Politechnika Wrocławska

Wydział Matematyki
Kierunek studiów: Matematyka stosowana
Specjalność: –

Praca dyplomowa – inżynierska

ZASTOSOWANIE STOCHASTYCZNEJ OPTYMALIZACJI DO GIER CZĘŚCIOWO OBSERWOWALNYCH

Aleksander Jakóbczyk

słowa kluczowe:
tutaj podajemy najważniejsze słowa kluczowe (łącznie nie powinny być dłuższe niż 150 znaków).

krótkie streszczenie:
Celem rozprawy jest wyznaczenie nieoczekiwanych strategii w grach częściowo obserwowalnych za pomocą metod stochastycznej optymalizacji. Praca będzie opierać się na wynikach Cauwet i Teytauda z 2018 roku, w których przedstawili nieoczekiwane strategie dla kilku klasycznych gier oraz kilku metod optymalizacji. Podjęta zostanie próba odtworzenia oraz rozszerzenia wyników na kolejną grę. Przeprowadzona zostanie analiza porównawcza dla różnych metod optymalizacji.

Opiekun pracy dyplomowej	dr inż. Andrzej Giniewicz	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław, rok 2022

Wrocław University
of Science and Technology

Faculty of Pure and Applied Mathematics
Field of study: Mathematics
Specialty: Theoretical Mathematics

Engineering Thesis

TYTUŁ PRACY DYPLOMOWEJ W JĘZYKU ANGIELSKIM

Aleksander Jakóbczyk

keywords:
tutaj podajemy najważniejsze słowa kluczowe w języku angielskim (łącznie nie powinny być dłuższe niż 150 znaków)

short summary:
Tutaj piszemy krótkie streszczenie pracy w języku angielskim (nie powinno być dłuższe niż 530 znaków).

Supervisor	dr inż. Andrzej Giniewicz	.....	.....
	Title/degree/name and surname	grade	signature

*For the purposes of archival thesis qualified to:**

a) category A (perpetual files)

b) category BE 50 (subject to expertise after 50 years)

** delete as appropriate*

stamp of the faculty

Wrocław, 2022

Spis treści

Wstęp

Rozdział 1

Definicje, lematy, twierdzenia, przykłady i wnioski

Celem algorytmów, które będziemy wykorzystywać, jest znalezienie optymalnej strategii dla graczy w dwuosobowych grach częściowo obserwowalnych. Aby matematycznie opisać pojęcia gry i strategii optymalnej, wprowadzimy kilka podstawowych definicji. Definicje dotyczące podstaw teorii gier pochodzą z prac [5],[6].

1.1 Czym jest gra?

Omówmy zatem co nazywamy gra w matematyce.

Gra w matematycznej teorii gie to sytuacja, w której gracze wybierają swoje strategie i otrzymują nagrody lub kary w zależności od podjętych decyzji oraz losowych zdarzeń. Teoria gier pozwala na modelowanie i analizowanie takich sytuacji oraz umożliwia znajdowanie rozwiązań, które są optymalne dla graczy.

Matematyczna definicja gry w teorii gier zazwyczaj zaczyna się od określenia następujących elementów:

- Zbioru graczy: Zbiór graczy to zbiór wszystkich graczy biorących udział w grze. Może to być zbiór składający się z dwóch lub więcej elementów, w zależności od typu gry.
- Zbioru strategii: Zbiór strategii to zbiór wszystkich możliwych strategii, które mogą być wybierane przez graczy. Strategia to plan działania gracza, który zakłada, jakie ruchy gracz zamierza podjąć w danej grze.
- Funkcji zwrotu: Funkcja zwrotu określa nagrody lub kary, które gracze otrzymują w zależności od ich strategii i losowych zdarzeń.
- Struktury informacji: Struktura informacji określa, jakie informacje są dostępne dla graczy w momencie podejmowania decyzji. Informacje mogą być pełne lub częściowe, co wpływa na możliwe strategie graczy i ich skuteczność.

Gry mogą być podzielone na kilka kategorii w zależności od kilku różnych kryteriów. Jednym z takich kryteriów jest moment, w którym gracze podejmują decyzje:

Definicja 1.1 (Gra w postaci strategicznej)**.** Jest to typ gry, w której gracze podejmują decyzje w tym samym momencie.

Definicja 1.2 (Gra w postaci ekstensywnej)**.** Jest to typ gry, w której gracze podejmują decyzje we wcześniej ustalonej kolejności.

Przykładami gier w postaci strategicznej są gry kamień papier nożyce, oszust czy też mora. Natomiast przykładami gier w postaci ekstensywnej są szachy, warcaby oraz go.

Gry możemy również dzielić ze względu na posiadaną wiedzę.

Definicja 1.3 (Gra z kompletną informacją)**.** Jest to typ gry, w której gracze mają informacje o możliwych przyszłych wynikach gry i o zbiorach możliwych strategii.

Definicja 1.4 (Gra częściowo obserwowalna)**.** Jest to przeciwnie gier z kompletną informacją.

Przykładami gier z kompletną informacją są szachy, warcaby oraz go. Natomiast przykładami gier częściowo obserwowanymi są wszelkie gry posiadające w rozgrywce pewien elementy losowe takie jak rzut kostką czy też dobieranie kart.

Istnieje jeszcze wiele innych podziałów gier ze wglądu na kategorie takie jak liczba graczy,zbiory dostępnych akcji, możliwość tworzenia koalicji i wiele innych.

1.2 Definicje i Oznaczenia

1.2.1 Strategie proste

Wprowadźmy podstawowe oznaczenia potrzebne nam do tego, aby móc zdefiniować czym jest strategia optymalna.

- $N = \{1, 2, \ldots, n\}$ – zbiór graczy,
- $A_i,\ i \in N$ – niepusty zbiór strategii czystych gracza i,
- $m_i = |A_i|$ – liczba strategi gracza i,
- $A = \prod_{i \in N} A_i$ – zbiór wszystkich strategii gry,
- $u_i : A \to \mathbb{R}$ – funkcja wypłaty gracza i,
- $a = (a_1, a_2, \ldots, a_n) = (a_i)_{i \in N},\ a_i \in A_i$ – profil gry w strategiach czystych,
- $u_i(a) = u_i(a, a_{-i})$ – wypłata gracza i z profilu a,
- $a_{-i} = (a_i)_{i \in N \setminus \{i\}}$, – profil wszystkich strategii poza strategią graca i.

Definicja 1.5 (Gra strategiczna)**.** Grą strategiczną nazywamy trójkę $GS = \langle N, (A_i)_{i \in N}, (u_i)_{i \in N} \rangle$.

Definicja 1.6 (Równowaga Nasha w strategiach czystych gry strategicznej)**.** Równowaga Nasha w strategiach czystych gry strategicznej jest to profil gry $a^* = (a_1^*, a_2^*, \ldots, a_N^*) \in A$, takim, że

$$\forall\, i \in N\ \forall\, a_i \in A_i \quad u_i(a_i^*, a_{-i}^*) \geqslant u_i(a_i, a_{-i}^*).$$

Zatem jest to profil gry, w którym istnieje strategia czysta dająca nie gorsze wyniki od dowolnej innej strategii czystej. Okazuje się jednak, że taki stan nie zawsze istnieje w strategiach czystych, np. w grze kamień papier nożyce strategia grania tylko kamienia daje gorsze rezultat przeciwko graniu tylko papieru. Podobnie ze strategią grania tylko nożyc i grania tylko papieru.

1.2.2 Strategie mieszane

Definicja 1.7 (Strategia mieszana)**.** Strategia mieszana σ_i graca i w grze strategicznej $GS = \langle N, (A_i)_{i\in N}, (u_i)_{i\in N}\rangle$. Nazywamy rozkład prawdopodobieństwa na zbiorze strategi czystych A_i:

$$\sigma_i = (\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{im_i})$$

gdzie σ_{ik} oznacza prawdopodobieństwo, że gracz i zagra strategie czysta $k \in A_i$.

Fakt 1.8. *Strategia czysta jest szczególnym przypadkiem strategij mieszanej, w którym prawdopodobieństwo zagrania jednej z dostępnych strategii wynosi 1.*

Wprowadźmy dodatkowe oznaczenia:

- $\Sigma_i = \left\{\sigma_i : A_i \to [0,1], \sum_{k=1}^{n} \sigma_{ik} = 1, \sigma_{ki} \geqslant 0\right\}$ – zbiór strategii mieszanych gracza i,
- $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ – profil gry,
- $u_i(\sigma) = u_i(\sigma_i, \sigma_{-i})$ – wypłata gracza i z profilu σ,
- $\sigma_{-i} = (\sigma_i)_{i\in N\setminus\{i\}}$, – profil wszystkich strategii poza strategią graca i.

Definicja 1.9 (Równowaga Nasha w strategiach mieszanej gry strategicznej)**.** Profil gry strategicznej σ_i^* jest Równowagą Nasha gdy:

$$\forall\, i \in N \;\; \forall\, \sigma_i \in \Sigma_i \quad u_i(\sigma_i^*, \sigma_{-i}^*) \geqslant u_i(\sigma_i, \sigma_{-i}^*)$$

Równowaga Nasha interpretujemy jako taki profil gry, w którym żaden z graczy nie opłaca się zmieniać swojej strategi, ponieważ nie skutkuje to zwiększeniem swoich zysków.

1.3 Twierdzenia

Algorytmy 1, 2, 3, 4, 5 wykorzystywane w poniższej pracy oparte są o dwa twierdzenia a dokładniej o szczególne przypadki wynikające z nierówności 1.10 i 1.12:

Twierdzenie 1.10 (Nierówność Hoffdinga)**.** *Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem niezależnych zmiennych losowych (i.i.d.) takim, że $a_i \leqslant X_i \leqslant b_i$, wtedy:*

$$S_t = \sum_{i=1}^{t} X_i, \quad c_i = b_i - a_i,$$

$$P(|S_t - \mathbb{E}(S_t)| \geqslant \epsilon) \leqslant 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right)$$

Lemat 1.11. *Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem niezależnych zmiennych losowych (i.i.d.) takim, że $0 \leqslant X_i \leqslant 1$, wtedy:*

$$\overline{X}_t = \frac{S_t}{t}, \quad \mu = \mathbb{E}(X_i), \quad P(|\overline{X}_t - \mu| \leqslant \epsilon) = 1 - \delta,$$

$$\epsilon \leqslant \sqrt{\frac{\ln(2/\delta)}{2t}}$$

Twierdzenie 1.12 (Empiryczna nierówność Bernsteina)**.** *Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem niezależnych zmiennych losowych (i.i.d.) takim, że $a \leqslant X_i \leqslant b$, wtedy:*

$$P(|\overline{X}_t - \mu| \geqslant \epsilon) \leqslant \delta, \quad \overline{\sigma}_t^2 = \frac{1}{t}\sum_{i=i}^{t}(X_i - \overline{X}_t)^2,$$
$$|\overline{X}_t - \mu| \leqslant \overline{\sigma}_t\sqrt{\frac{2\ln(3/\delta)}{t}} + \frac{3R\ln(3/\delta)}{t}$$

Lemat 1.13. *Niech $X_1, X_2, \ldots, X_t$ będzie ciągiem niezależnych zmiennych losowych (i.i.d.) takim, że $0 \leqslant X_i \leqslant 1$, wtedy:*

$$P(|\overline{X}_t - \mu| \leqslant \epsilon) = 1 - \delta,$$
$$\epsilon \leqslant \overline{\sigma}_t\sqrt{\frac{2\ln(3/\delta)}{t}} + \frac{3\ln(3/\delta)}{t}$$

1.4 Problem porównań wielokrotnych

Załóżmy, że z prawdopodobieństwem $1 - \delta$ chcemy ustalić, który z dwóch graczy, p_1 i p_2 jest lepszy. W tym celu będziemy przeprowadzać testy statystyczne, dla których prawdopodobieństwo pomyłki k-tego testu wynosi δ_k. Testy te będą kontynuowane aż do momentu, gdy jeden z graczy zwycięży większość przeważająca liczbę razy.

Po przeprowadzeniu n takich testów:

$$P(\text{Chociaż jeden z } n \text{ testów się pomylił}) \overset{(*)}{\leqslant} \sum_{k=1}^{n}\delta_k \implies$$
$$P(\text{Żaden test się nie pomylił}) \leqslant 1 - \sum_{k=1}^{n}\delta_k,$$

gdzie nierówność oznaczona $(*)$ wynika z faktu, że $P(X + Y) \leqslant P(X) + P(Y)$.

Aby ostateczne prawdopodobieństwo popełnienia błędu było mniejsze niż δ, konieczne jest wprowadzenie odpowiedniej korekty. Możemy zastosować jedna z dwóch poprawek:

1.4.1 Niech n będzie maksymalną liczbą testów jaką pozwalamy wykonać, aby wyznaczyć lepszego gracza. Wtedy $\delta_k = \frac{\delta}{n}$.

1.4.2 Niech δ_k spełnia nierówność $\delta \geqslant \sum_{k=1}^{\infty}\delta_k$. Wtedy niezależnie od ilości przeprowadzonych testów, ostateczne prawdopodobieństwo pomyłki będzie nie większe niż δ.

Rozdział 2

Algorytmy

2.1 Algorytmy wyścigowe

Do algorytmów wykorzystujących korekty 1.4.1 i 1.4.2 są algorytmy wyścigowe (Racing Algorytms) [cytat]. Dwoma najpopularniejszymi typami algorytmów ratingowych są „Hoffding race" oraz „Bernstein race". Oparte są one odpowiednio o Twierdzenie 1.10 i Twierdzenie 1.13 oraz korektę 1.4.2. Mają one jednak pewną wadę, mogą one wymagać bardzo dużej liczby iteracji, a gdy poziom umiejętności porównywanych graczy jest sobie równy (prawdopodobieństwo wygranej wynosi 50%), wtedy z prawdopodobieństwem równym $1-\delta$ algorytm nigdy się nie zatrzyma.

Aby rozwiązać problemy związane z klasycznymi algorytmami wyścigowymi, wprowadzamy tzw. Limited Racing algorytm. Załóżmy zatem dodatkowy warunek, który mówi, że przerywamy działanie algorytmu, gdy empiryczna wartość oczekiwana z prawdopodobieństwem większym bądź równym $1-\delta$ jest znana z dokładnością co do zadanego ϵ. W naszej pracy przyjmiemy $\epsilon = 0.01$ i $\delta = 0.05$.

2.1.1 Algorytm Limited Empirical Bernstein Race (LEBR)

Klasyczny algorytm racingowy opiera się na szeregu ϵ_t, spełniającej warunek, że zdarzenie $\mathcal{E} = \{|\overline{X}_t - \mu| \leqslant \epsilon_t,\ t \in \mathbb{N}^+\}$ występuje z prawdopodobieństwem nie mniejszym niż $1-\delta$. Dodatkowo niech δ_k będzie dodatnim szeregiem spełniającym $\delta \geqslant \sum_{k=1}^{\infty} \delta_k$. Wtedy korzystając z Lematu 1.13

$$\epsilon_t = \overline{\sigma}_t \sqrt{\frac{2\ln(3/\delta_t)}{t}} + \frac{3\ln(3/\delta_t)}{t}.$$

Ponieważ δ_k sumuje się co najmniej do δ, a $(\overline{X}_t - \epsilon_t, \overline{X}_t + \epsilon_t)$ jest przedziałem ufności dla μ o współczynniku ufności $1-\delta_t$ otrzymanym z Lematu 1.13, oznacza to, że zdarzenie $\mathcal{E}$ występuję z prawdopodobieństw nie mniejszym niż $1-\delta$. Identyczny rezultat otrzymujemy stosując ϵ_t oparte o Lemat 1.11 jednak zmienia się wtedy postać ϵ_t. W pracy [1] algorytm opierał się o szereg $\delta_t = \frac{c\delta}{t^2},\ c = \frac{6}{\pi^2}$. Pseudokod algorytmu, przedstawiony jako Algorytm 1, opiera się na rozgrywaniu t gi obliczeniu górnej granicy $\mathrm{UB} = \min_{1\leqslant k\leqslant t}(\overline{X}_k + c_k)$ oraz dolną granice $\mathrm{LB} = \max(0, \max_{1\leqslant k\leqslant t}(\overline{X}_k - c_k))$. Algorytm kończy działanie, gdy różnica między UB a LB jest mniejsza niż 2ϵ. Otrzymane w ten sposób $\overline{X}$ z prawdopodobieństwem nie mniejszym niż $1-\delta$ jest bliskie wartości μ z dokładnością co zadanego ϵ. tu skonczyłem sprawdzanie

Algorithm 1 LEBR

Ensure: precision ϵ, probabilitys δ_k
  $LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 0, \quad n \leftarrow 1$
  while $UB - LB > 2\epsilon$ **do**
    $t \leftarrow t + 1$
    Obtain X_t
    $\delta_n \leftarrow \frac{6\delta}{\pi^2 n^2}$
    $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n}} + \frac{3\ln(3/\delta_n)}{n}$
    $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
    $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
    $n \leftarrow n + 1$
  end while
  return $\overline{X_t}$

2.1.2 Improved LEBR (ILEBR)

Algorytm 1 opiera się na korekcie 1.4.2 która zakłada możliwie nieskończoną ilość testów. Jednak dołożenie ograniczenia na temat żądanej dokładności rzędu ϵ pozwala nam znaleźć maksymalną ilość testów jaką należy wykonać aby z prawdopodobieństwem nie mniejszym niż $1 - \delta$ empiryczna wartość oczekiwana była równa teoretycznej wartości oczekiwanej z dokładnością co do ϵ. Niech $\delta_k = \frac{\delta}{n_{max}}$ gdzie n_{max} oznacza maksymalną potrzebną ilość testów. Z Lematu 1.11 niezależnie od odchylenia standardowego zmiennej losowej X_i jesteśmy w stanie oszacować n_{max} rozwiązując analitycznie poniższe równie 2.1

$$\epsilon = \sqrt{\frac{\ln(2n_{max}/\delta)}{2n_{max}}}. \tag{2.1}$$

Dla $\epsilon = 0.01$ i $\delta = 0.05$ rozwiązaniem numerycznym równania 2.1 jest $n_{max} = 74539.85$. Oznacza to że maksymalna ilość gier jaką należy rozegrać między dwoma graczami aby z prawdopodobieństwem nie mniejszym niż $1-0.05$ oszacować prawdopodobieństwo wygrania gry przez pierwszego gracza z dokładniejsi co do 0.01 wynosi $\lceil n_{max} \rceil = 74540$.

Podobnie możemy postąpić w przypadku nierówności Bernsteina. Na początku jednak musimy oszacować z góry $\overline{\sigma}_t$. Przyjmijmy, że X_i jest zmienną z rozkładu zero-jedynkowego. Wtedy maksymalna możliwa wariancja dla takiej zmiennej losowej wynosi $\sigma^2 = 0.25$. Jest to również maksymalna możliwa wartość dla naszej empirycznej wariancji. Podstawiając wtedy $\delta_n = \frac{0.05}{n}, \epsilon = 0.01$ do Lematu 1.13 otrzymujemy

$$0.01 \leqslant \sqrt{\frac{\ln(\frac{3n_{max}}{0.05})}{2n_{max}}} + \frac{3\ln(\frac{3n_{max}}{0.05})}{n_{max}}. \tag{2.2}$$

Rozwiązaniem numerycznym równania (2.2) jest $n_{max} = 86329$. Jednak granice oparte o nierówność Bernsteina zalezą od wariancji. Sprawdźmy zatem jak wygląda n_{max} w zalotności od σ^2. Z Rysunku 2.1 widzimy zatem, że algorytm oparty o nierówność Bernsteina radzi sobie lepiej w przypadku gdy niskiej wariancji. Co więcej niezależnie od ilości wykonywanych testów, dla $\delta_n = \frac{0.05}{74540}$ maksymalna ilość gier po której z prawdopodobieństwem $1 - 0.05$ wiemy że $|\overline{X_i} - \mu| \leqslant 0.01$ wynosi 74540. Oznacza to że do Algorytmu 1 możemy zastosować funkcje ϵ_k opartą o $\delta_k = \frac{\delta}{n_{max}}$ oraz dołożyć nowy warunek zatrzymania algorytmu gdy liczba wykonanych testów przekroczy n_{max}. Dotychczas uwzględnione poprawki przedstawione sa przy Algorytmu 2.

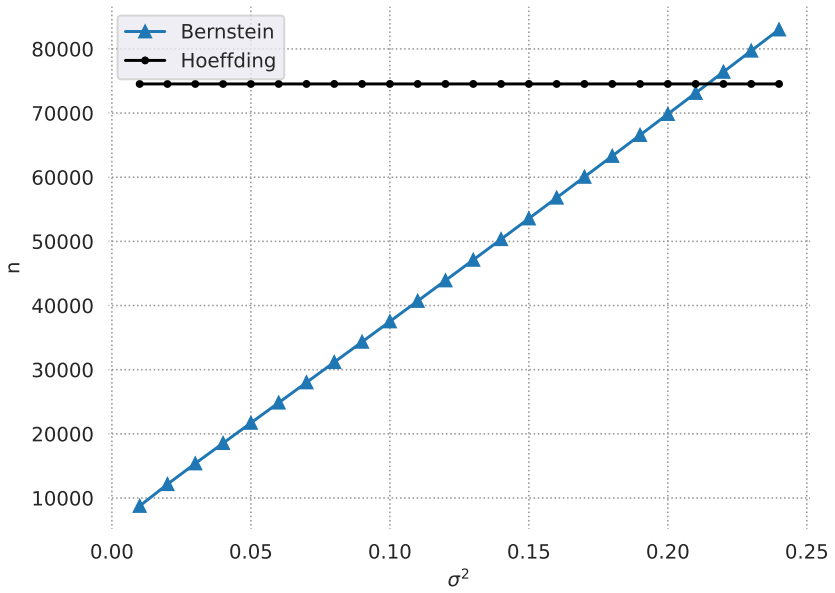

Figure 2.1: Wykres maksymalnej potrzebnej ilości testów w zależności od wariancji zmiennych losowych X_i w przypadku gdy liczba testów jest równa liczbie gier ($t = n$) dla $\epsilon = 0.01$ i $\delta = 0.05$.

Algorithm 2 ILEBR 1

Ensure: precision ϵ, probability δ
  $LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 0, \quad n \leftarrow 1$
  Find n_{max} such as $\epsilon = \sqrt{\frac{\ln(2n_{max}/\delta)}{2n_{max}}}$
  $\delta_n = \delta / n_{max}$
  while $UB - LB > 2\epsilon$ or $n \leqslant n_{max}$ **do**
    $t \leftarrow t + 1$
    Obtain X_t
    $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n}} + \frac{3\ln(3/\delta_n)}{n}$
    $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
    $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
    $n \leftarrow n + 1$
  end while
  return $\overline{X_t}$

Algorytmy 1 i 2 przeprowadzały testy po każdej rozegranej grzęz, skutkuje to wytopieniem bardzkiej ilości testów. Zamiast przeprowadzać test po każdej rozegranej grze, niech test odbywa się w momencie gdy liczba rozegranych gier będzie równa pewnej funkcji zależnej od ilości przeprowadzonych testów. Na podstawie artykuł [4] wiemy że funkcja która pozwoli nam na polepszenie wyników jest $t = n^2$. Przeprowadźmy identyczną procedure jak w przypadku równań (2.1) i (2.2). Tym razem jednak zamiast stosować $t = n$ do Lematów 1.11 i 1.13 zastosujmy $t = n^2$.

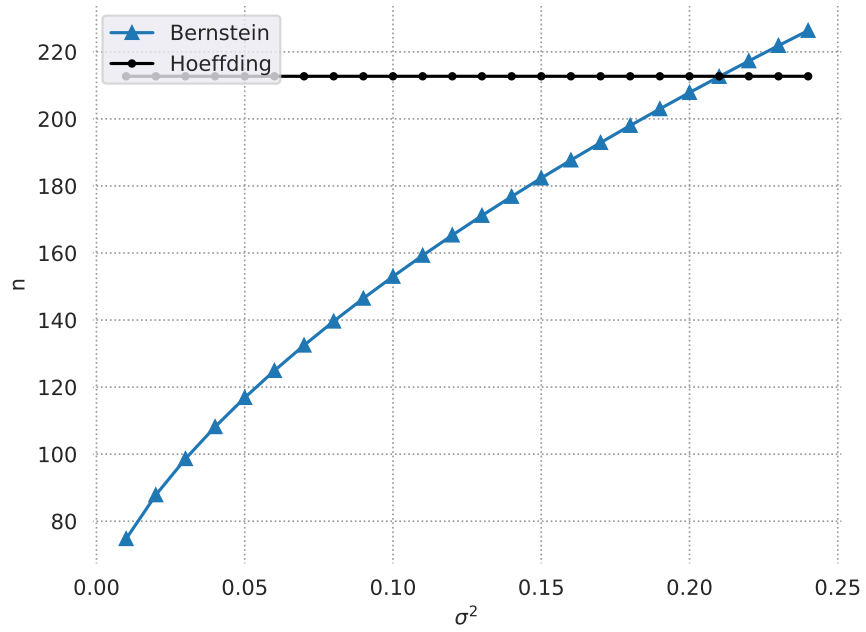

Figure 2.2: Wykres maksymalnej potrzebnej ilości testów w zależności od wariancji zmiennych losowych X_i w przypadku gdy liczba rozegranych gier jest równa liczbie przeprowadzonych testów do kwadratu ($t = n^2$) dla $\epsilon = 0.01$ i $\delta = 0.05$.

W przypadku granicy opartej o nierówność Hoffdina otrzymujemy

$$\epsilon = \sqrt{\frac{\ln(2n_{max}/\delta)}{2n_{max}^2}} \implies t_{max} = \lceil n_{max} \rceil^2 = \lceil 213 \rceil^2 = 45369. \tag{2.3}$$

W przypadku granicy opartej o nierówność Bernsteina otrzymujemy

$$0.01 \leqslant \sqrt{\frac{\ln(\frac{3n_{max}}{0.05})}{n_{max}^2} + \frac{3\ln(\frac{3n_{max}}{0.05})^2}{n_{max}}} \implies t_{max} = \lceil n_{max} \rceil^2 = \lceil 230.751^2 \rceil = 53247. \tag{2.4}$$

Porównując ze sobą uzyskane wartość z wzoru (2.1) oraz (2.3) widzimy że udało się zmniejszyć maksymalną ilość gier jakie należny wykonać z 74540 do 45369. Wprowadźmy zatem nowo uzyskane poprawki trwożąc Algorytm 3.

Algorithm 3 ILEBR 2

Ensure: precision ϵ, probability δ
  $LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 0, \quad n \leftarrow 1$
  Find n_{max} such as $\epsilon = \sqrt{\frac{\ln(2n_{max}/\delta)}{2n_{max}^2}}$
  $\delta_n = \delta / n_{max}$
  while $UB - LB > 2\epsilon$ or $n \leqslant n_{max}$ **do**
    repeat
      $t \leftarrow t + 1$
      Obtain X_t
    until $t = n^2$
    $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n^2}} + \frac{3\ln(3/\delta_n)}{n^2}$
    $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
    $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
    $n \leftarrow n + 1$
  end while
  return $\overline{X_t}$

Do tej pory Wszystkie stosowane algorytmy wyznaczały nam prawdopodobieństwo wygranej pierwszego gracza. Jednak nam nie zależny na tym dokładnie znać prawdopodobieństwo wygranej graczy. Głównym celem algorytmów jest odnalezienie lepszego gracza. Pozwala nam to modyfikacje Algorytmu 3 o nowe warunki zatrzymania.

Algorithm 4: ILEBR* 1

Ensure: precision ϵ, probability δ
  $LB \leftarrow 0, \quad UB \leftarrow \infty, \quad t \leftarrow 1, \quad n \leftarrow 0$
  Find n_{max} such as $\epsilon = \sqrt{\frac{\ln(2n_{max}/\delta)}{2n_{max}^2}}$
  $\delta_n = \delta / n_{max}$
  while $(UB - LB > 2\epsilon$ or $n < n_{max} + 1)$ and $(UB > 0.5$ or $LB < 0.5)$ **do**
    repeat
      $t \leftarrow t + 1$
      Obtain X_t
    until $t = n^2$
    $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{n^2}} + \frac{3\ln(3/\delta_n)}{n^2}$
    $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
    $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
    $n \leftarrow n + 1$
  end while
  if $LB > 0.5$ **then**
    return p_1 win
  else if $UB < 0.5$ **then**
    return p_2 win
  else if $\overline{X_n} > 0.5$ **then**
    return p_1 win
  else
    return p_2 win
  end if

Do tej pory staraliśmy się ograniczyć z góry maksymalną ilość rozgrywanych gier. Jednak oczywistym faktem jest że nie ma sensu testowania który z graczy jest lepszy w monecie gdy ilość rozegranych gier jest niewystarczająco mały. Jadnak jak wyznaczyć nasze minimalne t po którym zaczniemy testować? Algorytm oparte o nierówność Bernsteina najszybciej wyznaczają wynik w przypadku gdy wariancja naszej zmiennej losowej wynosi 0 (jeden z graczy zawsze wygrywa). Dla Algorytmu **??** interesuje nas monet od którego będziemy w stanie rozróżnić kiedy dolna bać górna granica przekroczy $\frac{1}{2}$. Zatem oszacujmy n_{min} korzystać z Lematu 1.13 dla $\overline{\sigma}_t = 0$ i $t = n^2$.

$$0.5 \leqslant \frac{3\ln(\frac{3n_{max}}{0.05})}{n_{min^2}} \implies t_{min} = \lceil n_{min} \rceil^2 = \lceil 7.53219 \rceil^2 = 64.$$

Oznacza to, że przy $\epsilon = 0.01$ i $\delta = 0.05$ w przypadku gdy jeden graczy wygrywa zawsze, będziemy w stanie to stwierdzić nie prędzej niż po 64 grach. Na tej podstawie wyznaczy nowe n_{max} uwzględniając pomijanie pierwsze niepotrzebne testy.

$$0.01 = \sqrt{\frac{\ln(2n_{max}/0.05)}{2(n_{max}+7)^2}} \implies t_{max} = \lceil n_{max} + 6 \rceil^2 = \lceil 205.289 + 7 \rceil^2 = 45369. \tag{2.5}$$

Chociaż równanie 2.5 nie zmniejszyło nam maksymalnej liczby testów jakie należy wykonać to pozwoliło nam ono na zmniejszenie ϵ_n. Oznacza to że stosują odroczenie pierwszych testów, pozwala nam na dokładniejsze oszacowanie naszej górnej i dolnej granicy w stosowanych algorytmach.

Algorithm 5: ILEBR* 2

Ensure: precision ϵ, probability δ
$LB \leftarrow 0, \quad UB \leftarrow \infty, \quad n \leftarrow 0$
Find n_{max} such as $\epsilon = \sqrt{\frac{\ln(2n_{max}/\delta)}{2n_{max}^2}}$
Find n_{min} such as $0.5 = \sqrt{\frac{\ln(2n_{max}/\delta)}{2(n_{min}+1)^2}}$
Obtain first $X_1, X_2, X_{n_{min}}$ games
$t \leftarrow n_{min}^2$
$\delta_n = \delta / n_{max}$
while $(UB - LB > 2\epsilon$ or $n \leqslant n_{max})$ and $(UB > 0.5$ or $LB < 0.5)$ **do**
  repeat
    $t \leftarrow t + 1$
    Obtain X_t
  until $t = n^2$
  $\epsilon_n \leftarrow \overline{\sigma}_n \sqrt{\frac{2\ln(3/\delta_n)}{(n+n_{min})^2}} + \frac{3\ln(3/\delta_n)}{(n+n_{min})^2}$
  $LB \leftarrow \max(LB, \overline{X_n} - \epsilon_n)$
  $UB \leftarrow \min(UB, \overline{X_n} + \epsilon_n)$
  $n \leftarrow n + 1$
end while
if $LB > 0.5$ **then**
  return p_1 win
else if $UB < 0.5$ **then**
  return p_2 win
else if $\overline{X_n} > 0.5$ **then**
  return p_1 win
else
  return p_2 win
end if

2.2 Moc testów statysytycznych

Mocą testu statystycznego nazywamy prawdopodobienstwo unikniecia popełnenia błędu drugiego rodzaju, czyli prawdopoodbienstwo przyjecie hipotezy zerowej gdy w rzeczywistosci jest ona fałszywa. W naszym przypadku jest to prawdopodobienstwo przyjecia za lepszego gracza osoby o niższym prawdopodobienstiwe wygranej. Wysoka moc testu jest pożadaną własciwoscia testu, jednak żadanie jaknajlepszej mocy może wiazać się z zwiększeniem próby badawczej.

Patrząc wyłączanie na Rysunek 2.3 możemy zobaczyć że wszystkie testu czują się dobrą mocą i działają bardzo dobrze do memento gdy jeden z graczy ma prawdopodobieństwo wygranej bliskie 0.495. Dodatkowo obszary ściemnione oznaczają 95% przedziały ufności.

Porównując ze sobą wyniki przedstawione na Rysunkach 2.3 i 2.4 możemy stwierdzić, że testem o najlepszej mocy jest testo oparty o algorytm LEBR, jednak wymaga on bardo dłużej liczby gier potrzebnych do rozegrania. Test statystyczny oparty o algorytm ILEBR ma gorsza moc ale pozwala nam na prawie dwukrotnie zmniejszenie liczby wymaganych gier. Dodatkowo widzimy że algorytmy ILEBR 2, ILEBR* i ILEBR* 2 cechują się mocą na bardzo podobnym poziomie jednak wersje algorytmy oznaczone * pozwalają nam bardo

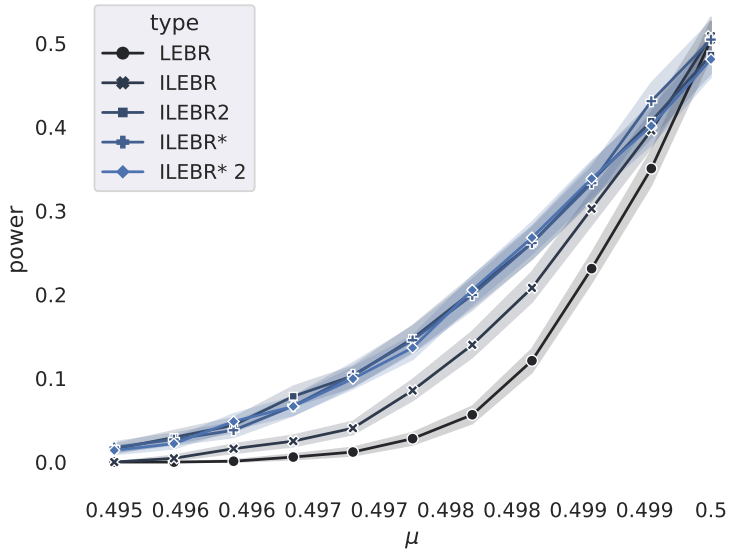

Figure 2.3: Wykres mocy testów statystycznych w zależności od wartosci oczekiwanej zmiennych losowych X_i dla $\epsilon = 0.01$ i $\delta = 0.05$.

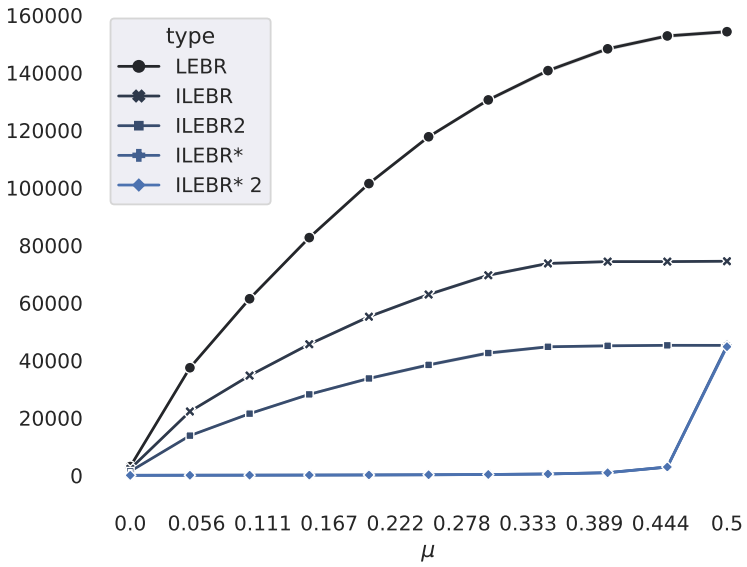

Figure 2.4: Wykres sredniej liczby gier potrzebej do rozegrania w zależności od wartosci oczekiwanej zmiennych losowych X_i dla $\epsilon = 0.01$ i $\delta = 0.05$.

duże ograniczenie liczby gier potrzebnej do rozegrania aby test mógł wyznaczyć lepszego gracza. W poniższej pracy będziemy stosować algorytm ILEBR* 2 ponieważ pozwoli on na znacznie niezwieszenie liczby porównań jakie będziemy przeprowadzacz w celu wyznaczenia lepszego gracza.

2.3 Algorytmy wyznaczania optymalnej strategi

Wszystkie algorytmy przedstawione w tym rozdziale są algorytmami genetycznymi [3]. Algorytm generacyjny to sposób tworzenia nowych rozwiązań dla danego problemu poprzez iteracyjne stosowanie procesu ewolucyjnego. W tym procesie tworzone są nowe rozwiązania, oceniane ich jakość i wybierane najlepsze z nich, aby stworzyć kolejną generację rozwiązań. Ten proces powtarza się, aż zostanie znalezione rozwiązanie spełniające określone kryteria. Algorytm generacyjny może być używany do rozwiązywania różnych rodzajów problemów, takich jak optymalizacja, uczenie maszynowe, tworzenie sztucznej inteligencji i wiele innych. Algorytmy ewolucyjne są często interpretowane jako odwzorowanie procesu ewolucyjnego w naturze, ze względu na swoje odpodobnienie do procesu selekcji naturalnej w którym najlepsze jednostki są wybierane do reprodukcji i tworzenia nowych pokoleń.

Ogólnie rzecz biorąc, algorytm generacyjny składa się z kilku kluczowych kroków:

- Inicjalizacja: Tworzenie początkowej zbioru rozwiązań dla danego problemu.
- Ocena: Ocena jakości każdego z rozwiązań za pomocą odpowiedniej funkcji celu lub innych miar.
- Selekcja: Wybieranie najlepszych rozwiązań do następnej generacji.
- Krzyżowanie: Łączenie najlepszych rozwiązań z poprzedniej generacji, aby stworzyć nowe rozwiązania dla następnej generacji.
- Mutacja: Losowa zmiana jednego lub więcej elementów w nowych rozwiązaniach, aby zapewnić różnorodność w następnej generacji.
- Powtarzanie: Powtarzanie kroków 2-5, aż zostanie znalezione rozwiązanie spełniające określone kryteria jakości lub osiągnięty zostanie maksymalny poziom iteracji.

Proces znajdowania potencjalnych rozwiązań polega na przeszukiwaniu przestrzeni wszystkich możliwych rozwiązań i wybieraniu tych, które dają najlepsze wyniki. W rzeczywistości jednak często nie mamy fizycznej możliwości sprawdzenia wszystkich możliwych rozwiązań lub ich sprawdzenie jest zbyt czasochłonne lub kosztowne. Dlatego w algorytmach ewolucyjnych często wykorzystuje się techniki probabilistyczne, które pomagają wybierać, tworzyć i wyszukiwać kolejne rozwiązania.

W niniejszej pracy zaprezentujemy 4 algorytmy, których celem jest znalezienie optymalnej strategii. Trzy pierwsze algorytmy zostaną opisane na podstawie pracy [1]]. Czwartym algorytmem jest proponowana przeze mnie metoda, która jest bardzo podobna do algorytmu trzeciego (patrz [referencja]). Opis tego algorytmu zostanie przedstawiony w dalszej części pracy.

2.3.1 Argorytm itereacyjny

Pierwszym algorytmem, który zostanie przedstawiony, jest algorytm iteracyjny. Jest to metoda bardzo intuicyjna, opierająca się na stopniowym zwiększaniu skuteczności strategii graczy poprzez porównywanie ich wyników. Gdy znajdziemy strategię, która daje lepsze wyniki niż ta poprzednia, staje się ona nowym punktem odniesienia (baseline). W oparciu o tę nową strategię algorytm będzie kontynuować proces optymalizacji wyników graczy. Nowa strategia jest akceptowana, jeśli wygrywa ona z prawdopodobieństwem większym niż 50% w porównaniu do poprzedniej strategii. Algorytm 6 jest przykładem algorytmem ewolucyjnym typu (1+1) [2].

Algorithm 6: Iterative algorithm

Ensure: precision ϵ, probability δ, random opponent x
  $\sigma \leftarrow 1$ ▷ Inital step-size
  while (termination criterion is not met) **do**
    for all $i = 1$ to lenght of x **do**
      $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Mutation
    end for
    repeat
      play game between x' and x
    until the limited Bernstein race of precision ϵ stop
    if x' better then x **then**
      $x \leftarrow x'$
      $\sigma \leftarrow 1.25\sigma$
    else
      $\sigma \leftarrow 0.84\sigma$
    end if
  end while
  return an approximation x of the optimal strategy

2.3.2 Real Coevolution algorytm

Kolejnym algorytmem ewolucyjnym, którego będziemy używać, jest algorytm koewolucyjny. W przypadku tej metody nowy punkt odniesienia jest wybierany w momencie, gdy nowo znaleziona strategia okazuje się lepsza niż wszystkie dotychczas wybrane strategie. Pseudokod algorytmu koewolucyjnego został przedstawiony jako Algorytm 7.

2.3.3 Aproxx Coevolution 1 algorytm

W przypadku dwóch poprzednich algorytmów nowe rozwiązanie jest tworzone na podstawie poprzednio znalezionego rozwiązania. Możemy jednak zastosować tzw. "podejście Paryskie" (Parisian approach). W tym podejściu zamiast porównywać nowo uzyskaną strategię z każdą poprzednio przyjętą, porównujemy ją tylko z jedną losowo wybraną strategią z populacji. Pseudokod algorytmu koewolucyjnego został przedstawiony jako Algorytm 8.

Algorithm 7: Real Coevolution

Ensure: precision ϵ, probability δ, random opponent x
  $\sigma \leftarrow 1$ ▷ Inital step-size
  $P \leftarrow \{x\}$ ▷ Best point population
  while (termination criterion is not met) **do**
    for all $i = 1$ to lenght of x **do**
      $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Mutation
    end for
    for all $i = 1$ to lenght of P **do**
      repeat
        play game between x' and P_i
      until each limited Bernstein race of precision ϵ stops
    end for
    if x' better then all points in P **then**
      $x \leftarrow x'$
      $P \leftarrow \{P, x'\}$
      $\sigma \leftarrow 1.25\sigma$
    else
      $\sigma \leftarrow 0.84\sigma$
    end if
  end while
  return an approximation x of the optimal strategy

Algorithm 8: Approximate Coevolution

Ensure: precision ϵ, probability δ, random opponent x
  $\sigma \leftarrow 1$ ▷ Inital step-size
  $P \leftarrow \{x\}$ ▷ Best point population
  while (termination criterion is not met) **do**
    for all $i = 1$ to lenght of x **do**
      $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Mutation
    end for
    Draw at random an integer $rand$ between 1 and the size of P
    repeat
      play game between x' and $rand^{\text{th}}$ individual of P
    until each limited Bernstein race of precision ϵ stops
    if x' better then all points in P **then**
      $x \leftarrow x'$
      $P \leftarrow \{P, x'\}$
      $\sigma \leftarrow 1.25\sigma$
    else
      $\sigma \leftarrow 0.84\sigma$
    end if
  end while
  return an approximation x of the optimal strategy

2.3.4 Aproxx Coevolution 2 algorytm

Ostatnim algorytmem, którym się zajmiemy, jest kolejny algorytm koewolucyjny. Tym razem, zamiast porównywać naszą strategię z jedną losowo wybraną strategią z populacji, tak jak to robiliśmy w przypadku Algorytmu 8, nasza strategia będzie testowana przeciwko losowo wybranemu przeciwnikowi.Pseudokod algorytmu koewolucyjnego został przedstawiony jako Algorytm 9.

Algorithm 9: Approximate Coevolution 2

Ensure: precision ϵ, probability δ, random opponent x
  $\sigma \leftarrow 1$ ▷ Inital step-size
  $P \leftarrow \{x\}$ ▷ Best point population
  while (termination criterion is not met) **do**
    for all $i = 1$ to lenght of x **do**
      $x'_i \leftarrow x_i + \sigma\mathcal{N}(0, 1)$ ▷ Mutation
    end for
    repeat play game between x' and random individual of P
    until each limited Bernstein race of precision ϵ stops
    if x' better then all points in P **then**
      $x \leftarrow x'$
      $P \leftarrow P, x'$
      $\sigma \leftarrow 2\sigma$
    else
      $\sigma \leftarrow 0.84\sigma$
    end if
  end while
  return an approximation x of the optimal strategy

Chapter 3

Gry

"Aby sprawdzić poprawność działania powyższych algorytmów, przetestujemy je na podstawie dwóch gier. Pierwszą z nich będzie popularna gra karciana zwana Wojną. Drugą grą, w której będziemy szukać optymalnej strategii, będzie gra o nazwie Rrrats. Przedstawmy najpierw zasady obowiązujące w obu tych grach.

3.1 Gra w wojnę

Wojna to prosta gra karciana, w której uczestnicy grają przeciwko sobie i używają talii standardowych kart do gry. Celem gry jest zdobycie wszystkich kart od przeciwnika.

Zasady gry są następujące:

- Gracze rozdają po 26 kart, tak aby każdy miał swoje ukryty "magazyny".
- Następnie jedna karta jest odkrywana z każdego magazynu i porównywana ze sobą. Gracz, który ma kartę o wyższej wartości, zabiera obie karty i dokłada je na koniec swojego magazynu. Jeśli karty są takie same, gracze rozgrywają "wojnę".
- W przy wojny, oboje gracze wykładają z magazynów najpierw jedną kartę rewersem do góry, a następnie odkrywają kolejną kartę rewersem do dołu. Ten gracz, który ma kartę o wyższej wartości, zabiera wszystkie karty i dokłada je do swojego magazynu. Jeśli karty są takie same, proces powtarza się, aż do momentu gdy jeden z graczy wygra.
- Gra kończy się w monecie któryś z graczy wygra wszystkie karty.

Same zasady gry nie definiują jednak tego w jaki sposób karty na koniec naszego "magazynu" mogą zostać umieszczane.

Wprowadźmy zatem 3 parametry nazwijmy je odpowiednio A, B, C, których będziemy używać do wyznaczania nieujemnych parametrów $\alpha = \exp(A)$, $\beta = \exp(B)$, $\gamma = \exp(C)$. Wtedy, umieśćmy k wygranych kart na końcu naszego "magazynu" niestepujący sposób:

- karty w kolejności malejącej z prawdopodobieństwem równym $\alpha/(\alpha+\beta+\gamma)$
- karty w kolejności rosnącej z prawdopodobieństwem równym $\beta/(\alpha+\beta+\gamma)$
- karty w kolejności losowej z prawdopodobieństwem równym $\gamma/(\alpha+\beta+\gamma)$

3.2 Rrrats!

Rrrats jest prosta grą kościaną typu ekstensywnego, a jej celem jest zdobycie przez poszczególnego gracza najwierniejszej liczby punktów. W każdej swojej turze gracz rzuca dwiema takomskim, aż do momentu gdy zostanie spełniony warunek stopu, bądź sam uzna że nie chce już konturować. Gra kończy się w momencie gdy z głównego stosu znikną wszystkie 31 żetonów (punktów). W grze ożywa się specjalnych których prawdopodobieństw uzyskania otwarci 0, 1, 2 wynosi odnowienie 3/6, 2/6, 1/6.

Zasady gry sa następujące:

- Grac w swojej turze może rzucić dwoma kosterskimi dowolną ilość razy
- Gracz wykonuje następujące akcje w zależności od uzyskanego wyniku:
  a) Jeśli na kostce wypadło 1, gracz bierze do "reki" żeton z stosu głównego.
  b) Jeśli wypadło 2, gracz kranie punkt przeciwnikowi i bierze go do swojej "reki". Jeśli to nie możliwe gracz bierze punkt z stosu głównego.
  c) Jeśli upadło 0, nic się nie dzieje.
  d) Jeśli na obu kostkach wypadło 0, gracz kończy swoją turę i odkłada wszystkie punkty jakie ma w "ręce".

  Przykład: Jeśli na kostkach wypadnie 0 i 1 gracz pobiera 1 punkt ze stosu głowniowe. Jeśli an kostkach wypadnie 1 i 1 gracz pobiera 2 punkty ze stosu głównego.
- Po każdym żucie gracz decyduje się na to czy grac dalej, czy zakończyć swoja torę.
- Jeśli gracz ma więcej niż 4 punktów w "rece" tura gracza automatycznie się kończy a uzyskaną nadwyżkę odkłada się do stosu głównego.
- Jeśli tura dobiegła końca, to gracz przenośni wszystkie punkty uzyskane na "ręce" do swojej puli puntów osobistych.
- Gra kończy się w momencie gdy liczba punktów na głowy stosie wyniesie 0.

W tym przypadku strategia której będziemy szukać będzie oparta o 8 parametrów $A_1,\ A_2, B_1,\ B_2, C_1,\ C_2, D_1,\ D_2$. Przy pomocy tych paramentów wyznaczymy kolejne 8 współczynników $\alpha_1 = \exp(A_1),\ \alpha_2 = \exp(A_2), \beta_1 = \exp(B_1),\ \beta_2 = \exp(B_2), \gamma_1 = \exp(C_1),\ \gamma_2 = \exp(C_2)$. Naszym celem jest wyznaczenie prawdopodobieństwa zdecydowania się na dalszy rzut kośćmi w zależność od ilości punktów posiadanych na "ręce". Wprowadźmy zatem zmienna losowa $Y|K = k,\ k = \{1, 2, 3\}$. Posłuży ona nam do wyznaczenia prawdopodobieństwo zdecydował się na dalszy rzut kośćmi ($Y = 1$) w zależności od ilości posiadanych punktów na "rece".

Same prawdopodobieństwa wprowadzonej ziemnej losowej określamy w następujący sposób:

$$P(Y = 1|K = 1) = \alpha_1/(\alpha_1 + \alpha_2),$$
$$P(Y = 1|K = 2) = \beta_1/(\beta_1 + \beta_2),$$
$$P(Y = 1|K = 3) = \gamma_1/(\gamma_1 + \gamma_2).$$

Chapter 4

Wyniki

Do przedstawienia wyników dziania przedstawionych algorytmów ożyjemy dwóch. Pierwsza z nich będzie tabela obrazująca prawdopodobieństwo granej strategi uzyskanych przez jazdy z algorytmów przeciwko strategi pocztowej. Droga tabele zobrazuje natomiast wyniki działania naszych strategii przeciwko strategią uzyskanym przez pozostałe algorytmy. Wyniki przedstawione na Tabelach 4.1 i 4.2 oraz Rysunkach 4.1 i 4.2 pokakują, że kazdemu z naszych alkorytmów odało sie zdaleśc strategie dającą sredio wikszy winnik od losowej strategi poczatkowej. Dodatkowo wyzaczone straetegie możemy zinterpretowac interpretować wprosty do zlozymnienia dla człowieka sposób. W grze wojnnie istnieje strategia prosta, jest nią układanie kart zawse w kolejnosci od nawiekszej do najmniejszej. Dodatkowo strategia ta ma 70% szans na wygranie przeciwko straetgi układania kart w sposób malejaćy, oraz 53% szans na wygranie przeciwko strategi losowej.

	iterawtive	coevolution	approx coevolution	approx coevolution 2
mean	0.541617	0.545225	0.530655	0.528750
std	0.005623	0.005158	0.004900	0.005244
min	0.525300	0.532900	0.517400	0.514000
25%	0.538300	0.541100	0.527600	0.525800
50%	0.540600	0.544600	0.530600	0.529200
75%	0.545575	0.548800	0.534700	0.532500
max	0.556100	0.556700	0.542200	0.540100

Table 4.1: Rezultaty uzyskanych strategi przeciwko strategi poczatkowej dla gry Rrrats!.

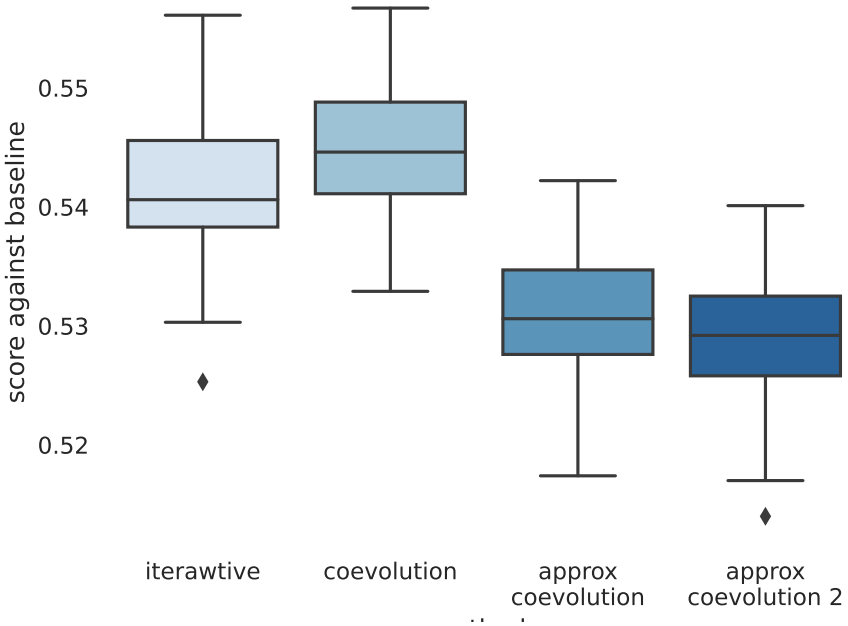

Figure 4.1: Wykresy pódełkowe przedwaiajace rezultaty uzyskanych strategi przeciwko strategi poczatkowej dla gry Wojna.

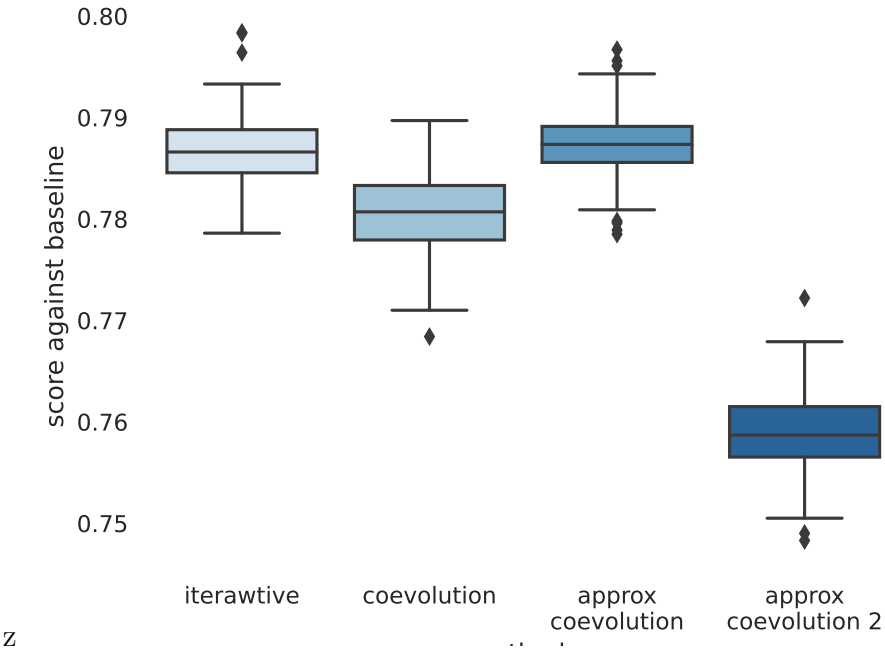

Figure 4.2: Wykresy pódełkowe przedwaiajace rezultaty uzyskanych strategi przeciwko strategi poczatkowej dla gry Rrrats.

	iterawtive	coevolution	approx coevolution	approx coevolution 2
mean	0.786629	0.780805	0.787310	0.759073
std	0.003581	0.004265	0.003586	0.004170
min	0.778600	0.768400	0.778500	0.748300
25%	0.784550	0.777925	0.785575	0.756525
50%	0.786600	0.780700	0.787350	0.758700
75%	0.788800	0.783300	0.789125	0.761500
max	0.798400	0.789700	0.796700	0.772200

Table 4.2: Rezultaty uzyskanych strategi przeciwko strategi poczatkowej dla gry Wojna.

4.1 ernstein race without maximum race length

Wykorzystując Lemat 1.13 oraz korektę 1.4.2 z $\delta_k = \frac{c\delta}{k^2}, c = \frac{6}{\pi^2}$ otrzymujemy, że dla ciągu $X_1, X_2, \ldots, X_t$ i.i.d. zmiennych losowych takim, że $0 \leqslant X_i \leqslant 1$

$$\epsilon_{t,k} \leqslant \overline{\sigma}_t \sqrt{\frac{2\ln(3/\delta_k)}{t}} + \frac{3\ln(3/\delta_k)}{t} = \overline{\sigma}_t \sqrt{\frac{2\ln(\frac{k^2\pi^2}{2\delta})}{t}} + \frac{3\ln(\frac{k^2\pi^2}{2\delta})}{t}$$

Wtedy $e_{t,k}$ interpretujemy jako maksymalną różnicę między empiryczną a teoretyczną wartością oczekiwaną po przeprowadzeniu k testów i rozegraniu t gier, z prawdopodobieństwem pomyłki równym

Lemat 4.1. *Niech liczba rozegranych gier będzie funkcja zależną od k ($f(k) = t$) oraz niech $\delta_k = \frac{\delta}{g(k)}$ gdzie $\delta \geqslant \sum_{k=1}^{\infty} \frac{\delta}{g(k)}$ i $\ln(g(k)) \in o(f(k))$. Wtedy $\lim_{k\to\infty} e_{f(k),k} = 0$*

Dowód Faktu 4.1.

$$0 \leqslant X_i \leqslant 1 \implies \overline{\sigma}_t^2 \leqslant \frac{1}{2}$$

$$\epsilon_{f(k),k} \leqslant \sqrt{\frac{\ln(\frac{3g(k)}{\delta})}{f(k)}} + \frac{3\ln(\frac{3g(k)}{\delta})}{f(k)}$$

Z założeń wiemy ze $ln(g(k)) \in o(f(k))$, zatem

$$\lim_{k\to\infty} \frac{\ln(\frac{3f(k)}{\delta})}{f(k)} = 0$$

Co ostatecznie z tw. o trzech ciągach daje nam

$$0 \leqslant \lim_{k\to\infty} e_{f(k),k} \leqslant 0 \implies \lim_{t\to\infty} e_{f(k),k} = 0$$

□

Lemat 4.1 pozwala nam na to aby ograniczyć ilość przeprowadzanych testów. Ponieważ podejście oparte o testowane który z graczy jest lepszy po każdej rozegranej grze prowadzi do nadmiarowej ilości wykonywanych testów. Przykładem funkcjami jakimi użyć do Lematu 4.1 są $f(k) = k^2$, $g(k) = \frac{6/\pi^2}{k^2}$. Teaki dobór funkcji oznacza ze k-ty test odbywa się gdy liczba przeprowadzonych gier wynosi k^2.

Podsumowanie

Podsumowanie w pracach matematycznych nie jest obligatoryjne. Warto jednak na zakończenie krótko napisać, co udało nam się zrobić w pracy, a czasem także o tym, czego nie udało się zrobić.

Dodatek

Dodatek w pracach matematycznych również nie jest wymagany. Można w nim przedstawić np. jakiś dłuższy dowód, który z pewnych przyczyn pominęliśmy we właściwej części pracy lub (np. w przypadku prac statystycznych) umieścić dane, które analizowaliśmy.

Bibliography

[1] Cauwet, M.-L., Teytaud, O. Surprising strategies obtained by stochastic optimization in partially observable games. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), IEEE, pp. 1–8.

[2] Droste, S., Jansen, T., Wegener, I. A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with boolean inputs. *Evolutionary Computation 6*, 2 (1998), 185–196.

[3] Figielska, E. Algorytmy ewolucyjne i ich zastosowania. 81–92.

[4] Heidrich-Meisner, V., Igel, C. Non-linearly increasing resampling in racing algorithms. In *European Symposium on Artificial Neural Networks* (2011), Evere, Belgium: d-side publications, pp. 465–470.

[5] Płatkowski, T. Wstęp do teorii gier. *Uniwersytet Warszawski* (2012).

[6] Prisner, E. *Game theory through examples*, vol. 46. American Mathematical Soc., 2014.