Badanie gier kooperacyjnych z niepełną informacją na przykładzie gry Hanabi

(A study on cooperative games with incomplete information based on the game of Hanabi)

Wojciech Jarząbek Jacek Leja

Praca inżynierska

Promotor: dr Paweł Rychlikowski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

31 stycznia 2020

Streszczenie

Od początku ery komputerów, gry planszowe były używane do testowania limitów sztucznej inteligencji. W ostatnich latach nadzwyczajną popularność, zarówno w środowiskach pasjonatów gier planszowych, jak i badaczy akademickich, zyskało Hanabi - w pełni kooperacyjna gra karciana z niepełną informacją, którą można określić mianem "kooperacyjnego pasjansa". W poniższej pracy przedstawiamy różnorodne metody tworzenia agentów grających w Hanabi, którzy, przy surowych ograniczeniach na czas wykonania ruchu, byliby zdolni do gry na poziomie ludzkiego gracza. W tym celu zaimplementowaliśmy siedmiu agentów regułowych o różnych stopniach zaawansowania wdrażanych strategii oraz agenta, który potrafi nauczyć się gry bez ingerencji człowieka. Sprawdziliśmy także, w jaki sposób strategie agentów wzajemnie na siebie oddziałują. Badania empiryczne wykazały, że stworzeni agenci potrafią grać na poziomie zbliżonym do ludzkiego, bez konieczności poświęcania wysokich nakładów mocy obliczeniowej. Pokazaliśmy również skuteczność metod uczących, które pozwoliły korzystającemu z nich agentowi na samoistne wypracowanie efektywnej strategii, osiągającej wyniki zbliżone do tych uzyskiwanych przez ludzi. Aby ułatwić dalszą analizę zagadnienia, a także umożliwić grę ze stworzonymi agentami, wykonaliśmy graficzny interfejs użytkownika. Praca daje podstawy do rozwijania problematyki wysokopoziomowej gry w Hanabi z ograniczeniami czasowymi i stanowi funkcjonalny prototyp.

Spis treści

1. Wprowadzenie **5**

1.1. Czym jest Hanabi? 5

1.2. Hanabi a sztuczna inteligencja 6

2. Reguły gry **7**

2.1. Wyjaśnienie zasad 8

2.2. Dodatkowe obserwacje 9

3. Strategie sztucznej inteligencji **11**

3.1. Podejście regułowe 11

3.2. Drzewa poszukiwań 12

3.3. Algorytmy uczące 12

3.4. Naginanie zasad gry 13

4. Funkcjonalna realizacja SI **15**

4.1. Cheater 16

4.2. SmartCheater 17

4.3. Erratic 19

4.4. StoppedClock 20

4.5. SimpleDistrustful 22

4.6. Distrustful 23

4.7. Trustful 27

4.8. BayesianTrustful 31

4.9. Reinforced 34

5. Niestandardowe modele rozgrywki **40**

5.1. Dwa oblicza drużyn Hanabi 40

5.2. Potyczki XY 40

6. Wnioski **44**

7. Dalsze badania **45**

A Korzystanie z projektu **46**

A.1. Instalacja zależności 46

A.1.1. Windows 46

A.1.2. MacOS 46

A.1.3. Ubuntu Linux 46

A.1.4. Inne systemy operacyjne 47

A.2. Uruchamianie projektu 47

A.3. Dodatkowe funkcje 47

Bibliografia **48**

Rozdział 1.

Wprowadzenie

1.1. Czym jest Hanabi?

Gry planszowe to forma rozrywki, która towarzyszy człowiekowi od tysięcy lat. Były one popularne już za czasów starożytnych, czego dowodzi chociażby malowidło z 3300 r. p.n.e, pochodzące z grobowca Merknery, na którym ukazano rozgrywkę Seneta. Przykładem może być także Królewska Gra z Ur, której egzemplarze odnaleziono w trakcie badań nad starożytną Mezopotamią. Choć gry te zostały w dzisiejszych czasach w znacznej mierze zapomniane, nie sposób nie wspomnieć o innych, które również istniały w starożytności, takich jak warcaby czy Go, a także o nieco młodszych szachach, które wciąż cieszą się ogromną i niesłabnącą popularnością.

Każdą z tych gier łączy aspekt rywalizacji: pod koniec rozgrywki jednoznacznie wyszczególnia się jednego lub więcej graczy, których określamy mianem zwycięzców, natomiast reszta - przegrywa. Inny schemat prezentują gry kooperacyjne, gdzie zadaniem nie jest pokonanie innych uczestników zabawy, a osiągnięcie wspólnego celu, który gwarantuje wygraną. Można powiedzieć, że przeciwnikiem graczy jest w tym przypadku sama gra, która swoją konstrukcją skłania do współpracy. Pierwsze gry tego typu powstały dopiero w drugiej połowie XX wieku i początkowo miały wyłącznie charakter edukacyjny. Wraz z popularyzacją tzw. "planszówek", gry kooperacyjne w znacznym stopniu zyskały na popularności, a ich forma wyewoluowała w kierunku rozrywki kładącej nacisk na aspekty towarzyskie, które ograniczają lub wręcz odrzucają współzawodnictwo. Przykładami takich gier mogą być Pandemic, Martwa Zima, a także Hanabi.

Hanabi (jap. fajerwerki) to w pełni kooperacyjna gra planszowa, która w 2013 roku wygrała prestiżową nagrodę Spiel des Jahres. Gracze wcielają się w niej w pracowników fabryki fajerwerków, w której omyłkowo zostały pomieszane ze sobą różne rodzaje prochu. Celem jest złożenie w odpowiedniej kolejności możliwie jak największej ilości sztucznych ogni, które gracze otrzymują poprzez dobieranie kart z potasowanej talii. Uczestnicy rozgrywki widzą karty, które są w posiadaniu innych graczy,

lecz nie mogą przypatrywać się tym, którymi sami dysponują. Dodatkowo, komunikacja odnosząca się do treści kart podlega restrykcyjnym zasadom i jest w znacznym stopniu ograniczona, co czyni rozgrywkę nietrywialną. Jakie strategie należy zatem zastosować, by wygrać? Jak można przełożyć je na świat algorytmów?

1.2. Hanabi a sztuczna inteligencja

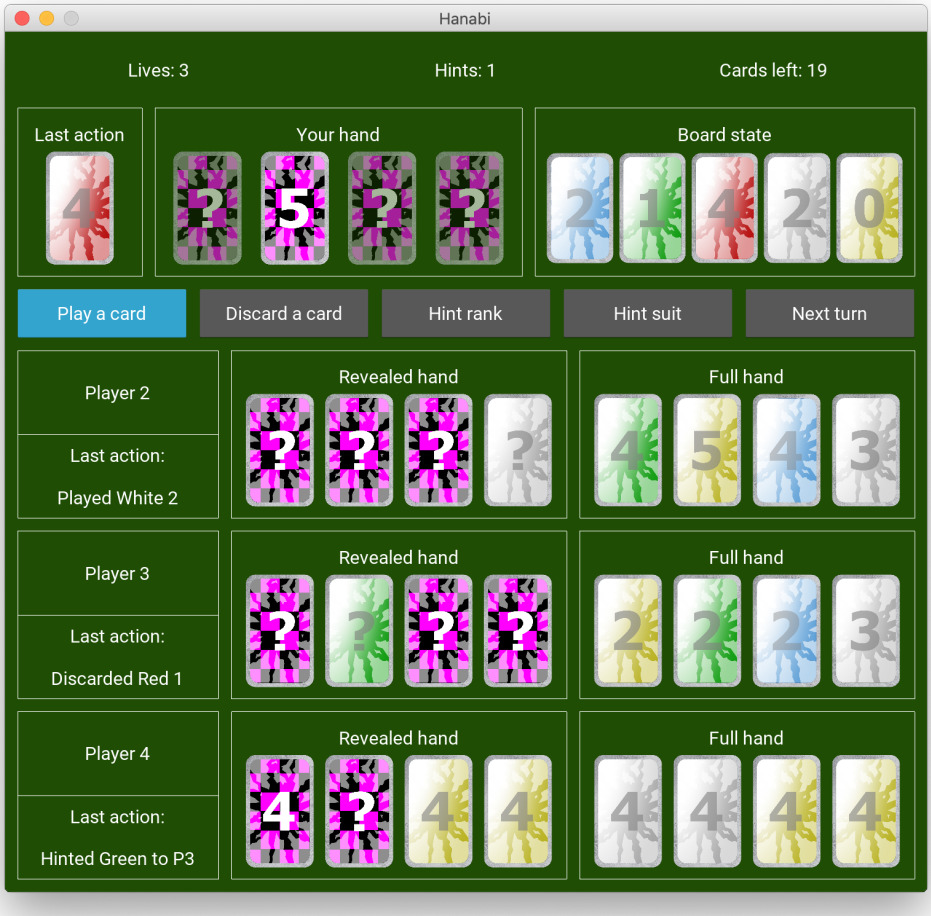
W teorii gier istnieje pojęcie perfekcyjnego zagrania, czyli pojedynczego ruchu zależnego od aktualnego etapu gry, prowadzącego do stanu rozgrywki maksymalizującego oczekiwany wynik - niezależnie od ruchów, które mogą w odpowiedzi wykonać inni gracze. Perfekcyjne zagrania są podstawą optymalnego planu działania, minimalizującego możliwe straty ponoszone w trakcie rozgrywania danej partii. Niestety, tak silna strategia - w przypadku złożonych gier - jest nieprawdopodobnie trudna do uzyskania ze względu na ogromną rozpiętość drzewa możliwych do uzyskania stanów rozgrywki. W praktyce używa się algorytmów: heurystycznych, regułowych, opartych na technikach uczących, nadużywających zasad gry lub siłowych. Przykładowo, słynny komputer Deep Blue, który w maju 1997 roku pokonał ówczesnego mistrza świata w szachach, Garrego Kasparova, nie posiadał optymalnej strategii. Używał on metody siłowej, wspomaganej algorytmem przeszukującym alfa-beta, rozpatrując wszystkie możliwe zagrania i wybierając te, które dawały mu największą przewagę lokalną. Takie działanie było możliwe z racji na ogromną moc superkomputera, który potrafił rozpatrywać 200 milionów ruchów na sekundę[1].

Stworzenie agentów sztucznej inteligencji do Hanabi to zadanie, które wymaga pokonania trudności niespotykanych w innych grach, takich jak: niepełnej informacji, losowości dobieranych kart, a także ograniczonych zasobów, m.in. w postaci podpowiedzi dla innych graczy. Agenci muszą współpracować, gdyż gracz, który odmawia kooperacji, może w kilku ruchach doprowadzić do przegranej całej grupy. Ważne jest, by nie marnować zasobów, a zatem sztuczna inteligencja musi być odpowiednio skoordynowana. Ponadto, znikoma ilość kart w talii nie pozwala na zbyt długą rozgrywkę - oznacza to zatem, że aby zdążyć z wygraną agenci muszą posiadać protokół komunikacji, który dopuszcza przekazywanie w obrębie zasad gry dodatkowych, implicytnych informacji, rozumianych przez pozostałych jej uczestników.

Niniejsza praca ma na celu zbadanie Hanabi jako gry kooperacyjnej z niepełną informacją. Będziemy analizować techniki tworzenia agentów sztucznej inteligencji grających w Hanabi, którzy wykonują możliwie najbardziej efektywne i zrozumiałe dla ludzi ruchy na tyle szybko, by umożliwić komfortową rozgrywkę z człowiekiem na komputerach osobistych.

Rozdział 2.

Reguły gry



Rysunek 2.1: Interfejs graficzny do gry Hanabi, utworzony na potrzeby projektu (twórca grafik: Jakub Podwysocki)

2.1. Wyjaśnienie zasad

Celem gry jest zdobycie możliwie największej ilości punktów poprzez poprawne zagrywanie kart. Maksymalna ilość możliwych do uzyskania punktów wynosi dwadzieścia pięć. Po zakończeniu gry ilość uzyskanych punktów oblicza się poprzez zsumowanie wartości najwyższych kart z każdego ze stosów odpowiedniego koloru.

Talia do gry składa się z pięćdziesięciu kart. Każda karta jest oznaczona jednym z pięciu kolorów (czerwony, żółty, niebieski, biały, zielony) oraz jedną z wartości z zakresu od 1 do 5. Dla każdego koloru istnieją po trzy karty o numerze 1, po dwie karty o numerach 2, 3 i 4, a także po jednej karcie o numerze 5.

Na początku gry talia jest tasowana. Gracze rozpoczynają rozgrywkę z ośmioma żetonami podpowiedzi i trzema żetonami życia. Żetony te są wspólne dla wszystkich uczestników rozgrywki. Jeżeli graczy jest dwóch lub trzech, każdy z nich dobiera po pięć zakrytych kart. Jeżeli jest ich czterech lub pięciu, dobierają po cztery zakryte karty. Następnie gracze po kolei wykonują swoje ruchy. Ruchu nie można pominąć. Ruch to wykonanie jednej z trzech następujących akcji:

1. Zagranie karty:

   Gracz deklaruje chęć wykonania zagrania, wybiera zakrytą kartę ze swojej ręki, a następnie wykłada ją na planszę w pozycji odkrytej. Zagranie może być poprawne lub niepoprawne. Karty muszą być zagrywane w kolejności rosnącej, zaczynając od jedynki, inaczej zagranie uważa się za niepoprawne. Przykładowo, jeśli na planszy nie ma żadnych kart, można poprawnie zagrać tylko te, które są oznaczone numerem 1. Jeżeli na planszy znajdują się wyłącznie jedna niebieska karta o numerze 1 i stos żółtych kart, spośród których największą wartość ma karta o numerze 4, można poprawnie zagrać niebieską kartę o numerze 2, żółtą kartę o numerze 5 lub dowolną kartę innego koloru o numerze 1. Jeżeli karta została zagrana poprawnie, jest ona dodawana do stosu o odpowiednim kolorze lub też rozpoczyna stos swojego koloru, jeżeli jest to karta o numerze 1. Dodatkowo, jeżeli zagrana karta ma numer 5, gracze otrzymują jeden żeton podpowiedzi (chyba, że mają ich już osiem - wtedy zagranie nie ma żadnego dodatkowego efektu). Jeżeli karta została zagrana niepoprawnie, jest ona usuwana z gry i nie jest dodawana do żadnego ze stosów, a gracze tracą jeden z żetonów życia. Po rozpatrzeniu efektów zagrania gracz dobiera zakrytą kartę z talii (jeżeli nie jest ona pusta).

2. Odrzucenie karty:

   Gracz deklaruje chęć odrzucenia karty, wybiera zakrytą kartę ze swojej ręki, a następnie wykłada ją na planszę w pozycji odkrytej. Karta ta jest usuwana z gry, bez dokładania jej do któregokolwiek ze stosów, a gracze otrzymują jeden żeton podpowiedzi (chyba, że mają ich już osiem - wtedy odrzucenie

karty nie ma żadnego dodatkowego efektu). Po rozpatrzeniu efektów akcji gracz dobiera zakrytą kartę z talii (jeżeli nie jest ona pusta).

3. Udzielenie podpowiedzi innemu graczowi:

   Gracz usuwa jeden z żetonów podpowiedzi, następnie wybiera innego uczestnika rozgrywki oraz jeden z dwóch rodzajów informacji, której chce mu udzielić: może wskazać wszystkie jego karty o wybranym kolorze lub wszystkie jego karty o wybranym numerze. Akcji tej nie można wykonać, jeśli w grze nie ma żadnych żetonów podpowiedzi, gdyż wiązałoby się to z koniecznością usunięcia żetonu podpowiedzi, który nie istnieje. Udzielanie graczom podpowiedzi dotyczących kart w jakikolwiek inny sposób jest zabronione.

Jeżeli któryś z graczy dobierze ostatnią kartę z talii, każdy uczestnik rozgrywa jedną dodatkową turę (wraz z graczem, który dobrał ostatnią kartę), a następnie gra się kończy.

Gra natychmiast kończy się, gdy zostanie utracony ostatni żeton życia lub gdy wszystkie stosy odpowiednich kolorów zostaną skompletowane (czyli na każdy z nich poprawnie położono kartę o numerze 5).

2.2. Dodatkowe obserwacje

Partie na dwóch graczy cechują się bardzo wysokim stopniem trudności. Uczestnicy rozgrywki znacznie częściej muszą radzić sobie ze skomplikowanymi sytuacjami, które wynikają z niekorzystnych rozdań. Fakt ten jest spotęgowany przez znikomą ilość dostępnych informacji (każdy z graczy widzi naraz tylko pięć kart swojego sojusznika), a także łatwość wpadnięcia w cykl, w którym jeden z graczy udziela podpowiedzi, a drugi zagrywa lub odrzuca karty.

Z powodu losowej natury gry, w niektórych rozdaniach nie jest możliwe zdobycie maksymalnej ilością punktów. Najprostsza taka sytuacja ma miejsce, gdy w rozgrywce na dwóch graczy jeden z nich dobierze same karty o numerze 5, drugi dobierze wyłącznie karty o numerze 4, a na górze talii znajdują się pozostałe karty o numerze 4. Aby uzyskać dostęp do kart o innych numerach, gracze muszą odrzucić (lub niepoprawnie zagrać) co najmniej sześć kart. Z zasady szufladkowej Dirchleta można wywnioskować, że wszystkie kopie co najmniej jednej z kart danego rodzaju zostaną bezpowrotnie usunięte z gry, bez umieszczania ich na stosie, co uniemożliwi wygraną.

Ze względu na ograniczony rozmiar talii, zagrywanie wyłącznie tych kart, o których posiada się komplet informacji, jest wysoce nieefektywne. Po rozdaniu kart talia zawiera od 30 do 40 kart, zależnie od liczby graczy. Aby uzyskać najwyższy wynik, należy zagrać aż 25 kart, a zagranie każdej z nich oznacza zmniejszenie rozmiaru talii o jeden. Oznacza to (zakładając, że gracze próbują uzyskać 25 punktów), że można wykonać maksymalnie od 10 do 17 ruchów, w których odrzuca się kartę, wliczając

tury po opróżnieniu talii. Po doliczeniu początkowych 8 żetonów maksymalna ilość możliwych do uzyskania podpowiedzi wynosi 25. Z tego wynika, że w grze na dwie osoby każda podpowiedź musi jednoznacznie ujawniać średnio po jednej karcie, lecz każda z nich potrzebuje dwóch podpowiedzi różnego rodzaju, by uzyskać pełną informację. Przy pięciu graczach każda podpowiedź musi średnio ujawniać już nie jedną, a prawie półtorej karty.

Ponadto, jeżeli chcemy odrzucać wyłącznie karty, które można bezpiecznie usunąć z gry, inni gracze muszą nam zakomunikować ich brak przydatności poprzez odpowiednie podpowiedzi (lub ich brak, co jest w znacznym stopniu utrudnione przez konieczność udzielania pełnej informacji o zawartości rąk współuczestników). Karty bezużyteczne mogą, lecz nie muszą zostać ujawnione w drodze przypadku, podczas ujawniania innych kart. W rezultacie, najczęściej przyjmowaną konwencją wśród graczy jest odrzucanie najstarszej karty w ręce: jeżeli żaden z uczestników rozgrywki nie ostrzega przed usunięciem jej z gry, istnieje wysokie prawdopodobieństwo, że jest ona nieprzydatna.

Rozdział 3.

Strategie sztucznej inteligencji

3.1. Podejście regułowe

Podczas realnej rozgrywki Hanabi gracze nie dysponują komputerem, który pomagałby im w wykonywaniu ruchów poprzez dokonywanie odpowiednich obliczeń. Zamiast tego korzystają oni z wiedzy nabytej w trakcie rozegranych już partii, modyfikując swoje strategie i rozszerzając je o kolejne elementy, aż do osiągnięcia zadowalającego ich poziomu wiedzy o grze. Proces ten w naturalny sposób prowadzi do wykształcenia konwencji (takich jak: "należy zawsze odrzucać najstarszą kartę w ręce"), a także do opracowania systemu reguł, pomocnych w uzyskiwaniu wysokich wyników (przykładowo: "jeżeli ktoś chce odrzucić ważną kartę, należy go powstrzymać poprzez udzielenie odpowiedniej podpowiedzi"). Zasady te można przełożyć na algorytmy, co prowadzi do utworzenia systemów regułowych, znanych także jako eksperckie.

Algorytmy regułowe to programy, naśladujące poprzez procesy decyzyjne wybory, których w danych sytuacjach mógłby dokonać człowiek. Są one najczęściej deterministyczne, co jest cechą szczególnie ważną w środowiskach, które wymagają koordynacji i przewidywania działań podejmowanych przez inne elementy systemu. W przypadku agentów sztucznej inteligencji, algorytmy regułowe powielają zachowania prawdziwych graczy.

Z racji na kooperacyjną konstrukcję Hanabi, sama emulacja typowych zachowań ludzkich graczy nie wystarcza, by wygrać. Potrzebna jest także odpowiednia koordynacja działań pomiędzy agentami: strategia. Gracze muszą zwracać uwagę na współuczestników rozgrywki i na bieżąco interpretować ich poczynania, by móc wywnioskować, jaka seria ruchów doprowadzi do najlepszej możliwej sytuacji. Prostym i efektywnym sposobem na zaimplementowanie strategii jest wyszczególnienie protokołu komunikacji pomiędzy agentami, który nadaje niektórym zagraniom dodatkowe znaczenie, rozumiane przez pozostałych graczy. Przykładowo, skuteczna może być zasada o następującej treści: "jeżeli inny gracz podpowiedział mi bez wy-

raźnej przyczyny jeden z kolorów, ujawniając w ten sposób kilka kart, najprawdopodobniej mogę je zagrać, w kolejności od lewej do prawej". Z tego powodu implementacja agentów regułowych wiąże się z koniecznością bardzo dobrej znajomości zasad rządzących grą.

3.2. Drzewa poszukiwań

Istnieją dwa główne czynniki, które sprawiają, że analiza stanu gry w Hanabi jest trudnym zadaniem. Są to: niepełna informacja o aktualnym etapie rozgrywki, a także losowość kart dobieranych z talii. Udowodniono, że nawet w uproszczonej wersji gry, w której uczestnicy mogą patrzeć na swoje karty, problem perfekcyjnego zagrania jest NP-kompletny[2]. Sprawia to, że próba rozwiązania zagadnienia w sposób siłowy jest nieefektywna. Fakt ten, połączony z trudnością opracowania funkcji oceniającej jakość zagrania, wyklucza użycie części możliwych rozwiązań problemu, takich jak algorytm alfa-beta.

Aby zredukować trudność znalezienia perfekcyjnego zagrania, grupa Facebook Research zaproponowała rozwiązanie bazujące na drzewie poszukiwań Monte Carlo[3]. Każdy z graczy dysponuje zbiorem predefiniowanych akcji, dostosowywanych odpowiednio do aktualnego stanu rozgrywki poprzez analizę prawdopodobieństwa zagrań, które mogą wykonać współuczestnicy. Algorytm bierze pod uwagę także szanse aktualnego gracza na posiadanie w ręce kart, które mogły zostać wylosowane z talii. Aby przyspieszyć działanie programu, głębokość drzewa poszukiwań jest ograniczana, a agenci wykonują predefiniowane akcje i nie eksplorują nowych opcji, jeśli wiązałoby się to z przekroczeniem zadanego limitu obliczeń.

Takie podejście pozwala na uzyskanie bardzo wysokich wyników, sięgających nawet 24.61 punktów w rozgrywce dla dwóch graczy. Działanie algorytmu jest jednak kosztowne obliczeniowo, nawet przy znacznym ograniczeniu zakresu dokonywanych poszukiwań. Do osiągnięcia tak wysokich punktacji potrzeba olbrzymich ilości obliczeń, które, z racji na możliwość ich zrównoleglenia, są najczęściej dokonywane na nowoczesnych kartach graficznych. Wyłączenie agentom możliwości dokonywania dodatkowych poszukiwań degeneruje ich do postaci agentów regułowych, którzy, choć na każdą akcję potrzebują zużycia istotnie mniejszej ilości zasobów, wciąż osiągają imponujący wynik 23 punktów.

3.3. Algorytmy uczące

Innym sposobem na zaimplementowanie programu grającego w Hanabi jest użycie algorytmów uczących, które łączą zalety podejść regułowych i poszukujących. Jak sugeruje nazwa, polegają one na symulowaniu procesu akumulacji doświadczenia, podobnego do tego doznawanego przez ludzkich graczy. W toku ewolucji agent

zdobywa wiedzę o środowisku, w którym operuje, dostosowując się do zmieniających warunków oraz wypracowując i udoskonalając sposoby radzenia sobie w zaprezentowanych sytuacjach. Tworzenie algorytmów uczących nie wymaga ani szerokiej wiedzy o zawiłościach zasad gry, ani kosztownych obliczeń, które byłyby wykonywane w trakcie rozgrywki.

Wadą tego algorytmu jest konieczność wyuczenia agenta odpowiednich zachowań. Odbywa się to poprzez zapewnienie mu zestawu danych, na których mógłby zdobyć doświadczenie. W zależności od uzyskiwanych wyników, decyzje algorytmu są nagradzane lub karane. Dobranie odpowiednio różnorodnego zbioru uczącego, w parze z funkcjami kwalifikującymi, pozwala programowi nie tylko na rozpoznawanie i radzenie sobie z najczęściej występującymi sytuacjami, ale i generalizację zachowań, potrzebną do wybrnięcia ze stanów gry, które nie zostały dotychczas napotkane.

Należy pamiętać, że nieodpowiedni dobór zbioru uczącego lub funkcji, które oceniają poczynania agenta, potrafi doprowadzić do anomalii w procesie zdobywania wiedzy. Jeżeli zestawy testowe będą zbyt homogeniczne i zbyt liczne w stosunku do osiągalnej liczby stanów rozgrywki, może dojść do przeuczenia modelu, z kolei zbyt krótka nauka nie przygotuje programu do nietypowych sytuacji. Nieprawidłowości w procedurach klasyfikujących, choć mniej zauważalne, także potrafią doprowadzić do niepożądanych sytuacji, tak jak miało to miejsce w przypadku programu grającego w produkcje na platformę Nintendo Entertainment System. Agent ten, nie chcąc doprowadzić do przegranej, nauczył się wstrzymywać rozgrywkę na zawsze[4].

3.4. Naginanie zasad gry

W oficjalnych zasadach gry nie ma wyszczególnionego przymusu udzielania podpowiedzi, które ujawniałyby jakiekolwiek karty. Jeżeli wybierzemy gracza, który nie posiada żadnych czerwonych kart i zdecydujemy się na podpowiedzenie mu czerwonego koloru, tura jest pomijana za cenę żetonu podpowiedzi. Choć taki ruch wydaje się nie mieć sensu, gdyż podpowiedź można wykorzystać w produktywny sposób, otwiera on możliwość poważnego nagięcia zasad gry. Jeżeli każdej z możliwych podpowiedzi przypiszemy unikatową wartość numeryczną, możemy za ich pomocą przekazywać innym graczom informacje liczbowe. Jest to powód, dla którego możliwość udzielania pustych podpowiedzi jest uznawana w społeczności graczy Hanabi za kontrowersyjną, toteż w niektórych edycjach gry została ona zakazana.

Fakt ten można wykorzystać do stworzenia agenta, który za pomocą pozornie bezwartościowych ruchów udziela podpowiedzi wszystkim graczom jednocześnie. Korzysta on ze słynnej zagadki logicznej, znanej jako problem więźniów i kapeluszy, odpowiednio uogólnionej i dopasowanej do liczby graczy. Agent, który rozgrywa aktualną turę, oblicza idealne zagrania dla innych uczestników rozgrywki, a następnie szyfruje je do postaci liczbowej. Kolejni gracze, znając podaną przez poprzednika

wartość, po rozpatrzeniu optymalnych zagrań innych graczy, są w stanie wywnioskować, jaki ruch powinni wykonać.

Według badań z 2017 roku[5], agent ten uzyskuje maksymalną ilość punktów średnio w 92% rozgrywanych gier, co jest wynikiem bliskim optymalnemu. To imponujący rezultat, zarówno ze względu na bardzo szybkie działanie algorytmu, jak i jego nadzwyczajną efektywność.

Niestety, taki sposób gry całkowicie zawodzi, gdy jeden z graczy wyłamie się z konwencji narzuconej przez protokół komunikacji. Dodatkowo, algorytm działa wyłącznie w rozgrywkach na czterech oraz pięciu graczy, gdyż głównym powodem jego sukcesu jest możliwość przekazywania w każdej z podpowiedzi maksymalnej ilości informacji, toteż zmniejszenie liczby graczy powoduje znaczne zredukowanie efektywności ruchów. Są to powody, dla których agent ten nie nadaje się do rozgrywki z człowiekiem.

Rozdział 4.

Funkcjonalna realizacja SI

W celu zbadania zaprezentowanych sposobów tworzenia sztucznej inteligencji, która potrafiłaby grać w Hanabi, zaimplementowaliśmy dziewięciu agentów. Aby umożliwić testowanie ich możliwości, opracowaliśmy także graficzny interfejs (patrz: Rysunek 2.1), pozwalający użytkownikowi na aktywne uczestnictwo w rozgrywce, złożonej z dowolnie dobranego przez niego składu graczy komputerowych.

Siedmiu agentów jest opartych o systemy regułowe, różniące się stopniem zaawansowania wdrażanych strategii. Aby usprawnić działanie jednego z agentów regułowych, użyliśmy bayesowskiej metody optymalizacji hiperparametrów, korzystającej z procesów gaussowskich. Ostatni z zaprezentowanych agentów stosuje algorytm uczący ze wzmocnieniem, oparty o funkcje heurystyczne generalizujące stan gry, wspomagane procesami decyzyjnymi Monte Carlo.

Systemy zasad zostały oparte o nasze własne spostrzeżenia, zdobyte poprzez obserwację rozgrywek ludzkich graczy, które uznaliśmy za kluczowe do gry na wysokim poziomie. Każdy kolejny agent regułowy bazuje na konceptach, które zostały użyte do zbudowania jego poprzedników, i usprawnia je. Takie działanie prowadzi do naturalnej progresji warstw abstrakcji prezentowanych pojęć.

Nomenklatura

μ — średnia arytmetyczna
$\tilde{\mu}$ — mediana
σ^2 — wariancja
σ — odchylenie standardowe

4.1. Cheater

Cheater jest agentem, którego strategia polega na bezmyślnym oszukiwaniu. Program oszukuje, dzięki czemu zawsze posiada komplet informacji o kartach, którymi dysponuje, lecz nie wykorzystuje dostępnej mu wiedzy w optymalny sposób: zagrywa karty, gdy to możliwe, w przeciwnym wypadku losowo je odrzuca.

Nowe pojęcia

- **Legalna akcja**:

  Jeżeli wybrana akcja może być wykonana w danym momencie rozgrywki, jest ona legalna, niezależnie od poprawności. Przykładowo: udzielenie informacji w momencie, w którym na planszy nie znajduje się ani jeden żeton podpowiedzi, jest nielegalne. Jeżeli akcja jest nielegalna, korespondująca z nią część algorytmu kończy się niepowodzeniem.

- **Ujawniona karta**:

  Jeżeli gracz dysponuje kartą, o której posiada komplet informacji, kartę tę określimy jako ujawnioną.

- **Bezpieczna akcja**:

  Jeżeli gracz może wykonać ruch, który nie doprowadzi do utraty żetonów życia, taką akcję określimy jako bezpieczną.

- **Bezpieczne zagranie**:

  Jeżeli gracz posiada ujawnioną kartę, którą może bezpiecznie (czyli poprawnie) położyć na planszy, wykonanie takiej akcji nazwiemy bezpiecznym zagraniem.

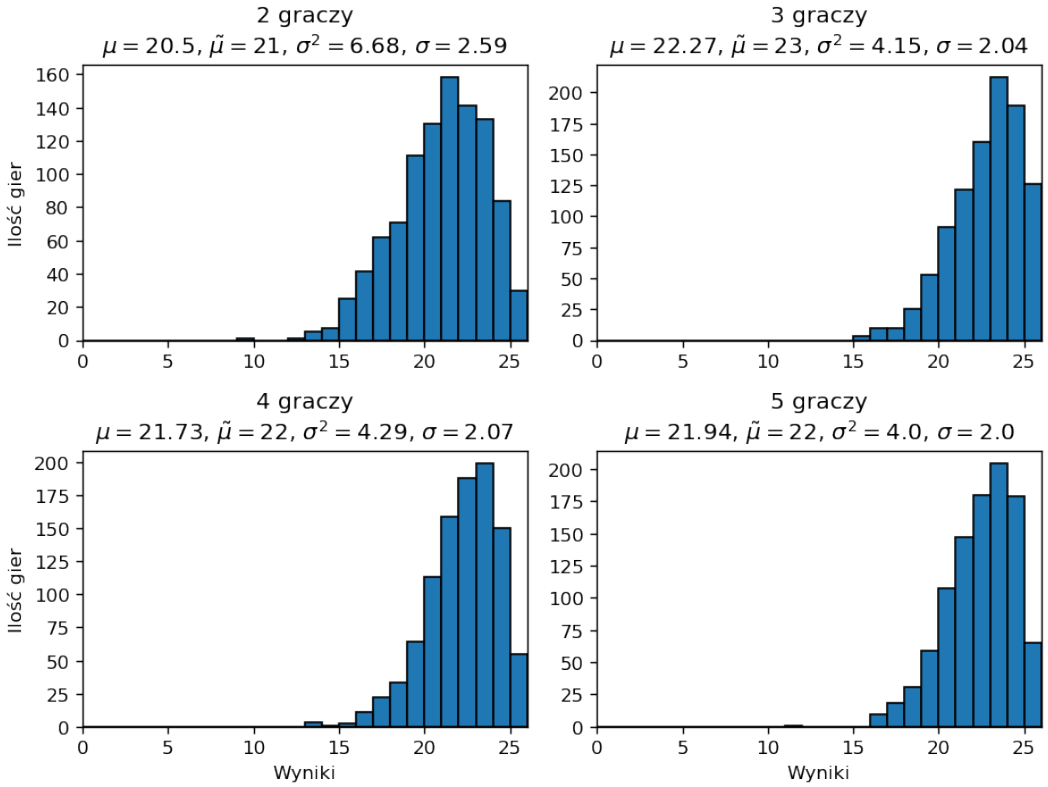
Schemat działania

1. Podejrzyj posiadane karty.

2. Jeżeli posiadasz bezpieczne zagranie, wykonaj je.

3. Odrzuć losową kartę.

Osiągi



Rysunek 4.1: Wyniki agenta Cheater (rozmiar próby: 1000 gier)

Uzyskane wyniki są bardzo dobre, lecz cechują się stosunkowo wysoką wariancją. Agenci są skłonni do odrzucania kluczowych kart i nie zwracają uwagi na fakt, że może to znacząco utrudnić lub nawet uniemożliwić dalsze zdobywanie punktów.

4.2. SmartCheater

SmartCheater usprawnia poczynania agenta Cheater. Program oszukuje, dzięki czemu zawsze posiada komplet informacji o kartach, którymi dysponuje, i wykorzystuje tę wiedzę w sposób bliski optymalnemu, stosując strategię przedłużającą rozgrywkę. Agent potrafi oceniać przydatność kart i za wszelką cenę unika ruchów, które utrudniają lub uniemożliwiają wygraną.

Nowe pojęcia

- **Zagranie lawinowe**:

  Jeżeli położenie karty na planszy pozwala innym graczom na wykonywanie nowych ruchów, które kontynuują dokładanie kart do stosów, takie zagranie

nazwiemy lawinowym. Przykładowo: jeżeli aktualny gracz poprawnie zagra czerwoną kartę o numerze 1, a dwaj kolejni gracze posiadają kolejno czerwone karty o numerach 2 i 3, zagranie jest lawinowe.

- **Bezużyteczna karta**:

  Jeżeli nie istnieje i nie będzie istniała możliwość poprawnego zagrania karty, jest ona bezużyteczna i może być odrzucona bez żadnych konsekwencji. Przykładowo: gdy stos kart koloru zielonego jest w pełni ułożony, nie można poprawnie zagrać żadnych dodatkowych kart koloru zielonego.

- **Karta krytyczna**:

  Jeżeli karta nie jest bezużyteczna i nie istnieją już inne jej kopie, które można by zagrać, karta jest krytyczna. Przykładowo: każda karta o numerze 5 jest krytyczna.

- **Spasowanie tury**:

  Jeżeli na planszy znajduje się co najmniej jeden żeton podpowiedzi, udzielenie losowej podpowiedzi dowolnemu z graczy jest równoznaczne ze spasowaniem tury, gdyż nie jest dobierana żadna nowa karta (chyba, że dobrano ostatnią kartę z talii).

Schemat działania

1. Podejrzyj posiadane karty.

2. Jeżeli posiadasz bezpieczne zagranie, wykonaj je. Pierwszeństwo nadawane jest zagraniom lawinowym, a w razie konfliktu wygrywa losowa karta o najniższym numerze.

3. Jeżeli możesz spasować turę, a ruch ten nie przeszkodzi żadnemu z pozostałych graczy w wykonaniu jednej z akcji od 2 do 5, zrób to.

4. Jeżeli posiadasz bezużyteczną kartę, odrzuć ją.

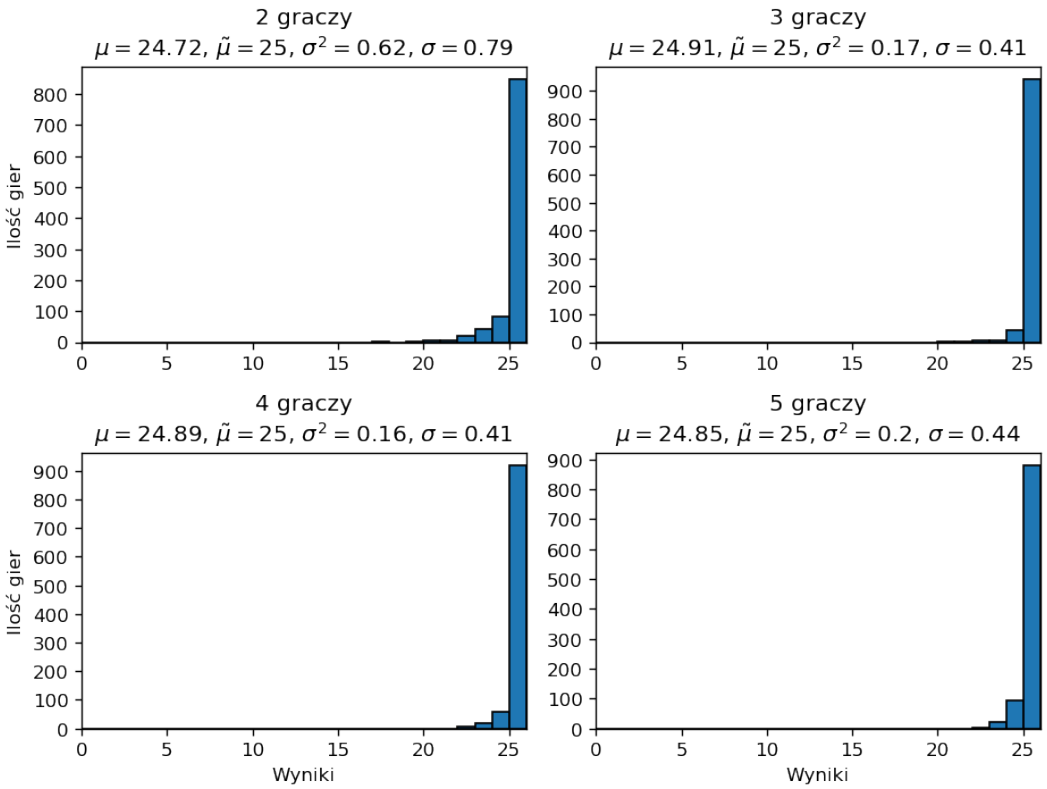
5. Jeżeli jest to możliwe, spasuj turę.

6. Jeżeli posiadasz kartę, której kopia znajduje się w ręce innego gracza, odrzuć ją.

7. Jeżeli posiadasz kartę, która nie jest krytyczna, odrzuć ją.

8. Odrzuć losową kartę krytyczną o najwyższym numerze.

Osiągi



Rysunek 4.2: Wyniki agenta SmartCheater (rozmiar próby: 1000 gier)

Jak można oczekiwać po agencie, który posuwa się do oszustwa bliskiego doskonałemu, jego wyniki są bardzo wysokie - nawet w przypadku partii na dwóch graczy, które cechują się największymi dysproporcjami w rozgrywce. Warto zauważyć, że istnieją rozdania cechujące się tak wysokim stopniem trudności, że nawet ten agent uzyskuje w nich mniej niż 20 punktów.

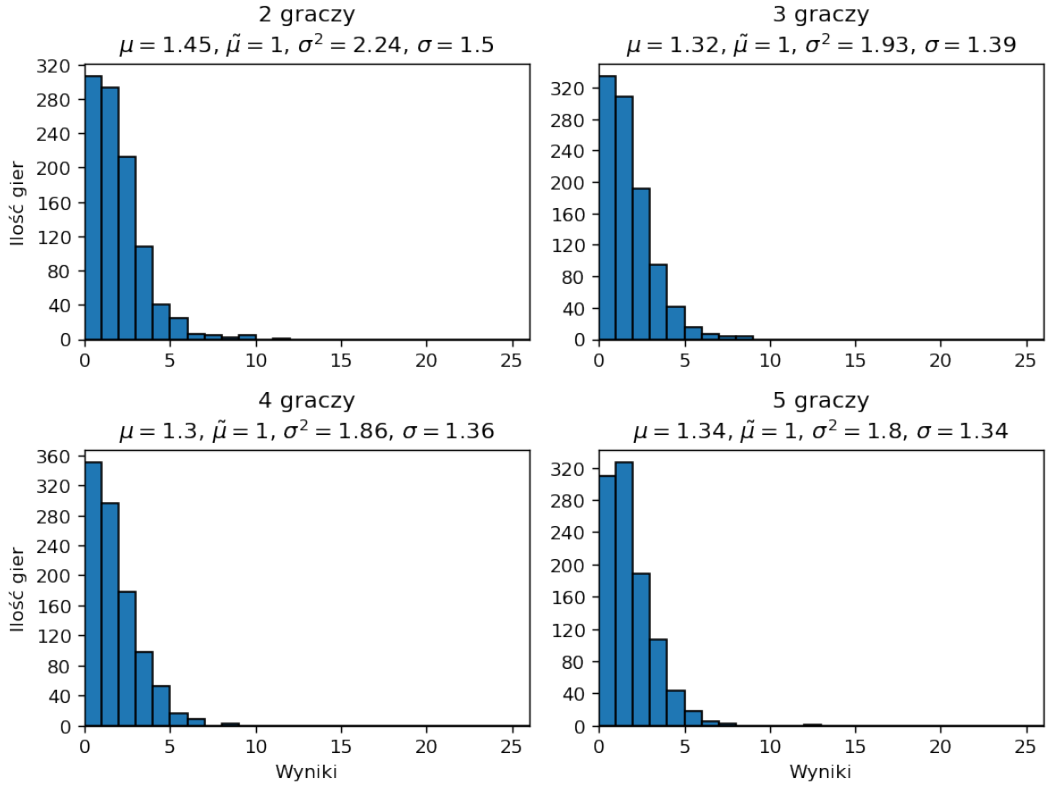
4.3. Erratic

Strategia agenta Erratic polega na wykonywaniu serii przypadkowych ruchów, które są legalne.

Schemat działania

- Wylosuj jedną z legalnych akcji (zagraj kartę, odrzuć kartę, udziel podpowiedzi innemu uczestnikowi rozgrywki).
- Losowo wybierz szczegóły ruchu, a następnie go wykonaj.

Osiągi



Rysunek 4.3: Wyniki agenta Erratic (rozmiar próby: 1000 gier)

Agent wykonujący wyłącznie losowe ruchy jest pierwszym tu opisanym, który nie oszukuje. Niestety, jak można było przewidywać, przegrywa on większość gier, uzyskując wynik 0 punktów. Jest to spowodowane brakiem mechanizmu, który powstrzymywałby go przed zagrywaniem kart, które nie są bezpieczne, toteż zagrania mają małą szansę na bycie poprawnymi. Choć istnieje szansa na wykonanie akcji udzielenia podpowiedzi, które w tym przypadku są bezużyteczne, ruch nie zaburza wyników, gdyż wykorzystanie żetonu podpowiedzi nie prowadzi do eskalacji progresu gry (jeżeli nie dobrano ostatniej karty z talii).

4.4. StoppedClock

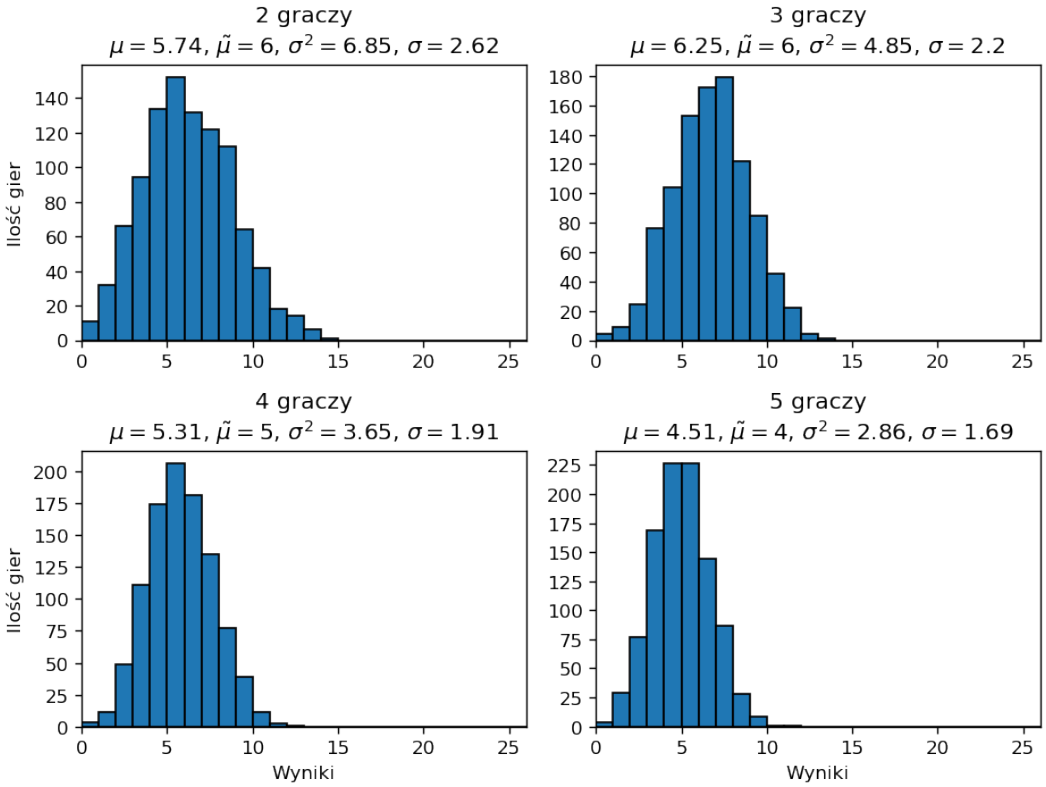
Agent StoppedClock rozszerza strategię agenta Erratic: wykonuje losowe ruchy, które nie mogą doprowadzić do utraty żetonów życia. Jego postępowanie, choć nadal bardzo chaotyczne, czasem prowadzi do wysokich punktacji, zgodnie z maksymą: “nawet zepsuty zegar dwa razy na dobę pokazuje właściwą godzinę”.

Schemat działania

Ruch agenta polega na zbadaniu, które z trzech dostępnych akcji są legalne, a następnie wylosowaniu i wykonaniu jednej z nich. Ponieważ tylko druga i trzecia akcja mogą wzajemnie się wykluczać, gracz zawsze będzie w stanie wykonać co najmniej jeden z ruchów. Dostępne akcje to:

- Jeżeli posiadasz bezpieczne zagranie, wykonaj je.
- Odrzuć losową kartę, priorytetyzując tę, która ma najmniej ujawnionych informacji. Akcja nie może być wykonana, jeżeli na planszy znajduje się osiem żetonów podpowiedzi.
- Wybierz losowego gracza, który posiada co najmniej jedną kartę o niepełnej informacji, wylosuj jedną, po czym losowo podpowiedz kolor lub numer wybranej karty, jeżeli nie zostały one uprzednio ujawnione. Jeżeli wszyscy gracze posiadają komplet informacji, spasuj turę.

Osiągi



Rysunek 4.4: Wyniki agenta StoppedClock (rozmiar próby: 1000 gier)

Agent ma nikłą szansę na uzyskanie wysokich wyników, a jego konstrukcja nigdy nie pozwoli na wygraną. Jest to spowodowane ograniczeniem, które umożliwia

programowi funkcjonowanie: wykonywane ruchy muszą być bezpieczne, co koliduje z ograniczoną liczbą żetonów podpowiedzi.

4.5. SimpleDistrustful

Agent SimpleDistrustful imituje postępowanie człowieka, który po raz pierwszy gra w Hanabi: wykonuje wyłącznie bezpieczne ruchy, lecz robi to umiejętnie, gdyż potrafi przeprowadzać pewne wnioskowania, które na jego miejscu mogłyby dokonać osoby o podstawowej znajomości zasad. Program nie udziela zbędnych podpowiedzi, stroni od odrzucania kart, które mogą się jeszcze przydać, a także rozumie, że niektóre karty zawsze można bezpiecznie zagrać lub odrzucić - mimo posiadania o nich niepełnej informacji.

Nowe pojęcia

- **Inferowalna karta**:

  Użyteczności niektórych kart można dociec poprzez analizę aktualnego stanu rozgrywki. Przykładowo: na początku gry zawsze można zagrać dowolną kartę z numerem 1, niezależnie od tego, czy jej kolor został ujawniony. Podobnie, jeżeli nie odrzucono ani nie zagrano żadnej karty o numerze 4, żadna taka karta nie może być krytyczna. Karty o takiej właściwości nazywamy inferowalnymi, czyli dającymi się wywnioskować. W pełni ujawnione karty są trywialnie inferowalne.

- **Wartościowa podpowiedź**:

  Zdarzają się sytuacje, w których udzielanie pewnych informacji jest niepożądane. Przykładowo: jeżeli dwóch lub więcej uczestników rozgrywki posiada kopię danej karty, warto skupić się na ujawnianiu wyłącznie jednej z nich. Wartościowa podpowiedź to taka, która jest lokalnie optymalna, gdyż unika powtarzania znanych informacji.

- **Najbliższy gracz**:

  Uczestnik rozgrywki, który po zakończeniu akcji aktualnego gracza wykona ruch najwcześniej.

Schemat działania

1. Jeżeli posiadasz inferowalnie bezpieczne zagranie, wykonaj je.

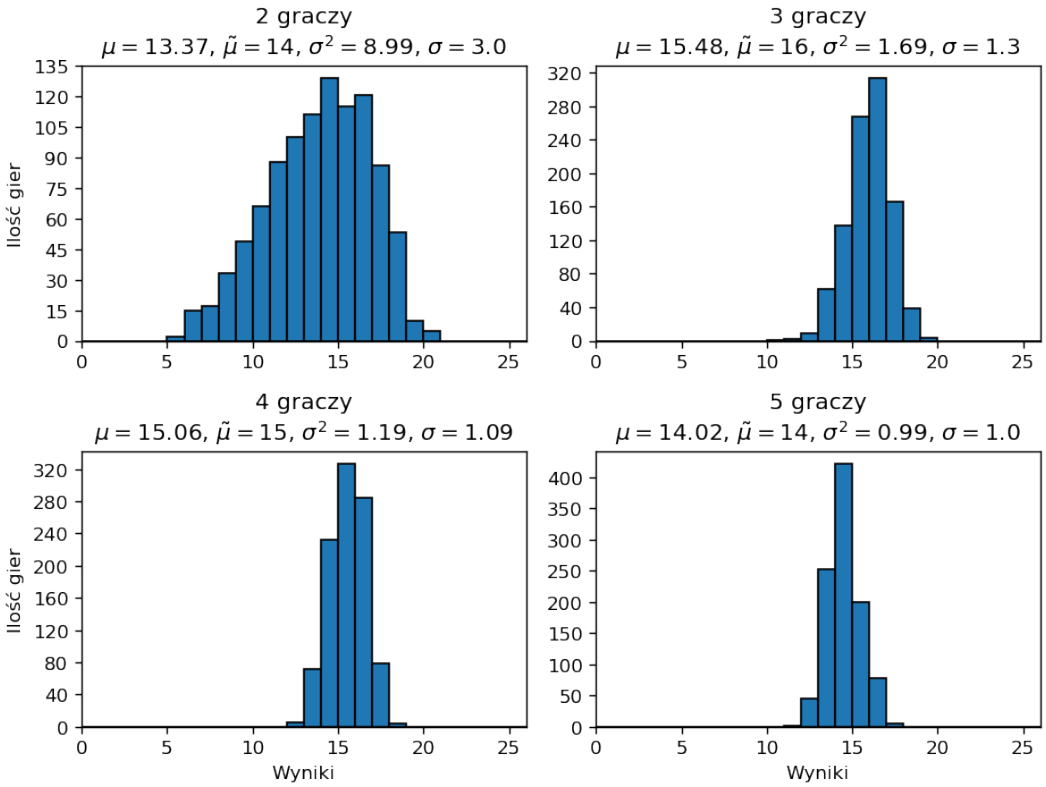
2. Jeżeli jest to możliwe, wybierz najbliższego gracza, któremu możesz udzielić wartościowej podpowiedzi, a następnie zrób to, nadając priorytet kartom o niższych numerach.

3. Jeżeli posiadasz nieujawnioną kartę, odrzuć ją.

4. Jeżeli posiadasz kartę, która nie jest inferowalnie krytyczna, odrzuć ją.

5. Odrzuć losową kartę.

Osiągi



Rysunek 4.5: Wyniki agenta SimpleDistrustful (rozmiar próby: 1000 gier)

Agent nie uzyskuje wysokich wyników, gdyż opiera swoje działania na przeciętnej strategii. Udzielanie wyłącznie wartościowych informacji skłania graczy do zagrywania kart, lecz często nie wystarcza do ochrony kart krytycznych - nieujawnione karty wymagają pewnej formy ochrony, gdyż mogły być dobrane dopiero niedawno. Szczególnie zawodzą wyniki rozgrywek dla dwóch graczy, które cechują się nadzwyczaj wysoką niestabilnością.

4.6. Distrustful

Agent Distrustful dopracowuje strategię stosowaną przez SimpleDistrustful. Program nadal gra zachowawczo, lecz ma do dyspozycji maksymalnie rozszerzony system dedukcji. Aby zminimalizować prawdopodobieństwo odrzucania kart krytycznych,

został on dodatkowo usprawniony o konwencję, która chroni najnowsze karty poprzez nadawanie im wysokich priorytetów. Aby zwiększyć potencjalną przydatność każdej z dokonywanych akcji, agent dynamicznie dostosowuje swój sposób postępowania do aktualnego stanu rozgrywki i zmienia priorytety manewrów, w zależności od ilości pozostałych żetonów podpowiedzi. Pozwala to na dopasowywanie przyjmowanej taktyki, przy jednoczesnym zachowaniu zasad ugruntowanych przez narzuconą strategię.

Nowe pojęcia

- **Pułap jakościowy**:

  Jedna z miar używanych w funkcjach oceniających przydatność ruchów: jeżeli uzyskana przez daną akcję nota jest niższa od pułapu, to jest ona dyskwalifikowana jako niewystarczająco pomocna.

- **Heurystycznie wartościowa akcja**:

  Niektóre akcje, takie jak bezpieczne zagranie, w oczywisty sposób mają korzystny wpływ na uzyskanie wyższego wyniku gry. Niestety, przypisanie części ruchów jednoznacznej wartości liczbowej, wskazującej na jej przydatność w kontekście danego momentu rozgrywki, jest trudnym zadaniem, gdyż pełna analiza możliwych następstw akcji wymagałaby wysokiego nakładu mocy obliczeniowej. Aby uniknąć tego problemu, można zastosować funkcje heurystyczne, które przybliżają wartości poszukiwanych ocen. Akcja, która uzyska najwyższą notę, a także będzie wyższa od zadanego pułapu jakościowego, jest nazywana heurystycznie wartościową.

- **Najstarsza (najmłodsza) karta**:

  Karta, która znajduje się w ręce danego gracza od największej (najmniejszej) ilości tur, w porównaniu do innych jego kart. W razie konfliktu za najstarszą (najmłodszą) kartę uznawana jest ta wysunięta najbardziej na lewo (prawo).

Schemat działania

Agent jest złożony z siedmiu modułów, które są rozważane po kolei w jednej z trzech konfiguracji. Każdy moduł, poza ostatnim w sekwencji, może zawieść. Ich kolejność zależy od ilości posiadanych żetonów podpowiedzi:

- Mniej niż 3 żetony: sekwencja 1, 2, 5, 3, 4, 7 (moduł 6 nie jest używany).
- Od 3 do 7 żetonów: sekwencja 1, 2, 3, 4, 5, 6, 7.
- Dokładnie 8 żetonów: sekwencja 1, 2, 3, 4, 6, 5, 7.

Używane moduły to:

1. "Necessary tip":

   Jeżeli najbliższy gracz planuje w kolejnym ruchu odrzucić kartę krytyczną, która mogłaby zostać poprawnie zagrana, powstrzymaj go poprzez udzielenie odpowiedniej podpowiedzi.

2. "Obvious play":

   Jeżeli posiadasz inferowalnie bezpieczne zagranie, wykonaj je, nadając priorytet kartom o niższych numerach.

3. "Good play tip":

   Jeżeli posiadasz co najmniej dwa żetony podpowiedzi i istnieje heurystycznie wartościowa podpowiedź, udziel jej, nadając priorytet tym, które umożliwią inferowalnie bezpieczne zagranie.

4. "Discard tip":

   Jeżeli posiadasz co najmniej dwa żetony podpowiedzi i istnieje heurystycznie wartościowa podpowiedź, która umożliwi innemu graczowi odrzucenie inferowalnie bezużytecznej karty, udziel jej.

5. "Obvious discard":

   Jeżeli posiadasz inferowalnie bezużyteczną kartę, odrzuć ją.

6. "Mediocre play tip":

   Jeżeli posiadasz co najmniej dwa żetony podpowiedzi, wykonaj akcję modułu numer 3 ("Good play tip"), ale ze znacznie niższym pułapem jakościowym.

7. "Guess discard":

   (a) Jeżeli posiadasz nieujawnione karty, odrzuć najstarszą z nich.
   (b) Jeżeli posiadasz karty, które nie są inferowalnie krytyczne, odrzuć najstarszą z nich.
   (c) Odrzuć najstarszą kartę.

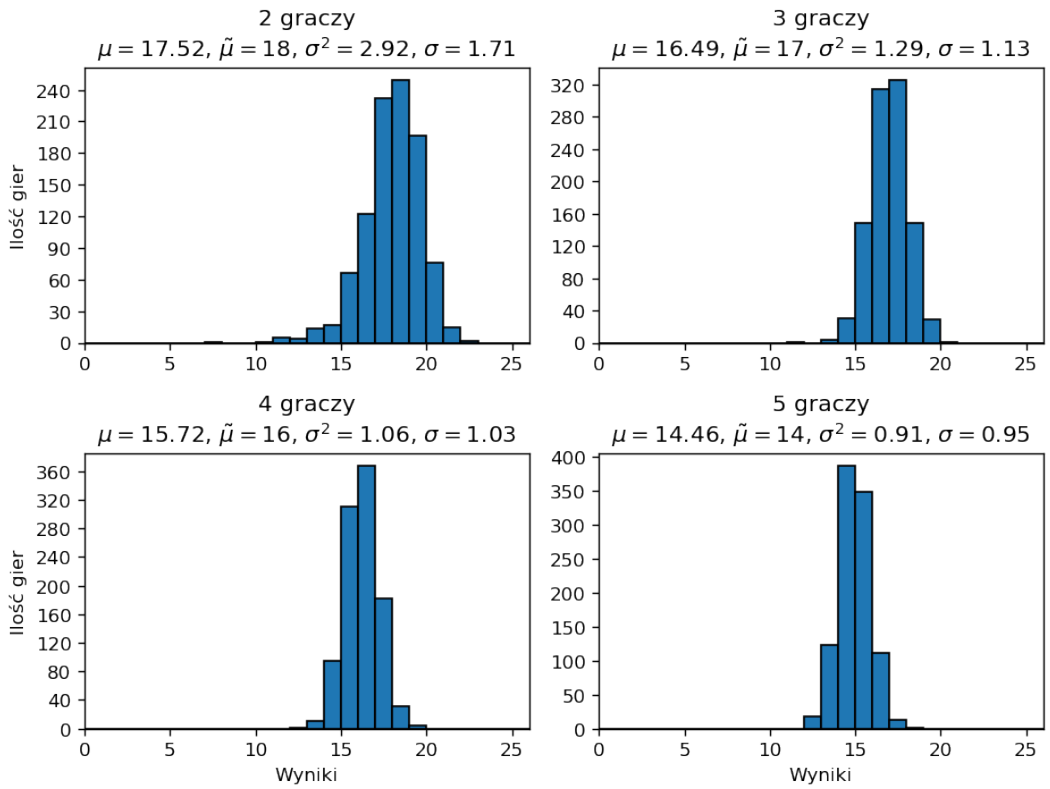
Zmiana kolejności manewrów pozwala na kontrolowanie priorytetów agenta: im mniej żetonów podpowiedzi znajduje się w grze, tym bardziej nagląca staje się potrzeba ich odzyskania. Nigdy nie wiadomo, czy i kiedy któryś z graczy dobierze kartę wymagającą natychmiastowej interwencji w formie dodatkowych informacji. Jest to powód, dla którego moduły, udzielające podpowiedzi do kart niebędących krytycznymi, zawodzą przy próbie wykorzystania ostatniego z żetonów.

Każda z kart jest oceniana z osobna przy użyciu funkcji heurystycznych. Za ostateczną wartość podpowiedzi przyjmuje się sumę ocen wszystkich kart, które zostaną za jej pomocą ujawnione. Wagi funkcji heurystycznych zostały dobrane eksperymentalnie. Na werdykt algorytmu wpływają między innymi następujące czynniki:

- stopień ujawnienia karty,
- numer karty,
- czy zagranie karty jest lawinowe,
- czy podpowiedź jest wartościowa,
- ilość kart, które muszą być zagrane, zanim zagranie karty będzie bezpieczne,
- odległość właściciela karty od gracza udzielającego podpowiedzi.

Początkowo, moduł numer 1 ("Necessary tip") przyjmował inną formę. Agent był szczególnie ostrożny i zawsze ostrzegał kolejnego gracza, jeżeli ten mógł odrzucić kartę krytyczną. W praktyce doprowadzało to do powstawania cykli, w których gracze naprzemiennie udzielali podpowiedzi i odrzucali karty, przez co częstotliwość zagrań drastycznie spadała. Choć pierwotny moduł znacznie pogarszał wyniki agenta Distrustful, jego zrewidowana wersja jest punktem centralnym kolejnego, bardziej zaawansowanego agenta.

Osiągi



Rysunek 4.6: Wyniki agenta Distrustful (rozmiar próby: 1000 gier)

Mimo znacznego wzmocnienia strategii agenta SimpleDistrustful, osiągane wyniki nadal nie zachwycają. Gracze udzielają heurystycznie wartościowych podpowiedzi, chronią krytyczne karty poprzez system wieku i oszczędzają żetony na wyjątkowe sytuacje - trudno jednak doszukiwać się między nimi głębszej współpracy, która jest kluczowa dla wysokich osiągów. Zagrywane są wyłącznie karty inferowalnie bezpieczne, co, nawet przy bardzo ostrożnej i wydajnej komunikacji, nie wystarcza do podjęcia równej walki z szybko kurczącą się ilością żetonów podpowiedzi. Aby przekroczyć wyczekiwany próg średniej ilości 20 punktów, potrzeba strategii, która dopuszcza pewne ryzyko.

4.7. Trustful

Poprzedników agenta Trustful nękają dwa zasadnicze problemy. Są to: trudność związana z efektywnym przekazywaniem informacji, a także wystrzeganie się zagrań, które nie są bezpieczne. Obie te przeszkody można pokonać poprzez ustanowienie protokołu komunikacji, który nadaje niektórym akcjom dodatkowego, implicytnego znaczenia, rozumianego przez pozostałych uczestników rozgrywki. Podpowiedzi udzielane przez agenta przekazują nie tylko informacje o kartach, ale i plan działania na kolejne tury. Aby częściowo rozwiązać problem ograniczonej ilości żetonów podpowiedzi, program potrafi, przy spełnieniu odpowiednich warunków, wykonywać ruchy, które nie są inferowalne, lecz mają bardzo wysokie prawdopodobieństwo bycia bezpiecznymi.

Nowe pojęcia

- **Obietnica**:

  Jeżeli agent udzieli informacji, która, zgodnie z protokołem komunikacji, wyznacza plan zagrywania kolejnych kart, taką podpowiedź nazywamy obietnicą.

- **Komunikat**:

  Podpowiedź, która nie jest obietnicą.

- **Podpowiedź ratunkowa**:

  Heurystycznie wartościowa podpowiedź, będąca obietnicą lub komunikatem, która jest rozważana wyłącznie dla najbliższego gracza.

- **Weto**:

  Akt unieważnienia efektu obietnicy karty. Jeżeli istnieje podpowiedź ratunkowa, która umożliwia danemu graczowi wykonanie jakiegoś ruchu, weto polega na jej udzieleniu, w przeciwnym wypadku weta dokonuje się poprzez ujawnienie danej karty.

Schemat działania

Agent posiada dwie kategorie podpowiedzi: obietnice oraz komunikaty. Ustalenie ich typu odbywa się według następującego schematu:

1. Jeżeli gracz otrzyma od swojego bezpośredniego poprzednika podpowiedź ratunkową obejmującą kartę, która miała zostać właśnie odrzucona, podpowiedź jest komunikatem.
2. Jeżeli gracz otrzyma od swojego bezpośredniego poprzednika dowolną podpowiedź, gdy jedna z kart objętych obietnicą miała zostać właśnie zagrana, ta karta jest wetowana, a typ podpowiedzi jest rozpatrywany ponownie, z pominięciem tego punktu schematu.
3. Jeżeli gracz otrzyma podpowiedź wskazującą na numer, którego poprawne zagranie byłoby inferowalnie niemożliwe w turze podpowiadającego gracza, podpowiedź jest komunikatem.
4. Podpowiedź jest obietnicą.

Agent jest złożony z dziesięciu modułów, które są rozważane po kolei w jednej z czterech konfiguracji. Każdy moduł, poza ostatnim w sekwencji, może zawieść. Ich kolejność zależy od ilości posiadanych żetonów podpowiedzi:

- Mniej niż 3 żetony: sekwencja 1, 2, 3, 9, 4, 5, 10 (moduły 6, 7, 8 nie są używane).
- Od 3 do 5 żetonów: sekwencja 1, 2, 3, 4, 5, 9, 10 (moduły 6, 7, 8 nie są używane).
- 6 lub 7 żetonów: sekwencja 1, 2, 3, 4, 5, 9, 6, 7, 10 (moduł 8 nie jest używany).
- Dokładnie 8 żetonów: sekwencja 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Używane moduły to:

1. "Necessary tip":

   Rozważ możliwe ruchy najbliższego gracza:

   (a) Jeżeli może on niepoprawnie zagrać kartę objętą obietnicą, lecz aktualnie posiadasz inferowalnie bezpieczne zagranie, które sprawia, że będzie ona zagrana poprawnie, wykonaj je - w przeciwnym wypadku zawetuj kartę.

   (b) Jeżeli może on odrzucić kartę krytyczną, lecz istnieje podpowiedź ratunkowa, która umożliwiłaby mu wykonanie innego ruchu, udziel jej - w przeciwnym wypadku ujawnij kartę, priorytetyzując informację o numerze.

2. "Obvious play":

   Jeżeli posiadasz inferowalnie bezpieczne zagranie, wykonaj je, nadając priorytet kartom o niższych numerach.

3. "Hinted play":

   Jeżeli posiadasz karty objęte obietnicą, których zagranie nie może być w oczywisty sposób niepoprawne, zagraj tę najbardziej wysuniętą na lewo, nadając priorytet kartom o ujawnionym numerze i wybierając spośród nich te, których numer jest najniższy.

4. "Play tip":

   Jeżeli istnieje heurystycznie wartościowa podpowiedź, która umożliwi innemu graczowi inferowalnie bezpieczne zagranie (lub zagranie bazujące na obietnicy), udziel jej.

5. "Discard tip":

   Jeżeli istnieje heurystycznie wartościowa podpowiedź, która umożliwi innemu graczowi odrzucenie inferowalnie bezużytecznej karty, udziel jej.

6. "Risky tip":

   Spróbuj wykonać akcję modułu numer 4 ("Play tip") z obniżoną karą za błędne obietnice i zwiększonym pułapem jakościowym.

7. "Save tip":

   Jeżeli istnieje heurystycznie wartościowa podpowiedź ujawniająca kartę krytyczną, którą można wykonać w trybie komunikatu, udziel jej, nadając priorytet graczom, którzy nie mają inferowalnie bezpiecznego zagrania lub dobrej obietnicy, a także preferując karty o niższych numerach.

8. "Information tip":

   Wykonaj akcję modułu numer 4 ("Play tip") ze zwiększoną nagrodą za udzielanie podpowiedzi w trybie komunikatu.

9. "Obvious discard":

   Jeżeli posiadasz inferowalnie bezużyteczną kartę, odrzuć ją.

10. "Guess discard":

    (a) Jeżeli posiadasz nieujawnione karty, odrzuć najstarszą z nich.

    (b) Jeżeli posiadasz karty, które nie są inferowalnie krytyczne, odrzuć najstarszą z nich.

    (c) Odrzuć najstarszą kartę.

Wagi funkcji heurystycznych ponownie zostały dobrane eksperymentalnie, lecz sposób oceny akcji znacznie różni się od tego znanego z agenta Distrustful. Z racji na ambiwalencję udzielanych typów podpowiedzi, należy uważać nie tylko na jakość przekazywanych informacji, ale i ich znaczenie w kontekście protokołu komunikacji.

Poza czynnikami wyszczególnionymi podczas analizy agenta Distrustful, na werdykt algorytmu wpływają między innymi:

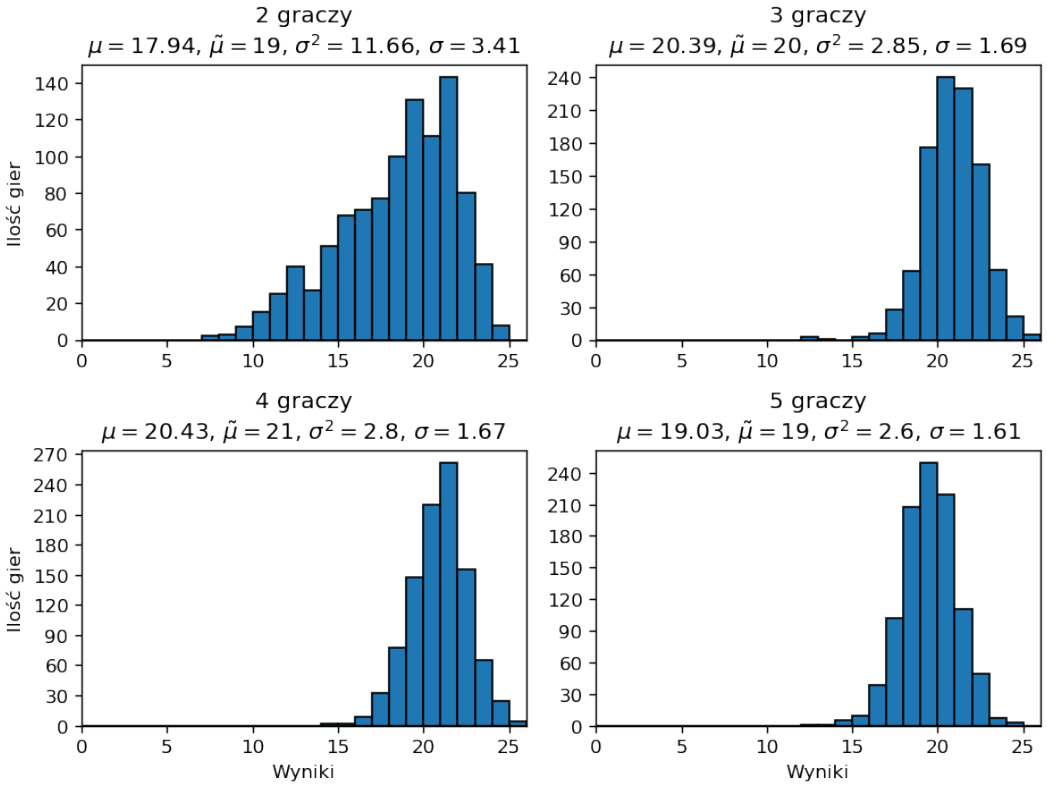
- czy podpowiedź jest obietnicą,
- czy komunikat zostanie uznany za obietnicę, mimo że tego nie chcemy,
- czy karta może stać się grywalna w turze danego gracza,
- czy karta jest pod działaniem efektu obietnicy,
- czy podpowiedź zniweczy przydatną obietnicę,
- czy podpowiedź jest ryzykowna,
- kolejność ujawnianych kart w ręce,
- ilość pozostałych żetonów życia.

Podpowiadanie kart w trybie obietnicy rzadko przebiega w bezproblemowy sposób: karty tworzące dobry porządek w ręce gracza to nieczęste zjawisko. Z tego powodu funkcje heurystyczne agresywnie zaniżają noty obietnic, których pierwszy element jest niepoprawny, lecz relaksują kary dla podpowiedzi umożliwiających co najmniej jedno bezpieczne zagranie.

Udzielanie podpowiedzi podlega pewnemu rygorowi, lecz czasami potencjalny zysk z udzielenia informacji o wielu kartach naraz przeważa nad ryzykiem związanym z niepoprawnością obietnicy. Ponieważ agent stara się zachowywać co najmniej jeden żeton podpowiedzi na takie przypadki, ryzyko niefortunnego zagrania drastycznie spada. Niestety, nadal może dojść do sytuacji, w których gracze tracą żetony życia poprzez niepoprawne zagrania. Aby temu zapobiec, agent ogranicza zaufanie do innych graczy, gdy w grze pozostał ostatni żeton życia - zagrywane są wyłącznie te karty objęte obietnicą, które zostały podpowiedziane pojedynczo. W ten sposób całkowicie eliminowane jest ryzyko niebezpiecznych zagrań, lecz agent nadal nie degeneruje swojej strategii do tej znanej z agenta Distrustful.

Strategia agenta nie jest przystosowana do rozgrywek dwuosobowych, gdyż udzielenie dowolnej podpowiedzi wetuje kartę, która miała szansę być właśnie zagrana - prowadzi to do konieczności omijania systemu wetowania obietnic. Można to osiągnąć wyłącznie poprzez wykonanie jednej z pozostałych akcji, co jest trudnym zadaniem, gdyż wiąże się ono z koniecznością dobrej znajomości własnych kart.

Osiągi



Rysunek 4.7: Wyniki agenta Trustful (rozmiar próby: 1000 gier)

Agent Trustful nie tylko potrafi przekroczyć próg 20 punktów, ale i zdarza mu się wygrać: zastosowanie protokołu implicytnej komunikacji pozwoliło wynieść osiągi sztucznej inteligencji na nowy poziom. Niestety, strategia programu nie jest przystosowana do rozgrywek dwuosobowych, co objawia się wysoką niestabilnością algorytmu. Wartym zaobserwowania faktem, który można odczytać z wykresu, jest, występujący od pewnego momentu rozgrywki, wykładniczy wzrost trudności uzyskiwania kolejnych punktów.

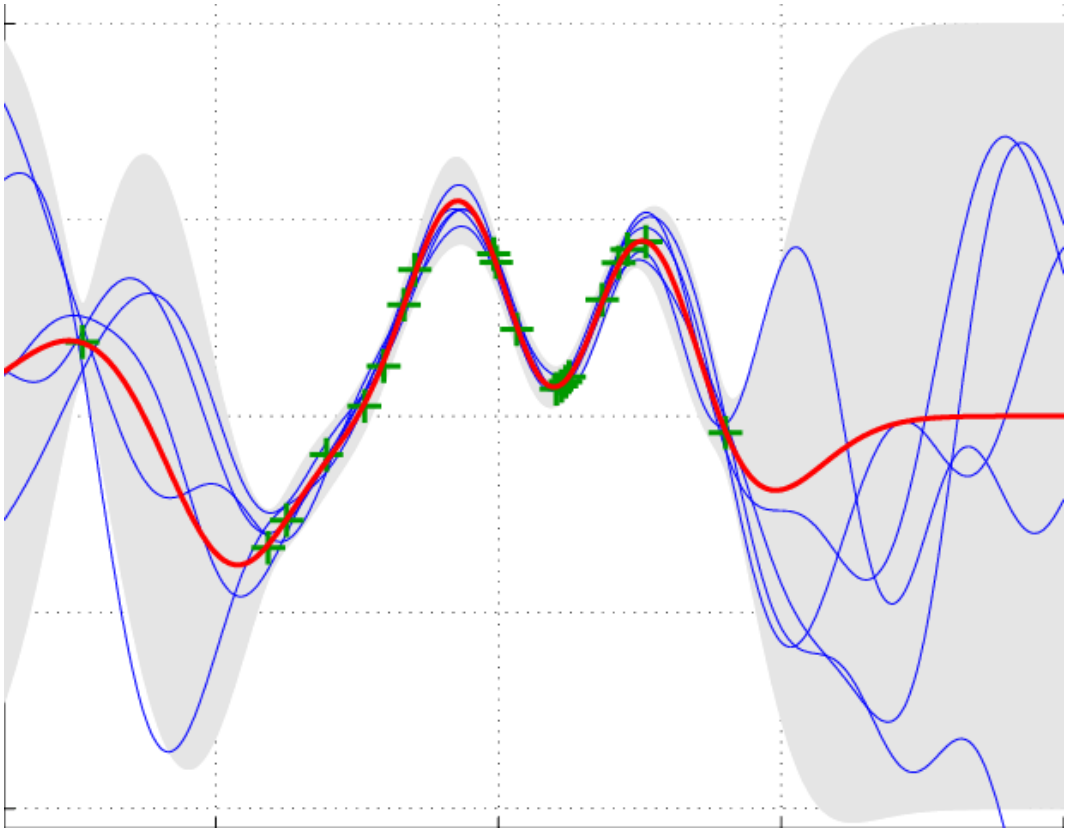
4.8. BayesianTrustful

Agent BayesianTrustful stanowi próbę odnalezienia nowych wag funkcji heurystycznych agenta Trustful, które pierwotnie zostały dobrane manualnie. W tym celu użyliśmy bayesowskiej metody optymalizacji hiperparametrów, korzystającej z procesów gaussowskich.

Podstawy teoretyczne

Bayesowska metoda optymalizacji to proces iteracyjny, oparty o twierdzenie Bayesa, który korzysta z wnioskowania statystycznego do analizy prawdopodobieństwa hipotez. Rozważmy przestrzeń hipotez $\mathbb{H}$, której elementom h przyporządkowane są początkowe prawdopodobieństwa $P(h)$, zwane *a priori*. Prawdopodobieństwo hipotezy $h \in \mathbb{H}$ po dokonaniu nowej obserwacji O można wyrazić wzorem $P(h|O)$, nazywanym *a posteriori*. Metoda bayesowska pozwala na aktualizację prawdopodobieństwa *a priori* hipotez w oparciu o nowe obserwacje. To działanie przekłada się na możliwość dokonywania wnioskowania statystycznego na temat rozkładu funkcji, która nie posiada wzoru jawnego.

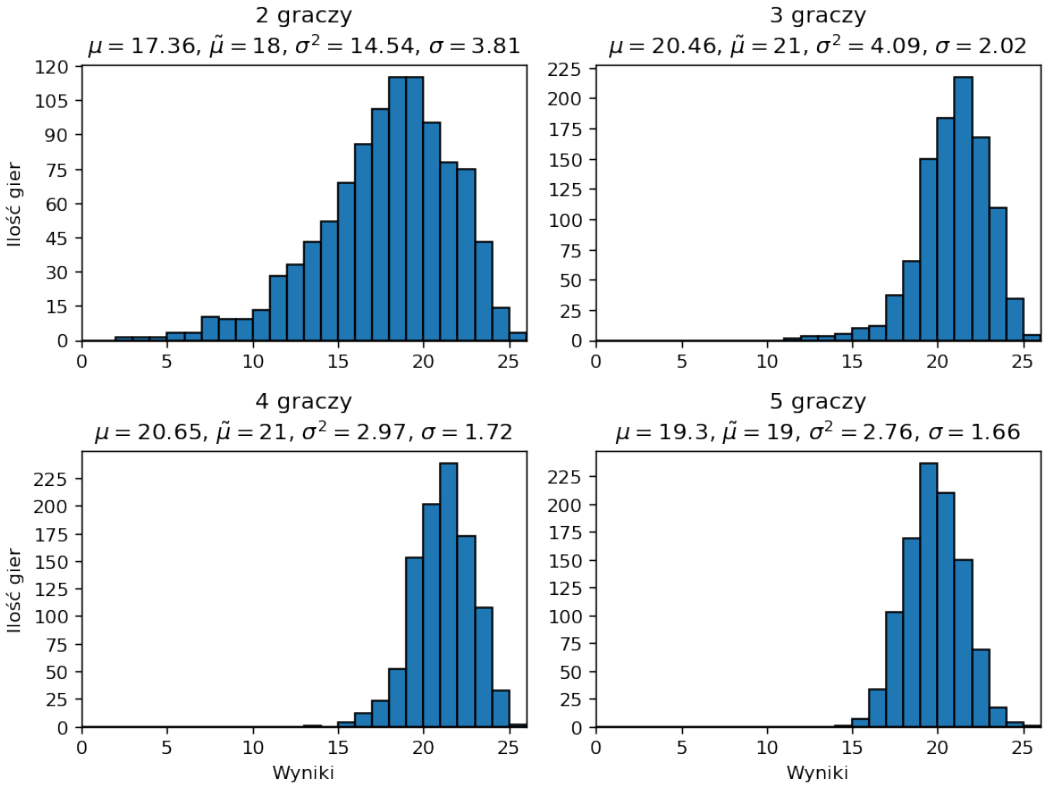
Proces gaussowski to proces stochastyczny, który pozwala na określenie łącznego rozkładu prawdopodobieństwa (oraz jego niepewności) wybranej rodziny nieparametrycznych funkcji $\mathbb{F}$. Polega on na konstrukcji prawdopodobieństw *a priori*, które można aktualizować poprzez kolejne obserwacje, co tworzy system przekonań o kształcie badanych funkcji.



Rysunek 4.8: Wizualizacja procesu gaussowskiego. Zielonymi plusami zaznaczono obserwacje, niebieskie krzywe to poprzednie przekonania *a posteriori*, zaś czerwona krzywa to ich średnia wartość. Zacieniona część wykresu wyznacza niepewność modelu. Żródło: [6]

Bayesowska metoda optymalizacji oparta o procesy gaussowskie polega na stworzeniu rozkładu *a priori* funkcji, której maksimum szukamy, a następnie próbkowaniu jej. Nowe obserwacje pozwalają na aktualizację prawdopodobieństw *a posteriori*. Wraz ze wzrostem ilości obserwacji przekonania algorytmu ulegają zmianie, a ponieważ proces gaussowski pozwala na identyfikację tych części funkcji, które najbardziej wymagają większej ilości obserwacji, metoda ta minimalizuje ilość próbkowań, potrzebnych do dalszych rozważań.

Osiągi



Rysunek 4.9: Wyniki agenta BayesianTrustful (rozmiar próby: 1000 gier)

Maksymalizowana była funkcja o 27 parametrach, odpowiadających wagom używanym w funkcjach heurystycznych. Jej wartość w punkcie została zdefiniowana jako średnia arytmetyczna wyników uzyskanych na przestrzeni 20 niezależnych rozgrywek czteroosobowych.

Algorytm optymalizujący, po zbadaniu kilku tysięcy próbek, znalazł nowy zestaw parametrów. Przeprowadzone testy wskazują na niewielki wzrost średniej ilości punktów dla grup trzech, czterech i pięciu graczy, kosztem zwiększonej wariancji. Ponieważ agent Trustful nie był projektowany do przeprowadzania rozgrywek dwuosobowych, lepsze dostosowanie wag, które pomagają przy bardziej licznych grupach, pogrąża strategię dla dwóch graczy.

Na podstawie analizy wartości nowych wag można dociec, w jaki sposób zmieniła się strategia agenta:

- Moduł "Discard tip", pomagający graczom w odrzucaniu bezużytecznych kart, udziela podpowiedzi wyłącznie w sytuacji, w której jest w stanie częściowo ujawnić jednocześnie co najmniej 4 karty (poprzednia wartość: 2).

- Moduł "Play tip", zarządzający podpowiedziami dotyczącymi zagrań, w znacznie większym stopniu nagradza zagrania lawinowe, udzialanie informacji dalszym graczom oraz podpowiedzi dotyczące kart o niższych numerach, jednocześnie drastycznie zwiększając kary za nieprawidłowe obietnice i marginalizując znaczenie podpowiedzi będących komunikatami.

- Moduł "Risky tip", pozwalający na udzielanie podpowiedzi będących nieprawidłowymi obietnicami, otrzymał karę o wysokości, która całkowicie wyklucza możliwość jego wykorzystania.

- Moduł "Information tip", pozwalający na udzielanie podpowiedzi będących komunikatami, ma zwiększony pułap jakościowy, co sprawia, że jest on znacznie rzadziej wykorzystywany.

Podsumowując: agent jest mniej skłonny do udzielania podpowiedzi będących komunikatami i agresywnie penalizuje nieprawidłowe obietnice. Choć prowadzi to do uzyskania wyższej średniej punktowej, takie postępowanie destabilizuje algorytm i powoduje wzrost wariancji obserwowanych wyników.

4.9. Reinforced

Poprzednicy agenta Reinforced korzystali ze ściśle określonych systemów regułowych. W przeciwieństwie do nich, agent Reinforced samodzielnie dopracowuje obieraną strategię poprzez sukcesywne rozgrywanie kolejnych partii i wyciąganie wniosków na temat wykonanych ruchów.

Na początku procesu uczenia, każdej z możliwych akcji agenta przydzielony jest taki sam priorytet, gdyż program nie wie, który z ruchów może być przydatny w danym momencie rozgrywki. Wiedza o tym, jakie zachowania są najbardziej pożądane, jest nabywana poprzez egzekwowanie systemu kar i nagród, nadawanych w trakcie gry, a także po jej zakończeniu. Na ich podstawie wyciągane są wnioski, które pozwalają na dokonywanie lepszych decyzji w przyszłości - zarówno w skali makro (wybieranie jednej z trzech akcji: zagrania, odrzucenia i podpowiedzi), jak i mikro (określanie szczegółów ruchu).

Schemat działania

Agent posiada cztery funkcje oceniające, które pozwalają na rozstrzyganie, które akcje powinny otrzymać najwyższe priorytety.

1. Rozstrzygnij, która z posiadanych kart powinna zostać zagrana.
2. Rozstrzygnij, która z posiadanych kart powinna zostać odrzucona.
3. Rozstrzygnij, która z możliwych do udzielenia podpowiedzi jest najbardziej przydatna, pomijając te z nich, które nie ujawniają żadnych informacji.
4. Sprawdź, które akcje zostały ocenione najlepiej, a następnie rozstrzygnij, która z nich powinna zostać wykonana.

Funkcje oceniające nie są narzucone odgórnie. Każda z nich wydaje werdykt w zależności od obecnego stanu gry, a także wartości, które zostały wyuczone na podstawie poprzednich decyzji, podjętych w zbliżonych warunkach. Mechanizm ten przypomina działanie algorytmu Q-learning.

Model uczący

Najwyższe osiągi i największą swobodę nauki można osiągnąć poprzez przekazywanie funkcjom oceniającym struktur, które dokładnie odwzorowują obserwowany stan rozgrywki. Dane, które musiałyby być zawarte w takim obiekcie, to między innymi: kolejność kart w rękach graczy, status stosów zagranych kart, listy odrzuconych oraz niedobranych kart, a także historia ruchów poszczególnych graczy. Niestety, ze względu na bardzo dużą ilość unikalnych stanów gry, użycie takiego obiektu jest niepraktyczne, gdyż długość procesu uczenia stałaby się nieakceptowalna. Należałoby rozważyć ponad 10^{12} stanów - zakładając, że potrafimy zbadać w ciągu jednej sekundy około 500 z nich, pojedyncze odwiedzenie każdego elementu drzewa stanów trwałoby ponad 63 lata. Aby zredukować złożoność algorytmu, zdecydowaliśmy się na upraszczanie stanów gry przy użyciu funkcji heurystycznych. Pod uwagę brane są wyłącznie informacje niezbędne do efektywnego rozpoznawania ogólnych sytuacji, w których może znaleźć się agent.

Informacje o stanie gry są przechowywane jako kolekcja wektorów, które zawierają zestaw danych o wybranym obiekcie gry. Informacje są przechowywane w formie liczbowej z ograniczonego zakresu, co pozwala na zredukowanie liczby stanów rozgrywki. Wyróżniamy trzy rodzaje kontenerów o (między innymi) następujących składowych:

1. Pojedyncze karty aktualnego gracza (około 10^5 unikalnych stanów):
    - stopień ujawnienia karty,

- czy istnieje ryzyko, że karta jest krytyczna,
- czy karta jest najstarsza,
- ilość kart, które zostały ujawnione w tym samym momencie, co rozpatrywana kartą,
- ilość stosów, których najwyższy numer jest niższy od numeru rozpatrywanej karty,
- ilość pozostałych żetonów podpowiedzi.

2. Możliwe do udzielenia podpowiedzi (około $4 * 10^5$ unikalnych stanów):
   - ilość ujawnianych kart, które można bezpiecznie zagrać,
   - ilość ujawnianych kart, które będzie można w przyszłości bezpiecznie zagrać,
   - ilość ujawnianych kart, które można inferowalnie bezpiecznie odrzucić,
   - ilość ujawnianych kart, które zostaną w pełni ujawnione,
   - czy podpowiedź obejmuje kartę, która może rozpocząć zagranie lawinowe,
   - czy podpowiedź obejmuje najstarszą kartę w ręce gracza,
   - czy podpowiedź obejmuje jakąkolwiek kartę krytyczną,
   - odległość gracza udzielającego podpowiedzi od gracza, któremu podpowiedź jest udzielana.
3. Możliwe do wykonania akcje (około 10^5 unikalnych stanów):
   - ilość pozostałych żetonów podpowiedzi,
   - ilość pozostałych żetonów życia,
   - czy talia jest pusta,
   - oceny najlepszych akcji z poszczególnych kategorii.

Składowe pierwszych dwóch kontenerów są od siebie całkowicie niezależne. Ponieważ wartości ostatniego kontenera zależą od werdyktu funkcji oceniających ruchy, dwa identyczne stany rozgrywki mogą zostać ocenione na różne sposoby. Powoduje to, że kontener akcji dostarcza agentowi najmniej pewnych informacji.

Funkcja oceniająca to mapowanie, które przyjmuje odpowiednie kontenery, a następnie zwraca numeryczną ocenę rozpatrywanego ruchu. Agent dysponuje czterema funkcjami oceniającymi: po jednej na każdą z akcji (zagraj kartę, odrzuć kartę, udziel podpowiedzi innemu uczestnikowi rozgrywki), a także ostatnią, która dokonuje ostatecznej decyzji o ruchu.

Początkowo, wartości liczbowe zwracane przez funkcje oceniające zawierają się w przedziale $[0, 1]$. Aby zachować balans pomiędzy potrzebą eksploracji nowych stanów i wysokimi osiągami, ostatecznie wyrażane są one za pomocą wzoru

$S + c\sqrt{\frac{\ln(N+1)}{p}}$, gdzie S to początkowa wartość funkcji oceniającej, c to parametr eksploracji (ustalony na $\sqrt{2}$), N to ilość odwiedzin aktualnego stanu gry, z kolei p to ilość kar, który otrzymał rozważany stan gry. Wzór ten jest odmianą formuły UCT ("Upper Confidence Bound 1 applied to trees"), używanej m.in. w algorytmie Monte Carlo Tree Search[7].

Po dokonaniu analizy heurystycznej ruchów, wykonywana jest akcja, która otrzymała najwyższą notę. Aby wymusić dodatkowe losowe błądzenia, agent ma stałą szansę na wykonanie arbitralnego ruchu, która wynosi 1%. Funkcjonalność ta jest aktywna wyłącznie wtedy, gdy program znajduje się w trybie nauki.

Na koniec każdej z tur, agent rozważa efekty wybranej akcji. Jeżeli program wykonał poprawne zagranie lub odrzucił bezużyteczną kartę, funkcje oceniające ten ruch są nagradzane. Bonus otrzymują także funkcje, które zdecydowały o udzieleniu podpowiedzi dotyczącej danej karty. Podobnie, jeżeli zagranie było niepoprawne lub została odrzucona karta krytyczna (bądź taka, którą można było zagrać), funkcje są karane. Z uwagi na trudność natychmiastowego oszacowania poprawności akcji udzielenia podpowiedzi, te ruchy są oceniane dopiero po zakończeniu rozgrywki.

W zależności od tego, czy po ukończeniu partii agent uzyskał wynik wyższy od mediany punktów z ostatniego tysiąca rozgrywek, wszystkie akcje dokonane w trakcie gry są nagradzane lub karane. Zmiana jest wprost proporcjonalna do różnicy pomiędzy uzyskanym wynikiem a medianą. Funkcja oceniająca, która dokonuje wyboru głównej akcji, korzysta z tego mechanizmu w największym stopniu.

Ponieważ funkcje oceniające są ze sobą silnie skorelowane, ocena stanu rozgrywki może diametralnie się zmieniać w zależności od aktualnie przyjętej strategii. Z tego powodu program pamięta dla każdego ze stanów jedynie 400 ostatnich kar i nagród, a funkcje nadają wyższe wagi tym z nich, które są najnowsze.

Opisany model uczący był tworzony empirycznie, za pomocą metody prób i błędów, a jego forma w znacznej mierze wynika z naszego zrozumienia gry Hanabi.

Osiągi

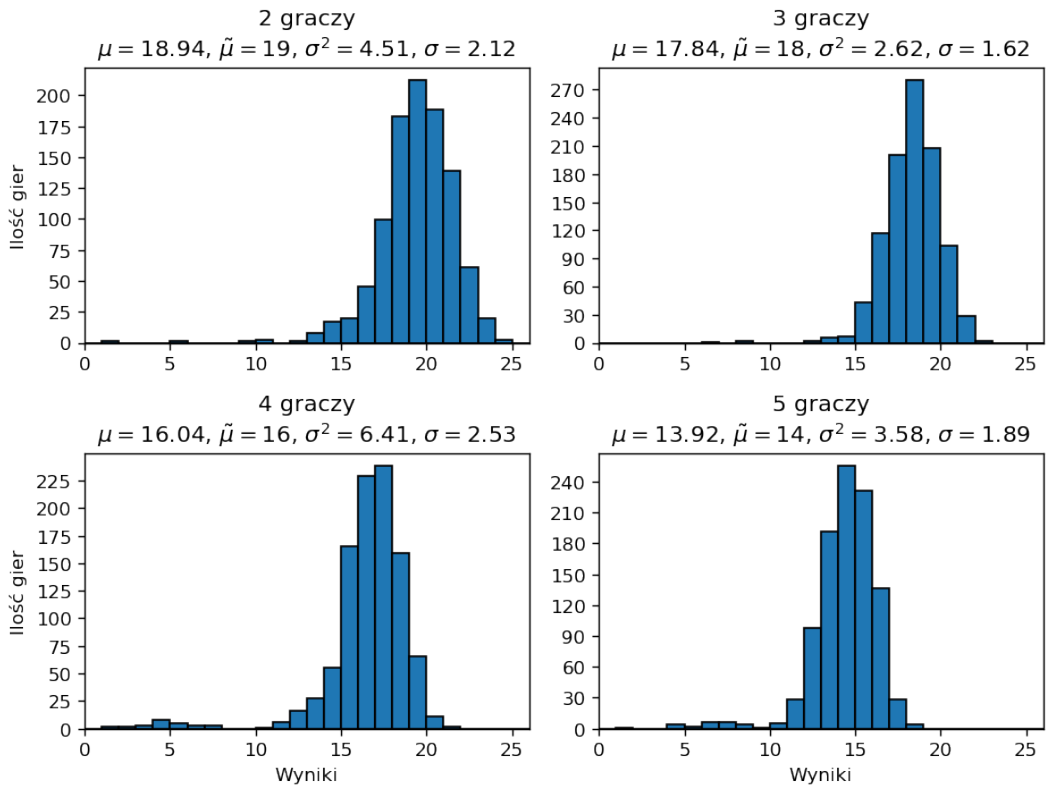
Aby unaocznić efektywność opisanych metod uczących, stworzyliśmy dwa skrajnie różne środowiska, w których agent wypracowywał swoje strategie. Agenci nazwani Reinforced2 rozgrywali partie dwuosobowe, zaś agenci Reinforced4 - czteroosobowe. W obu przypadkach kolejne partie były rozgrywane do momentu, gdy osiągi agentów przestały wykazywać się szczególnymi zmianami. Cały proces, w trakcie którego agenci sumarycznie rozegrali blisko pół miliona gier, zajął około dwunastu godzin.

Podjęliśmy nieudaną próbę utworzenia trzeciego środowiska, w którym pojedynczy agent Reinforced uczył się gry z czterema agentami Trustful. Ostatecznie nie doszło do jego realizacji, gdyż włączenie do rozgrywki stosunkowo powolnych

agentów Trustful spowalniało proces nauki prawie dziesięciokrotnie.

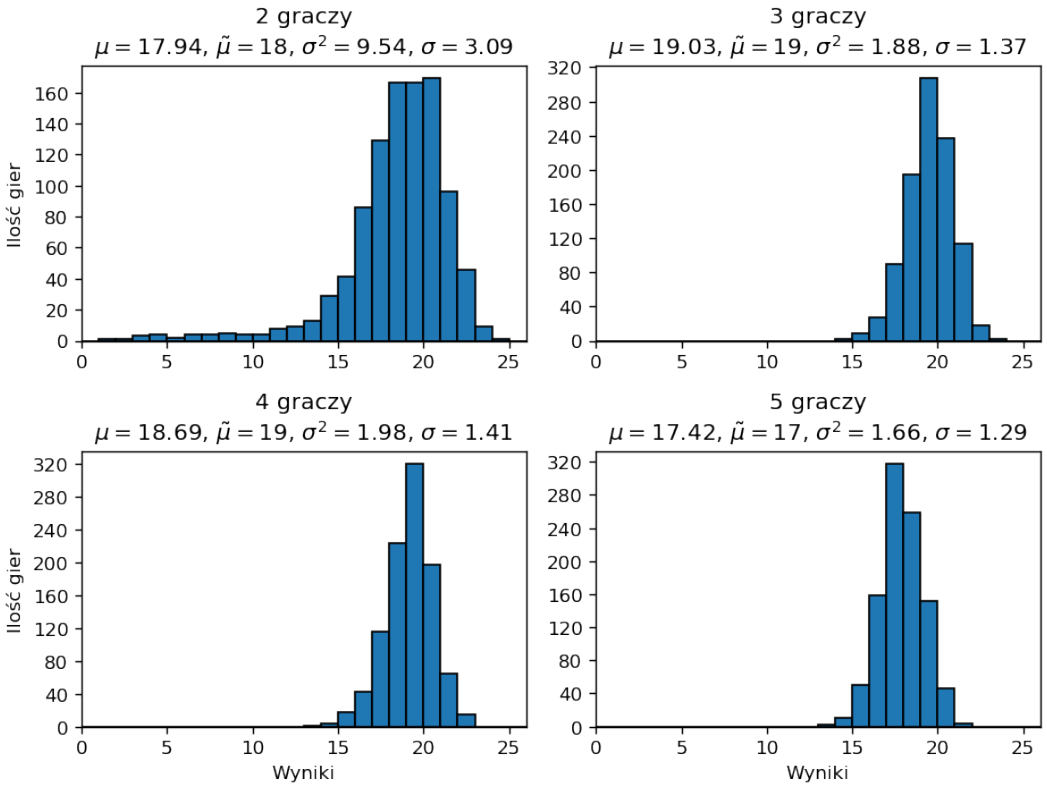
Strategia wypracowana przez agentów jest wyjątkowo trudna do zrozumienia, gdyż proces uczenia doprowadził do powstania ezoterycznego protokołu komunikacji. Manualna analiza kilku rozgrywek pozwoliła jednak na wyszczególnienie kilku powtarzających się zachowań:

- Agenci preferują udzielanie obiektywnie wartościowych podpowiedzi ponad zagrywanie własnych kart, nawet, jeśli są one bezpieczne, o ile są w posiadaniu wystarczająco wysokiej ilości żetonów podpowiedzi. Jest to szczególnie widoczne w zachowaniu agenta Reinforced2.
- Udzielenie informacji o pojedynczej karcie jest najczęściej odbierane przez pozostałych agentów jako sygnał do jej zagrania.
- Agenci często udzielają podpowiedzi, które ujawniają wiele kart jednocześnie. Pozostali uczestnicy rozgrywki nie uznają takiej podpowiedzi za jakikolwiek sygnał, w zamian oczekując na nowe informacje. Ponownie, jest to technika wykorzystywana głównie przez agenta Reinforced2.
- Gdy agent nie posiada w ręce kart, które można inferowalnie bezpiecznie odrzucić, najczęściej odrzuca nie najstarszą, a najmłodszą kartę.



Rysunek 4.10: Wyniki agenta Reinforced2 (rozmiar próby: 1000 gier)

Agent Reinforced2, który trenował granie partii dwuosobowych, osiąga w nich najwyższe wyniki spośród wszystkich przedstawionych agentów, którzy nie oszukują. Jednocześnie, mimo chaotycznego procesu formułowania strategii, zachowuje on zaskakująco wysoką stabilność. Niestety, jak można było przewidywać, wraz ze wzrostem liczby graczy agent uzyskuje coraz niższe wyniki. Ponadto, nigdy nie udaje mu się zdobyć maksymalnej ilości 25 punktów.



Rysunek 4.11: Wyniki agenta Reinforced4 (rozmiar próby: 1000 gier)

W przypadku agenta Reinforced4, wypracowana strategia jest znacznie mniej podatna na zmiany w liczbie graczy. Choć agent nie osiąga w partiach dwuosobowych wyników, które pozwoliłyby mu konkurować z bliźniaczym algorytmem, pokonuje w nich pozostałych agentów, którzy nie oszukują. W pozostałych testach sytuuje się on pomiędzy agentami Distrustful a Trustful, jednocześnie wyróżniając się bardzo wysoką stabilnością wdrażanych strategii.

Rozdział 5.

Niestandardowe modele rozgrywki

5.1. Dwa oblicza drużyn Hanabi

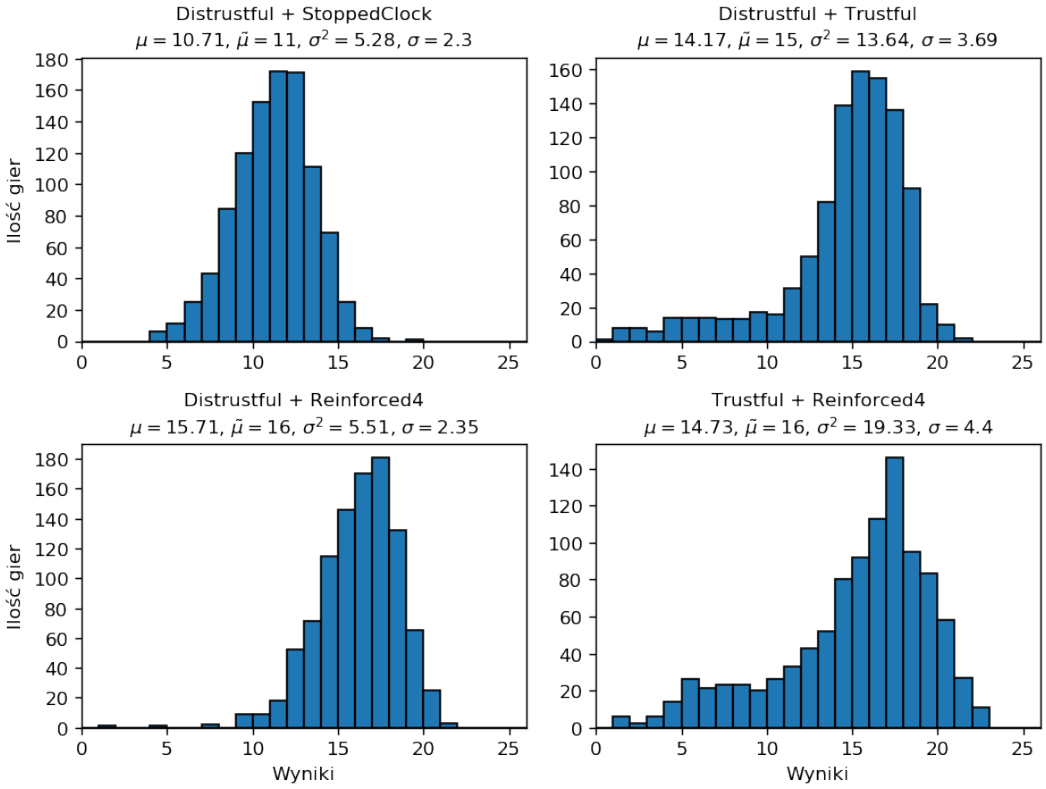
Prezentowanie możliwości dotychczas przedstawionych agentów polegało na pomyśle umieszczania ich w homogenicznych środowiskach, na które składało się wiele kopii jednego agenta. Gwarantowało to, że wszyscy uczestnicy rozgrywki posiadali jednolitą strategię. W naturalny sposób nasuwa się pytanie: co może się wydarzyć, kiedy do partii zasiądą gracze o różnych schematach działania?

Rozgrywki mieszane, znane także jako potyczki XY, stanowią ciekawy problem w kontekście sztucznej inteligencji grającej w Hanabi, gdyż wymagają pogodzenia dwóch antytecznych konceptów: konieczności współpracy oraz braku wspólnego planu działania. Niepewność poczynań innych graczy zmienia problematykę całej gry: implicytna komunikacja staje się niemożliwa, a gracze w każdej chwili mogą wykonać akcje, które nie mają sensu z punktu widzenia współuczestników rozgrywki, co skutecznie paraliżuje większość strategii. Dobrym sposobem na uzyskanie wysokich wyników nie jest już kooperacja, a maksymalne ograniczenie zaufania do innych graczy. W przypadku agentów sztucznej inteligencji, którzy nie potrafią dostosowywać się do zmieniających warunków tak, jak robią to ludzie, zredefiniowanie formuły rozgrywki może prowadzić do zaskakujących rezultatów.

5.2. Potyczki XY

Przeprowadziliśmy testy rozgrywek, w których gracze występowali w różnych konfiguracjach. Ponieważ istnieją dziesiątki możliwych ustawień, wybraliśmy z nich te, które uznaliśmy za najciekawsze. Partie na pięciu graczy niewiele różnią się od tych na czterech graczy, toteż zostały one pominięte.

Partie na dwóch graczy



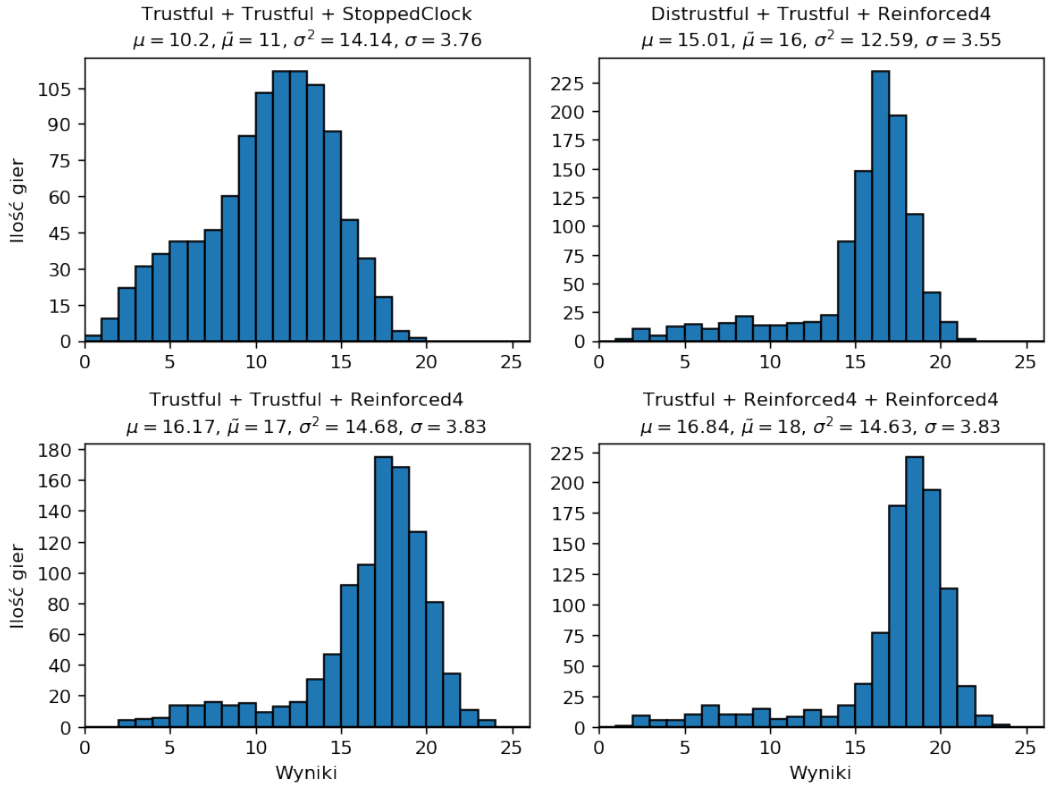
Rysunek 5.1: Wyniki potyczek XY dla dwóch graczy (rozmiar próby: 1000 gier)

Agenci Distrustful oraz StoppedClock używają strategii, które dobrze ze sobą współgrają. Zagrywane są wyłącznie karty bezpieczne, a pomiędzy graczami nie istnieje żaden protokół komunikacji, który mógłby zostać przypadkowo złamany. Ich połączenie skutkuje dosyć stabilnymi wynikami, mimo przypadkowości akcji dokonywanych przez agenta StoppedClock.

Jak można było przewidywać, protokół komunikacji agenta Trustful źle współgra z graczami, którzy go nie stosują: agent Distrustful udziela podpowiedzi, które są nadinterpretowywane przez Trustful, co prowadzi do niepoprawnych zagrań. Problem potęguje się, gdy drugim uczestnikiem rozgrywki jest agent Reinforced4 - posługuje się on nietypową strategią, co czasami prowadzi do zaskakująco wysokich wyników, lecz w większości przypadków kończy się dezorientacją agenta Trustful i szybką przegraną.

Połączenie agentów Distrustful oraz Reinforced4 daje ciekawy rezultat: mimo różnych strategii, średnia osiąganych wyników oscyluje blisko osiągów agenta Distrustful. Zachowanie Reinforced4, z powodu braku innych graczy, którzy rozumieliby implicytne informacje, zdegenerowało się do poziomu bliskiego agentowi Distrustful.

Partie na trzech graczy



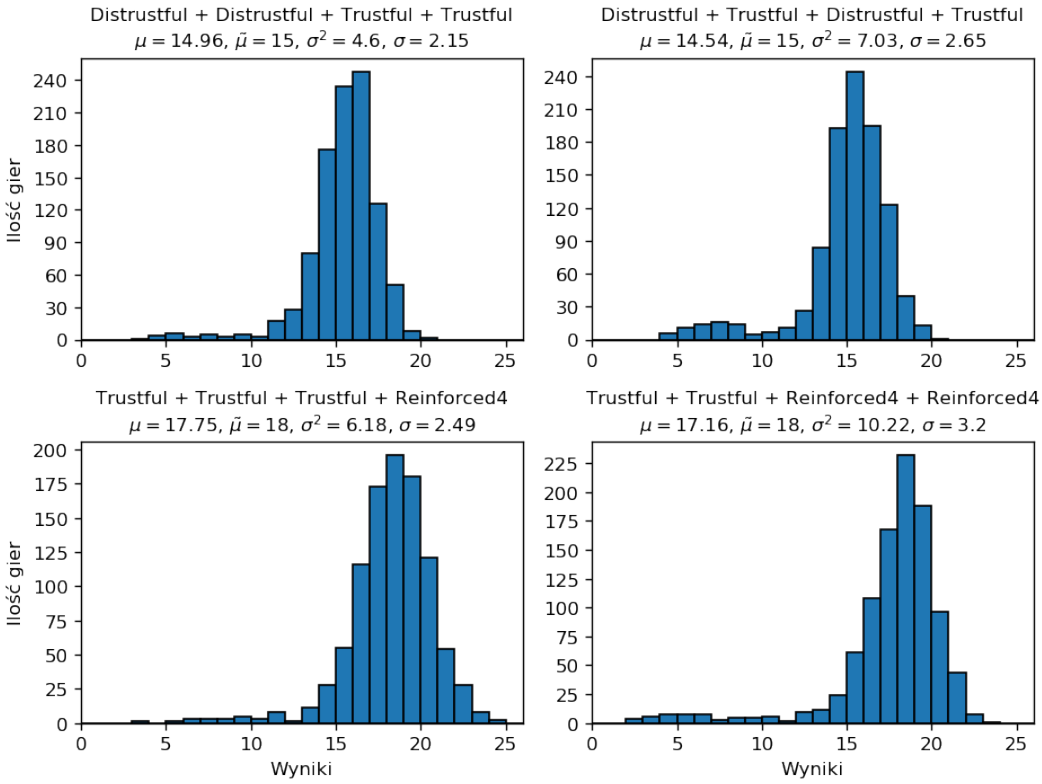
Rysunek 5.2: Wyniki potyczek XY dla trzech graczy (rozmiar próby: 1000 gier)

Nawet, gdy w grze z agentami Trustful znajduje się tylko jeden gracz, który łamie zasady protokołu komunikacji, losowe podpowiedzi prowadzą do porażającej ilości szybkich porażek. Po raz kolejny potwierdza się fakt, że agent Trustful nie sprawdza się w potyczkach XY: jego kooperacyjna strategia prowadzi do powstawania błędnych przekonań, których inni agenci nie są w stanie na czas skorygować, by zapobiec nieszczęściu.

Podobna, chociaż mniej ekstremalna sytuacja ma miejsce, gdy dwóch agentów Trustful próbuje współpracować z agentem Reinforced4. Rozumie on część protokołu komunikacji, gdyż stosuje technikę podpowiadania pojedynczych kart, lecz nie ma pojęcia o systemie obietnic. Często doprowadza to do wetowania kolejnych kart i zmusza drugiego agenta Trustful do udzielania dodatkowych informacji. Rozgrywki, w których uczestniczy dwóch agentów Reinforced4 oraz jeden agent Trustful to przypadek symetryczny, gdyż tym razem to Trustful rozumie tylko część protokołu komunikacji.

Wymieszanie trzech różnych agentów okazuje się nie być tragiczne w skutkach: w porównaniu do rozgrywki na dwóch graczy, w której uczestniczyli wyłącznie Trustful oraz Reinforced4, dodatkowy gracz w postaci agenta Distrustful pełni rolę arbitra, który balansuje niedobory w przekazywanych informacjach.

Partie na czterech graczy



Rysunek 5.3: Wyniki potyczek XY dla czterech graczy (rozmiar próby: 1000 gier)

Okazuje się, że nie tylko dobór agentów, ale i ich kolejność może mieć znaczenie dla przebiegu gry. Ustawienie agentów Trustful obok siebie skutkuje możliwością korekty obietnic, które zostały nieświadomie wystosowane przez pozostałych agentów uczestniczących w rozgrywce. Ułożenie agentów naprzemiennie nie pozwala na zastosowanie tej części protokołu komunikacji, co objawia się niższymi wynikami, a także większą niestabilnością uzyskiwanych wyników.

Wraz ze wzrostem liczby graczy, różnice pomiędzy protokołami komunikacji są bardziej oczywiste - nawet, gdy mają one pewne cechy wspólne. Zastąpienie jednego z agentów Trustful przez Reinforced4 powoduje znaczne zwiększenie wariancji ilości uzyskiwanych punktów. Fenomen ten nie występował w partiach na trzech graczy, gdzie wymiana Trustful-Reinforced4 skutkowała mało istotnymi zmianami.

Podsumowanie

Agenci, którzy wykorzystują wysublimowane strategie, są w znacznie większym stopniu podatni na niekorzystne sytuacje, wynikające ze specyfiki mieszanych rozgrywek. W potyczkach XY największą stabilność uzyskują strategie bazowane na wnioskowaniu, które nie polegają na sygnałach od innych graczy.

Rozdział 6.

Wnioski

Problem gry w Hanabi to złożone zagadnienie, które nie bez powodu rozbudza ciekawość zarówno pasjonatów gier planszowych, jak i matematyków czy informatyków. Jego trudność można określić mianem interdyscyplinarnej: w zależności od sposobu postawienia problemu, techniki jego analizy oraz rozwiązania diametralnie się różnią. Stworzenie agentów sztucznej inteligencji, którzy potrafiliby grać w Hanabi na poziomie mistrzowskim, jest wyjątkowo skomplikowanym zadaniem, wymagającym zastosowania niestandardowych rozwiązań.

Zaprezentowani agenci sztucznej inteligencji nie osiągnęli wyników, które mogłyby konkurować z programami wykorzystującymi złożone obliczeniowo techniki. Stanowią oni jednak dowód możliwości tworzenia szybkich i skutecznych rozwiązań zawiłych problemów algorytmicznych, korzystających nie z kolosalnej mocy współczesnych komputerów, lecz dobrej znajomości zadania, z którym dane jest się nam mierzyć.

Tworzenie agentów sztuczej inteligencji grających w Hanabi to zagadnienie, które unaocznia potęgę ludzkiej intuicji. Algorytmy regułowe, choć w znacznej mierze zdeprecjonowane i zastąpione przez nowe techniki tworzenia sztucznej inteligencji, przypominają, że pewne problemy można rozwiązywać nie tylko z pomocą wyspecjalizowanych narzędzi, ale i nietuzinkowych pomysłów.

Rozdział 7.

Dalsze badania

Choć agenci Reinforced2 oraz Reinforced4 osiągają zadowalające wyniki, mogą one zostać usprawnione poprzez rozszerzenie lub modyfikację uproszczonych stanów gry. Trudność tego zadania jest bezpośrednio powiązana z problemem wykładniczego przyrostu długości procesu nauki, który postępuje wraz z dodawaniem kolejnych składowych obiektów stanów.

W pracy nie zbadano możliwości stworzenia agenta, który dostosowywałby swój sposób postępowania w zależności od akcji, których dokonywaliby inni uczestnicy rozgrywki. Algorytm, który potrafiłby dynamicznie dostosowywać swoją strategię, mógłby osiągać bardzo wysokie wyniki zarówno w potyczkach XY, jak i grach, w których uczestniczy człowiek.

Protokół komunikacji agenta Trustful został oparty o nasze własne spostrzeżenia, zdobyte podczas obserwacji rozgrywek ludzkich graczy. Nie wykluczamy istnienia bardziej zaawansowanych strategii (przykładowo: opartych o obserwacje zachowań profesjonalnych graczy Hanabi), które pozwoliłyby na uzyskanie wyższych wyników.

Dodatek A

Korzystanie z projektu

A.1. Instalacja zależności

Aby skorzystać ze wszystkich możliwości programu, potrzebne będzie zainstalowanie środowiska uruchomieniowego Python 3. Projekt był testowany przy użyciu wersji 3.7.4 na następujących systemach operacyjnych:

- Windows 10, kompilacja 17763
- MacOS Catalina 10.15.2
- Ubuntu 18.04.3 LTS

A.1.1. Windows

Należy otworzyć powłokę CMD lub PowerShell, przejść do głównego katalogu projektu, a następnie wykonać następującą komendę:

```
$ python3 -m pip install -r requirements-windows.txt
```

A.1.2. MacOS

Należy otworzyć jedną z powłok systemu Unix (przykładowo: bash, zsh), przejść do głównego katalogu projektu, a następnie wykonać następującą komendę:

```
$ python3 -m pip install -r requirements-macos.txt
```

A.1.3. Ubuntu Linux

Należy otworzyć jedną z powłok systemu Unix (przykładowo: bash, zsh), a następnie wykonać następujące komendy:

```
$ sudo add-apt-repository ppa:kivy-team/kivy
$ sudo apt update
$ sudo apt install python3-pip python3-tk python3-kivy python3-sdl2
$ python3 -m pip install bayesian-optimization
```

A.1.4. Inne systemy operacyjne

Należy zainstalować biblioteki *bayesian-optimization* oraz *Kivy*[1]. Instalacji pierwszej z nich można dokonać poprzez wykonanie w powłoce systemowej następującej komendy:

```
$ python3 -m pip install bayesian-optimization
```

A.2. Uruchamianie projektu

Aby uruchomić graficzny interfejs użytkownika, należy otworzyć powłokę systemu operacyjnego, przejść do głównego katalogu projektu, a następnie wykonać następującą komendę:

```
$ python3 hanabi.py
```

Spowoduje to pojawienie się menu wyboru agentów, które pozwala na rozpoczęcie nowej rozgrywki w Hanabi.

A.3. Dodatkowe funkcje

Główny katalog projektu zawiera programy, które pozwalają na kontynuowanie jego rozwoju, a także powtórzenie przeprowadzonych eksperymentów:

- Aby rozpocząć proces uczenia własnego agenta Reinforced, wystarczy uruchomić plik o nazwie *learning_example.py*.
- Program odpowiedzialny za optymalizację hiperparametrów metodą bayesowską zawarty jest w pliku *bayesian_optimization.py*.
- Wyniki eksperymentów dokonywanych na agentach można znaleźć w notatniku *benchmark.ipynb*. Do otworzenia pliku potrzebne będzie zainstalowanie programu Jupyter Notebook[2].

[1] Instrukcja instalacji biblioteki Kivy: **link** (term. wiz. 30.01.2020)
[2] Instrukcja instalacji programu Jupyter Notebook: **link** (term. wiz. 30.01.2020)

Bibliografia

[1] M. Campbell, A. J. Hoane Jr., F. Hsu, *Deep Blue*, 2002. URL: **link** (term. wiz. 30.01.2020)

[2] J.-F Baffier i in., *Hanabi is NP-complete, Even for Cheaters who Look at Their Cards*, 2017. URL: **link** (term. wiz. 30.01.2020)

[3] A. Lerer, H. Hu, J. Foerster, N. Brown, *Building AI that can master complex cooperative games with hidden information*, 2019. URL: **link** (term. wiz. 30.01.2020)

[4] T. Murphy, *The First Level of Super Mario Bros. is Easy with Lexicographic Orderings and Time Travel . . . after that it gets a little tricky.*, 2013. URL: **link** (term. wiz. 30.01.2020)

[5] B. Bouzy, *Playing Hanabi Near-Optimally*, 2017. URL: **link** (term. wiz. 30.01.2020)

[6] Fernando Pérez-Cruz i in., *Gaussian Processes for Nonlinear Signal Processing: An Overview of Recent Advances*, 2013. URL: **link** (term. wiz. 30.01.2020)

[7] L. Kocsis, C. Szepesvári, *Bandit Based Monte-Carlo Planning*, 2006. URL: **link** (term. wiz. 30.01.2020)

[8] N. Bard i in., *The Hanabi Challenge: A New Frontier for AI Research*, 2019. URL: **link** (term. wiz. 30.01.2020)