Studium licencjackie

Kierunek: Metody Ilociowe w Ekonomii i Systemy Informacyjne

Forma studiów: Stacjonarne

Imie i nazwisko autora: Kacper Mordarski

Nr albumu: 101247

<tytu>

Praca licencjacka napisana

w Katedrze Matematyki i Ekonomii Matematycznej

pod kierunkiem naukowym

dr hab. Michaa Ramszy

Warszawa 2022

Contents

1 Introduction **5**

1.1 General Introduction 5

1.2 Description of The Paper 5

2 Elements of the noncooperative game theory **7**

3 Elements of the graph theory **8**

4 Cooperation among the players **9**

4.1 A brief description of Snowdrift and Prisoners Dilemma games 9

4.2 Replicatory dynamics 10

5 Algorithms and simulations **11**

6 Algorithm **11**

6.1 Introduction to the algorithm 11

6.2 Functions 11

6.3 Description of the algorithm 12

6.4 Simulations description 14

6.5 Compiling LaTeXfiles 15

7 Results and discussion **16**

1 Introduction

1.1 General Introduction

This paper aims to replicate, to some degree, the seminal paper of **?** 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 poniej to do poprawki. Trzeba tutaj napisa taki standardowy wstp machany rkami. Koniecznie trzeba si pozby “The paper”, to jednak jest nie do przyjcia.

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.

1.2 Description of The Paper

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 opisa jaka jest struktura caej pracy, w sensie w tym rozdziale to jest to, w tamtym rozdziale to jest tamto. Robimy to na kocu.

2 Elements of the noncooperative game theory

» Tutaj trzeba opisa wszystkie koncepcje matematyczne zwizane z teori gier. Wydaje si, e to co jest nam potrzebne to definicja gry w postaci normalnej i definicja równowagi Nasha. Na koniec zamieci przykady pojedynczych gier (ale niekoniecznie PD i SD. Poza tym warto chyba jednak opisa jaki ogólny framework, w którym gra jest populacyjna, co w takiej sytuacji jest mierzone i jak to ewoluuje w czasie. Myl, e tutaj mog Panu pomóc. Dodaem tutaj dwa cytowania typowe dla modelu: **??**.

3 Elements of the graph theory

» Tutaj musimy opisa wybrane elementy teorii grafów. Myl, e pojcie grafu nieskierowanego oraz wybrane grafy losowe, razem z cytowaniami. Tutaj mona zaczy przykady zarówno grafów losowych ale równie maych grafów losowych tak aby byo jasne jaka jest struktura tych grafów. Doda oczywicie odpowiednie cytowania.

4 Cooperation among the players

» W tym miejscu w kadym rozdziale trzeba wpisa 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:

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

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 powyej grach w one-shot games. Pokaza, e jest albo nie mam kooperacji. Tutaj równie wprowadzamy gr na grafach losowych, a wic musimy opisa na jakich grafach, itd. Tutaj równie opis uczenia si ale w sensie algorytmu matematycznego nie implementacji.

4.2 Replicatory dynamics

1. The site x is updated.
2. A neighbor y is drawn at random among all k_x neighbors
3. if cumulative payoffs of y (P_y) are greater than cumulative payoffs of x (P_x), the chosen neighbor takes over site x with probability (P_i) given below.

5 Algorithms and simulations

This chapter contains a description of the algorithm used in the simulation.
» To co powyej do rozszerzenia. Generalnie ten rozdzia ma zawiera implementacj koncepcji matematycznej z poprzedniego rozdziau.

6 Algorithm

6.1 Introduction to the algorithm

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.

6.2 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.

6.3 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:

1. Each pair of connected players (vertices of a graph) engage in a single round of a given game. It means that function GAMES is applied throughout the entire array of edges, adjusting their accumulated payoffs.
2. One player is chosen randomly.
3. An attempt to change strategy is being embarked on. It indicates that the function CHECKSTRAT is applied to the player chosen in the previous step.
4. If the current iteration is higher than 1000, it calculates the share of cooperators in the entire population and then passes it onto a corresponding value of a vector TRACK.
5. After 1 100 iterations (generations), it ends, and then the next iteration of the outside loop is triggered.

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 ARRPATHS — vector of length 100, used to store the results of the simulations.

The procedure followed by this loop is as shown below:

1. It generates the random Barabasi-Albert graph with N vertices and average connectivity equal to Z. Both parameters are chosen deliberately at the beginning of the code.

2. The next step is constructing an array of strategies. This array is of length 1000 (number of players). The strategies are represented by numbers 1 (cooperation) and 2 (defection). The choice of these numbers is strict because later on, we use them to index the payoffs matrix. The process of creating such an array is following:

   (a) Creating two vectors of length 500, one filled with ones, and one filled with twos.

   (b) Concatenate those two vectors to get a vector of length 1000.

   (c) Shuffle the values in a vector and save them as STRATEGIES.

3. For the inner loop to conduct games, we need to transform our SF NOC into two arrays — one containing edges (tuples of players) and the other containing strategies for each edge. To achieve this, we apply the following steps:

   (a) We use function COLLECT on the function EDGES used on the Barabasi-Albert model, which results in an iterable, however not easily callable, array of edges.

   (b) Then we map a function TRANSFORM onto the previously obtained object. This produces us an array of tuples, each tuple representing one edge.

   (c) Lastly, we map function STRAT onto an array of tuples representing the edges.

4. The array of accumulated payoffs is created — initially with all values equal to zero (after each game, the results are added to the corresponding values in this array).

5. A global variable is created. This variable is named TRACK and it is a vector of length 100. This vector is used by the inner loop to store the proportion of cooperators in the population.

6. The final step before triggering the inner loop is creating ARRPATH, which will be used to store 100 TRACK variables. If it is the first iteration of this loop, we create a global variable named ARRPATH, which is an array. If, on the other hand, it is not the first iteration, we are using the function PUSH! to add the next TRACK to our existing array.

6.4 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.

» Poniej wrzuciem przykadowy algorytm w pseudokodzie. Myl, e to jest to co chcemy wykorzysta aby zapisa sam algorytm. To oczywicie musi by potem opisane w tekcie.

6.5 Compiling LaTeXfiles

Data: this text

Result: how to write algorithm with LaTeX2e

initialization;

while *not at end of this document* **do**

    read current;

    if *understand* **then**

        go to next section;

        current section becomes this one;

    else

        go back to the beginning of current section;

    end

end

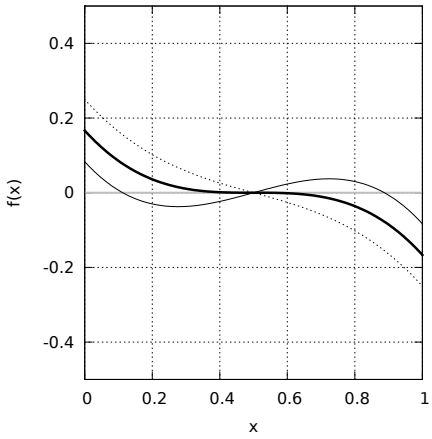
Algorithm 1: How to write algorithms

7 Results and discussion

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

8 Conclussions

» Tutaj na samym kocu dopisujemy dlaczego i co zrobilimy oraz jakie mamy dalsze plany na badania.



List of Tables

List of Figures

Streszczenie

Tutaj zamieszczaj Pastwo streszczenie pracy. Streszczenie powinno by dugoci okoo pó strony.