Data: 29.05.2020

Imię i nazwisko: Janusz Domaradzki

Nr albumu: 249024

Dane prowadzącego: mgr inż. Marta Emirsajłow

Termin zajęć: Piątek 13:15-14:45

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 3 – Gry & AI

1. Wprowadzenie

Celem projektu było stworzenie aplikacji będącej grą i wykorzystującej swego rodzaju algorytm sztucznej inteligencji. W tym projekcie gra została wyposażona w graficzną oprawę.

Stworzona gra:

- Warcaby

Algorytm wykorzystany w aplikacji:

- Minmax

2. Opis algorytmu

Algorytm MinMax wywodzi się z twierdzenia o grze o sumie stałej. Jeżeli dwie osoby grają przeciwko sobie, to poprawa sytuacji gracza pierwszego oznacza proporcjonalne pogorszenie się sytuacji gracza drugiego. Używając tej informacji można przypisać danej sytuacji na planszy konkretną wartość, używając odpowiedniej funkcji heurystycznej. Wartość ta będzie dodatnia, jeśli sytuacja jest korzystna dla gracza, a ujemna – gdy jest niekorzystna.

3. Opis gry

W warcaby gra się na planszy 8 na 8 pól. Każde z nich może być puste lub zawierać grywalną figurę: biały lub czarny pion oraz białą lub czarną damę. Każde pole ma zatem 5 możliwych stanów, dlatego do stworzenia planszy została wykorzystana tablica wyliczeniowa. Klasa ‘Plansza’ posiada również metody umożliwiające wygenerowanie wszystkich możliwych ruchów dla danego piona lub całego koloru, sprawdzenie możliwości bicia dla piona/koloru, wykonanie ruchu, funkcję heurystyczną oceniającą sytuację na planszy oraz funkcję sprawdzającą stan gry.

Do przechowania ruchów zaimplementowane zostały dwie struktury: ‘Wspolrzedne’ oraz ‘Ruch’. Pierwsza z nich zawiera współrzędne piona przed i

po wykonaniu ruchu. Druga zawiera wszystkie koordynaty dla danego ruchu (gdy możliwe są więcej niż jedno zbicie).
Graficzna reprezentacja została wykonana przy użyciu biblioteki SFML 2.5, a tekstury wykonano przy użyciu programu Paint.

Przyjęte zasady:
1. Bicie jest obowiązkowe
2. Piony mogą poruszać się tylko do przodu, ale dozwolone jest bicie wstecz
3. Jeśli po biciu można wykonać kolejne, to również ono jest obowiązkowe
4. Nie ma obowiązku wyboru najdłuższej sekwencji bić
5. Przy dojściu do ostatniego rzędu pion jest promowany na damkę, chyba że ma dostępne bicie – wtedy promocja nie następuje
6. Damka po biciu może przemieszczać się dalej, dopóki nie napotka przeszkody
7. Koniec gry następuje, gdy jeden z graczy nie ma możliwości wykonania następnego ruchu
8. Pat występuje gdy obaj gracze wykonają po 15 ruchów damkami bez zmiany ilości figur na planszy

4. Wykorzystane techniki AI

- Drzewo możliwych stanów

  Dla drzewa przeszukań, zawierającego wszystkie możliwe stany planszy dla *n* następnych ruchów, została zaimplementowana klasa pojedynczego elementu takiego drzewa. Konstruktor zawiera obiekt planszy, maksymalną głębokość drzewa, kolor gracza oraz ostatnio wykonany ruch jako argument opcjonalny. Przy tworzeniu drzewa najpierw trzeba utworzyć jego korzeń, t.j. pierwszy element, który otrzymuje aktualną planszę gry. Pierwszym elementem jest ruch wybrany przez algorytm Minmax. Konstruktor elementu jest funkcją rekurencyjną, wywołującą się aż do osiągnięcia maksymalnej głębokości drzewa, albo gdy po wykonaniu ruchu następuje koniec gry. Po spełnieniu jednego z warunków zakończenia pogłębiania drzewa zostaje wywołana funkcja heurystyczna, która ocenia ostateczną sytuację na planszy

- Funkcja Heurystyczna

  Funkcja heurystyczna pozwala na przybliżenie rozwiązania danego problemu bez zapewnienia gwarancji poprawności uzyskanego wyniku. Jest przydatna wtedy, gdy właściwy algorytm jest zbyt złożony, kosztowny lub nieznany. Od implementacji tej funkcji zależy efektywność algorytmu wybierania najlepszych ruchów.
  Piony i damki, które nie mogą zostać zbite (przylegające do bocznej ściany) były dodatkowo punktowane. Również wyprowadzanie pionów z rzędów początkowych, ponieważ piony poruszające się w zwartej formie są moim zdaniem bardziej efektywne. Każdy pion z możliwością bicia również otrzymuje dodatkowe punkty.

Przydzielanie punktów za określone czynniki na planszy:

1. Wygrana 10000 pkt
2. Przegrana -10000 pkt
3. Remis 0 pkt
4. Pion +7 pkt
5. Damka +25 pkt
6. Figura przy bocznej ścianie +2 pkt
7. Figura oddalona od ściany bocznej o 1 pole +1 pkt
8. Pion w rzędzie dalszym niż 1 +2 pkt
9. Pion w rzędzie dalszym niż 2 +1 pkt
10. Pion w rzędzie dalszym niż 3 +1 pkt
11. Każda figura z możliwością bicia +15 pkt

- Algorytm MinMax

Algorytm ten jest metodą wybierania ruchu, pozwalającego na minimalizację strat lub maksymalizację zysków. Dla danej sytuacji na planszy, po stworzeniu drzewa wszystkich możliwych układów dla *n* następnych ruchów i przypisaniu każdej z nich wartości przy użyciu funkcji heurystycznej, możemy wybrać ruch dający nam największe korzyści. Dla wskazanego gracza algorytm będzie dążył do maksymalizacji zysków, zaś przeciwnik podejmie starań zminimalizowania tych zysków.
Parametrem algorytmu MinMax jest korzeń drzewa stanów. Po jego otrzymaniu sprawdza, czy aktualny gracz dąży do minimalizacji czy maksymalizacji zysków i zależnie od tego wybiera ruch o najmniejszej bądź największej wartości. W tym celu algorytm wywołuje się rekurencyjnie aż do osiągnięcia warunku podstawowego, dzięki któremu może zwrócić konkretną wartość.
Przy założeniu, że dla każdego ruchu można wykonać *n* następnych, to złożoność obliczeniowa algorytmu wyniesie $O(n^m)$, gdzie *m* to maksymalna głębokość rekurencji. Ponieważ w warcabach liczba możliwych ruchów jest zmienna, określenie dokładnej złożoności obliczeniowej jest niemożliwe.

- Cięcia Alfa-Beta

Jest to modyfikacja do algorytmu MinMax. Zakłada ona dodanie dwóch zmiennych, przechowujących minimalną (β) i maksymalną (α) wartość, jakie MinMax może zapewnić na danej głębokości drzewa lub wyżej. Na początku algorytmu $\alpha = \infty$ i $\beta = -\infty$, wartości te są korygowane w dalszych etapach działania algorytmu. Na ich podstawie przy sprawdzaniu kolejnego poddrzewa węzła minimalizującego $\alpha \geq \beta$ algorytm może odciąć dane poddrzewo (w tym przypadku odcięte zostanie β), ponieważ już wie, że maksymalna wartość tego poddrzewa przekroczy minimalną, możliwą obecnie do zagwarantowania przez węzeł. Wpływa to korzystnie na złożoność algorytmu – pozwala osiągnąć lepszy czas wyszukiwania optymalnego ruchu, lub zwiększyć głębokość rekursji. W ten sposób, przy identycznych założeniach

wspomnianych w opisie algorytmu MinMax, złożoność obliczeniowa algorytmu wynosi $O(n^{m/2})$.

5. Wnioski

Algorytm radzi sobie bardzo dobrze w rozgrywce. Niekiedy można odczuć powtarzalność wykonywanych ruchów, zwłaszcza na początku rozgrywki.

Złożoność algorytmu jest duża, głównie przez ilość ruchów będących do rozpatrzenia.

Algorytm nie jest niepokonany, skuteczność jego działania uzależniona jest od dokładności przybliżenia danego systemu przez funkcję heurystyczną. Obecnie zaimplementowana jest z pewnością uboga, a przez to daleka od idealnego ocenienia planszy.

Testy były wykonywane metodą prób i błędów. Dotyczyły głównie reakcji AI na dane wagi czynników. Algorytm sprawia trudności człowiekowi w jego pokonaniu i jest sam w stanie wygrać.