PSZT

Grzegorz Olechwierowicz, Mikołaj Florkiewicz
Michał Sypetkowski

2016-12-21

1 Problem

Zadanie polega na stworzeniu automatycznego gracza w warcaby opierającego swoją strategię na algorytmach ewolucyjnych. W celach weryfikacji należy porównać go do podejścia opartego na MiniMax

2 Rozwiązanie

2.1 Algorytm ewolucyjny

Jeden wektor w tym algorytmie jest reprezentowany przez tablicę współczynników (IlośćOsiągalnychPozycjiNaPlanszy x 3).
Każda z 3 wartości w tym ostatnim wymiarze odpowiada na pytanie: "Ile dodać do punktacji planszy gdy na danym polu stoi: [0] - swój pionek [1] - wrogi pionek [2] - pole jest puste ?"
Algorytm nie uwzględnia damek (nie rozróżnia ich od zwykłych pionków). Mutacja jest dokonywana losowo.
Uczenie jest zaimplementowane w oddzielnym pliku.
Funkcja celu to winRatio = (ilośćGierWygranych + dodatkowePunkty)/ ilośćWszystkichGier gdzie dodatkowe punkty liczą się tak:
ilośćRemisów * 0.5
Dodatkowo program umożliwia dodanie flagi "-e", która zmieni funkcję celu, dodając następujące wartości:
+ 0.05 * ilośćPozostałychPionkówPoWygranej + 0.15 * ilośćPozostałychDamekPoWygranej Selekcja - algorytm 1+1

2.2 Mini-max

Jest to standardowy algorytm w którym maksymalizujemy własną, heurystyczną funkcję oceniającą sytuację na planszy, a minimalizujemy przeciwnika. W naszym przypadku funkcja celu jest to różnica ilości własnych pionków od ilości pionków przeciwnika. Algorytm nie implementuje alfa-beta w celu odcinania części drzewa gry. Aktualna implementacja umożliwia przewidywanie 4-5 ruchów w przód, bez większych "zgrzytów" na dzisiejszych komputerach.

3 Sprawy techniczne

Algorytmy są zaimplementowane w języku Python. Dostępne jest również GUI, które pokazuje aktualną grę. W celach porównawczych efektywnościu obu wyżej wymienionych rozwiązań GUI nie jest dostępne.