Studium licencjackie

Kierunek: Metody Ilościowe w Ekonomii i Systemy Informacyjne

Forma studiów: Stacjonarne

Imie i nazwisko autora: Kacper Mordarski

Nr albumu: 101247

Cooperation in the PD and SD games on preferential attachment graphs

Praca licencjacka napisana

w Katedrze Matematyki i Ekonomii Matematycznej

pod kierunkiem naukowym

dr hab. Michała Ramszy

Warszawa 2022

Contents

1 Introduction

This paper aims to replicate, to some degree, the seminal paper of Santos and Pacheco (2005) on the emergence of cooperation. The secondary goal of this research is to provide the publicly available code allowing replication of the actual results.

» To co jest poniżej to do poprawki. Trzeba tutaj napisać taki standardowy wstęp machany rękami. Koniecznie trzeba się pozbyć “The paper”, to jednak jest nie do przyjęcia.

either prove or disprove the research paper with title **"Scale-Free Networks Provide a Unifying Framework for the Emergence of Cooperation"** written by F.C. Santos and J. M. Pacheco and published in Physical Review Letters nr 95 (later on reffered to as "The Paper"). Regardless of the outcome this work is going to be useful because the above mentioned authors did not provide the code that was used to conduct the necessary simulations. Therefore goal is not only to disprove the above mantioned paper but also to provide readers with clear and easy to follow code.

The Paper was published in “Physical Review Letters” on the 26th of August 2005 and, according to Google Scholar, has been since cited over 1600 times. It clearly shows the magnitude of the said paper and its groundbreaking character. This paper presents the results of simulations conducted by the authors and their implications for evolutionary game theory.

The crucial thing to understand is that the authors of The Paper changed the approach to modeling such games by applying Scale-Free Networks of Contacts (later on referred to as "SF NOCs"). Its innovativeness lies in the never used before degree distribution of said graph. Before being used graphs had a degree distribution with a single peak. It means that every "player" could interact only with a fixed number of other "players". SF NOCs are said to perform better at modeling the actual, existing societies and networks. It complies with the rules of growth and preferential attachment (rich gets richer). When we analyze some actual networks, for example, the Twitter network, we observe that those with a bigger count of "followers" are more likely to gain new ones than accounts with a low count of followers.

One of the goals of The Paper was to compare the results of simulations on different kinds of graphs. According to The Paper, players that occupy vertices of SF NOC are much more likely to cooperate than on any other graph. Those results came up both in the Snowdrift game (later on referred to as SG) and Prisoners Dilemma game (later on referred to as PD).

» Tutaj opisać jaka jest struktura całej pracy, w sensie w tym rozdziale to jest to, w tamtym rozdziale to jest tamto. Robimy to na końcu.

2 Elements of the noncooperative game theory

The current chapter introduces the most fundamental concepts in the game theory and the appropriate notation. These concepts and notation are used later in subsequent chapters. The exposition follows Fudenberg and Tirole (1991), Gibbons (1992).

The noncooperative game theory is the basis for this dissertation. We call a game a noncooperative game if all of the game's participants (players) act in their own best interest and, therefore, compete with each other. Moreover, they need to be rational — choose a strategy based on its optimality. Games in this paper are presented as normal-form games. The definition of a norm-form game requires a couple of elements.

First of all, let us define a set of players $i \in I$, where I is a finite set of natural numbers, i. e., $I = \{1, 2, \ldots, N\}$. For each player i, we define the pure strategy space S_i consisting of $k_i \in \mathbb{N}$ pure strategies. The sequence of pure strategies $s = (s_1, \ldots, s_N) \in \prod_{i \in I} S_i = S$ is called the strategy profile. Lastly, we define a payoff functions, denoted as $u_i(s)$. A function $u_i(s)$ maps a strategy profile s into player's payoff, that is $u_i : \prod_{i \in I} S_i \to \mathbb{R}$.

The last two assumptions of a normal-form game are that (a) players act independently and (b) the perfect knowledge is assumed. The former assumption means that players have no knowledge of other players' choices while making a decision. The latter means, roughly, that all players know the structure of the game and it is perfect knowledge.

The fundamental concept of the game theory is the concept of equilibrium. The most widely used and accepted concept of equilibrium is Nash equilibrium, cf. Nash (1951). The general idea of the Nash equilibrium concept is that there is a strategy profile in which none of the players have an incentive to alter their strategy. Given a profile of strategies s, no single player can change a strategy to get a higher payoff.

Technically, we define the Nash equilibrium as follows. Let S_i be be the set of all possible strategies for a player i, and let $s^* = (s_i^*, s_{-i}^*)$ be a strategy profile, where s_{-1}^* denotes strategies of all other players except of i. For the strategy profile s^* to be the (weak) Nash equilibrium, it must satisfy the inequality:

$$\forall s_i \in S_i : \; u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*).$$

For the Nash equilibrium to be strict, it must satisfy the strict inequality:

$$\forall s_i \in S_i : \; u_i(s_i^*, s_{-i}^*) > u_i(s_i, s_{-i}^*).$$

It is important to note that obtaining a Nash equilibrium does not guarantee the solution's optimality. To be specific, it means that payoffs may not be Pareto optimal (Woźny 2012). Thus, there may exist a strategy profile yielding higher payoffs for at least one player with other players' payoffs not lower.

We conclude this chapter with two simple examples of normal-form games and their Nash equilibria. We start with the "Battle of sexes" game. The game described in Luce and Raiffa (1989) is a classical decision analysis problem. Let us consider two music critics. They agreed to attend one of the two excellent concerts happening in their whereabouts. They can either go to a Shostakovich or Stravinsky concert.

One critic — let us call him player one — would prefer to go to the Shostakovich concert. Meanwhile, the other critic, let us call him player two, would instead attend the Stravinsky concert. Unfortunately, they have no way of contacting each other, and therefore, the decision of where to go must be undertaken simultaneously. We have two players, each with two possible strategies: $\{Shostakovich, Stravinsky\}$. Each of them achieves a payoff of 1 if they get to go to their place of choice. Furthermore, if they both choose to go to the same event, they gain utility from spending time together. We can represent this game in form of the following payoff matrix:

	$Shostakovich$	$Stravinsky$
$Shostakovich$	$3, 2$	$1, 1$
$Stravinsky$	$0, 0$	$2, 3$

In the matrix, player one chooses the row strategy, and player two chooses the column strategy. Each cell represents two payoffs — the left one is the first player's payoff, and the right one represents the second player's payoff.

There are two Nash equilibriums in this game. They occur for the following strategy profiles: $(Shostakovich, Shostakovich)$ and $(Stravinsky, Stravinsky)$.

The second example concerns the "Stag Hunt" game. The stag hunt game originates from the work of Jean-Jacques Rousseau, cf. Rousseau (1985). It represents the conflict between social cooperation and conflict.

The idea behind the game is that two hunters can either hunt for a stag or a hare. Moreover, none of the hunters can single-handedly take down a stag — it is a job for two. If either hunter chooses to go for a stag and the other selects a hare, they get payoffs equal to 0 and 4, respectively. If they both go for a hare, each gets a payoff of 2 — there are only so many hares that they can provide an accumulated utility of 4. However, if they both decide to go for a stag, they get a payoff equal to 5. Therefore, the payoff matrix looks as follows:

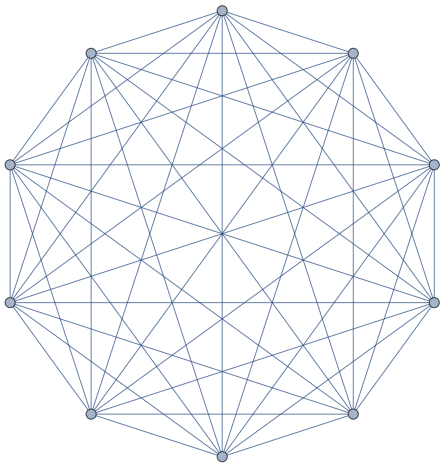
	$Stag$	$Hare$
$Stag$	$5, 5$	$0, 4$
$Hare$	$4, 0$	$2, 2$

This game also has two Nash equilibriums — strategy profiles $(Stag, Stag)$ and $(Hare, Hare)$. Clearly, the equilibrium $(Stag, Stag)$ is Pareto optimal.

In the remaining of this paper, we only use two-player normal-form games without mixed strategies. Hence, the above introduction is sufficient for our purposes.

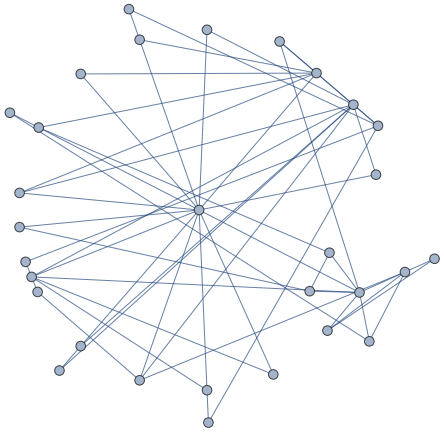
3 Elements of the graph theory

The next important element that we use in the current paper is the graph theory. We can define a graph as two sets — one containing objects called vertices and the other containing pairs of vertices called edges, cf. Ross and Wright (1985). A single vertex is usually denoted by a natural number. It can stand for various objects, but in this paper, we use the vertices of a graph to represent players. The edges epitomize relations between vertices. In our case, the connected vertices stand for connected players. Thus, these players can engage in a game. We can distinguish between two types of a graph — directed and undirected ones. As we use only undirected graphs, only these graphs are discussed below.



(a) Regular graph with $n = 10$ vertices

(b) BA graph with $n = 30$ and $k = 1$



(c) BA graph with $n = 30$ and $k = 2$

Figure 1: Example of regular and BA graphs. *Source:* own calculations

Since in an undirected graph edges represent relations with no regard to their direction, we can denote the relation between vertices a and b either as $\{$a,b$\}$ or $\{$b,a$\}$. As an example of an undirected graph, we can think of a graph in which vertices represent humans and edges represent relations between humans, e.g., being friends or connected in a given social network.

Since these relationships work both ways (if subject a is a friend of subject b, it implies that subject b is a friend of subject a), we only need to use a set $\{a, b\}$ to denote an edge. We do not use directed graphs in this paper. Henceforth, as a graph, we refer to an undirected graph.

Let us define a neighborhood of a vertex in a graph. The neighborhood of vertex v in a graph G is a set of vertices adjacent to the vertex v. We call two vertices adjacent when they are connected. The graph induced by a neighborhood of v is called a neighborhood graph. This concept is crucial for our work because it will allow us to identify all of the vertices able to engage in a game with a previously chosen player.

The degree of a vertex v is a number of all vertices connected to the v. The important characteristic of a graph is its' degree distribution. The degree distribution is the probability distribution of degrees in a graph. This characteristic varies throughout the different types of graphs. For a regular graph, it is a single value with the probability of 1 because, in a regular graph, all of the vertices have the same degree. Figure 1a show an example of a regular graph.

In our work, we use a type of graph called Scale-Free Network. This type of graph is generated according to the Barabasi-Albert Model, cf. Albert and Barabási (2002). The scale-free networks are built in a sequence of steps. We start with a single vertex and, in each step, add a single vertex. The new vertex is connected to the previous vertices with the probability proportional to the vertices' degrees. More precisely, the probability of attaching a new vertex to an already existing vertex v is given as:

$$p_v = \frac{k_v}{\sum_j k_j},$$

where k_v is a degree of vertex v and $\sum_j k_j$ is a sum of degrees of all existing nodes (which, since our graph is an undirected graph, is equal to twice the amount of existing edges).

This rule favorites the vertices with higher connectivity and causes the graph to have so-called hubs. A hub is a term for a vertex with a considerably higher degree than other vertices. Thanks to this rule, the degree distribution of a graph is given by the formula:

$$P(k) \backsim k^{-3}.$$

Figures 1b–1b show example of scale-free networks. More precisely, figure 1b shows an example of a scale-free network built by adding only one edge at every step. Figure 1b show an example of a scale-free network created by adding two edges at every step.

4 Cooperation among the players

» W tym miejscu w każdym rozdziale trzeba wpisać krótkie streszczenie co jest w danym rozdziale.

4.1 A brief description of Snowdrift and Prisoners Dilemma games

Prisoners Dilemma Game. The Prisoners Dilemma game is a widely known problem in game theory and decision analysis. It shows situations in which the outcome is not optimal, even though players act in their own best interest. In the scope of our analysis, it is essential to note that in the PD game, the best strategy is to defect, regardless of the opponent's choice. The game is parameterized as follows:

	C	D
C	R, R	S, T
D	T, S	P, P

where the values are given as follows:

$$T = b > 1, \quad R = 1,$$
$$P = 0, \quad S = 0,$$
$$1 < b \leq 2.$$

We see that the above restrictions order the parameters as follows:

$$T > R > P = S.$$

Snowdrift Game. The Snowdrift game represents a metaphor for cooperative interactions between players. Contrary to the PD game, the Snowdrift game stimulates cooperative behavior amongst players. It doesn't make the deflection strategy inapplicable, but the game's payoffs encourage cooperative behavior more than the payoffs of the PD game. The optimal strategy is to cooperate when the other defects and to defect when the other cooperates. The game is parametrize as follows:

	C	D
C	R, R	T, S
D	S, T	P, P

where the parameters' values are:

$$T = \beta > 1, \quad R = \beta - \frac{1}{2},$$
$$S = 1 - \beta, \quad P = 0.$$

We see that the above restrictions order the parameters as follows:

$$T > R > P > S.$$

Nash equilibria. » Tutaj analiza równowag Nasha w opisanych powyżej grach w one-shot games. Pokazać, że jest albo nie mam kooperacji. Tutaj również wprowadzamy grę na grafach losowych, a więc musimy opisać na jakich grafach, itd. Tutaj również opis uczenia się ale w sensie algorytmu matematycznego nie implementacji.

5 Algorithms and simulations

This chapter contains a description of the algorithm used in the simulation.
» To co powyżej do rozszerzenia. Generalnie ten rozdział ma zawierać implementację koncepcji matematycznej z poprzedniego rozdziału.

In this section of the dissertation, I'm going to describe the key elements of the algorithm used to replicate the results obtained in The Paper. My understanding of the mechanisms on which the algorithms are based is limited to the rather vague and unclear description in The Paper.

5.1 Functions

In order to perform necessary calculations I had to define the following functions:

1. `Transform` — This function is used to map strategies onto a vector of arrays. As an input this function takes one of the edges of a SF NOC, as an output it returns a vector of length two (two vertices connected with an edge).

2. `Strat` — This function is used to map previously distributed strategies onto a vector of edges. As an input it takes an edge and as an output it returnes a vector of length two (two strategies previously attributed to the vertices).

3. `Games` — This function is used to evaluate the results of games played between players (vertices connected with an edge). As an input this function takes a vector of edges, vector of strategies and a vector of accumulated payoffs. It returns an adjusted vector of accumulated payoffs.

4. `CheckStrat` — This function fulfills a number of tasks. It takes as an input a randomly chosen vertex, vector of strategies, vector of accumulated payoffs, and the SF NOC. It identifies all neighbors of the previously mentioned vertex, then shuffles them, and then looks for a neighbor with a different strategy. Then it proceeds to change the strategy of the vertex with lower accumulated payoffs. The probability of the transition is described in section 1.5.

5.2 Description of the algorithm

The main algorithm is the fundamental element of this dissertation. It consists of 3 nested loops executing instructions necessary to conduct simulations, on which my research is based. I will be describing those loops in an inside-out order.

The first loop is responsible for evaluating the results of the games, potentially changing the strategies of players, and keeping track of the proportion of the strategies used by players. To run properly, it requires previously set parametrization and a set of edges of the Barabasi-Albert graph. As a result, it produces a vector of length 100 in which elements are proportions of coop/def strategies, measured after each generation (1001–1100) in the population of players. This loop is repeated 2 000 times, which gives us a total of 2 200 000 generations for one Barabasi-Albert graph parametrization for a given game. The loop itself operates as follows:

Data: Barabasi-Albert Moedel, array of edges, array of strategies, array of accumulated payoffs (initially equal to 0), empty vector of length 100, parametrization of the games

Result: Vector of length 100 containing coop/def proportion for the population

initialization;

for *k in generations* **do**

  Play a single round of a given game between all possible pairs of players by applying function `games` on a vector of edges.;

  Randomly choose one vertex. ;

  Attempt to change the strategy of the previously chosen player (or one of his neighbors) by applying function `CheckStrat`. ;

  if *k is greater than the number of generations minus 100* **then**

    Evaluate the coop/def proportion in the entire population;

    Save the result to the appropriate value of a vector.;

  else

  end

end

Algorithm 1: The inner loop

The outer loop to the one described above is responsible for "resetting" the Barabasi-Albert graph. Since separate simulations are supposed to be carried out on a randomly generated SF NOC (but with the same parametrization), we need to conduct 100 of them (for each payoff parametrization); this loop is an indispensable element of the algorithm. The crucial fact to note is that this loop is also responsible for creating an object named `ArrPath` — vector of length 100, used to store the results of the simulations. The procedure followed by this loop is as shown below:

Data: Barabasi-Albert parametrization

Result: Scale-Free Network built by Barabasi-Albert Model, list of edges, list of strategies, vector of accumulated payoffs, ArrPath — an array of length 100 storing results of the simulations

initialization;

for *j in simulations* **do**

- Create a Scale-Free Network by using the Barabasi-Albert model with previously set parameters.;
- Create an array of length equal to the number of players, containing their strategies. ;
- Extract an array of edges from a previously created graph by mapping function `transform` onto a list of edges. ;
- Produce an array of tuples representing strategies for every player by mapping a function `strat` onto an array of edges. ;
- Create an array of accumulated payoffs (initially filled with zeros). ;
- Generate a vector of length 100 filled with zeros, called `track`. This vector will be later on used to store coop/def proportions. ;
- **if** *j equals to 1* **then**
  - Create an object called `ArrPath` with one element equal to `track`;
- **else**
  - Merge the current `track` with `ArrPath`. ;
- **end**
- Initiate the inner loop described above. ;

end

Algorithm 2: Second loop

The outer loop is in charge of setting the parametrs of the games. Since we want to obtain 20 data points for a single Barabasi-Albert Model parametrization, we need to iterate 20 times executing the previously described loops. The payoffs for our games are of range $[1, 2)$ and $(0, 1]$ for PD and SD respectively, we need to divide the span (which is equal to 1 in both cases) by the number of data points, and we gain a result of 0.05. Thus each iteration adds 0.05 to the payoffs of both games. Furthermore, this loop is also responsible for creating and object called `DataToSave` that will store our results. Atfer the loop ends, the `DataToSave` object is saved into the local repository.

Data: None

Result: Parametrization for the games, `DataToSave` — an array of vectors that will be used to store and ultimately save our results.

initialization;

for *i in number of parametrizations* **do**

  if *i is equal to 1* **then**

    Create an array of length 20 — `DataToSave`. ;

  else

    Store the current `ArrPath` as a value of `DataToSave`. ;

  end

  Set up payoffs of the games based on the current iteration. In each iteration, add 0.05 to the current values of the payoffs. ;

  Create a matrix of payoffs for the games with current parametrization. ;

  Initiate the second loop. ;

  Save `DataToSave` to the local repository. ;

end

Algorithm 3: The outer loop

5.3 Simulations description

Because of the nature of this paper (an attempt to clone the results of The Paper), our simulations must be conducted in strict accordance with the methods outlined in The Paper. Therefore it is only natural that we must follow each step with the utmost care and diligence. According to The Paper, we must conduct 100 simulations for each parametrization. Each simulation is performed following those steps: Where the parametrization is as follows:

1. Setting up the parameters which are needed to create SF NOCs (such as the number of final vertices (population size), the average connectivity, etc.).

2. Choosing the parameters of the game which is to be simulated (either PD or SG).

3. Creating the randomly generated SF NOC (we use Barabasi - Albert model to do that). The SF NOC must be created in compliance with preferential attachment and growth rules.

4. Randomly distributing strategies amongst the population (SF NOC in this particular case). Each vertex can either get a cooperation or deflection strategy.

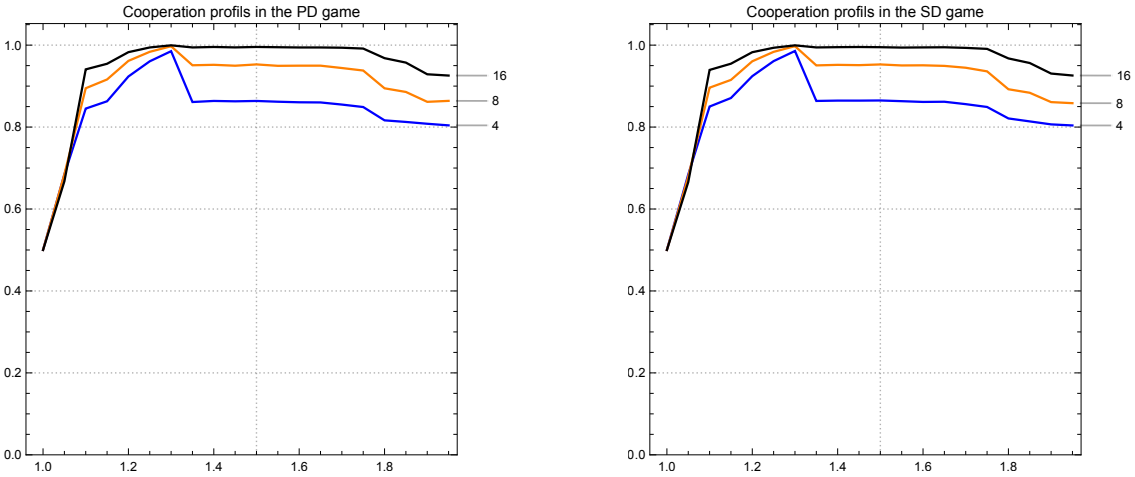
5. Each pair of cooperator-deflectors engages in a round of a given game. In compliance with replicators dynamics, we keep track of cumulative payoffs for both strategies so that "players" can adjust their strategies throughout the population. This step is repeated 11 000 times, each time is called "generation". The first 10 000 is the so-called "transient time".

6. We collect results (equilibrium frequencies of cooperators and defectors) by averaging over the last 1 000 generations.

» Poniżej wrzuciłem przykładowy algorytm w pseudokodzie. Myślę, że to jest to co chcemy wykorzystać aby zapisać sam algorytm. To oczywiście musi być potem opisane w tekście.

6 Results and discussion

» Tutaj wpisujemy uzyskane wyniki oraz dyskutujemy je, np. porównujemy do wyników z oryginalnego artykułu.



(a) PD game

(b) SD game

Figure 2: Cooperation profiles for the PD and SD games. *Source:* own calculations

7 Conclussions

» Tutaj na samym końcu dopisujemy dlaczego i co zrobiliśmy oraz jakie mamy dalsze plany na badania.

References

Albert, R. and Barabási, A.-L. (2002), 'Statistical mechanics of complex networks', *Reviews of modern physics* **74**(1), 47.

Fudenberg, D. and Tirole, J. (1991), *Game theory*, MIT press.

Gibbons, R. S. (1992), Game theory for applied economists, *in* 'Game Theory for Applied Economists', Princeton University Press.

Luce, R. D. and Raiffa, H. (1989), *Games and decisions: Introduction and critical survey*.

Nash, J. (1951), 'Non-cooperative games', *Annals of mathematics* pp. 286–295.

Ross, K. A. and Wright, C. R. (1985), *Discrete mathematics*.

Rousseau, J.-J. (1985), *A discourse on inequality*.

Santos, F. C. and Pacheco, J. M. (2005), 'Scale-free networks provide a unifying framework for the emergence of cooperation', *Physical review letters* **95**(9), 098104.

Woźny, L. (2012), 'Lecture notes on microeconomics'.

List of Tables

List of Figures

Streszczenie

Tutaj zamieszczają Państwo streszczenie pracy. Streszczenie powinno być długości około pół strony.