TURNIEJ AXELRODA

Treść projektu

„Dylemat więźnia jest prostym modelem z zakresu Teorii Gier. Przedstawia on sytuację, w której złapano dwie osoby podejrzane o udział w napadzie (nazwijmy je A i B). Każdej z nich (osobno) prokurator proponuje umowę: jeśli przedstawisz nam dowody na tego drugiego, to zwolnimy cię z odbywania kary, a ten drugi pójdzie siedzieć na długo.

Każdy z przestępców ma zatem dwie możliwości: zachować się lojalnie (fair) w stosunku do kolegi i nic nie mówić policji, albo nielojalnie (unfair) i zeznawać przeciwko niemu.

Możliwe są zatem następujące sytuacje:

1. Jeden z graczy zachowuje się nie fair (donosi), a drugi fair (milczy). Można to zapisać jako A=U, B=F. W takim przypadku A wychodzi na wolność (0 lat w więzieniu), zaś B – zostaje skazany na 10 lat.
2. Sytuacja wygląda podobnie, ale A=F, B=U. W takiej sytuacji A otrzyma karę 10 lat, a B – wychodzi na wolność.
3. Obaj gracze zachowują się fair (A=F, B=F). W tym przypadku proces jest procesem poszlakowym i każdy z nich pójdzie siedzieć na 3 lata.
4. Obaj gracze zachowują się unfair (donoszą na siebie wzajemnie, A=U, B=U), więc policja dysponuje twardymi dowodami – choć sąd nieco złagodzi wyrok. W tej sytuacji obaj zostaną skazani na 7 lat.

Jednokrotny dylemat więźnia polega zatem na przydziale kar, w zależności od postępowania obu graczy.

Możemy na przykład napisać funkcję, która będzie podawała karę dla gracza – zaś jej parametrami wejściowymi będą decyzje obu graczy. Dla uproszczenia przyjmiemy, że zachowanie unfair zakodujemy jako -1, a fair jako 1.

W związku z tym funkcja może wyglądać tak:

**int myPenalty ($myDecision, $hisDecision) {**

**if ($myDecision==-1) and ($hisDecision==-1) $result = 7;**

**if ($myDecision==1) and ($hisDecision==1) $result = 3;**

**if ($myDecision==-1) and ($hisDecision==1) $result = 0;**

**if ($myDecision==1) and ($hisDecision==-1) $result = 10;**

**return $result;**

**}**

Oczywiście jest to tylko przykład takiego rozwiązania. Jeśli mamy dwóch graczy, A i B, a ich decyzje oznaczymy jako $a i $b to kary dla każdego z nich wyliczymy tak:

**$a = 1; $b = -1;**

**$wyrokA = myPenalty($a, $b);**

**$wyrokB = myPenalty($b, $a);**

Oczywiście pojedynczy dylemat więźnia jest mało interesujący. Rzecz się komplikuje jeśli gracze A i B grają w dylemat wielokrotnie. Nazywa się to iterowanym dylematem więźnia.

W takiej sytuacji mówimy o STRATEGII, czyli o sposobie, w jaki gracz podejmuje decyzję w kolejnych grach.

Na przykład, strategia „zawsze unfair” – to bardzo prosta strategia, która mówi: niezależnie od wszystkiego, zawsze będę donosił. Strategia losowa to strategia, w której za każdym razem rzucam monetą i wybór donoszę-nie donoszę jest zupełnym przypadkiem. O wiele ciekawsze są jednak strategie, które zawierają pamięć i podejmują decyzję na podstawie dotychczas rozegranych gier.

Tzw. turniej Axelroda polega na przygotowaniu programów graczy grających różnymi strategiami, a następnie na rozgrywaniu pomiędzy nimi (systemem każdy z każdym) wielokrotnie powtarzanych

iterowanych dylematów (np. 1000 razy). ”

Warunki zaliczenia

„Napisz aplikację symulującą Turniej Axelroda dla co najmniej 6 różnych strategii. Programy grają ze sobą 1000 razy (każdy z każdym). Pokaż wyniki w postaci aplikacji internetowej i opisz własne wnioski.

Zaliczenie polega na przygotowaniu aplikacji, rozegraniu turnieju Axelroda i opracowaniu wyników. Aplikację należy zzipować, a wnioski przedstawić w PDF.

Zaliczenie należy wysłać na w.moscibrodzki@gmail.com w terminie do 31 maja z tytułem TURNIEJ AXELRODA. W treści maila: imiona i nazwiska autorów.

Aplikację można robić w zespołach max 4 osobowych.”

Opis implementacji

Interfejs użytkownika

Interfejs jest prosty i intuicyjny, a na jego wygląd składają się dwie główne części. Pierwszą z nich jest panel wyboru strategii dla każdego z graczy. W tej części znajdziemy również przycisk, po kliknięciu którego, symulacja wykona się łącznie 1000 razy na podstawie wybranych wcześniej strategii. W drugiej części są wykresy, które pokazują wyniki za pomocą dwóch różnych tabel

1. Panel wyboru strategii:

Użytkownik może wybrać jedną strategię dla każdego gracza z rozwijanych list, które również po wybraniu strategii pokazują opis danego rozwiązania. Dostępne strategie to.:

- Zawsze fair

- Zawsze nie fair

- Fair na 70%

- Nie fair na 70%

- Losowa (50/50)

- Oko za oko

- Do pierwszej zdrady

- Na zmianę

- Wybaczający czasem zdradę

2. Sekcja wyników:

W tej sekcji, jak wyżej wspomnieliśmy, znajdują się w tej aplikacji dwa rodzaje wykresów, a należą do nich:

- Wykres słupkowy, który przedstawia czas spędzony w więzieniu przez obu graczy

- Wykres liniowy, który pokazuje częstotliwość dokonywania zdrady przez każdego gracza w kolejnych rundach.

Logika aplikacji

Aplikacja została zrobiona w języku JavaScript i HTML z użyciem biblioteki Chart.js do wizualizacji wyników, a w celu nadania spójnej estetyki użyliśmy CSS.

1. Wybór strategii:

- Funkcja `getPlayerMove(strategy, state)` zwraca ruch gracza na podstawie wybranej strategii i stanu gry.

U tutaj warto zaznaczyć że status gry (czyli u nas `state`) zawiera informacje o poprzednich ruchach i o wszystkich innych niezbędnych danych, które głównie przydają się do strategii takich jak "oko za oko" czy "do pierwszej zdrady", gdzie wymaga to brania pod uwagę ruchu drugiego potencjalnego więźnia.

2. Symulacja rozgrywek:

Symulacja, tak jak wyżej wspomnieliśmy, wykonuje się przez 1000 rund dla wybranej pary strategii i zaczyna się po kliknięciu przycisku "Rozpocznij grę". Aplikacja sama dokonuje obliczeń i w zależności od tego, jakie wybory zostaną dokonane wykonuje dane obliczenia:

|  |  |
| --- | --- |
| Gdy obaj gracze współpracują | Brak kar |
| Współpraca jednego gracza i zdrada drugiego | Zdradzający unika kary, współpracujący dostaje 10 lat |
| Zdrada obu graczy | Obaj gracze dostają po 3 lata |

3. Prezentacja wyników:

Najprzyjemniejszą dla oka sekcją jest właśnie sekcja wyników, która od razu się odświeża w przypadku wybrania innej strategii, więc od razu można porównać, jak te strategie się od siebie różnią.

Wnioski

Przeprowadzenie Turnieju Axelroda za pomocą tej aplikacji pozwala na właściwie względnie łatwą obserwację efektywności różnych strategii, a oto wnioski, które wyciągnęliśmy na podstawie symulacji:

1. **Zawsze fair (alwaysFairStrategy)**
   * **Opis strategii**: Gracz zawsze współpracuje
   * **Wnioski**: Strategia "Zawsze fair" sprawdza się dobrze w grze z innymi strategiami, które mają tendencję do współpracy (a najlepiej jak drugi gracz też stosuje tą strategię). Jednak jest bardzo podatna na stratę, ponieważ nie ma sytuacji takiej, że dostaje tylko 3 lata, tylko albo nie dostaje nic, albo dostaje 10 lat
2. **Zawsze nie fair (alwaysDefectStrategy)**
   * **Opis strategii**: Gracz zawsze zdradza
   * **Wnioski**: Ta strategia przynosi największe korzyści, gdy gra przeciwko współpracującym strategiom, ale gdy obaj gracze stosują tę strategię, prowadzi to do wielu lat więzienia dla obu stron. Jest efektywna w krótkim okresie, ale długoterminowo nie opłaca się w starciu z innymi strategiami zdradzającymi, ale na te współpracujące już się sprawdza idealnie
3. **Fair na 70% (fair70Strategy)**
   * **Opis strategii**: Gracz współpracuje w 70% przypadków
   * **Wnioski**: Strategia ta stara się znaleźć równowagę między współpracą a zdradą. Jest mniej podatna na więzienie niż "Zawsze fair", ale może nadal ponosić wysokie koszty w starciu z bardziej agresywnymi strategiami
4. **Nie fair na 70% (defect70Strategy)**
   * **Opis strategii**: Gracz zdradza w 70% przypadków
   * **Wnioski**: To jest odpowiednik poprzedniej strategii, tylko że po drugiej stronie. Strategia ta może osiągać korzyści przeciwko współpracującym strategiom, ale również naraża się na wzajemne zdrady, co prowadzi do wysokich kar więzienia
5. **Losowa (randomStrategy)**
   * **Opis strategii**: Gracz losowo wybiera współpracę lub zdradę
   * **Wnioski**: Strategia losowa jest nieprzewidywalna i trudna do przewidzenia przez przeciwników, ale na wykresach widać jednak, że przy 1000 rozgrywek, wykres ze zdradami przypomina linię prostą, a w starciu z drugą taką samą strategią, obie strony dostają podobne kary. Dzięki tej strategii możemy czasami sobie korzyści dzięki swojej nieprzewidywalności, ale brakuje jej spójności i optymalizacji, co skutkuje zmiennymi wynikami
6. **Oko za oko (titForTatStrategy)**
   * **Opis strategii**: Gracz zaczyna od współpracy, a potem naśladuje poprzedni ruch przeciwnika
   * **Wnioski**: Z naszych obserwacji wynika że jest to jedna z najbardziej skutecznych strategii, ponieważ naszą decyzję dokonujemy na podstawie ruchu drugiego gracza, a jako że pierwszą rundę zaczyna zawsze od współpracy, to gdy druga osoba korzysta też z logicznych strategii, minimalizuje swój czas, który może potencjalnie spędzić w więzieniu. Również w przypadku zdrady, odzywa się tym samym, więc ma tym samym dodatkowe zabezpieczenie
7. **Do pierwszej zdrady (cooperateAfterFirstDefectStrategy)**
   * **Opis strategii**: Współpracuje zawsze, chyba że przeciwnik pierwszy raz zdradzi, wtedy zawsze zdradza
   * **Wnioski**: Podczas próbowania tej metody przeciwko wszystkim innym strategią, właściwie tylko przeciwko strategii „zawsze nie fair”, czas w więzieniu miał większy, o dokładnie 10 lat. W innych przypadkach było widać zdecydowaną przewagę tej strategii ponad inne.
8. **Na zmianę (alternateStrategy)**
   * **Opis strategii**: Naprzemiennie współpracuje i zdradza
   * **Wnioski**: Strategia ta jest bardzo przewidywalna i może być łatwo eksploatowana przez bardziej złożone strategie. Jest mało efektywna przeciwko strategiom, w których decyzje są podejmowane na podstawie wyborów drugiego gracza
9. **Wybaczający czasem zdradę (generousTitForTatStrategy)**
   * **Opis strategii**: Współpracuje, ale czasami wybacza zdradę
   * **Wnioski**: Strategia już nie jest tak przewidywalna, ponieważ element losowości, który pozwala na częściowe wybaczanie zdrad. Może być efektywna w dłuższym okresie, bo zachęca przeciwnika do współpracy, jednak niestety może być nadal średnia przeciwko bardziej agresywnym strategiom

Jak widać każda z powyższych strategii ma swoje mocne i słabe strony. W zależności od wyboru strategii przeciwnika, różne strategie mogą być bardziej lub mniej efektywne. Też naszym zdaniem użycie 2 rodzajów wykresów sprawia, że łatwiej jest przeanalizować jak różne strategie działają względem siebie.