Filip Materak

249482

Piątek 13:15-15:00

E08-06j

mgr Marta Emirsajłow

Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 3
Gry & AI
Warcaby

1 Wstęp teoretyczny - Algorytm MinMax

Algorytm MinMax wywodzi się z twierdzenia o grze o sumie stałej. W ścisłej definicji: dla każdej dwuosobowej gry o sumie zerowej istnieje wartość V i mieszana strategia dla każdego gracza, takie, że (a) – biorąc pod uwagę strategię gracza drugiego, najlepszą możliwą spłatą dla gracza pierwszego jest V, i (b) – biorąc pod uwagę strategię gracza pierwszego, najlepszą możliwą spłatą dla gracza drugiego jest -V.

2 Opis tworzonej gry

Gra warcaby klasyczne (określana też warcabami brazylijskimi) rozgrywana jest na planszy o rozmiarze 8 x 8 pól pokolorowanych na przemian na kolor jasny i ciemny. Każde pole może być: puste, zawierać czerwony / czarny pion, lub czerwoną / czarną damkę. Otrzymujemy więc 5 możliwych stanów dla każdego pola, dlatego do stworzenia planszy wykorzystałem tablicę typów wyliczeniowych. Klasa 'Board' posiada poza tym metody umożliwiające wygenerowanie wszystkich możliwych ruchów dla danego piona lub całego koloru, sprawdzenie czy występuje możliwość bicia dla piona / koloru, wykonanie ruchu, funkcję heurystyczną oceniającą sytuacje na planszy oraz funkcję sprawdzającą stan gry. Graficzna reprezentacja gry została wykonana przy użyciu biblioteki SFML 2.5.1, a textury wykonano przy użyciu programu Gimp.

3 Zastosowane algorytmy AI

3.1 Drzewo możliwych stanów

Dla drzewa przeszukań, zawierającego wszystkie możliwe stany planszy dla n następnych ruchów, zaimplementowałem klasę pojedynczego elementu takiego drzewa. W konstruktorze przyjmuje on obiekt planszy, maksymalną głębokość drzewa, kolor, którym gra gracz oraz jako argument opcjonalny przyjmuje on wykonany ostatnio ruch. Aby utworzyć drzewo należy utworzyć jego korzeń t.j. pierwszy element, który otrzymuje aktualną plansze gry. Do pierwszego elementu nie podajemy jaki ruch został wykonany, ponieważ znajdzie się w nim ruch wybrany przez algorytm Min-max. Konstruktor elementu jest funkcją rekurencyjną, która wywołuje się, aż do osiągnięcia maksymalnej głębokości drzewa, lub gdy po wykonanym ruchu następuje zakończenie gry. Konstruktor pobiera z planszy wszystkie możliwe do wykonania dla danego gracza ruchy, następnie dla każdego ruchu tworzy potomka, któremu przekazuje: ruch, obiekt planszy po wykonaniu danego ruchu, oraz kolor pionków, które będą wykonywały ruch następne. Jeśli zostanie spełniony jeden z warunków zakończenia pogłębiania drzewa, wywołana zostaje funkcja heurystyczna, oceniająca ostateczną sytuacje na planszy.

3.2 Algorytm MinMax

Algorytm ten jest metodą wybierania ruchu, który pozwala na minimalizację możliwych strat lub maksymalizację zysków. Jeśli dla danej sytuacji na planszy stworzymy drzewo wszystkich możliwych układów dla n-następnych ruchów i każdej przypiszemy wartość przy użyciu określonej funkcji heurystycznej, to możemy wybrać taki ruch, który daje nam największe korzyści. Gracz, dla którego wywołany jest w/w algorytm będzie dążył do maksymalizacji zysków, natomiast przeciwnik będzie się starał te zyski zminimalizować. Algorytm MinMax dostaje jako parametr korzeń drzewa stanów. Następnie sprawdza, czy aktualny gracz dąży do maksymalizacji, czy do minimalizacji zysków i zależnie od tego wybiera ruch o największej / najmniejszej wartości. Aby to uzyskać, wywołuje się rekurencyjnie aż do osiągnięcia warunku podstawowego, dzięki któremu może zwrócić konkretną wartość. Jeśli założymy, że dla każdego ruchu można wykonać n -następnych, to złożoność obliczeniowa algorytmu wyniesie $O(n^m)$, gdzie m to maksymalna głębokość rekurencji. W warcabach liczba możliwych ruchów jest zmienna, więc nie da się dokładnie określić złożoności obliczeniowej algorytmu

3.3 Funkcja heurystyczna

Funkcja heurystyczna pozwala na przybliżenie rozwiązania danego problemu bez zapewnienia gwarancji poprawności uzyskanego wyniku. Używa się jej wtedy, gdy właściwy algorytm jest zbyt złożony, kosztowny lub nieznany. Od implementacji tej funkcji zależy efektywność całego algorytmu wybierania najlepszych ru- chów. Oczywistym czynnikiem jest ilość posiadanych pionów oraz damek. Starałem się punktować dodatkowo piony 'bezpieczne', t.j. takie, które nie mogą zostać zbite (przylegają do bocznej ścia- ny planszy). Dodatkowe punkty przyznaję też za wyprowadzenie piona z rzędów początkowych, ponieważ uważam, że piony poruszające się w bardziej zwartej formacji są efektywniejsze. Podsta- wową mechaniką warcabów jest wykonywanie zbić, dlatego każdy pion z możliwością bicia również otrzymuje dodatkowe punkty.

4 Podsumowanie

Algorytm radzi sobie bardzo dobrze w rozgrywce. Po większej liczbie rozgrywek można zauważyć powtarzalność w początkowych ruchach algorytmu. Rozgrywka dwóch algorytmów grających przeciwko sobie za każdym razem wygląda jednakowo. Złożoność algorytmu jest dosyć duża, ze względu na ilość ruchów, które algorytm ma do rozpatrzenia. Program napotyka problemy przy zbyt dużej ilości możliwych ruchów co przy mniejszej mocy obliczeniowej komputera może zmniejszać płynność gry. Algorytm nie jest niepokonany, skuteczność jego działania jest uzależniona od tego, jak dokładnie funkcja heurystyczna przybliża dany system, a obecnie zaimplementowana jest z pewnością uboga i daleka od idealnej oceny planszy. Mimo wszystko algorytm sprawia trudności i jest w stanie pokonać człowieka.

5 Literatura

Podczas wykonywania projektu wspierałem się nastepującymi stronami:

1. Minimax Wikipedia
2. Geeksforgeeks minimax algotithm
3. Checkers-Game-using-C.
4. Michał Mielnicki - Funkcja oceniająca do algorytmu minimaksu w grze warcaby