algorytmy rozwiązywania gier o sumie zerowej

Sztuczna inteligencja i inżynieria wiedzy

Rafał Behrendt 246643

10.05.2020

# Implementacja

Zadaniem było implementacja gry Mankala wraz z algorytmami Minimax i alfa-beta cięć. Zasady gry: <https://pl.wikipedia.org/wiki/Mankala>. Gra wykonana w silniku Unity – narzędzie do prostego tworzenia gier. Nad rozgrywką i obsługą interfejsu panuje obiekt GameController. Istnieje instancja gracza, który może wykonywać ruchy tylko jak jest jego kolejka oraz rozgrywka trwa. Obiekt AI dziedziczy po obiekcie Player i ma dodatkowo zaimplementowane metody rozwiązywania gry – maksymalizacji minimalnego zysku. Obsługę aktualnego stanu planszy sprawuje GameServer, który zmienia stan planszy według ustalonych zasad i nasłuchuje na ruchy graczy.

Zaimplementowane zostały dwie metody rozwiązywania gier – algorytm Minimax oraz algorytm alf-beta cięć. W pierwszym kroku obiekt AI buduje drzewo o zadanej głębokości, na podstawie dozwolonych ruchów. Następnie w zależności od przyjętej metody wykonuje algorytm Minimax, lub alfa-beta cięć. Algorytm Minimax przechodzi w dół drzewa i wybiera w danej gałęzi dziecko:

* O najniższym możliwym wyniku – jeżeli na danym poziomie jest ruch gracza
* O najwyższym możliwym wyniku – jeżeli na danym poziomie jest ruch przeciwnika

Jak że przeciwnik zawsze wybierze najlepszy rezultat dla siebie (a tym samym najgorszy dla aktualnego gracza) powyższą metodą maksymalizowany jest minimalny wynik.

Algorytm alfa-beta cięć z kolei pomija gałęzie których nie musi rozpatrywać. Jeżeli w kolejnej gałęzi jeden z wyników jest niższy niż poprzedni, a jest ruch przeciwnika, to nie musimy rozpatrywać reszty ruchów w tej gałęzi. Podobnie robimy w przypadku naszego ruchu – jeżeli w naszym ruchu wynik jest niższy, nie ma sensu wybierać tej gałęzi.

Ocena ruchu polega na wybraniu najbardziej opłacalnego. W programie zostały zaimplementowane następujące heurystyki:

* Różnica między wynikami graczy
* Premiowanie przejmowania kamieni
* Premiowanie ruchów które kończą się zwycięstwem gracza

Różnica między wynikami gracza jest podstawową heurystyką. Obowiązuje ona zawsze, a do niej przypisane mogą zostać premie za przejmowanie kamieni (przejęcie kamieni jest warte więcej punktów niż w rzeczywistości) i/lub kończenie gry, gdy zwycięstwo jest widoczne w drzewie.

# Badania

Przeprowadzonych zostało 20 pomiarów prędkości działania algorytmu minimax i alfa -beta dla różnych algorytmów i heurystyk. Zapisane zostały czasy wykonania i liczba ruchów.

## Różnica między wynikami graczy:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Minmax** | |  |  | **Alfabeta** | |
| **Lp.** | **Czas [ms]** | **Liczba ruchów** |  | **Lp.** | **Czas [ms]** | **Liczba ruchów** |
| **1** | **54** | **19** |  | **1** | **10** | **19** |
| **2** | **38** | **21** |  | **2** | **11** | **18** |
| **3** | **37** | **21** |  | **3** | **10** | **22** |
| **4** | **78** | **17** |  | **4** | **12** | **15** |
| **5** | **84** | **32** |  | **5** | **9** | **19** |
| **6** | **30** | **16** |  | **6** | **22** | **37** |
| **7** | **29** | **16** |  | **7** | **2** | **11** |
| **8** | **62** | **25** |  | **8** | **14** | **14** |
| **9** | **106** | **20** |  | **9** | **11** | **20** |
| **10** | **73** | **24** |  | **10** | **2** | **12** |
| **11** | **30** | **20** |  | **11** | **18** | **24** |
| **12** | **144** | **22** |  | **12** | **20** | **39** |
| **13** | **28** | **22** |  | **13** | **16** | **25** |
| **14** | **45** | **20** |  | **14** | **16** | **21** |
| **15** | **24** | **22** |  | **15** | **3** | **17** |
| **16** | **40** | **21** |  | **16** | **13** | **21** |
| **17** | **41** | **15** |  | **17** | **7** | **11** |
| **18** | **65** | **20** |  | **18** | **11** | **27** |
| **19** | **54** | **34** |  | **19** | **15** | **32** |
| **20** | **51** | **30** |  | **20** | **9** | **30** |
| **Średnia** | **55,65** | **21,85** |  | **Średnia** | **11,55** | **21,7** |