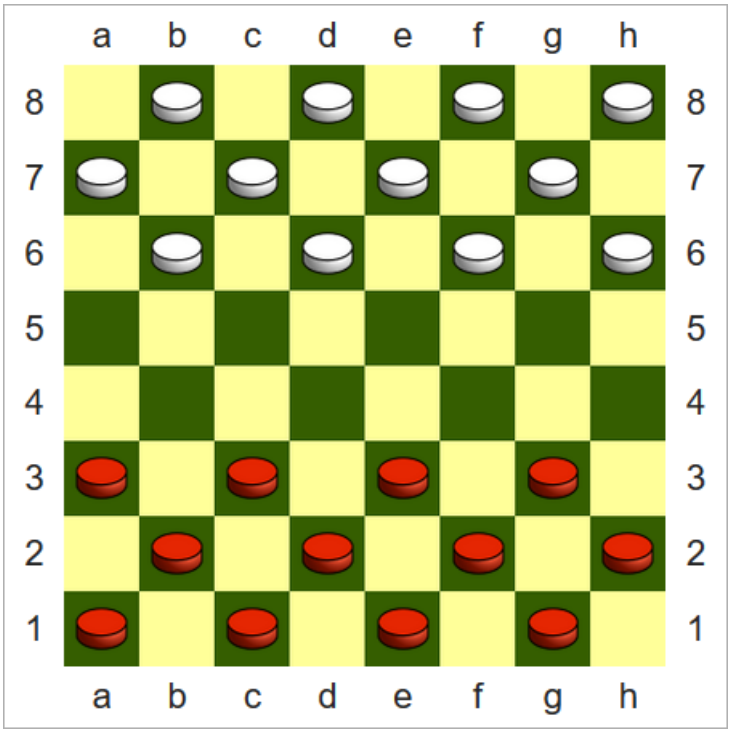
Temat projektu

Zaprojektować i zaimplementować komputerową grę w warcaby. Zastosować algorytm minimaksowy z przycinaniem alfa-beta. Zbudować graf gry z użyciem tablicy transpozycji i haszowaniem Zobrista.

1. Wstęp

Zasady gry w warcaby

Warcaby to dwuosobowa gra planszowa, rozgrywana na warcabnicy, bądź szachownicy, posiadającej 8x8 pól (używane są 32 ciemniejsze), z wykorzystaniem 24 pionków, po 12 danego koloru dla każdego z graczy. Występują dwa typy figur – piony oraz damki. Pierwszy ruch wykonuje gracz grający białymi figurami, kolejne ruchy następują naprzemiennie.



Ryc. 1 Plansza do gry w warcaby

Cel gry to zbicie wszystkich figur przeciwnika albo zablokowanie tych, które pozostają na planszy tak, aby nie mogły wykonać ruchu. W przypadku 15 ruchów każdego z graczy bez zmniejszenia liczby figur na planszy następuje remis. Piony mogą poruszać się po przekątnej do przodu o jedno pole, o ile nie zajmuje go inna figura. Bicia można dokonać, gdy po przekątnej na sąsiednim polu znajduje się pion lub damka przeciwnika, a za nim (również po przekątnej) wolne pole. W ciągu jednego ruchu można bić wielokrotnie, o ile na polu po zakończeniu poprzedniego bicia są spełnione wyżej wymienione warunki. Bicie jest obowiązkowe.

Pion staje się damką, gdy zakończy swój ruch w ostatnim rzędzie planszy. Damka może poruszać się o dowolną liczbę pól po przekątnej w przód lub w tył. Bicie damką można wykonać z dowolnej odległości, o ile za bitym pionem jest wolne pole. Po zbiciu pionu damka może zatrzymać się na wolnym polu w dowolnej odległości, bądź kontynuować bicie. Podczas bicia nie można przeskoczyć dwukrotnie tego samego pionu.

Hashowanie Zobrista

Jak zostało już wcześniej wytłumaczone, gra w warcaby toczy się na 32 polach. Aby zrealizować aktualny stan planszy będziemy więc potrzebować tablicy o 32 wierszach (1 na każde pole) i 5 kolumnach (1 na każdy możliwy stan pola – możliwe stany: pole puste, stoi biały/czarny pion, stoi biała/czarna damka).

Następnie wypełnimy tablicę wylosowanymi wartościami z $\sim U(0, R)$ (*gdzie* $R = 2^{64}$). Dzięki tak uzupełnionej tablicy będziemy mogli zrealizować zapamiętywanie stanu gry w postaci funkcji:

$$\mathrm{h}(w) = \sum_{i=0}^{31} t[i]\big[p[i]\big] \; mod \; R$$

Gdzie:

- $\mathrm{h}(w)$ - wynik funkcji hashującej dla danego stanu gry
- t - tablica wartości dla stanów każdego z pól
- i - indeks pola
- $p[i]$ - zakodowany stan pola
- R - górny zakres losowanych liczb (w naszym przypadku 2^{64})

Dla tak obliczonego stanu, każdy następny ruch powoduje zmianę wartości dwóch pól. To pozwala nam obliczyć nowy stan (na podstawie starego) poprzez operacje dodania stanu aktualnego dwóch zmienionych pól i odjęcie stanów poprzednich dla tych dwóch pól, modulo 2^{64}. Co istotne, jako że rozlosowywane wartości tablicy trzymającej aktualny stan gry były wybierane spośród liczb od 0 do 2^{64}, operacja dodawania i odejmowania będzie zrealizowana przy użyciu funkcji *XOR*.

Tak obliczony wynik funkcji hashującej jest kluczem/indeksem w tabeli transpozycji.

Tabela transpozycji

Obliczony wynik sumy stanowi klucz/indeks w tabeli transpozycji. **Tabela transpozycji** jest to "słownik" pod którym dla danego stanu gry (czyli dla danej wartości obliczonej z funkcji hashującej) zawarte są parametry algorytmu min-max oraz głębokość przeszukiwania.

Funkcja, która będzie oceniała nasz obecny stan gry ma następujący wzór:

$$S_{h(w)} = p_g - p_p + 5 * d_g - 5 * d_p + 3 * b_g - 6 * b_p + r_g - r_p$$

Gdzie:

- $S_{h(w)}$ – wynik funkcji (im większy, tym lepsza sytuacja)
- p_g, p_p – liczba pionów gracza, przeciwnika
- d_g, d_p – liczba damek gracza, przeciwnika
- b_g, b_p – liczba możliwych bić przez gracza, przeciwnika
- r_g, r_p – liczba możliwych do wykonania ruchów gracza, przeciwnika

Algorytm min-max to algorytm wyboru posunięcia, który zaimplementujemy w naszym programie. Polega on na analizie możliwych sekwencji ruchów odległych do najbliższych. Analizie towarzyszy założenie, że obaj przeciwnicy będą wybierali ruchy najbardziej korzystne dla siebie. Zatem, każdemu kolejnemu stanowi przypisywana jest ocena najlepszego (jeśli pora na ruch gracza) lub najgorszego (jeśli rusza się oponent) stanu, do którego prowadzi ruch. Algorytm min-max z przycinaniem α-β będzie polegał na nieanalizowaniu podgrafu wcześniej już analizowanego. Zostanie to zrealizowane za pomocą sprawdzania w tabeli transpozycji czy danego wierzchołka nie analizowaliśmy już wcześniej. Jeżeli tak, to zwrócimy wynik dla niego bez głębszej analizy. Powstanie nam w ten sposób graf, w którym dla jednego układu figur będzie jeden wierzchołek.

Głębokość przeszukiwania oznacza w oparciu o ile kroków w przód algorytm ocenił szansę wygrania partii.

```
funkcja minimax-alfabeta(stan, głębokość)
     zwróć alfabeta(stan, głębokość, -∞, +∞)
funkcja alfabeta(stan, głębokość, α, β)
     jeżeli tabela_transpozycji(stan)
          jeżeli tabela_transpozycji(stan).głębokość ≥ głębokość
               jeżeli w stanie gra gracz
                    zwróć β
               jeżeli w stanie gra przeciwnik
                    zwróć α
     jeżeli stan jest końcowy lub głębokość=0
          zwróć S(stan)
     jeżeli w stanie gra gracz // maksymalizujący
          dla każdego potomka stanu
               α := max(α, alfabeta(potomek, głębokość-1, α, β))
               jeżeli α ≥ β
                    zwróć β
          zwróć α
     jeżeli w stanie gra oponent // minimalizujący
          dla każdego potomka stanu
               β := min(β, alfabeta(potomek, głębokość-1, α, β))
               jeżeli α ≥ β
                    zwróć α
          zwróć β
```

Tab. 1 Algorytm MIN-MAX z przycinaniem α-β

Główny algorytm programu

Główny algorytm programu będzie polegał na wykonywaniu na zmianę ruchów komputera oraz gracza. Gracz – użytkownik programu – będzie sterował jedną ze stron. Każdy ruch komputera będzie polegał na przeanalizowaniu po kolei możliwych ruchów oraz wybraniu najkorzystniejszego poprzez porównanie wartości w tabeli transpozycji. W przypadku braku wartości w tabeli transpozycji, zostanie przeanalizowana sytuacja według algorytmu alfa-beta, a wynik zapisany do tabeli pod indeksem obliczonego hashu dla stanu gry. Jeżeli parametry poprzedniej analizy sytuacji (alfa, beta lub głębokość przeszukiwania) nie będą wystarczająco satysfakcjonujące do podjęcia decyzji (na przykład przy porównywaniu wyniku z głębokością przeszukania równą 15 oraz wyniku z głębokością przeszukania równą 3, możemy chcieć jeszcze raz przeanalizować ten drugi ruch, aby wyniki były możliwe do porównania – zostanie to precyzyjniej określone w późniejszym etapie wykonania projektu), sytuacja również zostanie przeanalizowana na nowo. Gra będzie się toczyła aż do zwycięstwa któregoś z graczy, bądź remisu.

2. Bibliografia

- Paweł Wawrzyński – „Podstawy sztucznej inteligencji”, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2014
- Warcaby – Wikipedia, https://pl.wikipedia.org/wiki/Warcaby
- Zobrist hashing – Wikipedia, https://en.wikipedia.org/wiki/Zobrist_hashing
- Algorytm alfa-beta – Wikipedia, https://pl.wikipedia.org/wiki/Algorytm_alfa-beta
- Algorytm min-max – Wikipedia, https://pl.wikipedia.org/wiki/Algorytm_min-max