**Metody Sztucznej Inteligencji 2**

—

# Zastosowanie Upper Confidence Bound Applied To Trees do stworzenia sztucznej inteligencji grającej w Taifho dla dwóch graczy

—

**Konspekt**

Paulina Przybyłek oraz Przemysław Chojecki

Numery albumu: 298837 oraz 298814

09.05.2022

# Spis treści


**1. Wstęp . . . . . 3**

1.1. Cel projektu . . . . . 3

1.2. Dzieje gry Taifho . . . . . 3

1.3. Opis gry Taifho . . . . . 4

1.4. Słownik pojęć . . . . . 5

**2. Algorytm MCTS . . . . . 6**

2.1. Motywacja . . . . . 6

2.2. MCTS . . . . . 7

2.2.1. Ocenianie pozycji nieterminalnych . . . . . 8

2.3. Podstawowy UCT . . . . . 8

2.4. Modyfikacja UCT - PUCT . . . . . 9

**3. Propozycja rozwiązania . . . . . 10**

3.1. Zastosowanie UCT . . . . . 10

3.1.1. UCT bez dodatkowych modyfikacji . . . . . 10

3.1.2. Wybrana modyfikacja UCT . . . . . 11

3.2. Podejście heurystyczne . . . . . 12

**4. Hipotezy badawcze . . . . . 15**

4.1. Przedstawienie hipotez . . . . . 15

4.2. Sposób weryfikacji hipotez . . . . . 16

**5. Implementacja i przebieg projektu . . . . . 17**

5.1. Ogólny projekt techniczny . . . . . 17

5.2. Harmonogram projektu . . . . . 17

# 1. Wstęp

Niniejszy dokument zawiera konspekt projektu zrealizowanego w ramach przedmiotu *Metody Sztucznej Inteligencji 2*.

Istotą projektu jest świadomy i celowy proces badawczy. W konspekcie zawarto wstępny plan badań dla projektu - od opisu zadania i planowanych do wykorzystania algorytmów po postawione hipotezy badawcze i projekt rozwiązania. Na późniejszym etapie projektu algorytmy zostaną zaimplementowane oraz sprawdzone zostaną opisane cele i hipotezy. Całość zadania zostanie zamknięta w raport podsumowujący prace, do którego część konspektu zostanie włączona.

## 1.1. Cel projektu

Celem tego projektu jest stworzenie algorytmu Sztucznej Inteligencji (ang. Artificial Inteligence, AI) do grania w grę Taifho. Projekt ten powstaje w celach badawczych, aby sprawdzić, jak dobrze w tą grę będą grać owe algorytmy. Dodatkowym celem jest porównanie ze sobą wykorzystywanych algorytmów, które są różnymi wersjami podejścia Upper Confidence Bound Applied To Trees. Celem ostatecznym jest sam rozwój autorów, aby mogli oni zapoznać się w szczegółach z działaniem algorytmów AI grających w gry dla dwóch grach z pełną informacją.

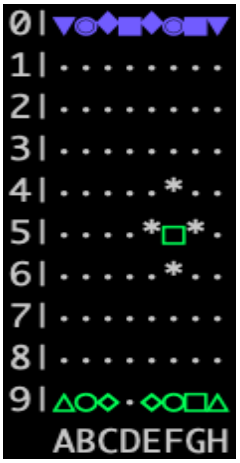
## 1.2. Dzieje gry Taifho

Taifho, zwana czasem szacho-warcaby, jest grą planszową dla dwóch bądź czterech graczy. Wywodzi się z Japonii, a w USA znana jest szerzej pod nazwą Traverse. W 1997 zawitała do europejskich sklepów z grami planszowymi właśnie pod nazwą Taifho będąc wydana przez szwedzką firmę *Alga*. Wygląd pudełka tego wydania wraz z opisem można zobaczyć pod linkiem *Strona internetowa z opisem zasad gry w Taifho z wydania planszowego z 1999 roku* b.d.

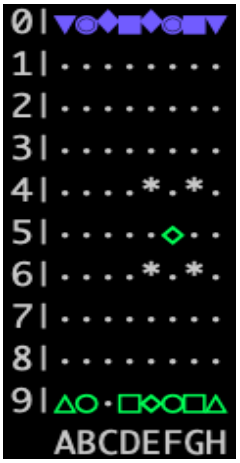
## 1.3. Opis gry Taifho

Wersja dla dwóch graczy rozgrywa się na planszy $8 \times 10$. Obaj zawodnicy mają po 8 pionków, po 2 z każdego rodzaju: koło, kwadrat, romb i trójkąt. Na początku rozgrywki zawodnicy ustawiają swoje pionki na odpowiadającej graczowi krawędzi planszy długości 8 (z tego powodu na początku rozgrywki środkowa część planszy 8x8 jest pusta). Wszystkie pionki pozostają na planszy do końca rozgrywki. Celem gracza jest ustawienie wszystkich swoich pionków na pozycji startowej gracza przeciwnego.

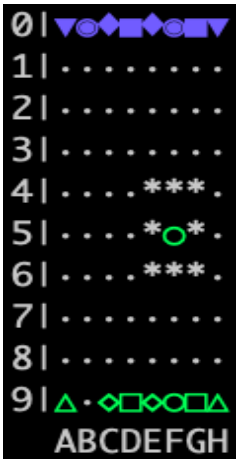
Gracze ruszają się na zmianę, a każdy pionek może poruszać się o jeden w kierunku w którym wskazuje bok figury, co pokazano za pomocą symboli * (gwiazdka) na Rysunkach od 1.1 do 1.5. Koło może poruszać się w dowolnym kierunku, kwadrat wzdłuż i wszerz, romb po ukosie, a trójkąt albo po ukosie do przodu, albo po prostej do tyłu. Ruchy te są dozwolone pod warunkiem, że na polu docelowym nie znajduje się inna bierka.



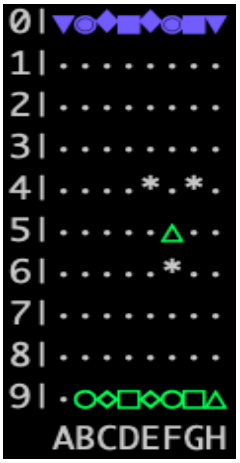

Rysunek 1.1: Legalne ruchy figury kwadrat; takie same dla obu graczy



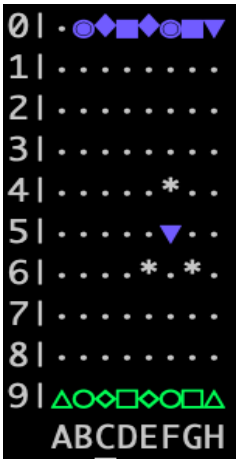

Rysunek 1.2: Legalne ruchy figury romb; takie same dla obu graczy




Rysunek 1.3: Legalne ruchy figury koło; takie same dla obu graczy




Rysunek 1.4: Legalne ruchy figury trójkąt ukazane dla gracza zielonego




Rysunek 1.5: Legalne ruchy figury trójkąt ukazane dla gracza niebieskiego

Istnieje specjalny ruch zwany *skokiem* i polega on na tym, że bierka może „przeskoczyć" inną bierkę pod warunkiem, że skok będzie wykonany o nieparzystą liczbę pól w kierunku poruszania się bierki oraz że „przeskakiwana" bierka będzie znajdować się dokładnie na środkowym polu skoku. Bierką „przeskakiwaną" może być zarówno bierka zawodnika wykonującego skok, jak i bierka przeciwna. Możliwym jest wykonanie kilku skoków w ramach jednego ruchu, ale muszą być one wykonane tą samą bierką. Nie można jednak kończyć ruchu w pozycji, z której bierka

zaczynała swoje skoki.

Ostatnią, wyjątkową zasadą jest zasada, którą nazywamy „fair-play", a która mówi: „nie można uniemożliwić przeciwnikowi zwycięstwa". W *Strona internetowa z opisem zasad gry w Taifho z wydania planszowego z 1999 roku* (b.d.) piszą po prostu „Players can not force a draw". Przyjęto następującą interpretację: jeśli gracz będzie blokował swoje pola startowe przed dostępem dla przeciwnika - natychmiast przegrywa. Dla przykładu sytuacja na Rysunku 1.6 jest pozycją terminalną, zwycięską dla gracza niebieskiego, mimo, że nie ustawił on wszystkich swoich bierek na polach startowych gracza zielonego.


Rysunek 1.6: Przykład zastosowania zasady „fair-play". Gracz zielony uniemożliwia zwycięstwo graczowi niebieskiemu, dlatego gra się kończy i wygrywa gracz niebieski.

W Projekcie założono, że gracze mają z góry ustalone startowe położenie bierek. Mianowicie: trójkąt, koło, romb, kwadrat, romb, koło, kwadrat, trójkąt. Dodatkowo nie będzie rozważana zasada „fair-play". Sprawdzone zostanie, czy brak rozważania tej zasady wpłynie na jakość grających algorytmów.

## 1.4. Słownik pojęć

**Węzeł** - pozycja gry rozumiana jako element drzewa gry

**Pozycja terminalna gry** - pozycja w której gra jest zakończona i znany jest jej wynik

**Bierka** - element gry. Obaj gracze mają swoje bierki. **Ruch** w grach planszowych zazwyczaj polega na zmianie pozycji jednej bierki należącej do gracza

**Turniej** - zestaw rozgrywek w których bierze udział wielu graczy grających przeciw sobie

# 2. Algorytm MCTS

W niniejszym rozdziale znajduje się skrócony opis algorytmu Monte-Carlo Tree Search (MCTS), który będzie wykorzystany jako główna część AI grającego w Taifho. Dodatkowo opisano algorytm Upper Confidence Bound Applied To Trees (UCT), który często jest z powodzeniem wykorzystywany jako jeden z etapów algorytmu MCTS. Na koniec rozdziału przedstawiono dodatkowe modyfikacje, które zostaną również zaimplementowane jako alternatywna wersja AI.

## 2.1. Motywacja

Oczywiste jest, że najlepszym sposobem na znalezienie optymalnego ruchu w danym stanie gry jest przeszukanie wszystkich możliwych rozwinięć aż do pozycji terminalnych, a potem zastosowanie zasady min-max. Niestety, dla wielu zadań tego typu podejście jest niemożliwe do zrealizowania, ze względu na wielkość drzewa gry.

Od algorytmu sztucznej inteligencji grającego w grę oczekuje się, że będzie wykonywał dobre ruchy, czyli takie, które najpewniej będą go doprowadzać do zwycięstwa. Wymaga się jednak, aby nie zajmowało mu to zbyt dużo czasu, gdyż chce się doczekać końca rozgrywki. Z tego powodu pożądanym zachowaniem byłoby, gdyby algorytm AI potrafił już po krótkim czasie działania mieć jakąś estymację jakości możliwych do wykonania ruchów. Natomiast z czasem działania by ową estymację poprawiał. Dzięki temu można by w dowolnym momencie wykonać ruch.

Algorytm MCTS jest przykładem algorytmu spełniającego powyższe wymagania. Jest on powszechnie używany w zadaniu szukania najlepszego ruchu w grach planszowych.

## 2.2. MCTS

MCTS był wprowadzony przez Browne i in. (2012) i odpowiada na potrzeby opisane w poprzedniej sekcji. Polega on na sekwencyjnym rozszerzaniu drzewa gry od korzenia (pozycji startowej), w dół drzewa. Algorytm ten zdobył ogromną popularność i jest powszechnie używanym podejściem do tego celu. Jest m.in. używany w znanym na cały świat algorytmie „AlphaGo" opracowanym przez Silver i in. (2016), który pokonał mistrza świata w grę Go w słynnym meczu z 2016 roku. O owym meczu można przeczytać w artykule Mullen (2016).

Algorytm składa się z czterech etapów: **selekcja**, **ekspansja**, **symulacja** i **propagacja wsteczna wyniku**. Zaczyna on swoje działanie z drzewem odwiedzonych pozycji, w którym znajduje się jedynie węzeł korzenia, czyli pozycja gry, na którym MCTS został wywołany. W każdym momencie działania algorytm trzyma swoje estymacje na temat jakości poszczególnych ruchów, a dokładniej, jakości pozycji, do których ruchy te prowadzą.

W **etapie selekcji** algorytm podróżuje w dół drzewa rozważonych pozycji w następujący sposób: jeżeli jest pozycja, do której reguły gry pozwalają dotrzeć w jednym kroku, ale MCTS jeszcze tam nie dotarł, to idzie tam i przechodzi do następnego etapu. W przeciwnym przypadku w jakiś sposób wybiera jaki ruch tym razem rozważyć i kontynuuje etap selekcji rekurencyjnie. UCT jest przykładową strategią jaką można wykorzystać w tym etapie. Jest on opisany w następnym podrozdziale.

W poprzednim etapie algorytm MCTS dotarł do nowej pozycji gry, w której nigdy nie był. **Etap ekspansji** polega na dodaniu tego węzła do drzewa rozważonych pozycji.

**Etap symulacji** dotyczy symulowania losowych ruchów obu graczy. Pozycje, do których MCTS dotrze w tym etapie, NIE są uznawane za „odwiedzone". Po dotarciu do pozycji końcowej znany jest wynik gry.

**Etap propagacji wstecznej wyniku** polega na zapamiętaniu przez algorytm wyniku gry otrzymanym w poprzednim kroku i wpisaniu go do wszystkich węzłów na ścieżce od korzenia, do nowo dodanego węzła w etapie ekspansji.

W każdej takiej czteroetapowej iteracji algorytmu MCTS rozważane drzewo gry rozszerza się o jeden węzeł, a estymacja jakości ruchów się poprawia. Działanie algorytmu MCTS można przerwać po dowolnej liczbie iteracji.

### 2.2.1. Ocenianie pozycji nieterminalnych

W niektórych grach istnieje naturalny mechanizm oceny pozycji nieterminalnej. Tak jest np. w Monopoly (liczba pieniędzy posiadanych przez graczy) lub Scrabble (liczba punktów posiadanych przez graczy, na stronie *scrabblemania.pl, oficjalne zasady gry* b.d. opisano, że „Celem gry jest uzyskanie jak najwyższego wyniku"). Jednakże wiele gier nie ma wbudowanych takich zasad.

Możliwość wstępnego ocenienia pozycji jest przydatna w praktycznym zastosowaniu algorytmu MCTS. Dla niektórych gier bowiem etap symulacji jest bardzo długi, co podaje w wątpliwość zależność znalezionego wyniku gry od pozycji dodanej w etapie ekspansji.

Można w takiej sytuacji skrócić etap symulacji i estymować jakość pozycji zgodnie z jakąś heurystyczną oceną. Takie podejście pozwoli dodatkowo na wykonanie większej liczby iteracji algorytmu MCTS.

## 2.3. Podstawowy UCT

Algorytm UCT jest modyfikacją strategii UCB (Peter Auer, Cesa-Bianchi i Fischer 2002) zaaplikowaną do podążania w dół drzewa. Strategia UCB opiera się na twierdzeniu dającemu optymalną strategię grania przy problemie wielorękiego bandyty (opisanym przez P. Auer i in. (1995)). Gra polega na tym, że mamy skończoną liczbę maszyn do grania, gdzie każda zwraca jakąś wypłatę z nieznanego graczowi rozkładu prawdopodobieństwa, być może innym dla każdej z maszyn. Zadaniem gracza jest zmaksymalizowanie osiąganej wypłaty.

Optymalna strategia UCT zaproponowana przez Kocsis i Szepesvári (2006) wygląda następująco: jeśli jakieś dziecko rozważanego węzła nie było jeszcze nigdy odwiedzone, to odwiedź je. Jeśli wszystkie dzieci były już odwiedzone, to wybierz to dziecko $a$, które maksymalizuje wartość:

$$g(a) = \hat{a} + C \cdot \sqrt{\frac{ln(N)}{N(a)}}$$

gdzie $\hat{a}$ to estymacja wartości dziecka $a$, $N$ jest liczbą odwiedzin rozważanego węzła, $N(a)$ jest liczbą odwiedzin dziecka $a$, natomiast $C > 0$ jest parametrem metody.

Algorytm ten można w naturalny sposób zastosować w etapie selekcji algorytmu MCTS przyjmując $\hat{a}$ jako średnią z otrzymanych do tej pory wypłat, oraz przechodząc do etapu symulacji w chwili trafienia w nowo odwiedzone dziecko.

## 2.4. Modyfikacja UCT - PUCT

Większość współczesnych implementacji opiera się na jakimś wariancie UCT, będącym modyfikacją podstawowego podejścia. Jedną ze znanych modyfikacji jest algorytm PUCT, będący zastosowaniem podejścia PUCB (Rosin 2011) do drzew. PUCB oznacza „Predictor" + UCB, gdyż modyfikuje pierwotną politykę wielorękich bandytów UCB, poprzez w przybliżeniu przewidywanie dobrego wyboru na początku sekwencji prób jednorękich bandytów.

Modyfikacja PUCT polega na przypisaniu pewnych wag $M_a$ do każdego dziecka $a$. Wszystkie wagi dzieci rozważanego węzła mają sumować się do wartości $\sum_{a \in \{dzieci\}} M_a = 1$. Następnie jest to wykorzystywane do obliczania kary:

$$m(N,a) = \begin{cases} \frac{2}{M_a} \cdot \sqrt{\frac{ln(N)}{N}} & gdy\ N > 1, \\ \frac{2}{M_a} & wpp \end{cases}$$

gdzie $N$ jest liczbą odwiedzin rozważanego węzła, $N(a)$ jest liczbą odwiedzin dziecka $a$, $M_a$ jest wagą przypisaną do dziecka $a$. Wówczas wzór wykorzystywany do wyboru dziecka wygląda następująco:

$$g(a) = \hat{a} + C \cdot \sqrt{\frac{ln(N)}{N(a)}} - m(N,a)$$

Wykorzystanie wag w UCT jest swego rodzaju predyktorem. Wybór odpowiednich predyktorów opisano dokładnie w sekcji trzeciej w artykule wprowadzającym PUCB (tamże).

Zastosowanie modyfikacji PUCT w niektórych rozwiązaniach AI odniosło znaczne korzyści. Pewne wariacje PUCT zostały użyte m.in. we wspomnianym już „AlphaGo" a także w „AlphaZero" czy „Leela Chess Zero".

# 3. Propozycja rozwiązania

Rozdział ten opisuje w ogólny sposób wybrane podejścia AI, które zostaną zaimplementowane w ramach projektu. Algorytmy te będą ze sobą porównane i ocenione względem siebie.

Zaproponowane zostały cztery wersje AI do grania w Taifho:

- podstawowa - UCT bez dodatkowych modyfikacji,
- PUCT - modyfikacji UCT,
- wykorzystująca heurystykę $h$,
- wykorzystująca heurystykę $h_G$.

## 3.1. Zastosowanie UCT

Podstawą wykorzystywanych algorytmów AI będzie MCTS z zastosowaną selekcją UCT, która w szczegółach omówiona została w rozdziale 2. W tej sekcji omówiono użycie tej metody do rozważanej gry.

### 3.1.1. UCT bez dodatkowych modyfikacji

Gra Taifho posiada bardzo ważną różnicę względem gier najczęściej rozważanych w kontekście MCTS, a mianowicie - same zasady nie popychają jej ku pozycjom terminalnym. W przypadku szachów co jakiś czas następuje bicie, natomiast w Go liczba ruchów jest ograniczona. Z tego powodu etap symulacji, mimo zachowania losowego, ma w tamtych grach tendencję do zmierzania ku pozycjom terminalnym. Gra Taifho takiej własności nie ma, gdyż trzy z czterech figur dają taką samą swobodę w poruszaniu się do przodu jak i do tyłu. Jedynie trójkąt ma średnio dwa razy więcej ruchów w przód. Z tego powodu należałoby przypuszczać, że etap ekspansji podstawowego algorytmu MCTS będzie trwał bardzo długo.

Dlatego też przedstawione przypuszczenia zostaną poddane sprawdzeniu czy są poprawne, a co za tym idzie, czy algorytm MCTS w podstawowej wersji będzie w stanie dotrzeć do odpo-

wiedniej liczby liści drzewa gry w sensownym czasie. Sposób w jaki to zostanie zrobione opisano w rozdziale 4.2.

Jeśli okaże się, że przedstawione przypuszczenie jest poprawne, oznaczać to będzie, że stworzony AI oparty na podstawowej wersji UCT nie będzie działało. Być może okaże się, że etap symulacji będzie jednak działał dość szybko. W takiej sytuacji stworzony AI oparty na podstawowej wersji UCT będzie porównany z pozostałymi wersjami.

### 3.1.2. Wybrana modyfikacja UCT

Niektóre modyfikacje UCT skupiają się na specjalnych ruchach typowych dla danych gier. Uważa się, że preferowanie takiego ruchu może polepszyć skuteczność algorytmu AI. Dla przykładu Wang i in. (2018) w swojej pracy o warcabach skupili się na ruchu tworzącym króla i na tej podstawie wypracowali odpowiednią modyfikację UCT.

W grze Taifho specjalny ruch zwany „skokiem" może przybliżyć znacząco bierkę gracza do przodu, co może przyśpieszyć zwycięstwo danego gracza. Zaproponowana w projekcie modyfikacja UCT chce preferować ten ruch nad innymi. Najbardziej oczekiwany byłby skok bierki do przodu o jak największą liczbę pól. Stąd każdemu dziecku $a$ należy przypisać odpowiednią wagę $M_a$ związaną z ruchem „skoku". Tego właśnie dotyczyć będzie heurystyka wyboru predyktorów.

Mając stan gry przed ruchem bierki i po wykonaniu możliwego ruchu tą bierką w daną stronę można obliczyć jej różnicę odległości od linii startowej:

$$d(a) = \mu_2 - \mu_1$$

gdzie $\mu_1$, $\mu_2$ oznaczają odpowiednio odległość ruszającej się bierki od linii startowej przed ruchem i po wykonaniu ruchu. Jeśli ruch nastąpił w poprzek to wartość wynosi 0, jeśli jest do przodu to jest dodatni, a jak bierka się cofnęła to ujemny.

Każda bierka ma kilka możliwych ruchów określanych jako dzieci, gdzie dla każdego dziecka $a$ należy wyliczyć odległość $d(a)$. Niech $D$ oznacza wektor tych wartości dla wszystkich dzieci, więc $D = (d(1), \ldots, d(K))$, gdzie $K$ to liczba dzieci. Rozważana jest sytuacja, gdy $K > 1$.

We wspomnianej już pracy Rosin (2011), wprowadzającej metodę PUBC, jedną z opisanych metod wyboru był „Uogólniony predyktor Bradleya-Terry'ego" (sekcja 3.1.1 w tamtej pracy). Zmodyfikowana wersja tego wzoru dla rozważanej gry wygląda następująco:

$$M_a = \frac{exp(\frac{1}{K}x_a)}{\sum_{i=1}^{K} exp(\frac{1}{K}x_i)}$$

gdzie wartości $x_j$ dla $1 \leqslant j \leqslant K$ związane są z ruchem bierki na podstawie liniowego przeskalowania wartości z wektora $D$ do przedziału $[\frac{1}{K^3}, 1 - \frac{1}{K}]$.

## 3.2. Podejście heurystyczne

W celu przeciwdziałaniu ogromnej liczbie ruchów losowych na drodze do pozycji terminalnej, wprowadzony zostanie nowy parametr *steps* sterujący, ile losowych kroków będzie wykonanych. Po ich wykonaniu pozycja zostanie oceniona za pomocą heurystycznej funkcji oceny.

Ocena pozycji $p$ będzie następować według następującego wzoru:

$$h(p) = \Big[ \sum_{i \in \{\text{bierki gracza}\}} b_i \Big] - \Big[ \sum_{j \in \{\text{bierki przeciwnika}\}} b_j \Big]$$

gdzie $b_i$ oznacza odległość bierki $i$ od jej linii startowej. Oznacza to, że jeśli bierki zawodnika będą na swoim docelowym miejscu, to pierwszy nawias będzie wynosił $8 \cdot 9 = 72$. Inny przykład, uwzględniający rozmieszczenie wszystkich bierek, można zobaczyć na rysunku 3.1, gdzie według heurystyki $h$ gracz niebieski ma przewagę.

Jednakże, jeśli przyjrzymy się pozycji na rysunku 3.1, to wcale nie jest to takie oczywiste, że gracz niebieski ma przewagę. Gracz niebieski bowiem zostawił swoje dwie bierki na początku planszy i prawdopodobnie później będzie miał trudności w efektywnym przyprowadzeniu ich do celu. Gracz zielony natomiast trzyma swoje bierki blisko, dzięki czemu będą one mogły sobie nawzajem pomagać.

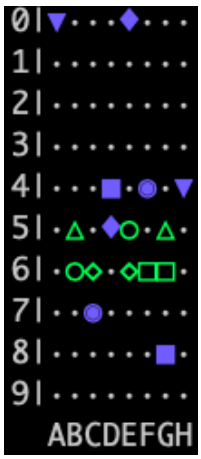
Wypracowano więc drugą heurystykę, która nieliniowo karze za odległość do celu. Ocena ta będzie następować według następującego wzoru:

$$h_G(p) = \Big[ \sum_{i \in \{\text{bierki gracza}\}} \frac{b_i \cdot log(9+G)}{log(b_i+G)} \Big] - \Big[ \sum_{j \in \{\text{bierki przeciwnika}\}} \frac{b_j \cdot log(9+G)}{log(b_j+G)} \Big]$$

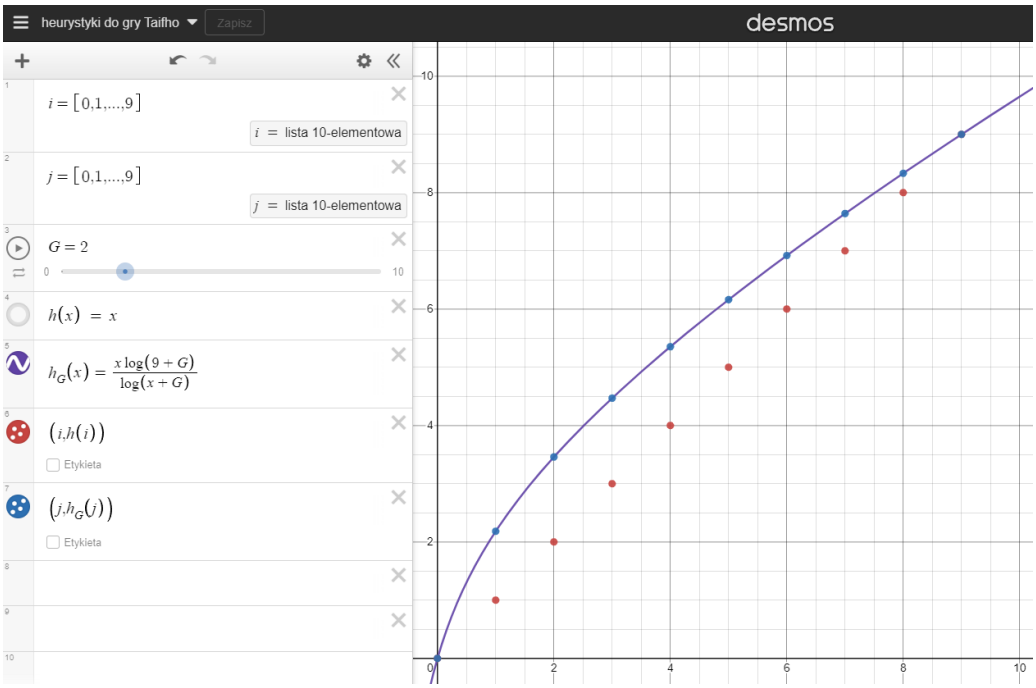
gdzie $b_i$ oznacza odległość bierki $i$ od jej linii startowej, a $G \in (1, \infty)$ jest parametrem, który im mniejszy, tym bardziej karane jest zostawianie bierek „z tyłu".

Wpływ parametru $G$ można prześledzić na stronie *Strona Desmos na której można zobaczyć jaki wpływ ma parametr G na heurystykę $h_G$* b.d., z której to również pochodzi rysunek 3.2 wizualizujący różnicę między $h$, a $h_2$. Warto zwrócić uwagę, że wcześniejsza heurystyka $h$ jest granicą wartości heurystyk $h_G$ przy parametrze $G$ zbiegającym do $\infty$. Heurystyka $h_2$ ocenia pozycję na rysunku 3.1 jako lekko przeważającą dla gracza zielonego.

Oczekiwanym jest więc, że heurystyka $h_G$ będzie bardziej preferować wspólne poruszanie się




Rysunek 3.1: Przykładowy stan gry. Ocena planszy z punktu widzenia gracza zielonego według heurystyki $h$ wynosi $h(p) = [5 \cdot 3 + 3 \cdot 4] - [2 \cdot 0 + 3 \cdot 4 + 5 + 7 + 8] = 27 - 32 = -5$, czyli niebieski ma przewagę. Natomiast według heurystyki $h_2$ ocena wynosi $h_2(p) \approx +0.21$, czyli gra jest wyrównana, a zielony ma lekką przewagę.




Rysunek 3.2: Zdjęcie ze *Strona Desmos na której można zobaczyć jaki wpływ ma parametr $G$ na heurystykę $h_G$* b.d. Czerwonymi kropkami zaznaczono heurystykę $h$, a niebieskimi heurystykę $h_G$ dla $G = 2$. Na stronie tej można się przekonać, że $h_1(0) > 2$, a pożądaną wartością jest 0, co jest spełnione dla każdego $G > 1$.

do przodu wieloma bierkami, zamiast zostawiać kilka z nich z tyłu. Jest tak dlatego, że dla $G = 2$, $h_2(0) = 0, h_2(1) \approx 2.18, h_2(2) \approx 3.46$. Oznacza to, że przejście jednej bierki z pozycji startowej na następną linię jest według $h_2$ warte około 2.18, natomiast przejście z pierwszej na

drugą około $3.46 - 2.18 = 1.28$. Heurystyka $h_G$ dla każdego $G > 1$ w ten sposób stopniowo coraz gorzej ocenia kolejne przejścia na następną linię.

# 4. Hipotezy badawcze

Jak wspomniano we wstępie, celem projektu jest zapoznanie się z działaniem algorytmów AI grających w gry dla dwóch graczy z pełną informacją poprzez implementację opisanych w poprzednim rozdziale podejść do gry Taifho, a także zrozumienie jak działają i jakie wyniki mogą osiągać. Projekt ten jest projektem badawczym, dlatego ważne jest przeprowadzenie właściwych eksperymentów i wyciągniecie na ich podstawie wartościowych wniosków. Z tego powodu, w tym rozdziale określono hipotezy, wybrane do zbadania podczas eksperymentów oraz sposób ich realizacji.

## 4.1. Przedstawienie hipotez

W planowanych eksperymentach zostaną sprawdzone następujące hipotezy:

1. Etap symulacji MCTS w podstawowym algorytmie UCT będzie działała na tyle długo, że podstawowy UCT będzie niepraktycznym algorytmem AI do gry Taifho.

2. Algorytm UCT będzie grał podobnie niezależnie, czy w implementacji będzie istnieć zasada „fair-play", czy też nie.

3. AI będzie grało najlepiej z wartością parametru $C$ inną niż teoretycznie najlepsze $C = \sqrt{2}$.

4. Najlepsza wartość parametru $G$ dla heurystyki $h_G$ będzie w przedziale $G \in [2, 5]$.

5. Skuteczność zaimplementowanych AI będzie następująca: Najgorzej będzie grał podstawowy UCT, lepiej będzie grał UCT z heurystyką $h$, jeszcze lepiej będzie grał UCT z heurystyką $h_G$, a najlepiej będzie grał UCT z modyfikacją PUCT.

6. Algorytmy AI będą w stanie pokonać ludzkiego gracza amatora.

## 4.2. Sposób weryfikacji hipotez

Dla każdej hipotezy przeprowadzony zostanie eksperyment, którego celem będzie potwierdzenie bądź odrzucenie hipotezy. W tej sekcji każdy taki eksperyment jest opisany i przypisany do odpowiedniej hipotezy swoim numerem.

1. Dla kilku losowych pozycji przeprowadzony zostanie proces błądzenia losowego taki, jak w podstawowym algorytmie MCTS w etapie symulacji. Sprawdzone zostanie jak dużo losowych ruchów będzie musiało być wykonanych przez algorytm, żeby osiągnąć stan terminalny oraz jak dużo czasu zajmie to komputerowi. Jeśli średni czas na znalezienie pozycji terminalnej przekroczy sekundę, to będzie oznaczało, że AI używające podstawowego MCTS w czasie 10 sekund wykona średnio jedynie 10 symulacji, co jest zdecydowanie za mało, aby algorytm taki mógł wykonać inteligentny ruch, przy naszych zasobach obliczeniowych.

2. Uruchomiony zostanie algorytm UCT jako gracz zielony na pozycji z Rysunku 1.6 w sytuacji braku zasady „fair-play". Sprawdzone zostanie, czy algorytm będzie dążył do utrzymania blokady, czy jednak umożliwi przeciwnikowi zwycięstwo.

3. Sprawdzone zostanie kilka AI z różnymi wartościami parametru $C$. Będą one grały przeciw sobie. Po rozgrywkach przeprowadzona zastanie analiza wyników i wyłoniona najlepsza wartość parametru $C$.

4. Sprawdzone zostanie kilka AI z różnymi wartościami parametru $G$. Będą one grały przeciw sobie. Po rozgrywkach przeprowadzona zastanie analiza wyników i wyłoniona najlepsza wartość parametru $G$.

5. Najlepsze wersje algorytmów AI każdego rodzaju będą ze sobą porównane w turnieju. Po rozgrywkach przeprowadzona zastanie analiza wyników i wyłoniona kolejność skuteczności algorytmów.

6. Zaimplementowane algorytmy AI zagrają z graczem kilka partii i ocenione zostanie kto jest w tą grę lepszy.

# 5. Implementacja i przebieg projektu

W tym rozdziale przedstawiono ogólny przegląd wykorzystywanej technologii i harmonogram projektu.

## 5.1. Ogólny projekt techniczny

Gra oraz wszystkie wersje algorytmy AI zaimplementowane będą jako pakiet do języka Python i będą dostępne na repozytorium GitHub pod adresem Przemysław b.d. Dzięki tej postaci, użytkownik będzie mógł w prosty sposób zagrać z zaimplementowanym AI. Szczegóły jak to zrobić znajdują się w pliku *README.md* na wspomnianym repozytorium. Na nim będą również dostępne wszystkie skrypty wykorzystane w tym projekcie do weryfikacji hipotez, analiz działania algorytmów, jak i wszystkie wykonane wizualizacje.

Głównym zadaniem jest własna implementacja algorytmów, dlatego też w projekcie wykorzystano jedynie bazowe biblioteki pythonowe, takie jak:

- NumPy - optymalna praca z wektorami liczb,

- matplotlib - wizualizacja,

- seaborn - wizualizacja.

## 5.2. Harmonogram projektu

W Tabeli 5.1 przedstawiono uogólniony harmonogram prac nad projektem skupiony na przyrostach, będących punktami kontrolnymi przebiegu prac.

| | |
|---|---|
| 2022-04-25 | Wstępny konspekt projektu |
| 2022-05-09 | Pełny konspekt projektu |
| 2022-05-23 | Wstępna implementacja |
| 2022-06-06 | Wstępny raport |
| 2022-06-13 | Oddanie projektu + prezentacja |

Tablica 5.1: Ogólny harmonogram prac

# Bibliografia


Auer, P. i in. (1995). „Gambling in a rigged casino: The adversarial multi-armed bandit problem". W: *Proceedings of IEEE 36th Annual Foundations of Computer Science*, s. 322–331. DOI: 10.1109/SFCS.1995.492488.

Auer, Peter, Nicolò Cesa-Bianchi i Paul Fischer (maj 2002). „Finite-time Analysis of the Multiarmed Bandit Problem". W: *Machine Learning* 47.2, s. 235–256. ISSN: 1573-0565. DOI: 10.1023/A:1013689704352. URL: https://doi.org/10.1023/A:1013689704352.

Browne, Cameron B. i in. (2012). „A Survey of Monte Carlo Tree Search Methods". W: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1, s. 1–43. DOI: 10.1109/TCIAIG.2012.2186810.

Kocsis, Levente i Csaba Szepesvári (2006). „Bandit Based Monte-Carlo Planning". W: *Machine Learning: ECML 2006*. Red. Johannes Fürnkranz, Tobias Scheffer i Myra Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 282–293. ISBN: 978-3-540-46056-5.

Mullen, Jethro (sty. 2016). *Computer scores big win against humans in ancient game of Go*. https://money.cnn.com/2016/01/28/technology/google-computer-program-beats-human-at-go/index.html.

Przemysław, Chojecki (b.d.). *GitHub page with the code for this project*. https://github.com/PrzeChoj/MSI2/. [Online; accessed 03-05-2022].

Rosin, Christopher D. (mar. 2011). „Multi-armed bandits with episode context". W: *Annals of Mathematics and Artificial Intelligence* 61.3, s. 203–230. ISSN: 1573-7470. DOI: 10.1007/s10472-011-9258-6. URL: https://doi.org/10.1007/s10472-011-9258-6.

*scrabblemania.pl, oficjalne zasady gry* (b.d.). https://scrabblemania.pl/oficjalne-zasady-gry-w-scrabble. [Online; accessed 03-05-2022].

Silver, David i in. (sty. 2016). „Mastering the game of Go with deep neural networks and tree search". W: *Nature* 529.7587, s. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

*Strona Desmos na której można zobaczyć jaki wpływ ma parametr G na heurystykę $h_G$* (b.d.). https://www.desmos.com/calculator/uhbknmltyg. [Online; accessed 03-05-2022].

*Strona internetowa z opisem zasad gry w Taifho z wydania planszowego z 1999 roku* (b.d.). https://bastardcafe.dk/games/taifho/. [Online; accessed 03-05-2022].

Wang, Yajie i in. (2018). „Application and Improvement of UCT in Computer Checkers". W: *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, s. 274–278. DOI: 10.1109/CCIS.2018.8691199.

# Wykaz skrótów

| | |
|---|---|
| AI | Artificial Inteligence |
| MCTS | Monte-Carlo Tree Search |
| UCB | Upper Confidence Bound |
| UCT | Upper Confidence Bound Applied To Trees |
| PUCB | Predictor UCB |
| PUCT | Predictor UCT |