Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Fizyki i Informatyki Stosowanej

KATEDRA INFORATYKI STOSOWANEJ I FIZYKI KOMPUTEROWEJ

PRACA INŻYNIERSKA

ERNEST JĘCZMIONEK

SYMULACJE EWOLUCJI KOALICJI MIESZANYCH

PROMOTOR:
prof. dr hab. Krzysztof Kułakowski

Kraków 2017

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

. . . . . .

PODPIS

AGH

University of Science and Technology in Krakow

Faculty of Physics and Applied Computer Science

Department of Applied Informatics and Computational Physics

Bachelor of Science Thesis

Ernest Jęczmionek

Simulations of the evolution of mixed coalitions

Supervisor:
Professor Krzysztof Kułakowski

Krakow 2017

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wprowadzenie

Teoria gier wielu osobom kojarzy się z opisem gier towarzyskich między dwojgiem graczy, lecz takie rozgrywki to rzadkość w naszym zróżnicowanym świecie, gdzie zwykle w grę ekonomiczną, społeczną czy polityczną angażuje się wielu uczestników. W niniejszej pracy zostaną umówione gry wieloosobowe o niepełnej informacji. W tym typie gier ważnym elementem strategii jest odpowiedni wybór koalicjantów. Oczywiście nie jest możliwe aby uwzględnić wszystkie czynniki mogące mieć wpływ na grę, ale zostaną przeanalizowane dwa równania, modelujące grę ewolucyjną, które mogłyby sterować graczami oraz dla jednego z nich przeprowadzona będzie analiza stabilności. Modelami gry użytymi w niniejszej pracy będzie gra 3-osobowa oraz gra wieloosobowa, w której gracze będą ustawieni w okręgu. Symulacje partii w przypadku gier 3-osobowych będą obrazowane jako trajektorie w trójwymiarowej przestrzeni prawdopodobieństw, natomiast dla gier wieloosobowych jako funkcje prawdopodobieństwa od czasu dla poszczególnych graczy.

2. Opis teoretyczny

2.1. Gra

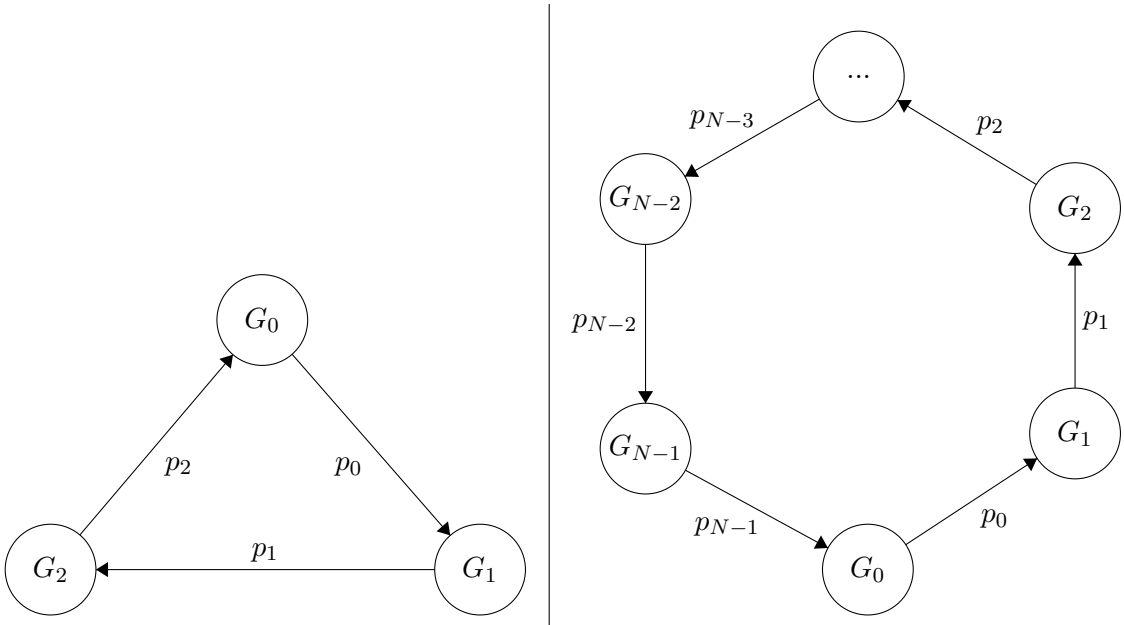
Niniejsza praca skupia się na grach wieloosobowych, których szczególnym przypadkiem jest gra 3-osobowa [8]. Model gry 3-osobowej pokazuje, że decyzje jednego z zawodników mają bezpośredni wpływ na zachowanie sąsiadów. W modelu gry N-osobowej decyzje graczy będą wpływać nie tylko na najbliższych sąsiadów, ale pośrednio także na decyzje innych graczy. Omawiane tutaj gry są grami o niepełnej informacji, w tej pracy oznacza to grę, w której nie wszyscy uczestnicy znają prawdopodobieństwo określonych decyzji przeciwników. Prawdopodobieństwo gry przeciwników będzie szacowane na podstawie obserwacji historii ich zagrań. Opisywane gry będą także grami ewolucyjnymi, gdzie gracze mają możliwość przewidywania i uczenia się na podstawie zachowań innych graczy.

2.2. Model gry

Jak wcześniej wspomniano będą użyte dwa modele gier. Pierwszym z nich będzie model gry 3-osobowej (Rys. 2.1a). Graczy nazwiemy odpowiednio G_0, G_1, G_2 (gracz zerowy, pierwszy, drugi). Zachowanie każdego z nich jest opisane prawdopodobieństwem próby wejścia w koalicję z graczem o wyższym indeksie. To prawdopodobieństwo oznaczone jest jako p_i (i jest indeksem gracza), przy czym dla G_2 gracz o wyższym indeksie to G_0. Prawdopodobieństwo zagrania, w celu nawiązania koalicji z graczem o niższym indeksie to $1 - p_i$. Analogicznie dla G_0 gracz o niższym indeksie to G_2. Ponieważ omawiana jest gra o niepełnej informacji żaden z zawodników nie ma dostępu do prawdopodobieństw innych graczy, lecz każdy ma dostęp do statystyki gry na którą składa się:

- l_p liczba rozegranych w grze partii
- n_i ilość zagrań G_i w celu zawiązania sojuszu z G_{i+1}. Ilość zagrań G_i aby nawiązać sojusz z G_{i-1} wyraża się jako $l_p - n_i$

Model gry N-osobowej (Rys. 2.1b) [2] będzie opisany takimi samymi danymi jak model gry 3-osobowej. Różnicą między nimi będzie ustawienie graczy w okręgu, co sprawia że skutki decyzji wykraczają poza najbliższych sąsiadów, dla gier o liczbie graczy większej niż 3. Gra 3-osobowa jest szczególnym przypadkiem gry N-osobowej, w którym konsekwencje zachowań jednego z naszych sąsiadów bezpośrednio wpływają na drugiego.



(a) 3-osobowej (b) N-osobowej

Rysunek 2.1: Modele gry. G w okręgu - gracz, strzałka z p - prawdopodobieństwo zagrania w celu nawiązania koalicji z graczem, na którego wskazuje strzałka

2.3. Równania standardowe

Pomysł na równania standardowe został zaczerpnięty ze wzoru na siłę w polu,

$$\overrightarrow{F}(\overrightarrow{r}) = -\nabla E_p(\overrightarrow{r}) \tag{2.1}$$

z tym że żądana jest maksymalizacja, a nie minimalizacja odpowiednika energii potencjalnej E_p. Z tego powodu należy zmienić znak. Siłę rozpisujemy jako:

$$F = \frac{\delta E_p}{\delta x} = m\frac{\delta^2 x}{\delta t^2} + \underbrace{\gamma\frac{\delta x}{\delta t}}_{\text{Siła tarcia}}$$

Zakładając że będziemy mieli do czynienia z ruchem przetłumionym, ze względu na duży współczynnik tarcia, w efekcie prowadzi do zależności:

$$\frac{\delta x}{\delta t} \sim \frac{\delta E_p}{\delta x} \tag{2.2}$$

Przemieszczenie zostanie zastąpione prawdopodobieństwem. W tym typie równań celem będzie maksymalizacja zysku(zyskiem jest wypłata) w czasie. Wyprowadzenie będzie przeprowadzone dla G_0, przy założeniu gry o pełnej informacji. Pozostałe dwa równania można wyprowadzić analogicznie. Wypłata dla G_0 dana jest przez:

$$W_0 = \underbrace{p_0(1-p_1)}_{\text{wypłata sojuszu } G_0 \text{ z } G_1} + \overbrace{(1-p_0)p_2}^{\text{wypłata sojuszu } G_0 \text{ z } G_2} \tag{2.3}$$

Do zależności 2.2 podstawiamy prawdopodobieństwa.

$$\frac{\delta p_0(t)}{\delta t} = \frac{\delta W_0(t)}{\delta p_0}$$

Iloraz różnicowy na zmianę prawdopodobieństwa od czasu daje:

$$\frac{p_0(t+\Delta t) - p_0(t)}{\Delta t} = 1 - p_1 - p_2 \tag{2.4}$$

W używanej dalej notacji:

$$\Delta p_0 = \Delta t(1 - p_1 - p_2) \tag{2.5}$$

W dalszej części będzie stosowane oznaczenie $\alpha = \Delta t$. Równanie wygląda obiecująco, gdyż człon $1-p_1$ jest prawdopodobieństwem zagrania G_1 aby zawiązać sojusz z G_0. Podobnie z prowadzi do sojuszu G_2 z G_0, który osłabia koalicję G_0 z G_1. Dlatego znajduje się ze znakiem minus w Δp_0.

W przypadku gry o niepełnej informacji należy zmodyfikować równanie 2.5. Opis użytych parametrów można znaleźć w podrozdziale 2.2.

$$\Delta p_i = \alpha \cdot (1 - \frac{n_{i+1}}{l_p} - \frac{n_{i-1}}{l_p}) \tag{2.6}$$

2.4. Równania replikatorów

Model dynamiki replikatorów jest najbardziej znanym różniczkowym modelem teorii gier ewolucyjnych [7][6], przez co może stanowić dobry wybór do sterowania zachowaniem graczy. Przyjmijmy prawdopodobieństwa $x = p_0$, $y = p_1$ oraz $z = p_2$ pamiętając, że prawdopodobieństwa przeciwników w symulacji są szacowane. Podstawowy wzór równania replikatorów wygląda następująco:

$$\dot{x} = x \cdot (W_x - \overline{W}) \tag{2.7}$$

gdzie $\dot{x}$ jest tempem zmiany udziału strategii x, W_x jest średnią wypłatą dla strategii x, natomiast $\overline{W}$ jest średnią wypłatą co daje:

$$\dot{x} = x \cdot (\overbrace{(1-y)}^{W_x} - \overbrace{(x(1-y) + (1-x)z)}^{\overline{W}})$$

$$\Downarrow$$

$$\dot{x} = x \cdot (1-x) \cdot (1-y-z)$$

Iloraz różnicowy zmiany prawdopodobieństwa od czasu ponownie wyznaczy Δp_0:

$$\dot{x}(t) = \frac{p_0(t+\Delta t) - p_0(t)}{\Delta t}$$

$$\dot{x}(t)\Delta t = p_0(t+\Delta t) - p_0(t) = \Delta p_0$$

$$\Delta p_0 = \dot{x}(t)\Delta t$$

W dalszej części tej pracy $\alpha = \Delta t$. Co dla gry 3-osobowej generuje równania:

$$\begin{aligned}
\Delta p_0 &= \alpha p_0 \cdot (1 - p_0) \cdot (1 - \frac{n_1}{l_p} - \frac{n_2}{l_p}) \\
\Delta p_1 &= \alpha p_1 \cdot (1 - p_1) \cdot (1 - \frac{n_2}{l_p} - \frac{n_0}{l_p}) \\
\Delta p_2 &= \alpha p_2 \cdot (1 - p_2) \cdot (1 - \frac{n_0}{l_p} - \frac{n_1}{l_p})
\end{aligned} \tag{2.8}$$

2.5. Ograniczenie prawdopodobieństwa

Wszystkie zmiany prawdopodobieństwa muszą zostać poddane funkcji ograniczającej.

$$p_i = ogr(p_i + \Delta p_i) \tag{2.9}$$

W przeciwnym razie równania z poprzednich podrozdziałów mogą wyjść poza przedział $< 0, 1 >$. Każde nowo obliczone prawdopodobieństwo podawane jest jako parametr do funkcji ogr, a dopiero jej rezultat jest przypisywany poszczególnym prawdopodobieństwom graczy. Funkcja używana aby zapewnić pozostanie prawdopodobieństwa w dziedzinie wygląda następująco:

$$ogr(p_i) = \begin{cases} 1 & \text{jeżeli } p_i > 1 \\ p_i & \text{jeżeli } 1 \geq p_i \geq 0 \\ 0 & \text{jeżeli } p_i < 0 \end{cases}$$

Wyjaśnienia wymaga wartość α, która została przyjęta jako 0.1. Jest to wartość przyjęta arbitralnie aby spowolnić rozgrywkę oraz umożliwić łatwiejszą zmianę koalicji.

2.6. Ograniczenie prawdopodobieństwa - przykłady

Równania standardowe Przeanalizowane zostaną równania standardowe dla gry 3-osobowej o prawdopodobieństwie początkowym $\frac{1}{2}$ i $\alpha = 1$. Gdzie N oznacza zagranie G_i mające na celu zawiązanie sojuszu z G_{i+1}, a P ilość zagranieG_i mające na celu zawiązanie sojuszu z G_{i-1}.

$$\begin{array}{cc}
G_0 = N, G_1 = P, G_2 = N & G_0 = N, G_1 = P, G_2 = P \\
\begin{cases} \Delta p_0 = (1 - 0 - 1) = 0 & p_0 = \frac{1}{2} \\ \Delta p_1 = (1 - 1 - 1) = -1 & p_1 = 0 \\ \Delta p_2 = (1 - 1 - 0) = 0 & p_2 = \frac{1}{2} \end{cases} & \begin{cases} \Delta p_0 = (1 - 0 - \frac{1}{2}) = 0 & p_0 = \frac{3}{4} \\ \Delta p_1 = (1 - \frac{1}{2} - 1) = -\frac{1}{2} & p_1 = 0 \\ \Delta p_2 = (1 - 0 - \frac{1}{2}) = \frac{1}{2} & p_2 = \frac{3}{4} \end{cases}
\end{array}$$

Schemat przedstawia dwie osobne tury: po lewej pierwszą, a po prawej drugą. Parametr $\alpha = 1$ powoduje szybkie zawiązanie mocnych koalicji, których przerwanie staje się mało prawdopodobne, co może praktycznie uniemożliwiać jakiekolwiek zmiany sojuszy.

Równania replikatorów Wartym rozważenia jest czy w równaniach replikatorów potrzebna będzie $\alpha < 1$, skoro posiadają one człon postaci $x(1-x)$ gasnący na krańcach przedziału prawdopodobieństwa.

Założenia zostały oparte na poprzednim przykładzie.

$$G_0 = N, G_1 = P, G_2 = N$$

$$\begin{cases} \Delta p_0 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - 1) = 0 & p_0 = \frac{1}{2} \\ \Delta p_1 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 1 - 1) = 0 & p_1 = \frac{1}{4} \\ \Delta p_2 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - 1) = 0 & p_2 = \frac{1}{2} \end{cases}$$

$$G_0 = N, G_1 = P, G_2 = P$$

$$\begin{cases} \Delta p_0 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - \frac{1}{2}) = \frac{1}{8} & p_0 = \frac{5}{8} \\ \Delta p_1 = \frac{1}{4} \cdot (1 - \frac{1}{4}) \cdot (1 - \frac{1}{2} - 1) = -\frac{1}{32} & p_1 = \frac{7}{32} \\ \Delta p_2 = \frac{1}{2} \cdot (1 - \frac{1}{2}) \cdot (1 - 0 - \frac{1}{2}) = \frac{1}{8} & p_2 = \frac{5}{8} \end{cases}$$

Jak widać, w wybranym przykładzie najszybsza zmiana zachodzi dla pierwszego gracza, który traci 25% z prawdopodobieństwa sojuszu z graczem o wyższym indeksie. W kolejnej partii nie widzimy już tak dużych zmian. Należy odpowiedzieć na pytanie czy jest to na tyle dużo, aby zastosować α taką jak w równaniach standardowych.

2.7. Rozwiązanie stacjonarne równań standardowych

Celem tego podrozdziału jest znalezienie rozwiązań stacjonarnych dla równań standardowych, czyli sytuacji w których układ dla długiego czasu dąży do punktu stałego. Do ustalenia punktów stałych potrzebny jest zanik dynamiki $\Delta p_i = 0$. Krok czasowy zostanie pominięty, gdyż nie daje wkładu do obliczeń. Należy rozwiązać układ równań, gdzie przyjęte jest $x = p_0$, $y = p_1$, $z = p_2$:

$$\begin{cases} 1 - y - z = 0 \\ 1 - x - z = 0 \\ 1 - x - y = 0 \end{cases} \Rightarrow p_0 = p_1 = p_2 = \frac{1}{2} \tag{2.10}$$

Z czego wynika że gra startująca w punkcie $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ nie powinna z niego wyjść. Byłoby tak gdyby prawdopodobieństwa użyte w symulacji były faktycznymi prawdopodobieństwami p_i, są one natomiast jedynie obserwacją zachowania pozostałych graczy. Są one dane jako $\frac{n_j}{l_p}$. Użycie prawdopodobieństw obserwowanych umożliwia opuszczenie punktu stałego.

2.8. Stabilność równań replikatorów

Stabilność równań replikatorów [5] wymaga dążenia do stałych wartości, gdy $t \Rightarrow \infty$. Do analizy zostanie przyjęty układ równań:

$$\begin{aligned} \dot{x} &= f(x, y, z) = \Delta p_0 = x(1 - x)(1 - y - z) \\ \dot{y} &= g(x, y, z) = \Delta p_1 = y(1 - y)(1 - x - z) \\ \dot{z} &= h(x, y, z) = \Delta p_2 = z(1 - z)(1 - x - y) \end{aligned} \tag{2.11}$$

Punkty stałe są to punkty, z których układ nie wyjdzie bez zewnętrznego czynnika. Do znalezienia punktów stałych wymagane jest znikanie pochodnych. Prowadzi to do układu równań.

$$\dot{x} = 0, \quad \dot{y} = 0, \quad \dot{z} = 0 \tag{2.12}$$

Z powyższych równań otrzymujemy punkty stałe (x_i^*, y_i^*, z_i^*), które zostały ponumerowane przy pomocy indeksu i.

$$(x_i^*, y_i^*, z_i^*) = \begin{cases} (0,0,0) & i = 0 \\ (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) & i = 1 \\ (1,1,1) & i = 2 \\ P_3\{0,1,\xi\} & i = 3, \text{ permutacja gdzie } \xi \in < 0,1 > \end{cases} \tag{2.13}$$

Znajomość punktów stałych pozwala na pytanie co do ich stabilności. Układ znajdujący się w pobliżu punktu stabilnego powinien zawsze do niego dążyć. Podczas analizy stabilności będą uwzględnione wszystkie powyższe punkty stałe, przy czym w ostatniej sytuacji będzie uwzględniony tylko przypadek $(0,1,\xi)$, ze względu na ich symetrię.

Stosując metodę linearyzacji tworzymy funkcje, które powinny osiągać 0 w punktach stabilnych.

$$\begin{aligned} X(t) &= x(t) - x^* \\ Y(t) &= y(t) - y^* \\ Z(t) &= z(t) - z^* \end{aligned} \tag{2.14}$$

Zróżniczkowanie funkcji 2.14 i rozwinięcie w szereg Taylora do pierwszej pochodnej dają:

$$\begin{aligned} \dot{X}(t) = \dot{x}(t) - \dot{x}^* = \dot{x}(t) &= f(x(t), y(t), z(t)) = f(x^* + X(t), y^* + Y(t), z^* + Z(t)) = \\ &= f(x^*, y^*, z^*) + \frac{\delta f(x^*, y^*, z^*)}{\delta x} X + \frac{\delta f(x^*, y^*, z^*)}{\delta y} Y + \frac{\delta f(x^*, y^*, z^*)}{\delta z} Z \\ \dot{Y}(t) = \dot{y}(t) - \dot{y}^* = \dot{y}(t) &= g(x(t), y(t), z(t)) = g(x^* + X(t), y^* + Y(t), z^* + Z(t)) = \\ &= g(x^*, y^*, z^*) + \frac{\delta g(x^*, y^*, z^*)}{\delta x} X + \frac{\delta g(x^*, y^*, z^*)}{\delta y} Y + \frac{\delta f(x^*, y^*, z^*)}{\delta z} Z \\ \dot{Z}(t) = \dot{z}(t) - \dot{z}^* = \dot{z}(t) &= h(x(t), y(t), z(t)) = h(x^* + X(t), y^* + Y(t), z^* + Z(t)) = \\ &= f(x^*, y^*, z^*) + \frac{\delta h(x^*, y^*, z^*)}{\delta x} X + \frac{\delta h(x^*, y^*, z^*)}{\delta y} Y + \frac{\delta h(x^*, y^*, z^*)}{\delta z} Z \end{aligned} \tag{2.15}$$

Wiadomo że w punktach stacjonarnych $\dot{x} = \dot{y} = \dot{z} = 0$, na podstawie czego można przyjąć, że $f(x^*, y^*, z^*) = g(x^*, y^*, z^*) = h(x^*, y^*, z^*) = 0$. Pierwszy i ostatni człon powyższych równań dają układ równań.

$$\begin{aligned} \dot{X} &= a_{11} X + a_{12} Y + a_{13} Z \\ \dot{Y} &= a_{21} X + a_{22} Y + a_{23} Z \\ \dot{Z} &= a_{31} X + a_{32} Y + a_{33} Z \end{aligned} \tag{2.16}$$

Macierz współczynników można zapisać jako macierz Jacobiego.

$$J_i = \left[\begin{array}{ccc} \frac{\delta\dot{x}}{\delta x} & \frac{\delta\dot{x}}{\delta y} & \frac{\delta\dot{x}}{\delta z} \\ \frac{\delta\dot{y}}{\delta x} & \frac{\delta\dot{y}}{\delta y} & \frac{\delta\dot{y}}{\delta y} \\ \frac{\delta\dot{z}}{\delta x} & \frac{\delta\dot{z}}{\delta y} & \frac{\delta\dot{z}}{\delta x} \end{array}\right] \Bigg| \begin{array}{l} x = x_i^* \\ y = y_i^* \\ z = z_i^* \end{array} = \left[\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}\right] \text{ dla } i\text{-tego punktu stałego}$$

Pochodne cząstkowe:

$$\begin{array}{ll} \frac{\delta\dot{x}}{\delta x} = 1 - y - z - 2x + 2xy + 2xz & \frac{\delta\dot{y}}{\delta x} = y^2 - y \\ \frac{\delta\dot{x}}{\delta y} = x^2 - x & \frac{\delta\dot{y}}{\delta y} = 1 - x - z - 2y + 2xy + 2yz \\ \frac{\delta\dot{x}}{\delta z} = x^2 - x & \frac{\delta\dot{y}}{\delta z} = y^2 - y \end{array}$$

$$\begin{array}{l} \frac{\delta\dot{z}}{\delta x} = z^2 - z \\ \frac{\delta\dot{z}}{\delta y} = z^2 - z \\ \frac{\delta\dot{z}}{\delta z} = 1 - x - y - 2z + 2xz + 2yz \end{array}$$

Do rozwiązania jest układ równań dany w zapisie macierzowym.

$$\left[\begin{array}{c} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{array}\right] = \left[\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}\right] \cdot \left[\begin{array}{c} X \\ Y \\ Z \end{array}\right] \tag{2.17}$$

W tym celu wyznaczymy wartości własne.

$$J_{i,\lambda} = J_i - \lambda I = \left|\begin{array}{ccc} \frac{\delta\dot{x}}{\delta x} - \lambda & \frac{\delta\dot{x}}{\delta y} & \frac{\delta\dot{x}}{\delta z} \\ \frac{\delta\dot{y}}{\delta x} & \frac{\delta\dot{y}}{\delta y} - \lambda & \frac{\delta\dot{y}}{\delta z} \\ \frac{\delta\dot{z}}{\delta x} & \frac{\delta\dot{z}}{\delta y} & \frac{\delta\dot{z}}{\delta z} - \lambda \end{array}\right| = 0 \tag{2.18}$$

$$\text{dla punktu stałego } \begin{array}{c} (0,0,0) \\ (\frac{1}{2},\frac{1}{2},\frac{1}{2}) \\ (1,1,1) \\ (0,1,\xi) \end{array} \text{ wartości własne to } \begin{array}{c} (1,1,1) \\ (-\frac{1}{2},\frac{1}{4},\frac{1}{4}) \\ (1,1,1) \\ (0,-z,z-1) \end{array}$$

W rozwiązaniu ogólnym układu 2.17 dla punktu stałego (x_i^*, y_i^*, z_i^*) wartości własne odpowiadające zmiennym x, y, z są oznaczone odpowiednio α_i, β_i, γ_i. Wektorami własnymi dla poszczególnych wartości własnych są V_i, natomiast C_i są współczynnikami rozwiązania jednorodnego układu równań różniczkowych.

$$\left[\begin{array}{c} X(t) \\ Y(t) \\ Z(t) \end{array}\right] = C_{\alpha_i} V_{\alpha_i} e^{\alpha_i t} + C_{\beta_i} V_{\beta_i} e^{\beta_i t} + C_{\gamma_i} V_{\gamma_i} e^{\gamma_i t} \tag{2.19}$$

Aby punkt był stabilny funkcje muszą dążyć w czasie do zera.

$$\lim_{t\to\infty} X(t) = 0 \quad \lim_{t\to\infty} Y(t) = 0 \quad \lim_{t\to\infty} Z(t) = 0 \tag{2.20}$$

Aby eksponenta malała musi posiadać ujemny wykładnik, z czego wynika że warunkiem stabilności jest:

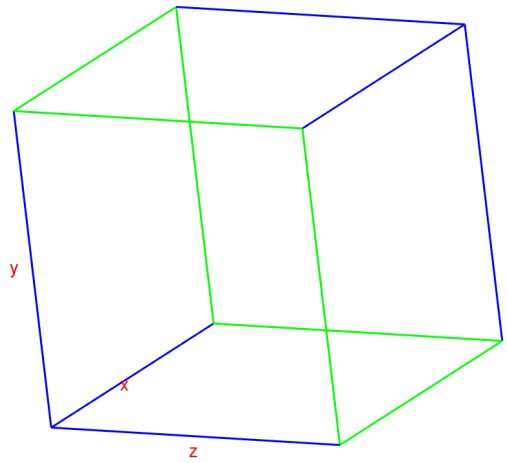
$$Re(\lambda_a) < 0 \wedge Re(\lambda_b) < 0 \wedge Re(\lambda_c) < 0 \tag{2.21}$$

Warunek 2.21 eliminuje wszystkie punkty stałe poza ostatnim. Pokazuje to że punkty o kombinacji $(0, 1, \xi)$ są marginalnie stabilne. Analiza nie wskazuje że są stabilne, lecz temu też nie przeczy.

Ponieważ metoda linearyzacji opisała punkty $(0, 1, \xi)$ jako marginalnie stabilne, spróbujemy przesądzić co do ich stabilności na podstawie analizy równania $\dot{z}$ - układ 2.11. Przyjmując sytuację gdy p_i dwóch z graczy przyjmie 0 lub 1. Przeanalizowane zostaną wszystkie przypadki pomimo, że dla omawianych punktów tylko wiersz drugi i trzeci ma znaczenie. Celem poniższej analizy jest sprawdzenie czy wspomniane punkty są stabilne po ich osiągnięciu. W pierwszym wierszu tabelki 2.1 widać, że G_0 dąży do zawarcia sojuszu z G_2, natomiast G_1 gra aby osiągnąć sojusz z G_0. W tej sytuacji prawdopodobieństwo z, że gracz G_2 zawiąże koalicję z G_0 będzie rosło. W kolejnych dwóch wierszach są stany ustalone. W wierszu drugim zmiana $\dot{z}$ nie ma znaczenia gdyż gracze G_0 oraz G_1 są w wyłącznej koalicji z G_2. Wiersz trzeci pokazuje przeciwną sytuację, w której jakakolwiek zmiana $\dot{z}$ nie wniesie nic do gry ze względu na trwały sojusz między G_0 a G_1. Te dwie sytuacje oznaczone są zielonymi krawędziami na rysunku 2.2. Ostatni wiersz pokazuje przypadek symetryczny do pierwszego, z tym że jedynym możliwym sojusznikiem dla G_2 jest G_1. Wiersz pierwszy i ostatni przedstawia sytuację oznaczoną niebieską krawędzią na rysunku 2.2, są to sytuacje w których zawiązanie sojuszu miałoby nastąpić przy parach prawdopodobieństw dwóch graczy $(0, 0)$ lub $(1, 1)$, co jest nierealne. Oczywiście omawiane teraz przykłady do osiągnięcia stabilności wymagałyby braku zmian decyzji pozostałych graczy. Musieli by oni znać realne prawdopodobieństwo przeciwników.

Tablica 2.1: Stabilność na krawędzi sześcianu

x	y	1-x-y	$\dot{z}$
0	0	1	$z \cdot (1 - z)$
0	1	0	0
1	0	0	0
1	1	-1	$-z \cdot (1 - z)$



Rysunek 2.2: Sześcian prawdopodobieństwa. Niebieskie krawędzie - niemożliwe koalicje. Zielone krawędzie - linie punktów stałych.

3. Implementacja symulacji

3.1. Opis problemu

Symulacja gry 3-osobowej będzie przedstawiona jako sześcian. Każdy z jego wierzchołków będzie miał współrzędną składającą się z kombinacji 0 i 1, co ma obrazować prawdopodobieństwo gracza. Odpowiednio prawdopodobieństwo gracza pierwszego przedstawia oś x, drugiego oś y, trzeciego oś z. Każda funkcja reprezentuje jedną instancję gry 3-osobowej. Rysowane funkcje będą przedstawiały prawdopodobieństwa graczy (p_i) wynikające z używanych równań i zachowania przeciwników.

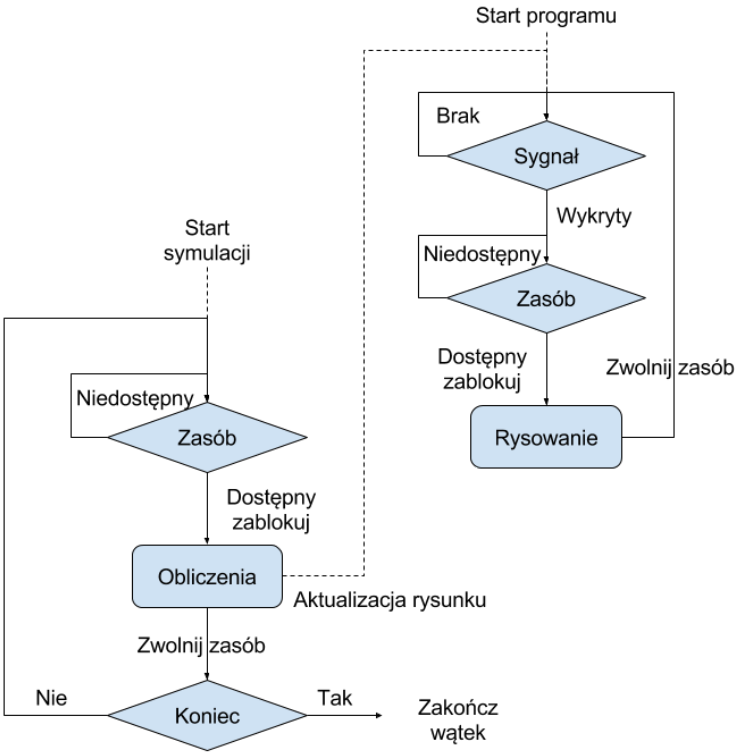
Symulacja gry N-osobowej, w której gracze ustawieni są w okręgu będzie obrazowana jako funkcje prawdopodobieństwa każdego gracza od numeru partii.

3.2. Środowisko QT

Do stworzenia programu wykorzystano QT Creator IDE z kilku powodów, które zaraz zostaną rozwinięte. Najważniejszą cechą środowiska jest udostępnienie go na kilku rodzajach licencji, z czego użyta tu została licencji LGPL. Pozwala ona na darmowe użytkowanie i modyfikowanie plików źródłowych, lecz udostępnienie własnego programu musi odbyć się na tej samej licencji. Kolejnym ważnym elementem jest multiplatformowość pozwalająca w łatwy sposób przenosić kod programu między systemami operacyjnymi, o ile nie zostały użyte biblioteki dostępne tylko na jeden z systemów. Kolejną z zalet jest łatwy i intuicyjny interfejs tworzenia graficznego interfejsu użytkownika, osoba mająca wcześniej styczność z na przykład biblioteką Swing Java'y nie powinna mieć problemu z zaadaptowaniem się do formularza QT Creatora. Używanie nowoczesnego języka C++ (w pracy użyto wersji 14) nie sprawia problemów. Przed użyciem klas z bibliotek standardowych C++ warto sprawdzić czy biblioteka QT nie zawiera specjalnych klas opakowujących je. Przykładem może być `pthread` i `QThread` [3].

3.3. Schemat programu

Rysunek 3.1 przedstawia poglądowy schemat działania programu[4]. Dzieląc rysunek na lewą i prawą część możemy wyodrębnić dwa wątki. Po lewej wątek odpowiadający za wykonanie symulacji, po prawej wątek odpowiadający za rysowanie. Określeniem *Zasób* są opisane współdzielone dane (opisane w podrozdziale 3.5) między wątkami, którymi jest historia symulacji.



Rysunek 3.1: Schemat programu

Podczas startu programu zostaje narysowany sześcian, po czym program przechodzi w stan nasłuchiwania na sygnały. Przykładowymi sygnałami mogą być zmiana rozmiaru okna aplikacji, bądź obrót figury. Po wykryciu takowego, następują próby zablokowania dostępu do zasobu dopóki nie zostanie to wykonane (zasób może być zablokowany przez inny wątek). Następnie na podstawie historii symulacji i/lub sygnału dokonywane jest nowe rysowanie. Po czym zasób zostaje zwolniony, a główny wątek (odpowiedzialny za rysowanie) wraca do stanu nasłuchiwania. Podczas startu symulacji (spowodowanej naciśnięciem przycisku `RUN`) identycznie jak poprzednio musi być uzyskany dostęp do zasobów. Dopiero po tym możliwe jest przeprowadzenie symulacji jednej partii dla każdej instancji gry. Następnie wyniki zostają zapisane do zasobów, po czym zasoby zostają zwolnione. Równocześnie do wątku głównego wysyłany jest sygnał mówiący o aktualizacji historii symulacji. Jeśli była to ostatnia planowana partia wątek symulacyjny kończy swoje działanie.

3.4. Sygnały, sloty i mutex

Komunikacja między klasami dziedziczącymi po `QObject` odbywa się przy pomocy połączonych ze sobą sygnałów i slotów [3]. Można na nie patrzeć jako funkcje (sloty) oraz wskaźniki do funkcji (sygnały). Sygnał może trafiać do wielu slotów oraz do slotu może docierać wiele sygnałów. Ważne jest aby typy parametrów w deklaracji sygnału pokrywały się z typami parametrów deklaracji slotu. W przypadku gdy sygnał będzie miał zbyt dużo parametrów dla slotu, nadmiarowe parametry zostaną pominięte.

W celu uniknięcia kolizji zapis-odczyt na współdzielonych zasobach została użyta klasa `mutex` [1]. Jest to semafor, nie chroni on danych przed dostępem z wielu źródeł, a jedynie informuje czy zasoby są obecnie używane czy nie. Nie ma niebezpieczeństwa wykonania operacji kolizyjnych, gdyż zasoby są wykorzystywane jedynie przez wewnętrzne struktury programu, które nie wykonują operacji na wspólnych zasobach jeśli semafor jest zablokowany.

3.5. Kod

W tym podrozdziale będą opisane kluczowe fragmenty kodu, niezbędne do zrozumienia implementacji programu. Współdzielonymi zasobami są:

```
1 template<typename T> using tup3= tuple<T,T,T>;
2 vector<vector<tup3<double>>> beginsp; //wektor historii punktów począ
    tkowych odcinków
3 vector<vector<tup3<double>>> endsp; //wektor historii punktów koń
    cowych odcinków
4 vector<tup3<int>>colorsp; //kolory funkcji
5 mutex points;
```

Interfejs klasy `Game` Cała symulacja przeprowadzana jest w klasie `Game`, dlatego zostanie teraz omówiona. Reprezentuje ona jedną instancję gry 3-osobowej. Parametrem konstruktora jest indeks tablicy `decision_funs`, której element zostaje przypisany do `decision`. `function` jest wrapperem biblioteki standardowej dla dowolnej funkcji, wyrażenia lambda, wyrażenia bind, funkcjonału lub wskaźników do funkcji. W poniższym kodzie `function` jest parametryzowany bezargumentowym, bezwynikowym zachowaniem. Funkcja `next` przeprowadza rozgrywkę jednej partii. Jej implementacja zostanie przedstawiona później. Zmienna `current` odpowiada $liczba_{partii}$, natomiast `nr[i]` odpowiada $nast_i$ (opisano w podrozdziale 2.2). Funkcja `checker` jest odpowiednikiem funkcji ogr (opisano w podrozdziale 2.5). Tablica `decision_funs` zawiera funkcje lambda mające dostęp do prywatnych zmiennych klasy `Game`. W tablicach `result` znajdują się obliczone Δp_i, które następnie poddawane są funkcji ograniczającej.

```
1 class Game{
2 public:
3     Game(int f);
4     tup3<double> next();
5     int getCurrent(); //zwraca numer bieżącej partii
6     tup3<double> prelast(); //zwraca pre
7 private:
8     array<double,3> p; //prawdopodobieństwa
9     tup3<double> pre; //ostatnia krotka prawdopodobieństw na
        podstawie, których było dokonane losowanie sojuszników
```

```
10      int current; //numer bieżącej partii
11      array<int,3> nr;
12      double checker(double r);
13      function<void()> decision; //dokonuje modyfikacji prawdopodobień
            stw
14      array<function<void()>,2>decision_funs={{
15        [this](){//standardowe
16          array<double,3> result= {
17              0.1*( 1 - static_cast<double>(nr[1])/current -
                    static_cast<double>(nr[2])/current),
18              0.1*( 1 - static_cast<double>(nr[2])/current -
                    static_cast<double>(nr[0])/current ),
19              0.1*( 1 - static_cast<double>(nr[0])/current -
                    static_cast<double>(nr[1])/current )
20          };
21          for(int i=0; i<3; i++)
22              p[i]= checker(p[i]+result[i]);
23        },
24        [this](){//replikatorów
25          array<double,3> result= {
26              0.1*(p[0]*(1-p[0])*(1-static_cast<double>(nr[1])/current-
                    static_cast<double>(nr[2])/current)),
27              0.1*(p[1]*(1-p[1])*(1-static_cast<double>(nr[0])/current-
                    static_cast<double>(nr[2])/current)),
28              0.1*(p[2]*(1-p[2])*(1-static_cast<double>(nr[0])/current-
                    static_cast<double>(nr[1])/current))
29          };
30          for(int i=0; i<3; i++)
31              p[i]= checker(p[i]+result[i]);
32        }
33      }};
34  };
```

Implementacja `Game::next` Tablica `choices` zawiera losowe wartości dla każdego z zawodników, na podstawie których dokonywane jest losowanie sojusznika.

```
1  tup3<double> Game::next(){
2      pre= make_tuple(p[0],p[1],p[2]);
3      current++;
```

```
4       array<double,3> choices= {static_cast<double>(rand())/RAND_MAX,
            static_cast<double>(rand())/RAND_MAX, static_cast<double>(rand
            ())/RAND_MAX};
5       for(int i=0; i<3; i++)
6           if(choices[i]<p[i])
7               nr[i]++;
8       decision();
9       return make_tuple(p[0],p[1],p[2]);
10  }
```

Implementacja wątku symulującego `QtConcurrent::run` jest funkcją uruchamiającą w osobnym wątku funkcję podaną jako parametr [3]. Zwraca ona obiekt `QFuture`, który nie może przerwać lub wstrzymać wątku. Poprzez `beginsp.resize(p)` tworzony jest wektor posiadający p pustych elementów(wektorów krotek), ponieważ już w pierwszej partii będziemy potrzebować osobnego wektora dla każdej instancji gry. Natomiast `beginsp[i].reserve(g)` rezerwuje pamięć dla g partii, gdyby ta funkcja nie została użyta prawdopodobnie wystąpiłaby realokacja pamięci, w celu zapewnienia jej ciągłości. Przekładałoby się to na spowolnienie działania programu. Napisałem prawdopodobnie, ponieważ standard języka nie precyzuje ilości pamięci domyślnie rezerwowanej przez wektor, co jest zostawione po stronie implementacji. Przed wykonaniem każdej partii wątek czeka na dostęp do zasobu. Funkcja `try_lock()` zwróci `true` w przypadku udanego przydzielenia zasobu, w przeciwnym razie zwróci `false`.

```
1   QtConcurrent::run(
2               [&]()->void{
3                   ui->pushButton_Run->setEnabled(false);
4                   const int g= nr_rounds;
5                   const int p= nr_players;
6                   const int f= fun;
7                   vector<unique_ptr<Game>> tab;
8                   std::default_random_engine generator;
9                   std::uniform_int_distribution<int> distribution
                        (0,255);
10                  auto r= bind(distribution, generator);
11                  clear_vectors();
12                  chrono::milliseconds d(delay);
13                  beginsp.resize(p);
14                  endsp.resize(p);
15                  for(int i=0; i<p;i++){
16                      tab.push_back(make_unique<Game>(f));
17                      colorsp.push_back(make_tuple( r(), r(), r() ));
```

```
18                      beginsp[i].reserve(g);
19                      endsp[i].reserve(g);
20                  }
21                  for(int j=0; j<g;j++){
22                      if(quit) return;
23                      while(!points.try_lock());
24                      for(int i=0; i<p;i++){
25                          beginsp[i].push_back(tab[i]->next());
26                          endsp[i].push_back(tab[i]->prelast());
27                      }
28                      points.unlock();
29                      emit copy();
30                      this_thread::sleep_for(d);
31                  }
32                  ui->pushButton_Run->setEnabled(true);
33              });
```

3.6. Rysowanie 3D

Rysowanie Rysowanie sześcianu oraz funkcji zostało wykonane jako rysowanie odcinków. Ze względu na ograniczony krok czasowy symulacji funkcje wyglądają na wygładzone. Środek sześcianu jest środkiem układu współrzędnych.

Rozszerzanie Do rozszerzania ekranu został użyta następująca funkcja[4] :

```
1  void MyGLWidget::resizeGL(int width, int height){
2      int side = qMin(width, height);
3      glViewport((width - side) / 2, (height - side) / 2, side, side);
4      glMatrixMode(GL_PROJECTION);
5      glLoadIdentity();
6  #ifdef QT_OPENGL_ES_1
7      glOrthof(-1, +1, -1, +1, 1.0, 15.0);
8  #else
9      glOrtho(-1, +1, -1, +1, 1.0, 15.0);
10 #endif
11     glMatrixMode(GL_MODELVIEW);
12 }
```

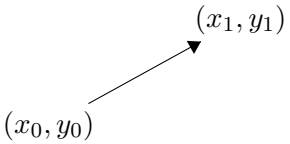
Oknem widgetu jest widoczny na ekranie prostokąt. Oknem roboczym widgetu jest obszar, na którym przeprowadzane są operacje. Oba okna nie muszą się pokrywać, mogą istnieć sytuacje, w których rysowane są elementy poza widokiem użytkownika. Posiadając żądane wymiary nowego okna należy wy-

znaczyć nowe położenie i wymiary nowego okna roboczego tak, aby zachować proporcje sześcianu oraz przedstawić go w całości. Argumentami funkcji `glViewport` są kolejno: współrzędne x i y lewego dolnego rogu widoku, szerokość i wysokość okna roboczego. Okno robocze musi być dopasowane do najkrótszego wymiaru widgetu, aby figury nie znalazły się poza oknem widgetu. Aby zapewnić wycentrowanie widoku roboczego należy dobrać punkt początkowy okna roboczego (lewy, dolny narożnik). Musi on mieć współrzędną punktu początkowego krótszej krawędzi jako 0, ponieważ chcemy wykorzystać ją całą. Współrzędna punktu początkowego dłuższej krawędzi okna roboczego musi być oddalona od punktu początkowego okna widgetu o połowę niewykorzystanej długości. Prowadzi to do wzoru na punkt początkowy okna roboczego:

$$p_0 = (\frac{1}{2}(\text{szerokość} - min(\text{szerokość}, \text{wysokość}), \frac{1}{2}(\text{wysokość} - min(\text{szerokość}, \text{wysokość})) \quad (3.1)$$

Mając okno robocze o odpowiednich wymiarach zostało tylko przesunięcie układu współrzędnych do centrum okna roboczego przy pomocy funkcji `glOrtho`.

Obracanie Kąt obrotu figury [4] jest wyznaczany na podstawie punktu, gdzie klawisz myszy został wciśnięty (x_0, y_0) oraz punktu w którym jest nadal wciśnięty (x_1, y_1). Lewy przycisk myszki ma priorytet nad prawym. Lewy przycisk myszy generuje obrót względem osi x i y o odpowiednio Δ y i Δ x stopni, natomiast prawy przycisk generuje obrót względem osi x i z o odpowiednio Δ y i Δ x stopni. Obrót jest ograniczony od 0° do 360°, wartości poza tym zakresem są do niego sprowadzane poprzez operacje $\pm 360^\circ$. Wymnażane delt przez czynnik mniejszy od 1 prowadzi do spowolnienia obrotu.



Rysunek 3.2: Obrót - ruch myszy

3.7. Makefile

Do rysowania wykresu funkcji prawdopodobieństw od numeru partii został użyty plik `Makefile`, który wykona kompilację, uruchomienie oraz narysowanie wykresu przy pomocy programu gnuplot. Aby uruchomić program należy podać argument: G - ilość partii do rozegrania oraz P - ilość graczy. Poniżej przykładowe polecenie do wykonania symulacji dla 100 partii rozegranych przez 20 zawodników.

```
make G=100 P=20
```

Podanie nierealnych wartości nie da wyniku. Główną różnicą w implementacja gry N-osobowej jest funkcja `Game::next()`.

```
1 void Multigame::next() {
2     ...
3     p[0]+= 0.1*(p[0]*(1-p[0])*(1-static_cast<double>(nr_r[1])/current
          -static_cast<double>(nr_r[players-1])/current));
4     for(int i=1; i<players-1; i++)
5         p[i]+= 0.1*(p[i]*(1-p[i])*(1-static_cast<double>(nr_r[i+1])/
              current-static_cast<double>(nr_r[i-1])/current));
```

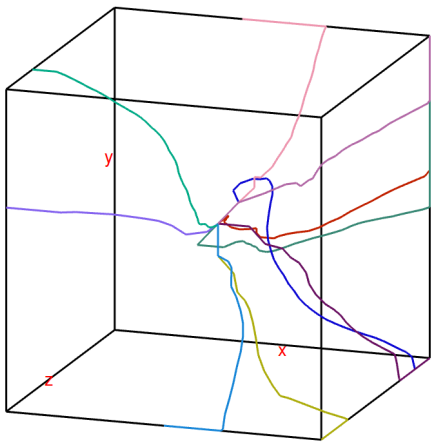
```
6       p[players-1]+= 0.1*(p[players-1]*(1-p[players-1])*(1-static_cast<
            double>(nr_r[0])/current-static_cast<double>(nr_r[players-2])/
            current));
7   }
```

Szacowanie prawdopodobieństw musi być wykonane w pętli, ze względu na nieznaną liczę graczy.

4. Wyniki

4.1. Gry 3-osobowe

Równania standardowe Analiza teoretyczna równań wskazała punkt $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ jako punkt stały. Wynika to z założenia gry o pełnej informacji. W symulacji to założenie nie jest spełnione, dlatego trajektoria opuszcza centrum sześcianu. Osiągnięcie przez funkcję krawędzi sześcianu oznacza zawiązanie koalicji, po czym funkcja przesuwa się po krawędzi dążąc do wierzchołka. Biorąc pod uwagę że dla par prawdopodobieństw przeciwników $(p_a, p_b) \in \{(0,1),(1,0)\}$ zmiana prawdopodobieństwa trzeciego gracza wynosi 0 należy stwierdzić, że funkcja powinna osiągać punkt stabilności na krawędzi i nie poruszać się dalej. Byłoby tak gdyby błąd szacowania prawdopodobieństw przeciwników był odpowiednio mały. Szacowanie prawdopodobieństw dane jest przez $\frac{n_j}{l_p}$. Wynika z tego, że wystarczy kilka zagrań przeciwników niezgodnych z zawartą koalicją, aby gracz szacujący ich prawdopodobieństwa nie mógł stwierdzić, że ich realne prawdopodobieństwa wynoszą 0 lub 1. Pokazuje to rysunek 4.1, na którym widać funkcje o wielu punktach przegięcia. Szybkość dążenia do wierzchołków będzie malała wraz z czasem spędzonym na krawędzi, gdyż

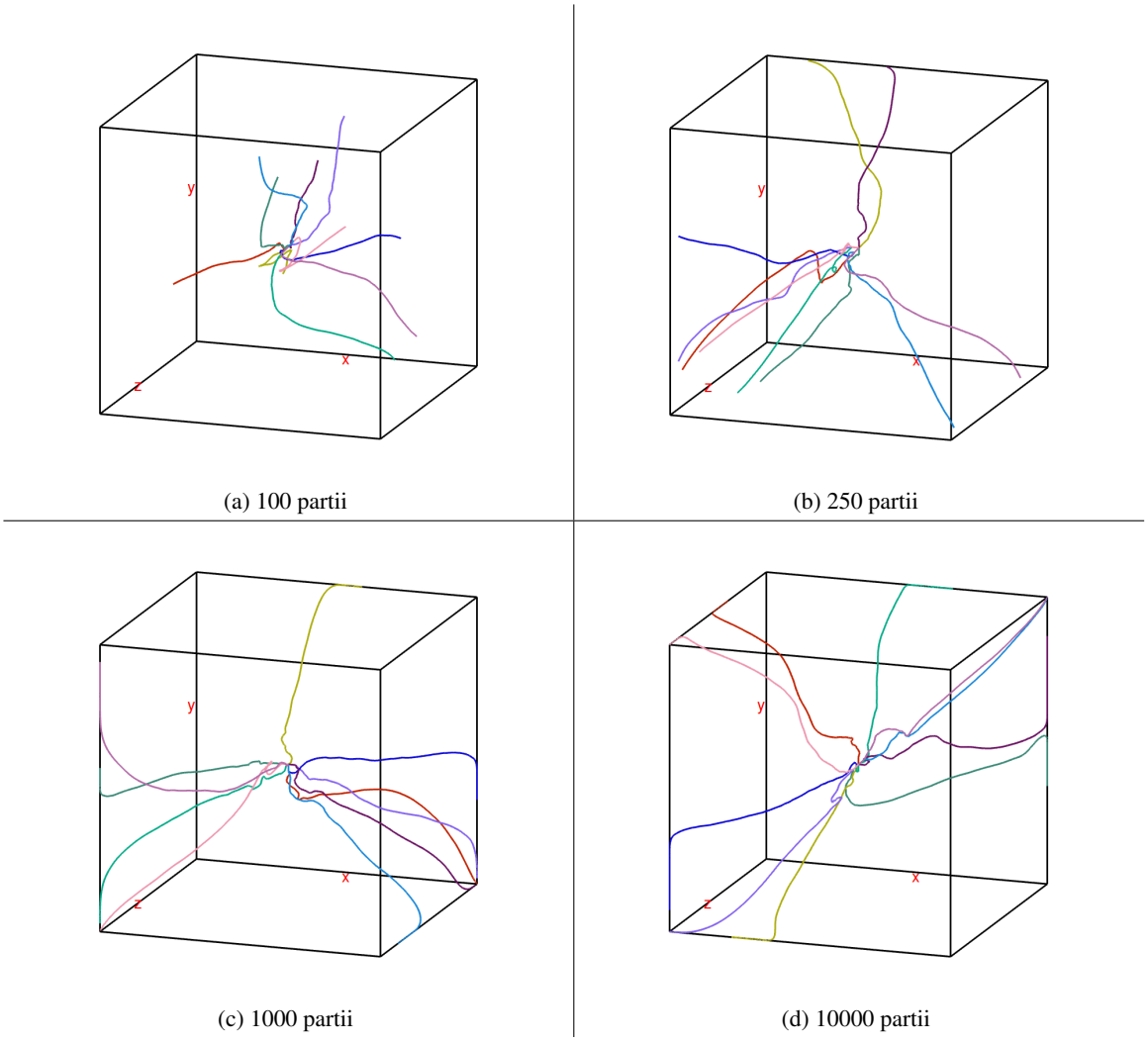


Rysunek 4.1: Równania standardowe: 100 partii, 10 instancji

$$\lim_{n_j \to l_p \wedge l_p \to \infty} \frac{n_j}{l_p} \in \{0, 1\}$$

W miarę rozgrywania kolejnych partii przewidywane prawdopodobieństwa będą zbliżały się do realnych. Dla realnego prawdopodobieństwa ruch po krawędzi nie występuje, więc z czasem prędkość ruchu będzie dążyć do 0.

Równania replikatorów Równania replikatorów charakteryzują się wolniejszymi zmianami gry w stosunku do równań standardowych. Przyczyną jest człon $x(1-x)$, którego maksimum wynosi 0.25. Daje to czterokrotnie mniejsze Δp w punkcie $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Na krańcach dziedziny $< 0,1 >$ wspomniany człon szybko dąży do 0, co znacznie opóźnia osiągnięcie koalicji. Wi-



(a) 100 partii

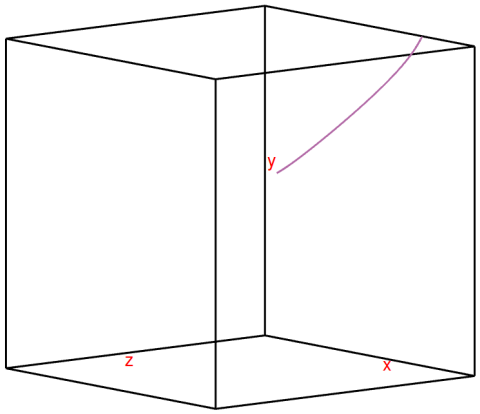
(b) 250 partii

(c) 1000 partii

(d) 10000 partii

Rysunek 4.2: Równania replikatorów: 10 instancji

dać to porównując rysunki 4.1 oraz 4.2a. Natomiast rysunki 4.1 oraz 4.2d pokazują liczbę partii potrzebną do osiągnięcia podobnych miejsc w przestrzeni sześcianu. Gra używająca równań replikatorów musiała wykonać ich 100 razy więcej niż gra używająca równań standardowych. Analiza stabilności pokazała, że 6 krawędzi sześcianu powinno być liniami punktów stabilnych. Rysunek 4.3 obrazujący grę o pełnej informacji wykazuje stabilność punktu na krawędzi. Natomiast symulacja przedstawiona na rysunku 4.2c pokazuje przemieszczanie się funkcji ku wierzchołkom. Powodem tego jest błąd gracza w szacowaniu prawdopodobieństwa przeciwników, wynikający z gry o niepełnej informacji. Analiza stabilności wskazuje, że punkty $(0,0,0)$ oraz $(1,1,1)$ nie są stabilne, co jest przyczyną tego, że żadna z symulacji do nich nie zmierza. Co więcej pierwszy i ostatni wiersz tabelki 2.1 nie może wystąpić w symulacji, jest to reprezentowane niebieskimi krawędziami na rysunku 2.2. Osiągnięcie punktu składającego się z permutacji $\{0,0,\xi\}$, gdzie $\xi \neq 1$ lub permutacji $\{1,1,\xi\}$, gdzie $\xi \neq 0$, nie jest możliwe w symulacji. Oznaczałoby to zawiązanie sojuszu pomiędzy graczami, których zagrania nie mają na celu zawiązania między nimi koalicji. Potwierdzają to symulacje z których żadna nie dochodzi do krawędzi wychodzących z punktów $(0,0,0)$ oraz $(1,1,1)$.

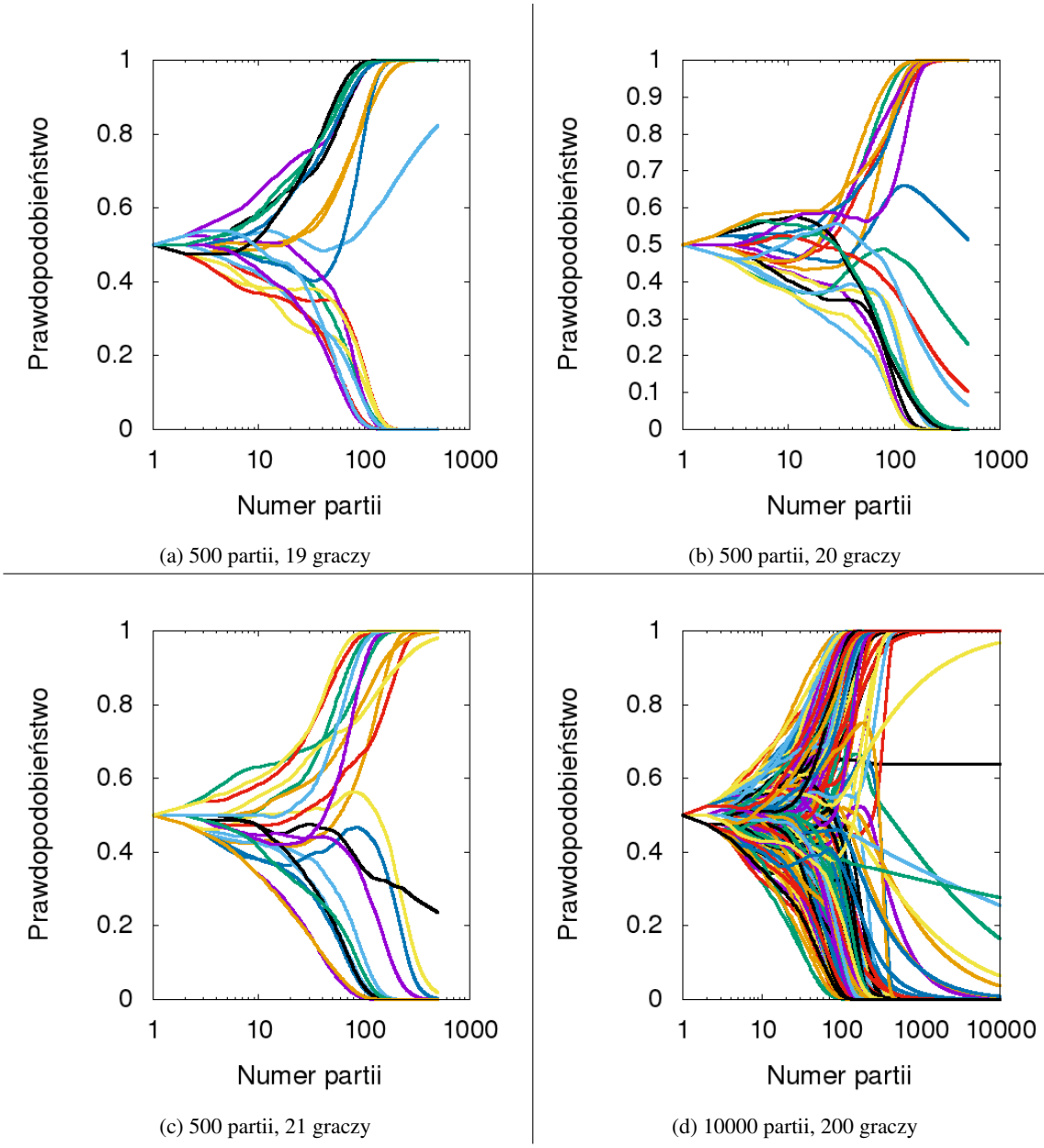


Rysunek 4.3: Równania replikatorów, gra o pełnej informacji: 300 partii, 10 instancji. Punkt początkowy $(0.55, 0.6, 0.4)$

4.2. Gry N-osobowe

W grze w której ustawionych w okręgu jest N graczy, spodziewamy się, że istotnym czynnikiem będzie parzystość ich liczby. Teoretyczna liczba graczy mogących nie znaleźć koalicjanta wynosi $< N \bmod 2, \lfloor \frac{N}{3} \rfloor >$. W grze o parzystej liczbie graczy zawsze istnieje rozwiązanie, które gwarantuje każdemu graczowi przynależność do koalicji. Natomiast w grach o nieparzystej liczbie graczy musi istnieć co najmniej jeden zawodnik, który nie zawrze sojuszu (posiadanie prawdopodobieństwa 0 lub 1, nie jest równoznaczne z byciem w sojuszu, co pokazały symulacje dla 3-graczy, kiedy funkcje dążyły do wierzchołków). Maksymalna liczba graczy bez sojuszu może być maksymalnie połową liczby graczy będących w sojuszy(maksymalnie co trzeci gracz może być bez sojuszy z zaokrągleniem w dół). Gdyby była większa oznaczałoby to, że istnieją pary sąsiadujących graczy, którzy nie są w żadnej koalicji. Taka sytuacja nie może mieć miejsca, ponieważ omawiane pary stałyby się koalicjami.

W przypadku występowania dużej ilości zawodników łatwiej można zaobserwować sytuacje, w których sąsiedzi zawodnika stosują w przewadze jedną taktykę. Co prowadzi do dokładnego szacowania ich prawdopodobieństw. Gracz wtedy nie dokonuje zmian w swoim zachowaniu, co wynika z tabelki 2.1. Przypadek taki można zaobserwować również na rysunku 4.4d, oznaczony kolorem czarnym.



(a) 500 partii, 19 graczy

(b) 500 partii, 20 graczy

(c) 500 partii, 21 graczy

(d) 10000 partii, 200 graczy

Rysunek 4.4: Gry N-osobowe

5. Podsumowanie

Celem pracy było przeprowadzenie symulacji powstawania koalicji przez stosowanie strategii mieszanych. Oczekiwane wyniki na podstawie analizy matematycznej nie odpowiadały ściśle wynikom uzyskanym z symulacji. W metodach analitycznych zakłada się bowiem pełną informację graczy, podczas gdy w symulacji zakładano grę o niepełnej informacji.

W celu zmniejszenia błędu szacowanego prawdopodobieństwa, można by uwzględniać do niego jedynie k-ostatnich gier. Spowodowałoby to trafne szacowanie prawdopodobieństwa podczas monotonicznej gry przeciwników w k partiach, czego skutkiem byłyby punkty stabilne na krawędziach po przeprowadzeniu na nich k partii. W niniejszej pracy szacowane prawdopodobieństwa nie mają możliwości dojścia do 0 lub 1, jeśli decyzje przeciwników nie były monotoniczne przez całą grę. Skutkiem tego jest przemieszczanie się funkcji po krawędziach sześcianu, co mogłoby ustać po k partiach jeśli tylko one uwzględniane byłyby w szacowaniu. Oczywiście szybkość poruszania się po krawędziach spada z czasem, lecz nigdy nie dojdzie do sytuacji w której spadnie do 0. Szacowanie prawdopodobieństw na podstawie tylko niektórych partii rodzi problem natury doboru wartości parametru k.

Zgodnie z analizą funkcje dążyły do krawędzi sześcianu, omijając punkty niestabilne oraz krawędzie reprezentujące niemożliwe do zawiązania koalicje zgodnie z rysunkiem 2.2.

Symulacje gier N-osobowych pokazały dążenie graczy do trwałych koalicji. Wartą przeanalizowania byłaby sytuacja losowej zmiany prawdopodobieństw części graczy w stanie ustalonym. Mogłoby to z czasem doprowadzić ustalenia nowego stanu ustalonego.

Używana tu metoda zastosowania koncepcji strategii mieszanych do określania koalicji jest owocem dyskusji autora z promotorem tej pracy.

Bibliografia

[1] C++ reference. http://en.cppreference.com/. 2017-12-26.

[2] Finite state machine designer. http://madebyevan.com/fsm/. 2017-12-20.

[3] Qt main site. http://doc.qt.io/. 2017-09-01.

[4] Qt5 tutorial opengl with qglwidget - 2017. http://www.bogotobogo.com/Qt/Qt5_OpenGL_QGLWidget.php. 2017-09-01.

[5] Stany stacjonarne i ich stabilność. http://prac.us.edu.pl/~ekonofiz//skrypty/StochDyn/ch1/chI023.html. 2017-12-26.

[6] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge, 1998.

[7] M. A. Nowak. *Evolutionary dynamics: exploring the equations of life*. The Belkan Press of Harvard University Press, Cambridge, MA and London, 2006.

[8] P. D. Straffin. *Teoria gier*. Wydawnictwo Naukowe SCHOLAR, Warszawa, 2001.