

Zastosowanie algorytmu UCT do stworzenia sztucznej
inteligencji grającej w *Connect4*
Wstępna dokumentacja projektu

Patryk Fijałkowski
Mateusz Burczaniuk

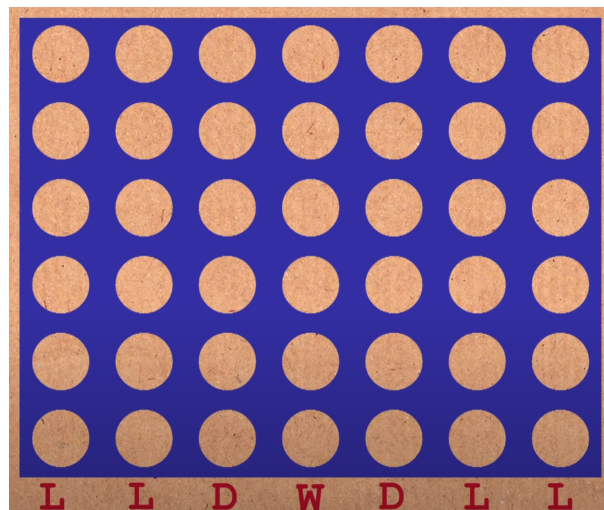
9 maja 2020

1 Opis *Connect4*

Connect4 jest planszową grą logiczną dla dwóch graczy. Rozgrywka jest prowadzona na planszy o szerokości siedmiu pól i długości sześciu pól. Gracze naprzemiennie wrzucają żetony do jednej z siedmiu kolumn, a żetony zajmują najniższą możliwą pozycję w danej kolumnie. Grę wygrywa ten z graczy, który ułoży cztery żetony w poziomie, pionie lub ukosie. W sytuacji w której plansza się zapełni, a nie utworzy się żadna czwórka, dochodzi do remisu.

Bazując na [1], wynik rozgrywki w *Connect4* jest przesądzony już po pierwszym ruchu przy założeniu, że gracze podejmują bezbłędne decyzje. Rysunek 1 przedstawia planszę z podpisanymi kolumnami, ilustrując jak pierwszy ruch wpływa na rozgrywkę. Jeżeli pierwszy gracz wrzuci swój żeton do jednej z kolumn podpisanych *L*, to przegra, *W* – wygra, a *D* – doprowadzi do remisu.

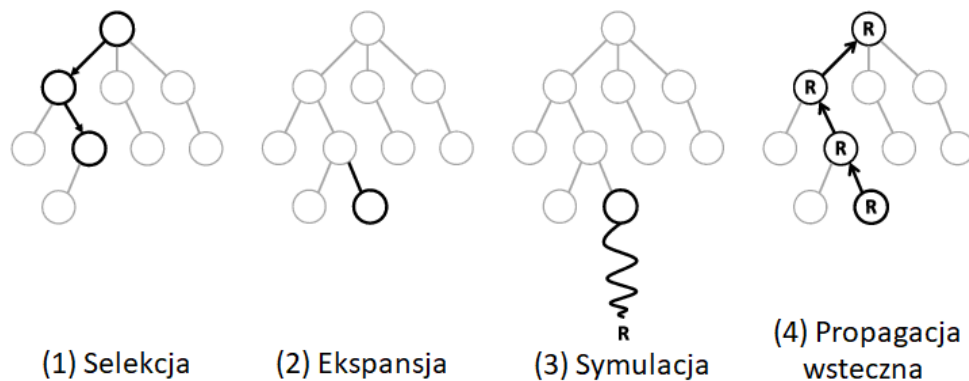
Powołując się na [1], wszystkich możliwych ustawień żetonów na planszy jest około $4.5 \cdot 10^{13}$. Jest to zbyt duża liczba ustawień, którymi miałby operować algorytm grający w *Connect4*. W związku z tym, stosuje się rozwiązania heurystyczne, takie jak Monte-Carlo Tree Search.



Rys. 1: Plansza *Connect4*

2 Algorytmy MCTS

Monte-Carlo Tree Search to heurystyka, której celem jest podejmowanie decyzji w pewnych zadaniach sztucznej inteligencji, na przykład wybieranie ruchów w grach. Metoda jest oparta na przeszukiwaniu możliwych stanów gry zapisanych w wierzchołkach drzewa i losowym symulowaniu rozgrywek. Algorytmy MCTS opierają się na rozbudowywaniu drzewa ze stanami gry przez iteracyjne wykonywanie czterech faz. Jednym z najpopularniejszych wariantów MCTS jest algorytm UCT. Pseudokod opisany w Listingu 1 oraz implementacja MCTS w projekcie bazują na [2]. Przykład działania algorytmu ze szczególnym uwzględnieniem kolejnych faz znajduje się na Rysunku 2.



Rys. 2: Fazy MCTS, źródło: [3]

1. **Faza selekcji** (wiersz 6 w listingu) – wybór pewnego liścia drzewa. Rozdział 2.1 opisuje jeden ze sposobów na wybranie wierzchołka w tej fazie.
2. **Faza ekspansji** (wiersz 7 w listingu) – utworzenie wierzchołków potomnych dla wierzchołka wybranego w fazie selekcji. Tworzone wierzchołki odpowiadają stanom możliwym do uzyskania przez wykonanie jednego ruchu ze stanu rodzica.
3. **Faza symulacji** (wiersz 10 w listingu) – rozegranie partii składającej się z losowych ruchów ze stanu jednego z wierzchołków utworzonych w poprzedniej fazie. Rozgrywana jest ona do końca, czyli do wyłonienia zwycięzcy lub spowodowania remisu, lub jest ucinana po pewnej liczbie ustalonych ruchów i wynik gry jest ewaluowany przez pewną funkcję.
4. **Faza propagacji wstecznej** (wiersz 11 w listingu) – aktualizacja informacji na temat wierzchołków na ścieżce od liścia, z którego rozpoczęto symulację, do korzenia drzewa. Główną przekazywaną wartością jest wynik symulacji.

```

1 def find_next_move(curr_state):
2     iterations_counter = 0
3     tree = initialize_tree(curr_state)
4
5     while iterations_counter < max_iterations_counter:
6         curr_node = select a leaf from tree
7         create child nodes from curr_node
8         if curr_node has children:
9             curr_node = random child of curr_node
10        playout_result = simulate random playout from curr_node
11        update tree according to playout_result
12        iterations_counter++
13
14    best_state = select best child(tree.root)
15    return best_state
16

```

Listing 1: Pseudokod algorytmu Monte Carlo Tree Search

2.1 Algorytm UCT

UCT jest wariantem metody MCTS, który stara się zachować równowagę między eksploatacją bardziej obiecujących ruchów a eksploracją tych rzadko odwiedzonych. Formuła, która odpowiada za wyznaczenie najbardziej obiecującego wierzchołka w fazie wyboru MCTS jest przedstawiona jako wyrażenie (1).

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}} \quad (1)$$

W wyrażeniu (1), indeks i odnosi się do liczby wykonanych przez algorytm iteracji, czyli czterech faz MCTS. W pierwszym składniku sumy wyrażenia (1), licznik w_i oznacza sumę wszystkich wypłat w danym węźle, a mianownik n_i oznacza liczbę rozegranych symulacji. Zatem ułamek ten przyjmuje wartości większe dla ruchów o większej średniej wygranej, co odpowiada ze eksploatację drzewa. Drugi składnik sumy wyrażenia (1) przyjmuje wartości większe dla wierzchołków, dla których wykonano mniej symulacji i odpowiada eksploracji drzewa. $N_i = \sum_i n_i$, a c jest parametrem eksploracji, który może być dostosowany do badanego problemu.

2.1.1 Usprawnienie UCB-Minimal

Przedstawione w [4] usprawnienie UCB-Minimal polega na zastosowaniu wzoru (2) w celu wyboru najbardziej obiecującego wierzchołka w fazie selekcji.

$$\frac{w_i}{n_i} + \frac{C_1}{n_i^{C_2}} \quad (2)$$

W wyrażeniu (2), C_1 jest parametrem balansującym eksploatację i eksplorację drzewa. Z kolei C_2 to parametr, którego zadaniem jest skorygować zmniejszenie wpływu eksploracji, które następuje z kolejnymi iteracjami algorytmu. Dobór parametrów powinien zostać

wykonany empirycznie – autorzy [4] sugerują wartości $C_1 = 2.5$, $C_2 = 1$ jako ogólnie optymalne. Przykładowo, badania autorów [5] wykazały, że najlepiej sprawdzają się wartości $C_1 = 8.4$, $C_2 = 1.8$.

2.1.2 Usprawnienie UCB-V

Przedstawione w [6] usprawnienie UCB-V opiera swoją skuteczność na wykorzystaniu wariancji wypłat powodując, że algorytm w fazie selekcji wybiera jak najlepsze i jak najmniej zróżnicowane ruchy w kontekście oczekiwanej wypłaty. UCB-V polega na zastosowaniu wzoru (3) w celu wyboru najbardziej obiecującego wierzchołka w fazie selekcji.

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

W wyrażeniu (3), σ_i^2 oznacza wariancję wypłat w danym węźle, a ε to tak zwana *funkcja eksploracji*. Zgodnie z sugestią autorów [6], za *funkcję eksploracji* przyjęta zostanie postać zaprezentowana w (4), gdzie ζ jest parametrem, który powinien zostać dopasowany empirycznie.

$$\varepsilon = \zeta \cdot \ln N_i \quad (4)$$

Autorzy [6] sugerują wartości parametrów $c = 1$, $\zeta = 1$ jako optymalne dla prezentowanego usprawnienia. Z drugiej strony, w zastosowaniu go do gry *Tron*, autorzy pracy [5] najlepsze efekty zaobserwowali dla wartości parametrów $c = 1.68$, $\zeta = 0.54$.

3 Podejście heurystyczne

Proponowane podejście heurystyczne jest podejściem zachłannym. Przy podejmowaniu decyzji o następnym ruchu algorytm będzie starał się ułożyć kolejny żeton tak, by utworzyć jak najwyższą kolumnę lub jak najszerszy rząd. Algorytm w pierwszej kolejności będzie sprawdzał, jak wysokie kolumny i jak szerokie rzędy może uzyskać (odrzucając te potencjalne rzędy i kolumny, które nie osiągną wielkości czterech żetonów). Następnie, algorytm wybierze największy rząd lub kolumnę. W przypadku więcej niż jednego rzędu czy kolumny o największym rozmiarze, wybrana zostanie pierwsza możliwość w kolejności analizowania.

4 Hipotezy badawcze

Projekt ma na celu zbadać, czy algorytmy z grupy MCTS można skutecznie zastosować do grania w *Connect4*. Jeśli tak jest, agent podejmujący decyzje przy użyciu tego rodzaju algorytmów powinien wygrywać co najmniej 75% rozgrywek z algorytmem zachłannym.

Ponadto, sprawdzone zostanie, jak liczba iteracji wpływa na jakość decyzji podejmowanych przez każdy z trzech wariantów algorytmu UCT. Przypuszcza się, że wraz ze wzrostem liczby iteracji, stosunek wygranych do rozegranych partii z algorytmem zachłannym powinien rosnać co najmniej liniowo, by od pewnej liczby iteracji wszystkie rozgrywane partie były wygrane.

Wyznaczone zostaną również optymalne wartości pięciu parametrów, wymienionych poniżej.

1. parametru c w algorytmie UCT,
2. parametrów C_1, C_2 w algorytmie UCT z usprawnieniem UCB-Minimal,
3. parametrów c, ζ w algorytmie UCT z usprawnieniem UCB-V.

W celu sprawdzenia przedstawionych hipotez zostaną przeprowadzone testy. Jako że na decyzje podejmowane przez algorytmy MCTS silnie wpływa czynnik losowości, dla każdej konfiguracji zostanie przeprowadzone 20 testów. W pierwszej kolejności wyznaczone zostaną najbardziej optymalne wartości parametrów przy ustalonej liczbie iteracji. W tym celu porównane zostaną liczby wygranych rozgrywek pomiędzy agentami korzystającymi z trzech wariantów MCTS i algorytmu zachłannego. Sprawdzone zostanie 5 wartości każdego parametru (co daje 25 konfiguracji w przypadku UCB-Minimal i UCB-V). W następnej kolejności przeprowadzone zostaną testy mające na celu sprawdzenie wpływu iteracji algorytmu na jego skuteczność w zależności od liczby iteracji (z wykorzystaniem najbardziej optymalnych parametrów).

5 Harmonogram działań

W tabeli 1 przedstawiono planowany harmonogram działań podczas pracy nad projektem.

Tab. 1: Harmonogram pracy

Deadline	Przygotowane zadania
06.05.2020	Stworzenie dokumentacji projektu Zaimplementowanie struktur potrzebnych do operowania na drzewach
13.05.2020	Zaimplementowanie logiki gry
20.05.2020	Zaimplementowanie algorytmu UCT Przygotowanie aplikacji okienkowej
27.05.2020	Zaimplementowanie dwóch wariantów UCT
03.06.2020	Zaimplementowanie algorytmu heurystycznego Stworzenie raportu
10.06.2020	Przeprowadzenie eksperymentów w celu weryfikacji hipotez Weryfikacja postawionych hipotez

6 Projekt techniczny

Projekt zostanie sporządzony przy użyciu języka *C#*. Nie jest planowane użycie żadnych specjalistycznych bibliotek tego języka. Projekt będzie zrealizowany jako aplikacja okienkowa stworzona przy użyciu biblioteki *Windows Forms*. Aplikacja będzie działać w dwóch trybach:

1. Komputer kontra komputer. Użytkownik aplikacji będzie wybierał rodzaj dwóch agentów, którzy zagrają w *Connect4*.
2. Człowiek kontra komputer. W tym trybie użytkownik wybierze rodzaj agenta, z którym będzie mógł przeprowadzić rozgrywkę.

Dostępne rodzaje agentów to:

- UCB1 - decyzje agenta podejmowane będą przy użyciu algorytmu UCT.
- UCB-M - decyzje agenta podejmowane będą przy użyciu algorytmu UCT z usprawnieniem opisanym w rozdziale 2.1.1.
- UCB-V - decyzje agenta podejmowane będą przy użyciu algorytmu UCT z usprawnieniem opisanym w rozdziale 2.1.2.
- GRDY - decyzje agenta podejmowane będą przy użyciu algorytmu zachłannego opisanego w rozdziale 3.
- RNDM - decyzje agenta będą podejmowane losowo.

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.