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

Dokumentacja końcowa

Patryk Fijałkowski

Mateusz Burczaniuk

8 czerwca 2020

1 Opis problemu

2 Sposób przeprowadzenia testów

Każdy z przeprowadzonych testów będzie w formie pełnej rozgrywki *Connect4* pomiędzy dwoma agentami. W celu oszacowania, jak dobre decyzje podejmował agent rozpoczynający rozgrywkę, zdefiniowano funkcję $REWARD$ opisaną równaniem (1). Funkcja jest zależna od liczby ruchów m w danej partii i przyjmuje wartości z zakresu $[-1; 0.8]$. Funkcja nie może przyjąć wartości większych od 0.8 ze względu na przewagę pierwszego gracza spowodowaną faktem, że rozpoczyna on rozgrywkę.

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

3 Weryfikacja hipotez

3.1 Optymalne parametry

W celu wyznaczenia najlepszych parametrów dla każdego z trzech analizowanych wariantów UCT, przeprowadzono rozgrywki z algorytmem heurystycznym. Dla każdego wariantu sprawdzono 20 próbnych konfiguracji parametrów, a każdą z konfiguracji sprawdzono dla 15 wartości ziarna generatora liczb losowych, by zmniejszyć wpływ losowości na działanie algorytmu. W każdym teście algorytm MCTS wykonywał 15.000 iteracji. Jako ocenę każdej konfiguracji przyjęto średnią arytmetyczną wartości funkcji $REWARD$ otrzymanych po zakończonych rozgrywkach. Wyniki testów ukazane są w tabelach 1 - 3.

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

Tab. 1: Ocena algorytmu UCB1 w zależności od parametru eksploracji

Jak widać w tabeli 1, algorytm UCB1 został najlepiej oceniony dla wartości parametru eksploracji $c = 2$. Wraz ze zwiększaniem i zmniejszaniem wartości parametru, algorytm był oceniany gorzej. Ponadto, w przypadku $c = 0$, kiedy algorytm eksploatował jedynie najbardziej obiecujące ruchy, podejmował najgorsze decyzje. Wartość sugerowana przez autorów algorytmu w [2] ($c = 1.41$) została oceniona nieznacznie gorzej względem $c = 2$.

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

Tab. 2: Ocena algorytmu UCB-V w zależności od parametrów c i ζ

Analizując tabelę 2, wnioskuje się, że algorytm działa najlepiej dla wartości ($c = 1.4, \zeta = 0.5$). Wartości sugerowane przez [5] i [6] zostały ocenione gorzej. Podczas testów używano wyłącznie sugerowanej *funkcji eksploracji*, przyjęto $\varepsilon = \zeta \cdot \ln N_i$.

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

Tab. 3: Ocena algorytmu UCB-Minimal w zależności od parametrów C_1 i C_2

Algorytm UCB-Minimal wypadł najgorzej w porównaniu – nawet najlepiej dobrane wartości parametrów C_1 i C_2 skutkowały ujemnym bilansem zwycięstw. Zgodnie z tabelą 3, algorytm gra najlepiej w konfiguracji ($C_1 = 11, C_2 = 1$). Z kolei referencyjne wartości parametrów, zaczerpnięte odpowiednio z [4] i [5], wiążą się z niższą oceną algorytmu.

Algorytm	Ocena
UCBV $(1.4, 0.5)$	0.560
UCB1 (2)	0.552
UCB1 (1.41)	0.529
UCBV $(2, 0.5)$	0.510
UCB1 (1.7)	0.484
UCBV $(1.7, 0.6)$	0.478
UCBV $(1.5, 0.5)$	0.462
UCBV $(0.9, 0.9)$	0.457
UCB1 (3)	0.438
UCBV $(1.1, 1.1)$	0.427

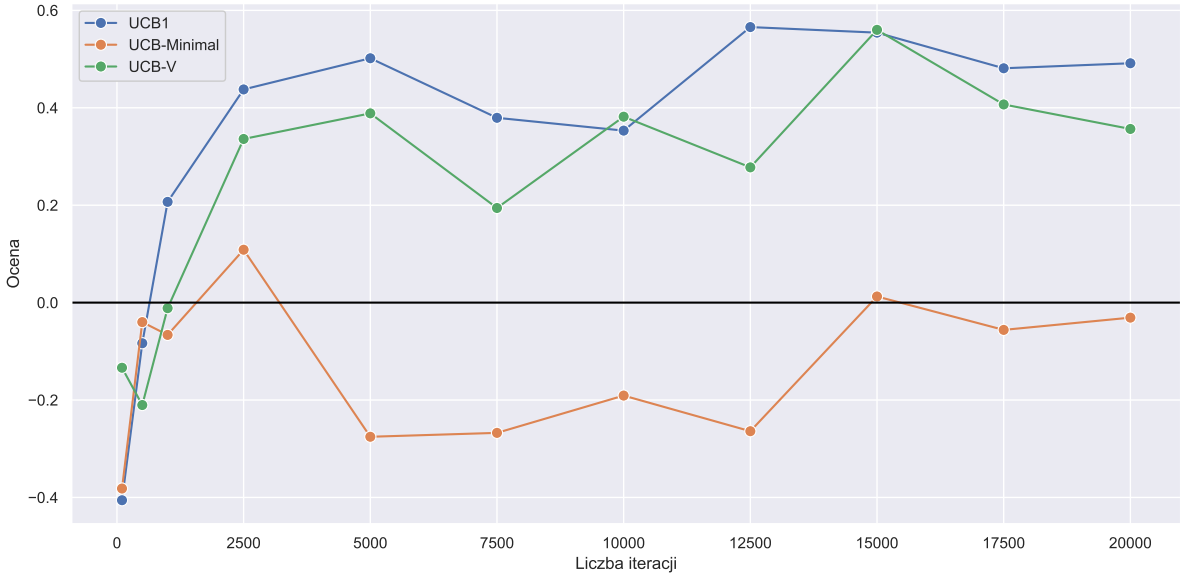
Tab. 4: Ocena najlepszych konfiguracji algorytmów

Tabela 4 prezentuje, który wariant UCT został najlepiej oceniony w rozgrywkach z algorytmem heurystycznym. W celu porównania skuteczności każdego z algorytmów, wyznaczono również średnie arytmetyczne wartości funkcji $REWARD$ po wszystkich rozgrywkach. Uzyskano odpowiednio oceny:

- UCB-V – 0.306,
- UCB1 – 0.112,
- UCB-Minimal – -0.472.

3.2 Wpływ iteracji

W celu sprawdzenia, jak liczba iteracji MCTS wpływa na poprawę decyzji algorytmu, przeprowadzono testy dla najbardziej optymalnych konfiguracji parametrów każdego wariantu: UCB1, UCB-V i UCB-Minimal. Oceniono rozgrywki z algorytmem heurystycznym po wykonaniu 100, 500, 1000, 2500, 5000, 7500, 10000, 12500, 15000, 17500 i 20000 iteracji.



Rys. 1: Wpływ liczby iteracji na jakość podejmowanych decyzji

Wyniki analiz zostały zaprezentowane na wykresie na rysunku 1. Ocena każdego z algorytmów osiąga stosunkowo wysokie wartości przy liczbie iteracji 2500, następnie maleje, by przy liczbie iteracji 15000 osiągnąć najwyższą wartość. Wartym zauważenia jest również fakt, że algorytm UCB-V osiąga relatywnie najlepsze wyniki przy najniższych zakresach iteracji.

3.3 Najlepszy wariant

	UCBV $(1.4, 0.5)$	UCB1 (2)	UCB1 (1.41)	UCBV $(2, 0.5)$
UCBV $(2, 0.5)$	0	0	0	
UCB1 (1.41)	0	0		
UCB1 (2)	0			
UCBV $(1.4, 0.5)$				

Tab. 5: Ocena najlepszych konfiguracji algorytmów

Najlepszy z wariantów zostanie wyłoniony na podstawie rozegrania partii każdy z każdym.

Literatura

[1] Victor Allis, *A Knowledge-based Approach of Connect-Four*, Department of Mathematics and Computer Science Vrije Universiteit Amsterdam, The Netherlands.

[2] Levente Kocsis, Csaba Szepesvári, *Bandit based Monte-Carlo Planning*, European Conference on Machine Learning, Berlin, Germany, September 18–22, 2006.

[3] Steven James, George Konidaris, Benjamin Rosman, *An Analysis of Monte Carlo Tree Search*, University of the Witwatersrand, Johannesburg, South Africa.

[4] 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.

[5] 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.

[6] Jean-Yves Audibert, Remi Munos, Csaba Szepesvári, *Tuning Bandit Algorithms in Stochastic Environments*, Algorithmic Learning Theory 18th International Conference, Sendai, Japan, October 1–4, 2007.