Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Metody Sztucznej Inteligencji 2

Projekt 2: Algorytm MCTS

Filip Suchorab, Jakub Winiarski

9 czerwca 2024

Streszczenie

Projekt przedstawia analizę i implementację algorytmu Monte Carlo Tree Search (MCTS) oraz jego modyfikacji w grze Connect 4. Celem pracy było zbadanie wpływu różnych wariantów MCTS na efektywność gry, porównując je również z prostym algorytmem heurystycznym inspirowanym ludzką strategią. Przedstawiono zarówno podstawowy algorytm MCTS, jak i jego wersje z tablicami transpozycji oraz modyfikacjami LGR (Last Good Reply).

W ramach projektu przetestowano sześć hipotez, które dotyczyły między innymi skuteczności poszczególnych modyfikacji MCTS w porównaniu do wersji podstawowej oraz do algorytmu heurystycznego. Eksperymenty przeprowadzono poprzez symulacje gier pomiędzy różnymi wariantami algorytmu MCTS oraz pomiędzy MCTS a algorytmem heurystycznym. Wyniki eksperymentów wykazały, że algorytmy MCTS z modyfikacjami LGR były bardziej efektywne niż podstawowy MCTS, natomiast tablice transpozycji nie przyniosły znaczącej poprawy.

Dodatkowo, algorytmy MCTS w każdej wersji znacząco przewyższały algorytm heurystyczny, co potwierdziło ich wyższość w podejmowaniu decyzji w grze Connect 4. Badania dowiodły, że modyfikacje LGR-2 były bardziej efektywne niż LGR-1.

W ramach części technicznej zaimplementowano GUI umożliwiające rozgrywkę człowieka przeciwko sztucznej inteligencji, które zostało wykorzystane do weryfikacji szóstej hipotezy w trakcie zajęć. Do testów pozostałych hipotez posłużył symulator umożliwiający wykonanie dużej liczby pojedynków pomiędzy dwoma algorytmami sztucznej inteligencji. Symulator zadbał o to żeby każdy algorytm zaczynał rozgrywkę taką samą liczbę razy, co zapewniło rzetelne warunki eksperymentu.

Podsumowując, praca wykazała, że algorytmy MCTS, zwłaszcza z modyfikacjami LGR, są skutecznymi narzędziami do gry w Connect 4, znacznie przewyższającymi podejście heurystyczne. Modyfikacje te poprawiają efektywność algorytmu, choć tablice transpozycji nie wnoszą istotnych korzyści. Wyniki te mogą być przydatne dla dalszych badań nad zastosowaniem MCTS w innych grach i problemach decyzyjnych.

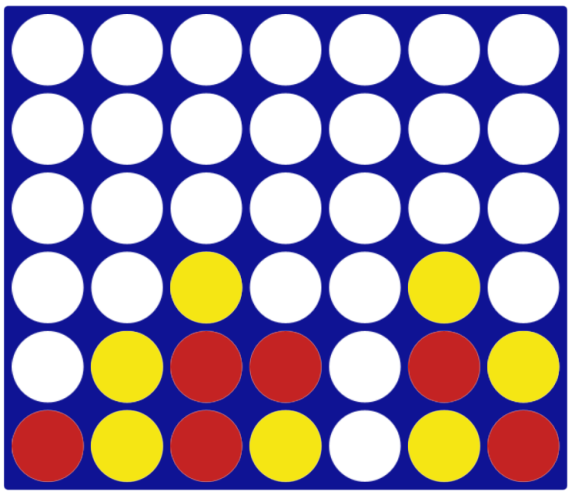
Spis treści

1 Opis problemu

Projekt ma na celu zaimplementowanie sztucznej inteligencji bazującej na algorytmie MCTS (Monte Carlo Tree Search [4]) w grze Connect 4. Głównym celem projektu jest zbadanie jakości działania algorytmu MCTS w zależności od zastosowanych modyfikacji. Zarówno algorytm w wersji podstawowej jak i jego modyfikacje zostały szczegółowo opisane w sekcji 2. Przed przystąpieniem do eksperymentu postawiono hipotezy opisane w sekcji 3, a następnie przeprowadzono eksperymentu mające na celu ocenę jakości poszczególnych wersji rozwiązania (sekcja 4).

1.1 Gra Connect 4

Connect 4 jest prostą grą, przypominającą kółko i krzyżyk. Dana jest plansza o szerokości 7 i wysokości 6. Dwóch graczy rzuca na zmianę swoje żetony. Żetony podlegają grawitacji i zatrzymują się w najniższym wolnym rzędzie w danej kolumnie. Wygrywa ten z graczy, który ułoży cztery swoje żetony w jednej linii - poziomo, pionowo lub na ukos. Przykładowy fragment przebiegu gry przedstawiono na rysunku 1. Współczynnik rozgałęzienia Connect 4 wynosi 7, a maksymalna długość gry to 42 tury.



Rysunek 1: Przykładowa rozgrywka Connect 4

2 Algorytmy

Głównym celem projektu jest zbadanie jakości działania algorytmu MCTS w zależności od zastosowanych modyfikacji. Ponadto w celu sprawdzenia jak algorytm MCTS radzi sobie z podejściem heurystycznym wyspecjalozowanym w danej grze zaproponowano algorytm heurystyczny, który testowano razem z algorytmem MCTS.

2.1 MCTS

Jest to algorytm probabilistyczny wybierający tę ścieżkę, która zwraca średnio najlepsze wyniki (względem wybranej metryki). Algorytm stosuje się często do wyboru ruchów w grach. W ogólnym pojęciu MCTS przeszukuje przestrzeń stanów gry reprezentowaną przez drzewo w celu znalezienia najlepszego ruchu. MCTS znajduje najlepszą decyzję balansując eksplorację i eksploatację w ramach ograniczonego budżetu np. czasu działania. Algorytm MCTS przeszukuje przestrzeń stanów gry iteracyjnie. Po wykonaniu odpowiedniej liczby iteracji algorytm zwraca najlepszą znalezioną akcję dla stanu S za pomocą wzoru:

$$a^* = arg \max_{a \in A(S)} Q(S, a),$$

gdzie $A(S)$ to zbiór możliwych akcji w stanie S, a $Q(S, a)$ to zależny od gry średni wynik wykonania akcji a w stanie S. Iteracyjność algorytmu przejawia się w estymowaniu $Q(S, a)$. W każdej kolejnej iteracji poprawiane jest oszacowanie $Q(S, a)$ dla jednej z akcji.
W podstawowej wersji MCTS wyróżniamy cztery fazy iteracyjne:

Selekcja - faza w której algorytm przeszukuje fragment drzewa gry dostępną w pamięci urządzenia. W każdym kolejnym stanie algorytm wybiera jedną z dostępnych akcji na podstawie metody pozwalającej balansować między eksploracją a eksploatacją. Podstawową metodą wyboru akcji jest jest Upper Conficence Bound (UCB). W tej metodzie akcja w stanie S jest wybierana na podstawie wzoru:

$$a^* = \arg\max_{a \in A(S)} \left\{ Q(S,a) + C\sqrt{\frac{\ln[N(S)]}{N(s,a)}} \right\},$$

gdzie $N(S)$ to liczba odwiedzeń stanu S, a $N(s,a)$ to liczba wyborów akcji a w stanie S w fazie selekcji dotychczasowych ieracji algorytmu. C to stała, która pozwala odpowiada za eksplorację algorytmu. Za podstawową wartość stałej C w literaturze uznaje się $\sqrt{2}$. Selekcja kończy się w momencie, gdy algorytm natrafi na stan terminalny gry lub na stan, którego akcje prowadzą do stanu nieodwiedzonego przez algorytm.

Ekspansja - faza następująca po fazie selekcji. Algorytm będąc w stanie S jeśli nie trafił na stan terminalny, dodaje do drzewa wszystkie stany dostępne do osiągnięcia przez wykonanie akcji $a \in A(S)$. Algorytm następnie wybiera pierwszy z dodanych do drzewa stanów i przechodzi do fazy symulacji. Jeśli algorytm w fazie selekcji trafił na stan terminalny to przechodzi do fazy propagacji wstecznej.

Symulacja - faza w której algorytm będący w stanie S przeprowadza symulację gry. Algorytm przechodzi do stanu terminalnego wykonując losowe akcje w kolejnych stanach pośrednich.

Propagacja wsteczna - na początku tej fazy algorytm znajduje się w stanie terminalnym S. Stan terminalny jest oceniany jako wartość $Q(S)$ zależnie od danego problemu. Dla gier dwuosobowych można przyjąć, że wygrana gracza wykonującego akcję to 1, remis to 0.5, a przegrana to 0. Ocena stanu terminalnego jest propagowana w górę drzewa. Podczas propagacji wszystkie stany odwiedzone w fazie selekcji i stan wybrany w fazie ekspansji mają aktualizowane średnie wyniki $Q(S,a)$.

2.2 MCTS z tablicami transpozycji

Tablice transpozycji[2] to odpowiedź na to, że przeszukując drzewo gry algorytm może dotrzeć do tego samego stanu różnymi ciągami akcji. W podstawowej wersji algorytmu MCTS takie stany byłyby niezależne. Takie podejście sprawia, że dany stan może być przeliczany wielokrotnie w trakcie działania algorytmu, spowalniając jego działanie. Tablice transpozycyjne to tablice zawierające informacje o odwiedzonych stanach w grze. Informacja ta w algorytmie MCTS to średni wynik $Q(S)$ dla danego stanu S. Jest ona wykorzystywana w fazie selekcji akcji a^* według wzoru:

$$a^* = \arg\max_{a \in A(S)} \left\{ Q(g(S,a)) + C\sqrt{\frac{\ln[N(S)]}{N(s,a)}} \right\},$$

gdzie $g(S,a)$ to stan osiągnięty po wykonaniu akcji a w stanie S.

Tablica transpozycji jest aktualizowana podczas fazy propagacji wstecznej. W przypadku tego projektu tablica transpozycji została zaimplementowana jako prosta tablica haszująca o ograniczonym rozmiarze.

2.3 MCTS z modyfikacją LGR

Metoda LGR (Last good reply)[3] zakłada, że akcje, które w pewnym stanie poprowadziły do zwycięstwa , powinny być dobre również w innych stanach. Po fazie symulacji zapisywane są informacje o akcjach - odpowiedziach na akcje przeciwnika, które prowadziły w danej symulacji do zwycięstwa. Kolejne symulacje mogą używać zapisanych informacji, preferując akcje będące wcześniej dobrymi odpowiedziami na akcje przeciwnika. Jeśli podczas symulacji akcja, zapisana jako dobra odpowiedź, nie jest w danym stanie legalna, wybierana jest losowa akcja z dostępnych.

Metoda może być rozszerzona do metody LGR-n, polegającej na zapisywaniu informacji nie tylko o odpowiedzi na jedną akcję, ale na zapisywaniu odpowiedzi na ciągi akcji o długości n. W takim wypadku przy wyborze akcji w fazie symulacji stosuje się metodę najdłuższego dopasowania, wybierając akcję, która była wcześniej dobrą odpowiedzią na ostatnie k wykonanych akcji.

2.4 Algorytm heurystyczny

W celu porównania MCTS z myśleniem człowieka zaproponowano algorytm heurystyczny wyspecjalizowany do gry w Connect 4. Algorytm w założeniu miał być prosty i bazować na wiedzy ludzi znających się dobrze na tej grze. Zaprojektowano algorytm 1, naśladujący myślenie człowieka. Algorytm jeśli może wygrać to wykonuje ruch wygrywający, jeśli istnieje groźba przegranej w jednym ruchu to algorytm blokuje przeciwnika, w przeciwnym wypadku algorytm próbuje grać ruch strategiczny, a ostatecznie ruch losowy. Dodatkowo każdy ruch sprawdzany jest pod kątem tego czy jego wykonanie nie daje ruchu wygrywającego przeciwnikowi.

Algorithm 1 Algorytm heurystyczny

```
1: if istnieje ruch wygrywający then
2:     return ruch wygrywający
3: else if przeciwnik ma ruch wygrywający then
4:     return ruch blokujący przeciwnika
5: else if możliwy bezpieczny ruch strategiczny then
6:     return ruch strategiczny
7: else
8:     return losowy bezpieczny ruch
9: end if
```

Ruchy strategiczne bazują na strategii przedstawionej w filmie Keitha Gallia[5] umieszczonego na portalu YouTube. Generowanie ruchu strategicznego bazuje na poniższych regułach:

1. Gdy przeciwnik wykonuje dwa ruchy obok siebie na dolny rząd zablokuj powstanie trójki,
2. Rozpoczynaj środkową kolumną i graj środkową kolumnę dopóki nie osiągnie wysokości 5,
3. Jako gracz pierwszy staraj się ustawić 3 żetony obok siebie w rzędach nieparzystych licząc od dołu,
4. Jako gracz drugi staraj się ustawić 3 żetony obok siebie w rzędach parzystych licząc od dołu.

3 Hipotezy

Testom zostały poddane cztery wersje algorytmu MCTS opisane w sekcji 2. Ponadto przetestowano podejście heurystyczne. Dzięki temu, że gra Connect 4 jest rozwiązana, eksperymenty uwzględniły również algorytm dokładny znajdujący strategię wygrywającą. Odniesienie do strategii wygrywającej pozwoli na dokładne porównanie algorytmów. Poniżej przedstawiono hipotezy, które zostały sprawdzone w ramach eksperymentu

Hipoteza 1 Algorytmy MCTS z modyfikacjami wygrają średnio więcej razy niż przegrają, grając przeciwko podstawowemu algorytmowi MCTS.

Hipoteza 2 Wszystkie algorytmy MCTS wygrają średnio więcej razy niż przegrają, grając przeciwko algorytmowi z podejściem heurystycznym.

Hipoteza 3 Algorytm MCTS z tablicami transpozycji i modyfikacją LGR wygra średnio więcej razy niż przegra, grając przeciwko algorytmom MCTS z jedną modyfikacją.

Hipoteza 4 Algorytm MCTS z tablicami transpozycji wraz ze wzrostem rozmiaru tablicy transpozycji będzie wykonywał więcej dobrych ruchów[1].

Hipoteza 5 Algorytm MCTS z modyfikacją LGR-2 wykona średnio więcej dobrych ruchów niż z modyfikacją LGR-1.

Hipoteza 6 Żaden z algorytmów MCTS nie przegra ani jednego pojedynku z prowadzącym podczas prezentacji na zajęciach :-)

[1]Dobry ruch - ruch niepogarszający szans na zwycięstwo. Dobre ruchy zostaną sprawdzone za pomocą narzędzia "Connect 4 solver" [6].

4 Wyniki eksperymentów

W poniższych eksperymentach algorytm MCTS i jego modyfikacje zostały przetestowane ze stałą liczbą iteracji równą 10^5.

4.1 Hipoteza 1

Treść

Algorytmy MCTS z modyfikacjami wygrają średnio więcej razy niż przegrają, grając przeciwko podstawowemu algorytmowi MCTS.

Metodologia

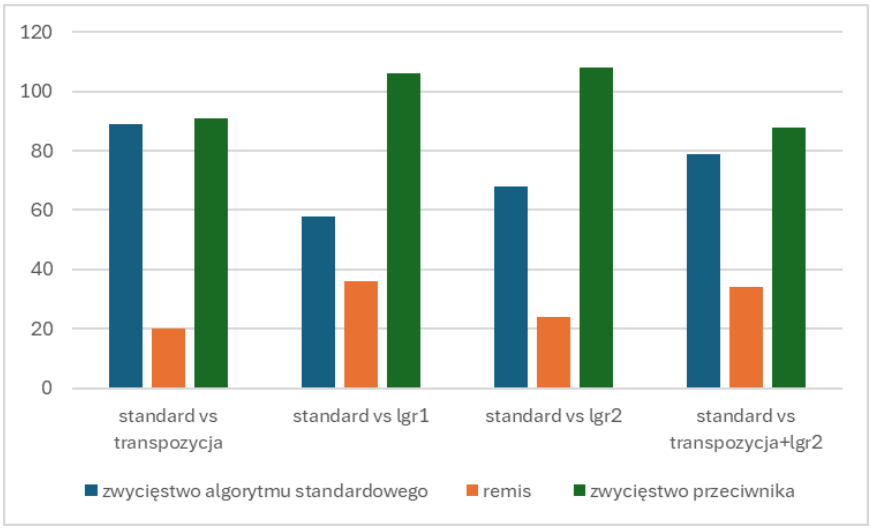
Eksperyment przeprowadzono za pomocą 200-krotnej symulacji gry dwóch algorytmów przeciwko sobie. Każdy z algorytmów w trakcie jednego pojedynku rozpoczynał 100 razy i 100 razy grał jako drugi gracz. Przetestowano pary standard vs transpozycja, standard vs lgr1, standard vs lgr2, standard vs transpozycja + lgr2. Prawdziwość hipotezy sprawdzono testem McNemara [1].

Wyniki

W tabeli 1 przedstawiono wyniki pojedynków z podziałem na to, kto zaczynał rozgrywkę. Na rysunku 2 przedstawiono sumaryczną liczbę zwycięstw i remisów w każdej parze (dla 200 iteracji).

1. zawodnik	Zwycięstwa 1.	Remisy	Zwycięstwa 2.	2. zawodnik
standard	14	10	76	**transpozycje**
transpozycje	15	10	75	**standard**
standard	11	3	86	**lgr1**
lgr1	20	33	47	**standard**
standard	14	1	85	**lgr2**
lgr2	23	23	54	**standard**
standard	24	1	75	**transpozycje+lgr2**
transpozycje+lgr2	23	23	54	**standard**

Tabela 1: Tabela porównawcza dla każdego pojedynku



Rysunek 2: Sumaryczne porównanie algorytmu standardowego z innymi algorytmami

Dla sumarycznego porównania każdej pary przeprowadzono test McNemara. Hipoteza w teście McNemara zakłada, że wartości przed i po wprowadzeniu modyfikacji powinny być takie same. Jeśli wartość uzyskana z testu jest większa niż wartość krytyczna dla danego poziomu istotności, hipotezę odrzucamy. W związku z tym postawiono hipotezę zerową

$$H_0 = \text{Algorytm standardowy i jego przeciwnik dają takie same rezultaty.}$$

$$H_1 = \text{Przeciwnik daje lepsze rezultaty niż algorytm standardowy.}$$

W tym doświadczeniu skorzystano z testu McNemara na poziomie istotności $0,05$ w następującej postaci

$$\chi^2 = \frac{(w_{standard} - w_{przeciwnik})^2}{w_{standard} + w_{przeciwnik}},$$

gdzie $w_{standard}$ - sumaryczna liczba wygranych algorytmu standardowego jako pierwszy i drugi zawodnik, $w_{przeciwnik}$ - sumaryczna liczba wygranych jego przeciwnika.

W tabeli 2 przedstawiono otrzymane wartości χ^2.

Pojedynek	χ^2
standard vs transpozycja	0.022
standard vs lgr1	14.05
standard vs lgr2	9.09
standard vs transpozycja+lgr2	0.485

Tabela 2: Test McNemara

Jak się okazuje, najlepiej w porównaniu z algorytmem standardowym poradziły sobie algorytmy lgr. Obliczono wartość krytyczną dla poziomu istotności $0,05$, $\chi = 3,841$. Ponieważ dla par 2 i 3, $\chi^2 < \chi$ widzimy, że możemy odrzucić H_0. Jednak dla par 1 i 4. Algorytmy dają bardzo zbliżone wyniki i wydaje się, że tablice transpozycji nie poprawiają znacząco działania algorytmu MCTS. Ponieważ dla dwóch par nie widać poprawy, musimy odrzucić hipotezę.

Wniosek

Hipoteza odrzucona.

4.2 Hipoteza 2

Treść

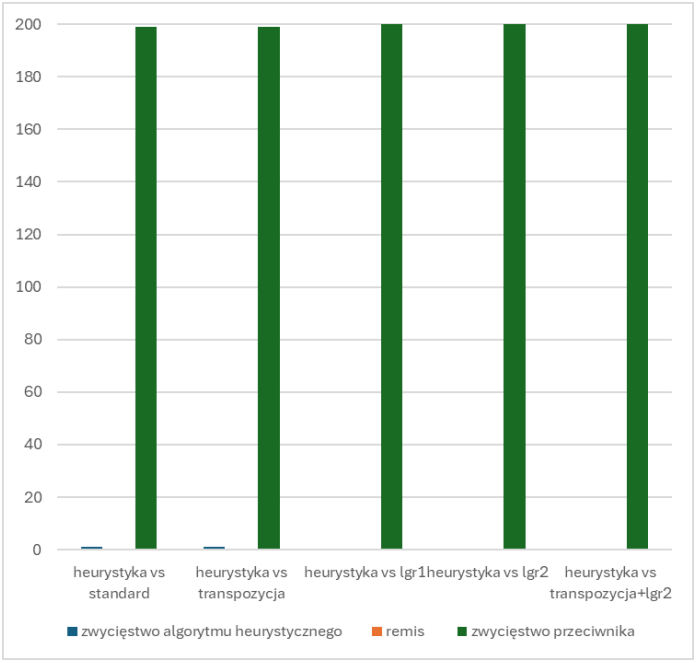
Wszystkie algorytmy MCTS wygrają średnio więcej razy niż przegrają, grając przeciwko algorytmowi z podejściem heurystycznym.

Metodologia

Eksperyment przeprowadzono za pomocą 200-krotnej symulacji gry dwóch algorytmów przeciwko sobie. Każdy z algorytmów w trakcie jednego pojedynku rozpoczynał 100 razy i 100 razy grał jako drugi gracz. Przetestowano pary heurystyka vs standard, heurystyka vs transpozycja, heurystyka vs lgr1, heursytyka vs lgr2, heurystyka vs transpozycja + lgr2. Wyniki okazały się na tyle oczywiste, że hipotezę potwierdzono empirycznie.

Wyniki

Na rysunku 3 przedstawiono zbiorcze wyniki porównujące heurstykę z innymi algorytmami - dla każdego pojedynku (200 iteracji) pokazano liczbę zwycięstw heurystyki, liczbę remisów oraz liczbę zwycięstw przeciwników.



Rysunek 3: Sumaryczne porównanie algorytmu standardowego z innymi algorytmami

Jak widać, wyniki doświadczenia są oczywiste. Tylko w dwóch przypadkach na tysiąc, algorytmowi heurystycznemu udało się wygrać z jakimkolwiek algorytmem MCTS (raz z standardowym, raz z tablicami transpozycji). Nie odnotowano remisów. W związku z tym podjęto decyzję o potwierdzeniu hipotezy.

Wniosek

Hipoteza potwierdzona

4.3 Hipoteza 3

Treść

Algorytm MCTS z tablicami transpozycji i modyfikacją LGR wygra średnio więcej razy niż przegra, grając przeciwko algorytmom MCTS z jedną modyfikacją.

Metodologia

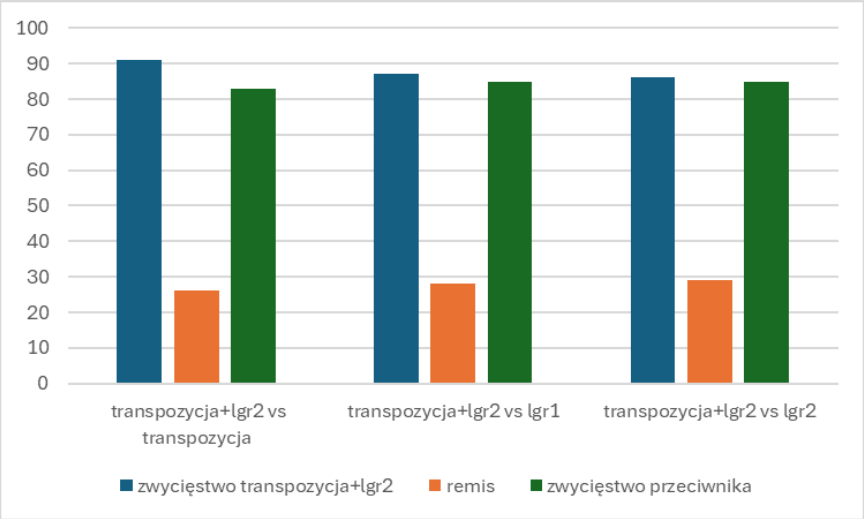
Eksperyment przeprowadzono za pomocą 200-krotnej symulacji gry dwóch algorytmów przeciwko sobie. Każdy z algorytmów w trakcie jednego pojedynku rozpoczynał 100 razy i 100 razy grał jako drugi gracz. Przetestowano pary transpozycja+lgr2 vs transpozycja, transpozycja+lgr2 vs lgr1, transpozycja+lgr2 vs lgr2. Prawdziwość hipotezy sprawdzono testem McNemara.

Wyniki

W tabeli 3 przedstawiono wyniki pojedynków z podziałem na to, kto zaczynał rozgrywkę. Na rysunku 4 przedstawiono sumaryczną liczbę zwycięstw i remisów w każdej parze (dla 200 iteracji).

1. zawodnik	Zwycięstwa 1.	Remisy	Zwycięstwa 2.	2. zawodnik
transpozycja+lgr2	15	23	62	**transpozycja**
transpozycja	21	3	76	**transpozycja+lgr2**
transpozycja+lgr2	60	16	24	**lgr1**
lgr1	61	12	27	**transpozycja+lgr2**
transpozycja+lgr2	61	15	24	**lgr2**
lgr2	61	14	25	**transpozycja+lgr2**

Tabela 3: Tabela porównawcza dla każdego pojedynku



Rysunek 4: Sumaryczne porównanie algorytmu standardowego z innymi algorytmami

Zgodnie z metodą opisaną szczegółowo w sekcji 4.1, przeprowadzono test McNemara na poziomie istotności $0,05$ wprowadzając hipotezę zerową i hipotezę alternatywną.

H_0 = Algorytm z tablicami transpozycji i modyfikacją lgr2 nie ma przewagi nad algorytmem z jedną modyfikacją.

H_1 = Algorytm z tablicami transpozycji i modyfikacją lgr2 ma przewagę nad algorytmem z jedną modyfikacją.

Wyniki przedstawiono w tabeli 4.

Pojedynek	χ^2
transpozycja+lgr2 vs transpozycja	0.368
transpozycja+lgr2 vs lgr1	0.023
transpozycja+lgr2 vs lgr2	0.0058

Tabela 4: Test McNemara

Jak wynika z tabeli 4, tablice transpozycji nie dają żadnej poprawy, spisują się gorzej od algorytmu z dwiema modyfikacjami. Obliczono wartość krytyczną dla poziomu istotności $0,05$, $\chi = 3,841$. Wobec tego, że dla każdej pary algorytmów $\chi^2 < \chi$, przyjmujemy H_0 w każdym przypadku. Co jednak istotne dla naszej hipotezy, nie ma istotnej różnicy pomiędzy algorytmami z modyfikacjami lgr, a algorytmem z dwiema modyfikacjami.

Wniosek

Hipoteza odrzucona.

4.4 Hipoteza 4

Treść

Algorytm MCTS z tablicami transpozycji wraz ze wzrostem rozmiaru tablicy transpozycji będzie wykonywał więcej dobrych ruchów.

Metodologia

W celu weryfikacji hipotezy wybrano sześć rozmiarów tablic transpozycji odpowiednio: 1E4, 2,5E4, 5E4, 1E5, 2, 5E5, 1E6, 8E6 liczonych ilością różnych pozycji o których informacje może trzymać tablica. Wytypowano 36 pozycji charakteryzujących się tym, że gracz wykonujący kolejny ruch ma możliwość zagrać ruch umożliwiający zwycięstwo lub remis oraz ruch przegrywający(zakładając idealną grę przeciwnika). Dla wybranych rozmiarów tablicy transpozycji pięciokrotnie uruchomiono algorytm MCTS ustawiając liczbę iteracji na 100 tysięcy i zliczono średnią ilość dobrych ruchów. W

celu dokładnego sprawdzenia hipotezy użyto testu korelacji Pearsona. Test na poziomie istotności $0,05$ sprawdza czy można odrzucić H_0 i przyjąć H_1 przedstawione następująco:

- H_0 : Nie ma liniowej zależności między ilością dobrych ruchów a rozmiarem tablicy transpozycji.
- H_1 : Istnieje taka zależność liniowa.

Wyniki

Średnią ilość dobrych ruchów w zależności od rozmiaru tablicy transpozycji przedstawiono w tabeli 5.

Rozmiar tabliy transpozycji	Średnia ilość dobrych ruchów
1E4	18
2,5E4	18
5E4	20
1E5	19
2,5E5	19
5E5	20
1E6	17
8E6	19

Tabela 5: Ilość dobrych ruchów w zależności od rozmiaru tablicy transpozycji

Wykonano test korelacji Pearsona. Otrzymane $p\text{-}value = 0,9$. W związku z tym nie ma podstaw aby odrzucić H_0.

Wniosek

Hipoteza odrzucona.

4.5 Hipoteza 5

Treść

Algorytm MCTS z modyfikacją LGR-2 wykona średnio więcej dobrych ruchów niż z modyfikacją LGR-1.

Metodologia

W celu weryfikacji hipotezy wykorzystano podobny schemat jak przy testowaniu hipotezy 4. Tym razem przetestowano ile dobrych ruchów z 36 zrobią algorytmy z modyfikacjami LGR-1 i LGR-2. Wyniki testów przedstawiono w tabeli 6.

Wyniki

Algorytm	Średnia ilość dobrych ruchów
LGR-1	21
LGR-2	20

Tabela 6: Ilość dobrych ruchów wykonanych przez algorytm MCTS z modyfikacjami LGR-1 i LGR-2

Wniosek

Z powodu tego, że różnica w ilości dobrych ruchów między modyfikacjami jest równa jeden oraz jest ona na korzyść modyfikacji LGR-1 hipoteza zostaje odrzucona.

4.6 Hipoteza 6

Treść

Żaden z algorytmów MCTS nie przegra ani jednego pojedynku z prowadzącym podczas prezentacji na zajęciach :-)

Metodologia

W trakcie prezentacji projektu na zajęciach prowadzący zostanie poproszony o rozegranie meczu towarzyskiego z wybranymi przez siebie algorytmami. Hipoteza zostanie zaprzeczona, jeśli prowadzącemu uda się wygrać chociaż jeden pojedynek.

Wyniki

Nie wiemy jaki był wynik eksperymentu, ponieważ raport jest pisany przed prezentacją, ale zgadujemy, że hipoteza została potwierdzona, ponieważ żadnemu z autorów projektu nie udało się wygrać z algorytmem.

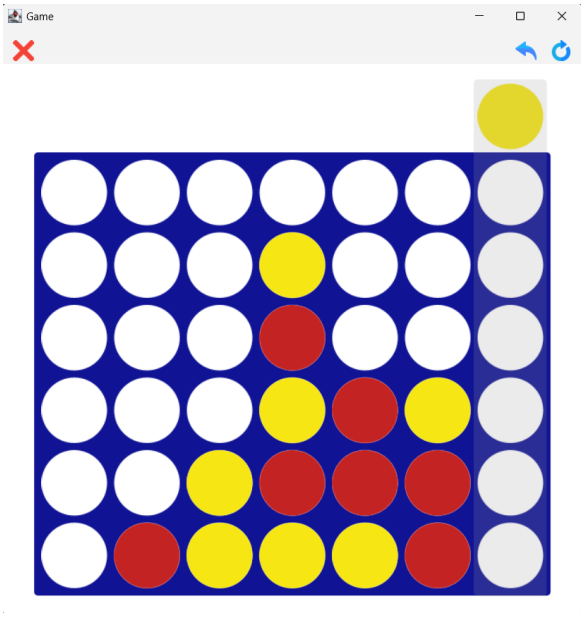
Wniosek

Hipoteza prawdopodobnie potwierdzona.

5 Analiza techniczna

Zarówno interfejs graficzny, jak i logikę aplikacji zdecydowano się zaimplementować przy pomocy języka Kotlin. Decyzja została podyktowana chęcią nauczenia się nowego języka, w którym nie są realizowane projekty na studia.

GUI zostało zaprojektowane przy pomocy frameworku Korge [7], który umożliwia prostą obsługę grafiki 2D oraz dodawanie animacji, które pozytywnie wpływają na UX. Udostępniono 2 tryby gry - człowiek vs człowiek oraz człowiek vs AI, w którym można wybrać przeciwko której wersji algorytmu chcemy się zmierzyć.



Rysunek 5: Okienko aplikacji

Interfejs zaprojektowano tak, aby był jak najbardziej intuicyjny i przyjemny. Ruchy sztucznej inteligencji są wykonywane z pewnym opóźnieniem tak aby użytkownik miał chwilę na przyjrzenie się jaki ruch wybiera AI, jednak odpowiedź zwracana przez MCTS jest błyskawiczna. Po zakończeniu rozgrywki, użytkownik otrzymuje informację zwrotną, tak aby przypadkowo nie przeoczyć porażki.

Rysunek 6: Widok skończonej gry

Oprócz GUI został zrealizowany również symulator umożliwiający rozgrywkę AI vs AI. Dzięki temu byliśmy w stanie porównać konkretne modyfikacje algorytmów i zweryfikować hipotezy.

6 Podsumowanie

Algorytm MCTS radzi sobie dobrze w grze connect 4. Jego duże możliwości modyfikacji stanowią ciekawy aspekt do znajdowania coraz to lepszych algorytmów do gier dwuosobowych. W ramach projektu przetestowano dwie modyfikacje algorytmu MCTS. Okazało się, że modyfikacja MCTS z tablicami transpozycji nie jest istotnie lepsza od podstawowej wersji, za to modyfikacja LGR jest istotnie lepsza co zostało potwierdzone testami statystycznymi.

Bibliografia

[1] Andrew C. Leon. "3.12 - Descriptive and Inferential Statistics". W: *Comprehensive Clinical Psychology*. Red. Alan S. Bellack i Michel Hersen. Oxford: Pergamon, 1998, s. 243–285. ISBN: 978-0-08-042707-2. DOI: https://doi.org/10.1016/B0080-4270(73)00264-9. URL: https://www.sciencedirect.com/science/article/pii/B0080427073002649.

[2] Benjamin E. Childs, James H. Brodeur i Levente Kocsis. "Transpositions and move groups in Monte Carlo tree search". W: (2008), s. 389–395. DOI: 10.1109/CIG.2008.5035667.

[3] Mandy Tak, Mark Winands i Yngvi Björnsson. "N-Grams and the Last-Good-Reply Policy Applied in General Game Playing". W: *Computational Intelligence and AI in Games, IEEE Transactions on* 4 (czer. 2012), s. 73–83. DOI: 10.1109/TCIAIG.2012.2200252.

[4] Maciej Świechowski i in. "Monte Carlo Tree Search: a review of recent modifications and applications". W: *Artificial Intelligence Review* 56.3 (lip. 2022), s. 2497–2562. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10228-y. URL: http://dx.doi.org/10.1007/s10462-022-10228-y.

[5] Keith Galli. *The Best Strategy to Win at Connect 4! (Odd Even Strategy)*. URL: https://www.youtube.com/watch?v=YqqcNjQMX18.

[6] Pascal Pons. *Connect 4 solver*. URL: https://connect4.gamesolver.org.

[7] Carlos Ballesteros Velasco. *Korge*. URL: https://docs.korge.org/getting-started/.