Leonard Feszczuk 249027
gr. Pt 13:15-15:00

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 3- Kółko I krzyżyk

1 **Wprowadzenie**

Celem projektu było stworzenie gry w oparciu o strategię Min-Max. Z kilku możliwych podanych gier, wybrano Kółko i krzyżyk, gdzie gracz posiada możliwość definiowania rozmiaru planszy (zawsze kwadratowej tj. 3x3, 4x4, itd.) wraz z ilością znaków potrzebnych by wygrać.

2 **Budowa programu**

Program składa się z:

- Pliku głównego main: ten plik odpowiada za menu i wywoływanie funkcji potrzebnych do działania programu.
- Plików gra.cpp i gra.hpp zawierających funkcje:
  Wyswietl() która wyświetla plansze w terminalu
  Wygrana() sprawdzająca czy spełniony jest warunek potrzebny do wygranej przechodzi ona po wierszach, kolumnach i na ukos.
  **Remis(**) sprawdza czy wolne są jeszcze jakieś pola jeżeli nie to zwraca true co oznacza wynik remis.
- Plików minimax.cpp i minimax.hpp zawierających funkcje:
  minimax() wdrażającą algorytm.
  minimaxalfabeta() wprowadzający usprawnienie prunningu do algorytmu.
  CzłowiekzKomputerem() przeprowadza rozgrywke użytkownik kontra AI
  DwochGraczy() jest to opcja pozwalająca przeprowadzić grę kółko i krzyżyk dla dwóch graczy wykonujących ruchy po kolej.

3 Algorytm Alfa-Beta

Algorytm przeszukujący, redukujący liczbę węzłów, które muszą być rozwiązywane w drzewach przeszukujących przez algorytm min-max. Jest to przeszukiwanie wykorzystywane w grach dwuosobowych, takich jak właśnie kółko i krzyżyk lub szachy. Warunkiem stopu jest znalezienie przynajmniej jednego rozwiązania czyniącego obecnie badaną opcję ruchu gorszą od poprzednich opcji. Wybranie takiej opcji ruchu nie przyniosłoby korzyści graczowi ruszającemu się, dlatego też nie ma potrzeby przeszukiwać dalej gałęzi drzewa tej opcji. Ta technika pozwala zaoszczędzić czas poszukiwania bez zmiany wyniku działania algorytmu.

Zapis w pseudokodzie:

minimax(pozycja, głębia, alfa, beta, graczMax):

jeżeli głębia == 0 lub koniec gry

    zwróć ocenę ruchu

jeżeli gracz Max

maxOcena = -INF

dla każdego możliwego ruchu

  ocena = minimax(nowyRuch, głębia-1, alfa, beta, fałsz)

  alfa = max(alfa, ocena)

  jeżeli alfa >= beta

    przerwij

zwróć maxOcena

w przeciwnym przypadku

minOcena = INF

dla każdego możliwego ruchu

```
        ocena = minimax(nowyRuch, głębia-1, alfa, beta,prawda)
        beta = max(beta, ocena)
        jeżeli alfa >= beta
            przerwij
zwróć minOcena
```

4 Wnioski

- Początkowo by oceniać wykonany ruch kopiowano planszę z każdym wywołaniem Min-Maxa. Powodowało to bardzo długie działanie algorytmu. Ostatecznie wybrano sposób by wykonywać ruchy na tej samej planszy, jednak cofać je po wykonaniu algorytmu i wyświetlać dopiero, gdy zostanie wybrany najbardziej optymalny ruch.
- Algorytm Min-Max świetnie radzi sobie z rozgrywką na tyle ile jest to możliwe przy tak prostej grze, nie popełnia błędów i wygrywa o ile jest taka możliwość
- Podstawowa wersja Min-Max potrzebowała bardzo dużo czasu dla wywołania ruchu komputera dla plansz większych niż 3x3. Dodanie algorytmu Alfa-Beta odcinającego niektóre z gałęzi drzewa znacząco zmniejszyło czas na wybranie odpowiedniego ruchu lecz nie usunęło problemu do końca ponieważ komputer i tak „myśli” na tyle długo że uniemożliwia to płynną rozgrywkę dla plansz większych niż 3x3. Dodatkowym usprawnieniem było wybranie ruchu inicjującego komputera.
- Dodatkowo dla większego usprawnienia działania programu możliwe byłoby dodanie bazy danych z każdym możliwym ruchem i jego wyceną.

Źródła:

[1] Geeks4geeks Minimax algorithm in game theory

[2] GeeksForGeeks
Minimax Algorithm in Game Theory— Set 3 (Tic-Tac-Toe AI –
Finding optimal move)

[3] Sebastian Lague
Algorithms Explained – minimax and alpha-beta pruning

[4] Kamal Rawat
Print elements of a matrix in diagonal order