Sztuczna inteligencja i inżynieria wiedzy
Lista 2

Jan Jakubik, Piotr Syga

25 marca 2024

1 Cel listy

Celem ćwiczenia jest praktyczne zapoznanie się z algorytmami grania w gry dwuosobowe - algorytmem Minimax oraz jego ulepszeniem z wykorzystaniem alfa-beta cięć.

2 Wprowadzenie teoretyczne

Poniżej znajdują się informacje pomocne w wykonaniu listy zadań. Zakłada się, że po wykonaniu student opanował zagadnienia teoretyczne.

2.1 Definicje

Definicja 1 (Drzewo decyzyjne)**.** *Skierowane drzewo, w którym każdy węzeł reprezentuje stan problemu, a każda krawędź reprezentuje działanie lub wybór, który prowadzi do kolejnego stanu nazywamy drzewem decyzyjnym.*

Drzewo decyzyjne może być wykorzystane do modelowania problemów, w których należy podejmować decyzje, uwzględniając wiele możliwych opcji.

W przypadku gier, drzewa decyzyjne są często używane do reprezentowania możliwych ruchów w grze. Każdy węzeł w drzewie reprezentuje stan gry, a krawędzie odpowiadają możliwym ruchom lub decyzjom, które gracz może podjąć. Łatwo zatem zauważyć, iż stopień wierzchołka odpowiada liczbie możliwych do wykonania ruchów, a zatem nie musi być stały na jednym poziomie drzewa. Gdy gracz dokonuje ruchu, przesuwamy się po drzewie do węzła odpowiadającego nowemu stanowi gry. Koniec gry następuje w momencie dotarcia do liścia. Długość ścieżki (odpowiadająca liczbie rund) może być różna dla różnych liści.

Definicja 2 (Gra o sumie zerowej)**.** *Grą nazywamy uporządkowaną trójkę* $G = (N, \mathcal{S}, U)$*, gdzie* N *oznacza liczbę graczy,* $\mathcal{S} = S_1, S_2, \ldots, S_N$ *oznacza zbiór skończonych zbiorów strategii graczy od* 1 *do* N*, a* $U = u_1, u_2, \ldots, u_N$ *jest zbiorem funkcji użyteczności (wypłaty), które przypisują wartość wyniku dla każdej*

możliwej kombinacji działań graczy. To znaczy $u_i : S_1 \times \cdots \times S_n \to \mathbb{R}$ *jest wypłatą gracza i przy założeniu strategii graczy.*
Jeśli $\forall s \in S_1 \times \cdots \times S_n, \sum_{i \in N} u_i(s) = 0$*, to grę nazywamy grą o sumie zerowej.*

2.2 Algorytm Minimax

Algorytm Minimax jest strategią przeszukiwania drzewa gry, która służy do podejmowania decyzji w grach dwuosobowych o sumie zerowej, takich jak szachy, warcaby czy go. Algorytm ten dąży do minimalizacji maksymalnej możliwej straty, której gracz może doznać, biorąc pod uwagę ruchy przeciwnika. Algorytm Minimax opiera się na przeglądaniu drzewa gry, gdzie każdy wierzchołek drzewa reprezentuje stan gry, a krawędzie pomiędzy nimi reprezentują ruchy graczy. Liściami będą wierzchołki reprezentujące zakończenie gry wygraną jednego z graczy. W praktyce, ze względu na liczbę wierzchołków wzrastającą wykładniczo z głębokością stosujemy limit głębokości oraz heursytyki oceny stanu gry, pozwalające nam ocenić niezakończoną grę w przypadku dojścia do tego limitu.

Główne kroki algorytmu Minimax:

1. Przeszukuj drzewo gry, zaczynając od korzenia.
2. Jeśli dotarliśmy do liścia lub zadanej głębokości, zwracamy wartość dla gracza, który aktualnie wykonuje ruch. Dla liścia jest to wynik gry, natomiast przy dojściu do limitu głębokości przeszukiwania - wartość heurystyki oceny.
3. W przeciwnym razie, będąc w aktualnym stanie gry, obliczamy wartość każdego z potomków wierzchołka rekurencyjnym wywołaniem Minimax (od kroku 2).
4. Obecnemu wierzchołkowi przypisujemy wartość maksymalną lub minimalną spośród jego potomków, w zależności od tego, którego gracza reprezentuje ten wierzchołek.

Algorytm Minimax można zoptymalizować za pomocą cięcia alfa-beta, które polega na zatrzymaniu rekurencyjnego przeglądania pewnej części drzewa, gdy określone warunki pozwalają na stwierdzenie, że dalsze przeglądanie nie ma już sensu.

2.3 Alfa-beta cięcie

Algorytm cięcia alfa-beta (*alpha-beta pruning*) jest rozwinięciem algorytmu Minimax, które pozwala na pominięcie rozpatrywania niepotrzebnych węzłów w drzewie gry a tym samym skrócenie czasu przeszukiwania. Algorytm cięcia alfa-beta działa poprzez eliminowanie gałęzi drzewa, które z pewnością nie są optymalne, ponieważ zawierają wyłącznie ruchy, które nie poprawią wartości funkcji wartości dla danego węzła. Algorytm ten pozwala znacznie zmniejszyć liczbę węzłów, które muszą zostać przeglądnięte podczas przeszukiwania drzewa gry.

Główne kroki algorytmu:

1. Uruchom algorytm Minimax na drzewie gry, przeszukując je rekurencyjnie od korzenia.

2. Przechowuj dwie wartości: alfa i beta, początkowo ustawione, odpowiednio, na minus i plus nieskończoność.

3. Dla obecnego wierzchołka drzewa, jeśli jest to wierzchołek poziomu maksymalizującego, wykonaj kroki 4.-6. W przeciwnym razie, wykonaj kroki 7.-9.

4. Dla każdego potomka wierzchołka, wywołaj algorytm cięcia alfa-beta rekurencyjnie (od kroku 3).

5. Jeśli wartość zwrócona przez algorytm jest większa niż alfa, zaktualizuj wartość alfa na wartość zwróconą.

6. Jeśli wartość beta jest mniejsza lub równa alfa, przerwij przeglądanie potomków i zwróć wartość alfa.

7. Dla każdego potomka wierzchołka, wywołaj algorytm cięcia alfa-beta rekurencyjnie (od kroku 3).

8. Jeśli wartość zwrócona przez algorytm jest mniejsza niż beta, zaktualizuj wartość beta na wartość zwróconą.

9. Jeśli beta jest mniejsza lub równa alfa, przerwij przeglądanie potomków i zwróć wartość beta.

3 Zadania

Halma to gra planszowa dla od dwóch do czterech graczy, rozgrywana na planszy o wymiarach 16x16 pól, zasady gry są opisane np. w: `https://pl.wikipedia.org/wiki/Halma`

Korzystając z wiadomości na tej liście oraz podanych na wykładzie, napisz program, który gra w Halmę. Program powinien na standardowym wejściu przyjmować 16 linii składających się z 16 odzielonych spacjami liczb – 1 oznacza pionek gracza pierwszego, 2 oznacza pionek gracza drugiego, 0 oznacza pole puste. Przyjmij, że na wejściu liczba '1' oraz '2' jest równa, następuje tura gracza '1'. Na standardowym wyjściu program powinien zwrócić 16 linii składających się z 16 liczb – 1 oznacza dysk gracza pierwszego, 2 oznacza dysk gracza drugiego, w 17 linii powinna być informacja o liczbie rund oraz o tym, który z graczy wygrał. Na standardowym wyjściu błędów powinna zostać zwrócona liczba odwiedzonych węzłów drzewa decyzyjnego oraz czas działania algorytmu.
Punktacja:

1. poprawne zdefiniowanie stanu gry i funkcji generującej możliwe ruchy dla danego stanu i gracza (10 punktów)

2. zbudowanie zbioru heurystyk oceny stanu gry dla każdego z graczy, każdy z graczy powinien mieć przynajmniej 3 różne strategie (20 punktów)

3. implementacja metody Minimax z punktu widzenia gracza 1 (30 punktów)

4. implementacja alfa-beta cięcia z punktu widzenia gracza 1 (40 punktów)

5. modyfikacja programu tak, by wykonywał tylko jeden ruch (z punktu widzenia gracza, którego jest kolej) co umożliwi rozegranie partii pomiędzy dwoma programami (np. dwoma wywołaniami tej samej aplikacji) oraz zastosowanie heurystyk adaptacyjnie zmieniających strategię gracza w celu osiągnięcia zwycięstwa (**dodatkowe** 20 punktów).

Do zadania przygotuj raport zawierający opis teoretyczny metody, formalne sformułowanie problemu z ograniczeniami, stanu gry i drzewa decyzyjnego oraz opis idei rozwiązania wraz z krótkim przykładem. W raporcie wskaż wykorzystane materiały źródłowe oraz krótko opisz biblioteki wykorzystane przy implementacji. W podsumowaniu raportu dodatkowo opisz napotkane problemy implementacyjne przy wykonywaniu zadania. Raport wyślij prowadzącemu przynajmniej na 24 godziny przed oddaniem listy.

4 Literatura

- Halma
- J.F. Nordstrom – Introduction to Game Theory
- M. Maschler et al., Game Theory
- N. Nisan et al., Algorithmic Game Theory
- M. J. Osborne, An Introduction to Game Theory