Zastosowanie algorytmu UCT do stworzenia sztucznej inteligencji grajacej w *Connect4*

Patryk Fijałkowski

Mateusz Burczaniuk

Zaimplementowane strategie agentów

- Random
- Greedy
- Algorytm UCT
- UCB-Minimal (usprawnienie do UCT)
- UCB-V (usprawnienie do UCT)

UCB-V

$$\frac{w_i}{n_i} + \sqrt{2\frac{\sigma_i^2 \cdot \varepsilon}{n_i}} + c\frac{3 \cdot \varepsilon}{n_i}$$

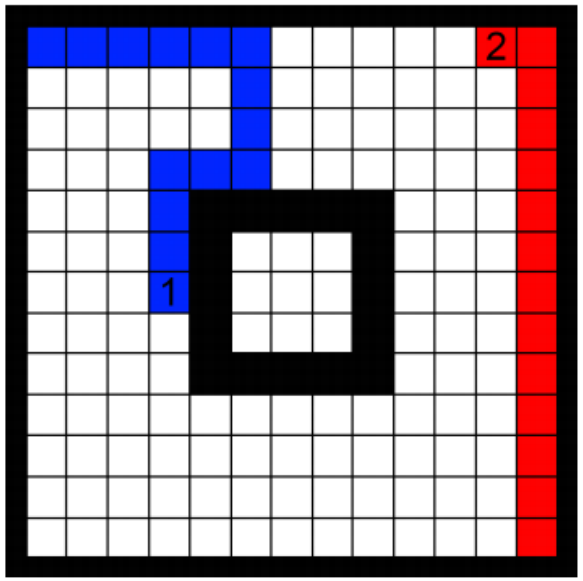
- Korzysta z wariancji (unikamy rozbieżności)
- Funkcja eksploracji

$$\varepsilon = \zeta \cdot \ln N_i$$

UCB-Minimal

$$\frac{w_i}{n_i} + \frac{C_1}{n_i^{C_2}}$$

- Głowne założenie: prostota
- C1 balansuje eksploatację i eksplorację
- C2 koryguje wpływ eksploracji
- Dobrze się sprawdza w grach o niskim branching factor (jak Tron)



Algorytm zachłanny

```
if(OpponentWinsNextTurn())
    BlockHim();
else
    PerformMostValuableMove();
```

```
private readonly static int[,] EVALUATION_TABLE = new int[,]{
        {3, 4, 5, 7, 5, 4, 3},
        {4, 6, 8, 10, 8, 6, 4},
        {5, 8, 11, 13, 11, 8, 5},
        {5, 8, 11, 13, 11, 8, 5},
        {4, 6, 8, 10, 8, 6, 4},
        {3, 4, 5, 7, 5, 4, 3}};
```

PLUSY:

- Łatwość implementacji
- Szybkość
- Nie popełnia najbardziej trywialnego błędu, więc nadaje się do benchmarkowania
- Jest deterministyczny, więc nadaje się do benchmarkowania

MINUSY:

- Przewiduje tylko 1 ruch przeciwnika do przodu

Jak przeprowadziliśmy badania?

$$REWARD(m) = \begin{cases} 0.8 \cdot (1 - \frac{m-7}{35}) & \text{jeśli agent wygrał} \\ \frac{m-7}{35} - 1 & \text{w p.p.} \end{cases}$$

- Funkcja REWARD przyjmująca wartośći z zakresu [-1; 0.8], zależna od liczby ruchów
- Dla każdego wariantu UCT – uruchomienie na 20 różnych seedach

Optymalne parametry - podsumowanie

Algorytm	Ocena	Odsetek wygranych
UCBV (1.4, 0.5)	0.560	100%
UCB1 (2)	0.552	100%
UCB1 (1.41)	0.529	100%
UCBV (2, 0.5)	0.510	100%
UCB1 (1.7)	0.484	93%
UCBV (1.7, 0.6)	0.478	87%
UCBV (1.5, 0.5)	0.462	93%
UCBV (0.9, 0.9)	0.457	87%
UCB1 (3)	0.438	80%
UCBV (1.1, 1.1)	0.427	80%

Algorytm	Średnia ocena	Średni odsetek wygranych
UCB-V	0.306	74%
UCB1	0.112	61%
UCB-Minimal	-0.472	18%

- Łącznie 10.000 rozgrywek
- Zawsze 15.000 iteracji MCTS
- Klęska UCB-Minimal

Optymalne parametry - szczegóły

Wartość c	Ocena
2	0.552
1.41	0.529
1.7	0.484
1.6	0.425
1.5	0.415
1.45	0.401
1	0.153
0.09	-0.448
0.01	-0.563
0	-0.702

UCB1

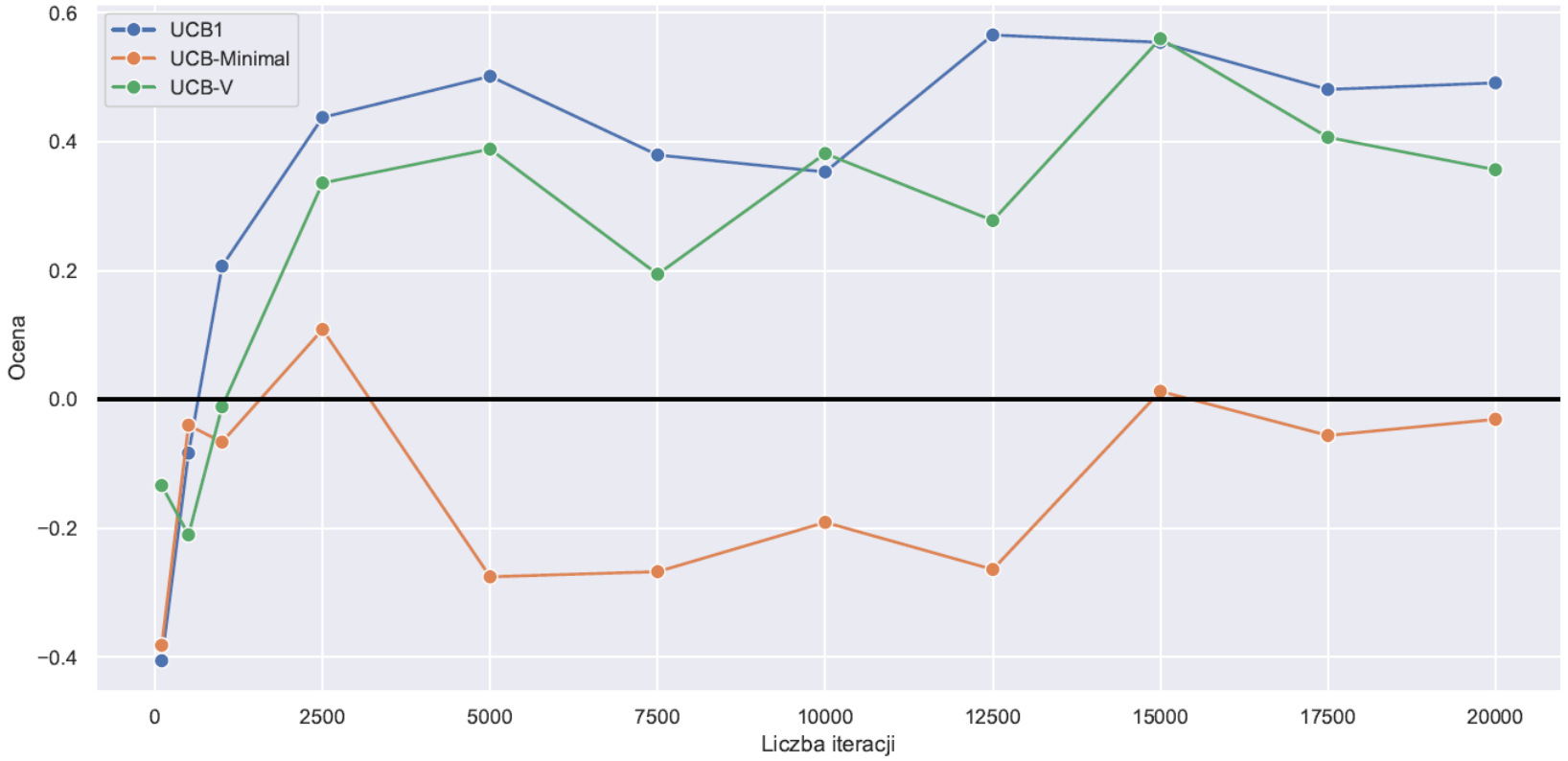
Wartość c	Wartość ζ	Ocena
1.4	0.5	0.560
2	0.5	0.510
1.68	0.54	0.494
1.7	0.6	0.478
1.5	0.5	0.462
0.9	0.9	0.457
1	1	0.366
1.5	0.4	0.289
120	30	-0.007
0.1	0.05	-0.513

UCB-V

Wartość C_1	Wartość C_2	Ocena
11	1	-0.091
2.5	1	-0.272
2.9	1.4	-0.289
12	5	-0.297
8.4	1.8	-0.349
3	2	-0.366
1.8	8.4	-0.452
3	3	-0.508
26	26	-0.522
9.4	2.8	-0.556

UCB-Minimal

Iteracje MCTS



Najlepszy wariant

- Zwycięzca: UCB1 z c=2, 15000 iteracji
- UCB-V często doprowadzał do remisów (13.4%)

Algorytm	Ocena
UCB1 (2)	0.3824
UCB-V (2, 0.5)	0.142
UCB-V (1.4, 0.5)	-0.2591
UCB1 (1.41)	-0.2653

	UCB-V (1.4, 0.5)	UCB1 (2)	UCB1 (1.41)	UCB-V (2, 0.5)
UCBV (2, 0.5)	0.1316	-0.0597	0.0701	
UCB1 (1.41)	-0.0336	-0.1616		
UCB1 (2)	0.1611			
UCB-V (1.4, 0.5)				

	UCB-V (1.4, 0.5)	UCB1 (2)	UCB1 (1.41)	UCB-V (2, 0.5)
UCB-V (2, 0.5)	7.5%	16.25%	16.25%	
UCB1 (1.41)	12.5%	5%		
UCB1 (2)	8.75%			
UCB-V (1.4, 0.5)				

Źródła

- Francis Maes, Louis Wehenkel, Damien Ernst, *Automatic Discovery of Ranking Formulas for Playing with Multi-armed Bandits*, European Workshop on Reinforcement Learning, Athens, Greece, September 9–11, 2011.
- Pierre Perick, David L. St-Pierre, Francis Maes, Damien Ernst, *Comparison of Different Selection Strategies in Monte-Carlo Tree Search for the Game of Tron*, IEEE Conference on Computational Intelligence and Games, Granada, Spain, September 12–15, 2012.
- Jean-Yves Audibert, Remi Munos, Csaba Szepesv´ari, *Tuning Bandit Algorithms in Stochastic Environments*, Algorithmic Learning Theory 18th International Conference, Sendai, Japan, October 1–4, 2007.

Dziękujemy za uwagę

☺