Tomasz Jankowiak 249006 1 VI 2020

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 3 – Gry & AI

Prowadzący:	mgr inż. Marta Emirsajłow
Termin zajęć:	Piątek, 9.15
Numer grupy projektowej:	E08-06i

1. Wprowadzenie

Sztuczna inteligencja (SI, ang. artificial intelligence, AI) jest działem informatyki zajmującym się tworzeniem modeli zachowań inteligentnych (procesów decydujących o inteligencji człowieka; sposobu myślenia i działania człowieka) oraz programów i systemów symulujących te zachowania. Są to metody i algorytmy pozwalające na symulowanie inteligentnego zachowania komputera, zbliżające jego sposób podejmowania decyzji do ludzkiego. Inaczej mówiąc, sztuczna inteligencja to nauka o tym, jak produkować maszyny wyposażone w niektóre cechy ludzkiego umysłu, takie jak umiejętność rozumienia języka, rozpoznawania obrazów, rozwiązywania problemów i uczenia się. Termin „sztuczna inteligencja" został wymyślony przez amerykańskiego informatyka Johna McCarthy'ego w 1955r. na konferencji naukowej w Dartmouth. Określił ją jako „naukę i inżynierię tworzenia inteligentnych maszyn". Badania nad sztuczną inteligencją wykorzystują narzędzia i ustalenia z wielu dziedzin. „Mądre" maszyny wykorzystujące SI mogą pomagać ludziom w codziennych pracach, czynić nasze życie łatwiejszym. Pozwalają oszczędzać czas i pieniądze.

Istnieje kilka rodzajów/ stopni sztucznej inteligencji. Obecnie potrafimy jedynie nakazać komputerowi naśladowanie zachowania inteligentnego, gdyż stopień skomplikowania ludzkiego mózgu (ponadto zasada działania ludzkiego mózgu wcale nie jest do końca rozumiana przez współczesną naukę) przewyższa najbardziej nawet złożony komputer. Pamiętajmy jednak, że wcale nie jest powiedziane, iż za kilka lat nie powstanie technologia, która pozwoli skonstruować ideowy odpowiednik ludzkiego mózgu i nauczyć go rozwiązywania problemów niedostępnych dla człowieka. Nie da się niestety dokładnie przewidzieć tego, jak szybki będzie rozwój sztucznej inteligencji.

Celem projektu było stworzenie gry, w której wykorzystane zostaną algorytmy sztucznej inteligencji. Wybrano grę kółko i krzyżyk i zastosowano w niej algorytm min-max, opisany w punkcie 2.

Program napisano w języku C++ i umieszczono w nim wyjaśniające komentarze do niektórych fragmentów kodu. Jest on podzielony na 3 pliki: *main.cpp*, *si.cpp* oraz *si.h*. Do kompilacji i testów użyto programu Code::Blocks. Nie wprowadzano wszystkich możliwych asercji i poprawek, ponieważ główną wagę przywiązano do algorytmu minimax, symulującego sztuczną inteligencję. Program nie posiada typowej wersji graficznej, jednak w terminalu jest wyświetlana plansza do gry, utworzona ze znaków. Przed rozpoczęciem rozgrywki gracz wybiera rozmiar kwadratowej planszy oraz liczbę znaków w linii potrzebną do wygranej. Gracz (człowiek) jest kółkiem i zaczyna, natomiast komputer w odpowiedzi po każdym ruchu stawia krzyżyk.

Do zapamiętania stanu gry może być wykorzystywana zwykła tablica, której indeksy odpowiadają pozycjom planszy. Po każdym ruchu trzeba sprawdzać, czy ktoś już nie wygrał lub nie nastąpił remis. Sama gra jest zwykłą pętlą, która prowokuje wykonanie ruchów. Warunkiem progresji pętli jest stan, w którym nikt jeszcze nie wygrał lub nie zremisował.

Testy zachowania SI przeprowadzono rozgrywając z komputerem kilka pojedynków, głównie na planszach 3x3 i 4x4.

2. Kółko i krzyżyk oraz stosowane techniki SI

a) Algorytm min-max

Szukając najlepszego możliwego ruchu, algorytm zakłada, iż przeciwnik wykona najlepszy ruch dla niego (czyli najgorszy dla nas), a my przygotowujemy najlepszą odpowiedź. Naszym celem będzie zatem wykonanie ruchu, który maksymalizuje dla nas wartość pozycji, po której przeciwnik wykonał swój najlepszy ruch (taki, który minimalizuje wartość dla niego). Badamy w ten sposób pewną liczbę poziomów (nietrudno zauważyć formę drzewa analizy), wartości z ostatnich poziomów są „wnoszone" do góry, wedle reguł min-max. Algorytm jest rekurencyjny i do analizy kolejnych ruchów wywołuje sam siebie. Na początku tworzymy drzewo wszystkich możliwych ruchów (stanów w grze) do pewnej głębokości ograniczonej przez moc obliczeniową komputera.

Złożoność czasowa algorytmu mini-max wynosi $\mathbf{O(b^m)}$, gdzie b to liczba dostępnych ruchów w każdym wierzchołku drzewa, a m maksymalna głębokość drzewa (maksymalna głębokość rekurencji). Z kolei złożoność pamięciowa jest równa $\mathbf{O(b \cdot m)}$.

Algorytm min-max w swojej podstawowej formie jest dość wolny. Istnieje poprawiona wersja algorytmu mini-max, która w praktyce często go zastępuje. Modyfikacja ta pozwala znacznie skrócić czas analizy (do $O(b^{m/2})$), eliminując zbędne porównania wartości pochodzących z poddrzew i tak niemających szansy na wyniesienie podczas propagacji wartości wg reguły mini-max. Jest ona powszechnie znana jako algorytm **cięć α-β**. Z kolei nie każdy algorytm przeszukiwania dobrze nadaje się do programowania określonych gier z uwagi na skomplikowaną obsługę struktur danych. Dobre algorytmy odszukiwania właściwej strategii gry są, niestety, bardzo złożone. Programiści zaczynają coraz częściej wykorzystywać szybkość współczesnych komputerów, co pozwala uprościć sam proces programowania poprzez stosowanie najprostszych algorytmów przeszukiwania typu brute-force (sprawdzenie wszystkich możliwości), co powoduje olbrzymią, często wykładniczą, złożoność obliczeniową.

John von Neumann, węgierski matematyk i informatyk, w 1928 roku udowodnił twierdzenie, że każda skończona gra dwuosobowa o sumie zerowej ma co najmniej jedno rozwiązanie, które określa wartość gry i optymalne strategie dla graczy.

Wracając do wcześniejszych rozważań, pamiętamy, iż metoda minimax wykorzystuje **heurystykę**, która zakłada, że wykonujemy najlepszy dla nas ruch (max), a przeciwnik wykonuje posunięcie najlepsze dla siebie, czyli najgorsze dla nas (min). Funkcja heurystyczna jest funkcją oceniającą wartość stanu gry w danym momencie. Tłumaczy ona stan gry na formę zrozumiałą dla komputera. Jest to metoda szukania rozwiązania optymalnego. Poprawnie ułożona funkcja heurystyczna jest konieczna do właściwego działania algorytmu min-max.

Powstaje pytanie, po czym poznajemy siłę naszej pozycji w danym etapie gry „kółko i krzyżyk”. W literaturze często pojawia się kryterium, wykorzystujące pojęcie **liczby linii otwartych dla danego gracza**, tzn. takich, które nie są blokowane przez przeciwnika i w związku z tym rokują nadzieję na skonstruowanie pełnej linii dającej zwycięstwo. Wartość tej liczby jest pomniejszana o liczbę linii otwartych dla przeciwnika. Porównania różnych wariantów i wyboru optymalnej możliwości dostarcza algorytm min-max.

3. Podsumowanie i wnioski

Podstawowa wersja algorytmu min-max okazała się mało wydajna dla większych plansz, i potrzeba było dużo czasu na wybranie ruchów. Zastosowanie cięć alfa-beta znacznie skróciło czas wybierania najlepszego ruchu, gdyż współczynnik rozgałęzienia był dwukrotnie mniejszy niż w metodzie minimaks. Mimo to zaimplementowany algorytm potrzebuje dużo czasu na podjęcie decyzji o najlepszym ruchu dla większych plansz. Ponadto, by ocenić wartość ruchu dla plansz większych niż 3x3 i o mniejszym warunku wygranej niż rozmiar planszy, potrzeba większej głębokości rekurencji niż dla standardowego warunku zwycięstwa.

Algorytm radzi sobie bardzo dobrze w rozgrywce. Ani razu nie udało mi się wygrać, a błędy były bezlitośnie wykorzystywane przez maszynę. Zatem udało się zrealizować zadanie, a program prawdopodobnie działa poprawnie.

4. Bibliografia

- P. Wróblewski – „Algorytmy, struktury danych i techniki programowania”, wydanie V, Helion 2015
- https://www.sztucznainteligencja.org.pl/
- https://pl.wikipedia.org/wiki/Sztuczna_inteligencja
- https://en.wikipedia.org/wiki/Artificial_intelligence
- https://www.hackerearth.com/blog/developers/minimax-algorithm-alpha-beta-pruning/
- https://pl.wikipedia.org/wiki/Algorytm_min-max
- https://en.wikipedia.org/wiki/Minimax
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/
- https://stackoverflow.com/
- https://zeszytysrn.ujk.edu.pl/wp-content/uploads/2016/05/zeszytsrn24.pdf#page=152
- https://el.us.edu.pl/ekonofizyka/index.php?title=Teoria_gier/Gry_dwuosobowe_suma_zero&oldid=7568