Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa licencjacka

na kierunku Matematyka

NAJPIERW WYGENERUJ STRONĘ TYTUŁOWĄ

KTÓRA ZNAJDUJE SIĘ

Numer albumu 000000

promotor

W KATALOGU title_page

konsultacje

I OTRZYMANEGO PDF-A NAZWIJ titlepage.pdf I WSTAW DO KATALOGU Z SZABLONEM!

WARSZAWA 2018

....

podpis promotora

....

podpis autora

Streszczenie

Wizualizacja drzewa stanów algorytmu UCT

Streszczam.

Lorem ipsum dolor sit amet, consetetur sadipscing elit, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Słowa kluczowe: slowo1, slowo2, ...

Abstract

Visualization of UCT trees

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Keywords: keyword1, keyword2, ...

Warszawa, dnia .....

Oświadczenie

Oświadczam, że pracę inżynierską pod tytułem „Wizualizacja drzewa stanów algorytmu UCT", której promotorem jest mgr inż. Jan Karwowski, wykonałam/wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Spis treści

1. Wykaz najważniejszych oznaczeń i skrótów

2. Wstęp i cel pracy

2.1. Opis problemu klienta

2.2. Cel biznesowy

Algorytm UCT, będący usprawnieniem MCTS, jest powszechnie stosowanym algorytmem w sztucznej inteligencji. Jest metodą analizującą obiecujące ruchy na podstawie generowanego drzewa, która równoważy eksploatację najbardziej korzystnych z eksploracją mniej korzystnych decyzji. Każdemu wierzchołkowi drzewa odpowiada pewien stan rozgrywki, z którego algorytm rozgrywa losowe symulacje, rozszerzając potem drzewo o kolejne możliwe stany. Sposób, w jaki rozrasta się opisywane drzewo, jest kluczowy dla podejmowania przez algorytm jak najlepszych decyzji.

Celem projektu jest stworzenie aplikacji pozwalającej na wizualizację drzew algorytmu UCT. Aplikacja będzie pozwalała na wizualizowanie drzew generowanych podczas rozgrywania dwóch przykładowych gier (pozwalając przetestować rozwiązanie). Aplikacja powinna pozwalać na wizualizację drzew, ich sekwencji i róznic między kolejnymi drzewami w sekwencji. Powinna być możliwość płynnego przybliżania/oddalania i przewijania wizualizacji oraz zapisu aktualnego stanu do pliku graficznego - wszystko, aby klient mógł wygodnie korzystać z naszego programu.

Taki produkt pozwoliłby zrozumieć klientowi ideę i sposób działania algorytmu UCT.

2.3. Założenia projektowe

2.3.1. Założenia funkcjonalne

2.3.2. Założenia niefunkcjonalne

3. Teoria

3.1. Algorytmy MCTS

MCTS (Monte Carlo Tree Search) - heurystyka podejmowania decyzji w pewnych zadaniach sztucznej inteligencji, np. ruchów w grach. Najczęściej MCTS opiera się na jakimś wariancie metody UCT.

3.1.1. Opis grupy algorytmów

Algorytm jest opisany w formie pseudokodu w listingu 3.1. Nasza implementacja będzie oparta o pracę **Bandit based Monte-Carlo Planning**[1].

Algorytm Monte Carlo Tree Search opiera się na rozbudowywaniu drzewa ze stanami gry, poprzez iteracyjne wykonywanie czterech faz, opisanych poniżej.

1. **Faza wyboru** (linia 6 w listingu) - wybranie najbardziej obiecującego wierzchołka do rozrostu drzewa. Istotny w tej fazie jest balans pomiędzy eksploracją ruchów przeanalizowanych najdokładniej oraz eksploatacją jeszcze niezbadanych.

2. **Faza rozrostu** (linia 7 w listingu) - utworzenie wierzchołków potomnych dla najbardziej obiecującego wierzchołka. Tworzone wierzchołki odpowiadają stanom możliwym do uzyskania poprzez wykonanie jednego ruchu ze stanu wierzchołka obiecującego.

3. **Faza symulacji** (linia 8 w listingu) - rozegranie losowej rozgrywki ze stanu jednego z utworzonych wierzchołków utworzonych w poprzedniej fazie.

4. **Faza propagacji wstecznej** (linia 9 w listingu) - aktualizacja informacji wierzchołków na ścieżce od wierzchołka, z którego rozpoczęto symulację, do korzenia drzewa.

[1] *Bandit based Monte-Carlo Planning* - Levente Kocsis, Csaba Szepesvári, Berlin, Germany, September 18–22, 2006

```
def find_next_move(curr_state):
    iterations_counter = 0
    tree = initialize_tree(curr_state)

    while iterations_counter < max_iterations_counter:
        # selection(tree.root)
        curr_node = select promising node based on UCT formula

        # expansion(node)
        create child nodes from node

        # simulation(node)
        playout_result = simulate random playout from curr_node

        # backpropagation(node, playout_result)
        update tree according to playout_result

        iterations_counter++

    best_state = select best child(tree.root)
    return best_state
```

Listing 3.1: Pseudokod algorytmu Monte Carlo Tree Search

3.2. Algorytm UCT

UCT (Upper Confidence Bound Applied to Trees) - algorytm przeszukujący drzewo stanów rozgrywki w poszukiwaniu najbardziej opłacalnych ruchów. Algorytm stara się zachować równowagę między eksploatacją ruchów po ruchach o wysokiej średniej wygranej a eksploracją tych mało sprawdzonych.

3.2.1. Opis algorytmu

Moduł **Algorytm** jest implementacją algorytmu Monte Carlo Tree Search, korzystającą z wariantu UCT. Odpowiedzialnością tego modułu jest wyznaczanie kolejnego ruchu na podstawie dostarczonego stanu gry. Opisywany moduł będzie odpowiadał za iteracyjne tworzenie drzewa stanów i przeszukiwanie go w celu wyznaczenia najbardziej korzystnego ruchu. Użytkownik będzie miał możliwość zmiany liczby iteracji algorytmu albo ograniczenie czasowe jego działania.

W listingu 3.1 operujemy na trzech istotnych zmiennych - **tree**, **curr_node** i **curr_state**. Odpowiedzialnością struktury opisującej **tree** jest przechowywanie korzenia drzewa oraz stanu wyjściowego rozgrywki, który jest tej samej struktury co zmienna **curr_state**. Struktura opisująca **curr_node** przechowuje wszystkie informacje na temat wierzchołka drzewa, wraz z referencjami do wierzchołków potomnych i rodzica.

3.2.2. Dodatkowe założenia

Aby gra była poprawnie obsłużona przez utworzony system, musi spełniać następujące założenia:

1. Rozgrywka jest prowadzona naprzemiennie przez dwóch graczy.
2. Każdy ruch ma jednoznaczny wpływ na dalszą rozgrywkę (rozgrywka jest deterministyczna).
3. Każdy z graczy ma dostęp do pełnej informacji o aktualnym stanie gry.

3.3. Algorytm wizualizacji drzewa

3.3.1. Określenie problematyki

3.3.2. Usprawniony algorytm Walkera

4. Implementacja

4.1. Wykorzystane technologie

4.2. Architektura i działanie systemu

4.2.1. Moduły

4.2.2. Główne komponenty aplikacji

4.2.3. Interfejs użytkownika

5. Instrukcje

5.1. Instrukcja instalacji

5.2. Instrukcja użytkownika

6. Podsumowanie i ocena

6.1. Uzyskane efekty

6.2. Kontynuacja pracy

6.3. Wydajność

6.4. Testy akceptacyjne

7. Wnioski

Bibliografia

[1] A. Author, *Title of a book*, Publisher, year, page–page.

[2] J. Bobkowski, S. Dobkowski, Jak stworzyć bibliografię w BibTeX-u, *Czasopismo nr*, rok, strona–strona.

[3] C. Brink, Power structures, *Algebra Universalis 30(2)*, 1993, 177–216.

[4] F. Burris, H. P. Sankappanavar, *A Course of Universal Algebra*, Springer-Verlag, Nowy Jork, 1981.

Wykaz symboli i skrótów

nzw.	nadzwyczajny
*	operator gwiazdka
~	tylda

Jak nie występują, usunąć.

Spis rysunków

Jak nie występują, usunąć.

Spis tabel

Jak nie występują, usunąć.

Spis załączników

1. Załącznik 1
2. Załącznik 2
3. Jak nie występują, usunąć rozdział.