



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH

FACULTAT D'INFORMÀTICA DE  
BARCELONA (FIB)

BACHELOR DEGREE IN INFORMATICS  
ENGINEERING

SPECIALIZATION IN COMPUTING

---

# Strategic Network Formation with Attacks and Immunization: Dynamics

---

*Author:*

CLAVEROL I GONZÁLEZ,  
Carla

*Supervisor:*

ÀLVAREZ FAURA,  
Carme

DEPARTMENT OF  
COMPUTER SCIENCE

Academic Year 2020-2021

Date 29/06/2021

## **Abstract**

In this project we study the dynamic behaviour of Strategic Network Formation Games with attacks and immunization introduced by Goyal et al. in [8, 9]. This model takes Reachability Network Formation Games introduced by Bala et al. in [4] and introduces an adversarial attack, as well as immunization against the attack. The benefit of each agent is the expected size of her connected component post-attack (as in the Reachability Network Formation Games, forming links has a cost), and agents may choose to immunize at some additional cost.

In this project we expand the experimental study conducted in [9] by Goyal et al. We do a systematical study of the topology of the swapstable equilibria found with different initial configurations.

We then extend the model to a new one in which we give more power to the adversary, and we do an experimental research on the swapstable equilibria found. We again focus on the topologies found with different initial configurations. Finally, we compare the results obtained with both models.

## Resum

En aquest projecte estudiem el comportament dinàmic dels *Strategic Network Formation Games with attacks and immunization* introduïts per Goyal et al. [8, 9]. Aquest model agafa els *Reachability Network Formation Games* introduïts per Bala et al. [4] i introdueix l'atac d'un adversari, així com immunització contra l'atac. El benefici de cada agent és el tamany esperat del seu component connex després de l'atac (com en els *Reachability Network Formation Games*, formar enllaços té un cost), i els agents poden escollir immunitzar-se amb un cost addicional.

En aquest projecte ampliem l'estudi experimental dut a terme en [9] per Goyal et al. Fem un estudi sistemàtic de la topologia dels *swapstable equilibria* trobats amb diferents configuracions inicials.

Després estenem el model a un de nou en què donem més poder a l'adversari, i fem una recerca experimental sobre els *swapstable equilibria* trobats. Ens tornem a centrar en les topologies trobades amb diferents configuracions inicials. Finalment, comparem els resultats obtinguts amb els dos models.

## Resumen

En este proyecto estudiamos el comportamiento dinámico de los *Strategic Network Formation Games with attacks and immunization* introducidos por Goyal et al. en [8, 9]. Este modelo coje los *Reachability Network Formation Games* introducidos por Bala et al. en [4] e introduce el ataque de un adversario, así como inmunización contra el ataque. El beneficio de cada agente es el tamaño esperado de su componente conexa deappués del ataque (como en los *Reachability Network Formation Games*, formar enlaces tiene un coste), y los agentes pueden escojer inumnizarse con un coste adicional.

En este proyecto ampliamos el estudio experimental llevado a cabo en [9] por Goyal et al. Hacemos un estudio sistemático de la topología de los *swapstable equilibria* encontrados con diferentes configuraciones iniciales.

Después extendemos el modelo a uno nuevo en el que damos más poder al adversario, y hacemos una investigación experimental sobre los *swapstable equilibria* encontrados. Nos volvemos a centrar en las topologías encontradas con diferentes configuraciones iniciales. Finalmente, comparamos los resultados obtenidos con los dos modelos.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>  | <b>10</b> |
| 1.1      | Motivation . . . . .   | 10        |
| 1.2      | Project contribution . . . . .   | 11        |
| 1.3      | Project outline . . . . .  | 11        |
| <b>2</b> | <b>Preliminaries: Strategic Games</b>  | <b>13</b> |
| 2.1      | Nash Equilibrium . . . . .   | 14        |
| 2.2      | Example 1: The Prisoner's Dilemma . . . . .  | 14        |
| 2.3      | Example 2: Matching Pennies . . . . .  | 15        |
| 2.4      | Dynamics Graph . . . . .   | 16        |
| 2.5      | Best Response Dynamics Algorithm . . . . .   | 16        |
| <b>3</b> | <b>Strategic Network Formation with a Single Attack and Immunization</b>                           | <b>18</b> |
| 3.1      | Definition . . . . .   | 18        |
| 3.1.1    | Swapstable Equilibrium . . . . .   | 21        |
| 3.1.2    | Swapstable Best Response Dynamics . . . . .  | 22        |
| 3.2      | Summary of previous results . . . . .  | 23        |
| 3.2.1    | Diversity of Equilibria, Sparsity, Connectivity and Social Welfare (Theoretical results) . . . . . | 23        |
| 3.2.2    | Simulations: Dynamics (Experimental results) . . . . .   | 25        |
| <b>4</b> | <b>New Simulations for a Single Attack and Immunization</b>  | <b>28</b> |
| <b>5</b> | <b>Extension of the model: Two attacks and immunization</b>  | <b>45</b> |
| 5.1      | Definition . . . . .   | 45        |
| 5.2      | Simulations: dynamics . . . . .  | 46        |
| 5.3      | Comparison with a single attack . . . . .  | 54        |
| <b>6</b> | <b>Conclusions and Future Work</b>   | <b>64</b> |
|          | <b>Bibliography</b>  | <b>66</b> |

|          |  |           |
|----------|--|-----------|
| <b>A</b> | <b>Project Management</b>              | <b>68</b> |
| A.1      | Temporal planning . . . . .            | 68        |
| A.1.1    | Tasks description . . . . .            | 68        |
| A.1.2    | Dedication to each task . . . . .      | 70        |
| A.2      | Budget . . . . .                       | 70        |
| A.2.1    | Direct costs . . . . .                 | 70        |
| A.2.2    | Indirect costs . . . . .               | 73        |
| A.2.3    | Unplanned costs . . . . .              | 74        |
| A.2.4    | Final budget . . . . .                 | 74        |
| A.3      | Sustainability report . . . . .        | 74        |
| A.3.1    | Environmental sustainability . . . . . | 75        |
| A.3.2    | Economic sustainability . . . . .      | 75        |
| A.3.3    | Social sustainability . . . . .        | 76        |
| <b>B</b> | <b>Code for Simulations</b>            | <b>77</b> |
| B.1      | Graph Class . . . . .                  | 77        |
| B.1.1    | Headers file . . . . .                 | 77        |
| B.1.2    | Source file . . . . .                  | 79        |
| B.2      | Model Class . . . . .                  | 81        |
| B.2.1    | Headers file . . . . .                 | 81        |
| B.2.2    | Source file . . . . .                  | 92        |
| B.3      | Main . . . . .                         | 104       |
| B.4      | Graph viewer . . . . .                 | 105       |

# List of Tables

|     |  |    |
|-----|--|----|
| 2.1 | The utilities of the four configurations of <i>The Prisoner's Dilemma</i>  | 15 |
| 2.2 | The utilities of the four configurations of <i>Matching Pennies</i>  | 16 |
| 4.1 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversary, for $n = 30$ , with different initial configurations and different $C_E$ and $C_I$ .                                      | 32 |
| 5.1 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversary which attacks two players, for $n = 30$ , with different initial configurations and different $C_E$ and $C_I$ .            | 49 |
| 5.2 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 0.5$ and $C_I = 0.5$ . | 55 |
| 5.3 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 0.5$ and $C_I = 2$ .   | 56 |
| 5.4 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 0.5$ and $C_I = 20$ .  | 57 |
| 5.5 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 0.5$ and $C_I = 50$ .  | 57 |
| 5.6 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 2$ and $C_I = 0.5$ .   | 58 |

|      |  |    |
|------|--|----|
| 5.7  | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 2$ and $C_I = 2$ . . . . .     | 59 |
| 5.8  | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 2$ and $C_I = 20$ . . . . .    | 60 |
| 5.9  | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 2$ and $C_I = 50$ . . . . .    | 61 |
| 5.10 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 10$ and $C_I = 0.05$ . . . . . | 61 |
| 5.11 | Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for $n = 30$ , with different initial configurations and $C_E = 10$ and $C_I = 0.5$ . . . . .  | 62 |
| A.1  | Number of hours spent on each task. . . . .  | 70 |
| A.2  | Number of hours spent on each task by each role. . . . .   | 71 |
| A.3  | Cost of the human resources. . . . .   | 71 |
| A.4  | Cost of hardware resources. . . . .  | 72 |
| A.5  | Cost of software resources. . . . .  | 73 |
| A.6  | Indirect costs. . . . .  | 73 |
| A.7  | Unplanned costs. . . . .   | 74 |
| A.8  | Final budget of the project. . . . .   | 74 |



# List of Figures

|      |   |    |
|------|---|----|
| 3.1  | Vulnerable regions $\mathcal{V}_1$ , $\mathcal{V}_2$ and $\mathcal{V}_3$ . Blue nodes denote $\mathcal{I}$ and red nodes denote $\mathcal{U}$ . . . . .   | 20 |
| 3.2  | Examples of Nash equilibria with respect to the maximum carnage adversary. (3.2a) Hub-spoke equilibrium, $C_E = 1$ and $C_I = 1$ ; (3.2b) forest equilibrium, $C_E = 1$ and $C_I = 9$ ; (3.2c) cycle equilibrium, $C_E = 1.5$ and $C_I = 3$ ; (3.2d) 4-petal flower equilibrium, $C_E = 0.1$ and $C_I =$ ; (3.2e) complete bipartite equilibrium, $C_E = 0.1$ and $C_I = 4$ . . . . . | 24 |
| 3.3  | Swapstable equilibria for $n = 50$ . (3.3a) $C_E = 0.5$ and $C_I = 2$ ; (3.3b) $C_E = 2$ and $C_I = 2$ ; (3.3c) $C_E = 0.5$ and $C_I = 20$ . . . . .  | 26 |
| 4.1  | Swapstable equilibrium for $n = 30$ , 30 initial edges, $p = 0.25$ , $C_E = 0.5$ and $C_I = 0.5$ . . . . .  | 33 |
| 4.2  | Swapstable equilibrium for $n = 30$ , 10 initial edges, $p = 0.25$ , $C_E = 0.5$ and $C_I = 2$ . . . . .  | 33 |
| 4.3  | Swapstable equilibrium for $n = 30$ , 30 initial edges, $p = 0.75$ , $C_E = 0.5$ and $C_I = 2$ . . . . .  | 34 |
| 4.4  | Swapstable equilibrium for $n = 30$ , 10 initial edges, $p = 0.50$ , $C_E = 0.5$ and $C_I = 20$ . . . . .   | 35 |
| 4.5  | Swapstable equilibrium for $n = 30$ , 30 initial edges, $p = 0.50$ , $C_E = 0.5$ and $C_I = 20$ . . . . .   | 36 |
| 4.6  | Swapstable equilibria for $n = 30$ , 10 initial edges, $p = 0.25$ and $C_I = 0.5$ . . . . .   | 37 |
| 4.7  | Swapstable equilibria for $n = 30$ , 10 initial edges, $p = 0.25$ and $C_I = 2$ . . . . .   | 37 |
| 4.8  | Swapstable equilibria for $n = 30$ , 10 initial edges, $p = 0.25$ and $C_I = 20$ . . . . .  | 38 |
| 4.9  | Swapstable equilibria for $n = 30$ , 10 initial edges and $p = 0.50$ . . . . .  | 39 |
| 4.10 | Swapstable equilibria for $n = 30$ , 30 initial edges, $p = 0.25$ and $C_I = 20$ . . . . .  | 40 |

|      |   |    |
|------|---|----|
| 4.11 | Swapstable equilibria for $n = 30$ , 30 initial edges, $p = 0.50$ and $C_I = 20$ . . . . .  | 40 |
| 4.12 | Swapstable equilibria for $n = 30$ , 30 initial edges, $p = 0.75$ and $C_I = 20$ . . . . .  | 41 |
| 4.13 | Swapstable equilibrium for $n = 30$ , 10 initial edges, $p = 0.25$ , $C_E = 2$ and $C_I = 50$ . . . . .   | 42 |
| 5.1  | Swapstable equilibrium for $n = 30$ , 10 initial edges, $p = 0.25$ , $C_E = 0.5$ and $C_I = 0.5$ , with respect to the maximum carnage adversary which attacks two nodes. . . . .   | 49 |
| 5.2  | Swapstable equilibrium for $n = 30$ , 10 initial edges, $p = 0.25$ , $C_E = 0.5$ and $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes. . . . .  | 50 |
| 5.3  | Swapstable equilibrium for $n = 30$ , 30 initial edges, $p = 0.75$ , $C_E = 0.5$ and $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes. . . . .  | 51 |
| 5.4  | Swapstable equilibria for $n = 30$ , 10 initial edges, $p = 0.25$ and $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes. . . . .   | 52 |
| 5.5  | Swapstable equilibria for $n = 30$ , 30 initial edges, $p = 0.75$ and $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes. . . . .   | 52 |
| 5.6  | Swapstable equilibrium for $n = 30$ , 30 initial edges, $p = 0.75$ , $C_E = 10$ and $C_I = 0.05$ , with respect to the maximum carnage adversary which attacks two nodes. . . . .   | 53 |
| 5.7  | (a) The cycle is a swapstable equilibrium when $C_E = 1.5$ and $C_I = 3$ and the adversary makes a single attack. (b) Swapstable equilibrium when $C_E = 1.5$ and $C_I = 3$ and the adversary makes two attacks, starting from the cycle. . . . . | 63 |

# Chapter 1

## Introduction

In this chapter we explain the motivation in conducting this research, the contribution of this project and how this manuscript is structured.

### 1.1 Motivation

*Game Theory* is a branch of applied mathematics that studies models where a set of agents interact with each other trying, each one of them, to get as much benefit as she can, but this benefit is affected by the choices of other agents. Game Theory has applications in areas such as economics, computer science, biology...

The intersection of Game Theory and Computer Science is called *Algorithmic Game Theory*, an example of which are the so-called *Network Creation Games*, which study models where agents receive benefits from being connected to others but also pay a cost to form these links. The classic Network Creation Games model is proposed by Fabrikant et al. [6]. In this model, agents take advantage of the shortest path to all the other agents. Another Network Creation Game model is the one introduced by Bala et al. in [4], the *Reachability Network Creation Game*, in which the focus is on reachability rather than on centrality: the agents have a benefit equal to the size of their connected component in the graph formed by the agents and their links.

In this project we focus on *Network Creation Games with attacks and immunization*. This model, presented by Goyal et al. in [8, 9], modifies the Reachability Network Creation Game by introducing an adversary which attacks a player. This player dies and the attack spreads through her neighbours, who die as well. Goyal et al. also introduce the concept of immuniza-

tion. Each player can choose whether to pay an immunization cost which would protect her from dying.

This model can be applied to the Internet, where a virus attacks agents. Agents may reduce the risks by paying a cost to buy an anti-virus. It can also be applied to biology, where the agents are the humans, and the links among them are physical proximity or contact. Then a biological virus (as a current example we have the COVID-19) attacks agents, who again may have the option to reduce risks via a vaccine.

In this work we study the dynamics of the model. This means the agents choose their actions in turns, and they change their actions sequentially until we arrive to what is called an *equilibrium*, a situation where no agent wants to change her strategy, because otherwise she would not increase her benefit.

We are interested in studying how the agents behave in different situations, which are given by the cost of forming links and immunizing and different initial configurations (with a certain number of links and immunized agents before starting the dynamics). We also wonder what would happen if the adversary had more power, attacking more agents.

## 1.2 Project contribution

The contribution of this project may be divided in two parts:

- The expansion of the experimental study of the dynamics of the Network Formation Game with Attack and Immunization model conducted in [9]. We do a more systematical study of the topology of the equilibria found.
- The introduction of an extension of the previously mentioned model, in which the adversary attacks two players: the *Network Formation Game with two Attacks and Immunization* model. We do an experimental study of the dynamics of this model, and we compare our results with the ones obtained with a single attack.

## 1.3 Project outline

We structure this document in the following way:

- In Chapter 2 we introduce the the necessary definitions and concepts of Game Theory to understand this project.
- In Chapter 3 we give a detailed description of the Network Formation Game with Attack and Immunization model presented in [8, 9] and we summarize the main results in the mentioned work.
- In Chapter 4 we do a experimental study of the dynamics of the model.
- In Chapter 5 we present the Network Formation Game with two Attacks and Immunization and we do a experimental study of the dynamics of this model. We also compare this model with the one with a single attack.
- Finally, in Chapter 6 we summarize the main contributions of the project and the open problems.
- In Appendix A we explain the planning of the project, its costs and its environmental, social and economic impact.
- In Appendix B we give the implementation of the code used to do the simulations.

## Chapter 2

# Preliminaries: Strategic Games

In this chapter we introduce the basic concepts of Game Theory needed to understand this project. Let us first define *Strategic Games*.

Strategic Games model the interactions between a set of individuals, called *players*. Each player has a set of possible *actions* to choose. Each player receives a *utility* (or pays a *cost*) depending on the actions of the whole set of players.

**Definition 1.** *Formally, a Strategic Game is defined as a tuple*

$$\Gamma = \langle V, (\mathcal{S}(i))_{i \in V}, (u_i)_{i \in V} \rangle$$

where

- $V = \{1, \dots, n\}$  is the set of players.
- $\mathcal{S}(i)$  is the set of actions that are available to player  $i \in V$ .

A strategy of player  $i$  consists in selecting an action  $s_i \in \mathcal{S}(i)$ . The strategies chosen by all the players form the strategy profile  $\mathbf{s} = (s_1, \dots, s_n)$ .

We denote the set of all possible strategy profiles as  $\mathcal{S}(\Gamma) = \mathcal{S}(1) \times \dots \times \mathcal{S}(n)$ .

For a strategy profile  $\mathbf{s}$ , we denote by  $s_{-i}$  the set of strategies  $(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$ .

Therefore,  $\mathbf{s} = (s_{-i}, s_i)$ . We denote by  $(s_{-i}, s'_i)$  the strategy profile obtained from  $\mathbf{s}$  in which player  $i$  has changed her strategy  $s_i$  for  $s'_i$ ,  $(s_{-i}, s'_i) = (s_1, \dots, s_{i-1}, s'_i, s_{i+1}, \dots, s_n)$

- For each player  $i \in V$ ,  $u_i : \mathcal{S}(\Gamma) \rightarrow \mathbb{R}$  is her utility function. The objective of each player is to maximize her utility (or to minimize her cost).

## 2.1 Nash Equilibrium

An important concept in relation to Strategic Games is the concept of *Nash Equilibrium (NE)*. In some games there is a strategy profile where no player has an incentive to change her strategy, because by doing this she would not strictly increase her current utility.

**Definition 2.** *Formally, given a strategic game  $\Gamma = \langle V, (\mathcal{S}(i))_{i \in V}, (u_i)_{i \in V} \rangle$  we say that a strategy profile  $\mathbf{s} \in \mathcal{S}(\Gamma)$  is a Nash Equilibrium if*

$$\forall i \in V, \forall s'_i \in \mathcal{S}(i), u_i(\mathbf{s}) \geq u_i(s_{-i}, s'_i)$$

. We denote by  $NE(\Gamma)$  the set of NE strategies.

Each player acts selfishly, so each one will choose a strategy that maximizes her utility given the strategies of the other players. We define the set of such strategies as the player's *Best Responses*.

**Definition 3.** *Formally, given a strategic game  $\Gamma = \langle V, (\mathcal{S}(i))_{i \in V}, (u_i)_{i \in V} \rangle$  and a strategy profile  $\mathbf{s} \in \mathcal{S}(\Gamma)$ , the set of Best responses of player  $i$  to  $s_{-i}$  is:*

$$BR(\Gamma, \mathbf{s}, i) = \{s_i \in \mathcal{S}(i) | \forall s'_i \in \mathcal{S}(i), u_i(s_{-i}, s_i) \geq u_i(s_{-i}, s'_i)\}$$

## 2.2 Example 1: The Prisoner's Dilemma

We will now present a classical example to clarify the concepts we have introduced, the *Prisoner's Dilemma*:

*Two crime suspects are judged simultaneously, with no means of communicating with one another. The police has evidences of them committing a minor crime, and they suspect the suspects have committed a major crime, but they do not have evidences unless one of them betrays the other one. If they both remain quiet, each one will spend 1 year in prison for the minor crime. If one of them betrays the other one, he will be free and his partner will be convicted for the major crime, so he will spend 4 years in prison. If they both betray their partner, they will be convicted for the major crime, but because of their collaboration, they will spend 3 years in prison.*

The game is represented by  $\Gamma = \langle V = \{1, 2\}, (\mathcal{S}(1), \mathcal{S}(2)), (u_1, u_2) \rangle$ , where

- The players of the game are the two criminals.

- $\mathcal{S}(1) = \mathcal{S}(2) = \{Quiet, Fink\}$ . Therefore,  $\mathcal{S}(\Gamma) = \mathcal{S}(1) \times \mathcal{S}(2) = \{(Quiet, Quiet), (Quiet, Fink), (Fink, Quiet), (Fink, Fink)\}$ .
- We can represent the utility of each player as the years lost, that is, the years they would have to spend in prison:  
 For player 1 we have that  $u_1(Quiet, Quiet) = -1$ ,  $u_1(Quiet, Fink) = -4$ ,  $u_1(Fink, Quiet) = 0$ ,  $u_1(Fink, Fink) = -3$ .  
 For player 2 we have that  $u_2(Quiet, Quiet) = -1$ ,  $u_2(Quiet, Fink) = 0$ ,  $u_2(Fink, Quiet) = -4$ ,  $u_2(Fink, Fink) = -3$ .  
 It can be represented using the following table:

|       | Quiet  | Fink   |
|-------|--------|--------|
| Quiet | -1, -1 | -4, 0  |
| Fink  | 0, -4  | -3, -3 |

Table 2.1: The utilities of the four configurations of *The Prisoner's Dilemma*

We can see that the only situation of equilibrium is when they both fink. Otherwise, at least one player can increase her utility going from staying quiet to finking. Even so, the social optimum occurs when they both stay quiet.

## 2.3 Example 2: Matching Pennies

In the Prisoner's Dilemma we have found a Nash Equilibrium. We will now see that not all games have Nash Equilibria. Let us present another game, *Matching Pennies*:

*There are two players, and each one chooses either the Head or the Tail of a coin. If the choices differ the first player pays the second player a dollar. If they are the same, the second player pays the first one a dollar.*

The game is represented by  $\Gamma = \langle V = \{1, 2\}, (\mathcal{S}(1), \mathcal{S}(2)), (u_1, u_2) \rangle$ , where

- There are two players.
- $\mathcal{S}(1) = \mathcal{S}(2) = \{Head, Tail\}$ . Therefore,  $\mathcal{S}(\Gamma) = \mathcal{S}(1) \times \mathcal{S}(2) = \{(Head, Head), (Head, Tail), (Tail, Head), (Tail, Tail)\}$ .



- We represent the utility of each player as the dollars she receives:

We can see it in the following table:

|      | Head  | Tail  |
|------|-------|-------|
| Head | 1, -1 | -1, 1 |
| Tail | -1, 1 | 1, -1 |

Table 2.2: The utilities of the four configurations of *Matching Pennies*

We can see that there is no situation of NE: if the strategy profile is (Head, Head) or (Tail, Tail) the player 2 can change her strategy and improve her utility and if the strategy profile is (Head, Tail) or (Tail, Head) the player 1 can change her strategy improving her utility.

## 2.4 Dynamics Graph

We now present the concept of the *dynamics graph* of a strategic game.

**Definition 4.** *The (best response) dynamics graph of a strategic game  $\Gamma = \langle V, (\mathcal{S}(i))_{i \in V}, (u_i)_{i \in V} \rangle$  is a directed graph where each node represents a strategy profile  $\mathbf{s}$  and there exists an edge  $(\mathbf{s}, (s_{-i}, s'_i))$  if  $s_i \notin BR(\Gamma, \mathbf{s}, i)$  and  $s'_i \in BR(\Gamma, \mathbf{s}, i)$ , for some player  $i \in V$ .*

A Nash equilibrium is a *sink* (a node with out-degree 0) in the graph.

Note that this graph can have cycles. In this case, the game may not have a Nash equilibrium. Nevertheless, yet in the case that the graph has cycles, there may exist a Nash equilibrium, that is, a sink outside the cycles.

## 2.5 Best Response Dynamics Algorithm

We can consider doing a local search in the dynamics graph: we start from an initial configuration (a strategy profile  $\mathbf{s}$ , represented by a node in the graph), and we follow up paths in the graph until we eventually arrive to a Nash equilibrium (a sink in the graph). This is the *best response dynamics algorithm*.

Note that in order to follow up paths we need to define a rule for choosing which node comes next in the path, that is, which  $(s_{-i}, s'_i), s'_i \in BR(\Gamma, \mathbf{s}, i), i \in V$ . To do that, we need to define in which player  $i$  we focus and, fixing  $i$ ,

which strategy  $s'_i \in BR(\Gamma, \mathbf{s}, i)$  we choose.

In order to define in which player we focus each time, we proceed in *rounds*, and in each round, we focus in one player.

## Chapter 3

# Strategic Network Formation with a Single Attack and Immunization

In this chapter we present the *Strategic Network Formation with Attacks and Immunization* model, defined in [8, 9], article in which we base our project. We also summarize the theoretical and experimental results presented in the mentioned work.

### 3.1 Definition

We first introduce the model presented in [8, 9].

In this model we take the *Reachability* Network Formation Game presented in [4] and we introduce an adversary, who attacks a player, and the possibility for the players to immunize, or protect, themselves.

We model our network as a graph  $G = (V, E)$ , where  $V = \{1, \dots, n\}$  is the set of players. Each player may purchase edges to other players at a cost of  $C_E > 0$  per edge and, in addition, she may also immunize herself at a cost of  $C_I > 0$ .

A strategy for player  $i$ ,  $s_i$ , is a pair consisting of the the set of players  $i$  buys an edge to, formally denoted as  $x_i \subseteq \{1, \dots, n\}$ , and her immunization choice, formally denoted as the binary variable  $y_i \in \{0, 1\}$  ( $y_i = 1$  when  $i$  immunizes). Therefore,  $s_i = (x_i, y_i)$ . In this model players do not need reciprocation in order to purchase an edge to another, but the connectivity

benefits and risks affect both players.

Fixing  $\mathbf{s} = (s_1, \dots, s_n)$  (the strategy profile for all players), the set of the edges purchased by all the players induces an undirected graph,  $G = (V, E)$ , and the set of immunized players forms a bipartition of the vertices,  $\mathcal{I} \subseteq V$ . A *game state* is the pair  $(G, \mathcal{I})$ .

The set of *vulnerable vertices*, that is, the players who do not immunize, is denoted as  $\mathcal{U} = V \setminus \mathcal{I}$ . A *vulnerable region*,  $\mathcal{V}_i$ , is a subset of vertices of  $\mathcal{U}$  that form a maximally connected component. The set of vulnerable regions is  $\mathcal{V} = \{\mathcal{V}_1, \dots, \mathcal{V}_k\}$ .

Fixing  $(G, \mathcal{I})$ , the adversary chooses to attack a vertex. If she attacks a vulnerable vertex  $v \in \mathcal{U}$ , then the attack spreads, killing  $v$  and all the vulnerable vertices reachable from  $v$ . The attack cannot spread through immunized vertices.

The adversary is specified by a function that defines a probability distribution over vulnerable regions. A vulnerable region with non-zero probability of attack is a *targeted region*,  $\mathcal{T}' \in \mathcal{T}$ , where  $\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_{k'}\}$  is the set of targeted regions.<sup>1</sup> The vulnerable vertices inside of a targeted region are called *targeted vertices*.

If there are no targeted regions, i.e.,  $\mathcal{T} = \emptyset$ , the adversary does not make any attack, so player  $i$ 's utility is the size of her connected component,  $CC_i$ , minus her expenses (edge purchases and immunization). When  $|\mathcal{T}| > 0$ , the expected utility of player  $i$  is the expected size of her connected component post-attack<sup>2</sup> minus her expenses. Formally, let  $Pr[\mathcal{T}']$  be the probability of attack to  $\mathcal{T}'$  and  $CC_i(\mathcal{T}')$  the size of player  $i$ 's connected component after the attack to  $\mathcal{T}'$ . Then the expected utility of  $i$  in the strategy profile  $\mathbf{s}$ ,  $u_i(\mathbf{s})$  is

$$u_i(\mathbf{s}) = \sum_{\mathcal{T}' \in \mathcal{T}} (Pr[\mathcal{T}'] CC_i(\mathcal{T}')) - |x_i| C_E - y_i C_I$$

Taking a strategy profile  $\mathbf{s}$ , the sum of the expected utilities of all the

---

<sup>1</sup>Since every targeted region is vulnerable,  $k' \leq k$ .

<sup>2</sup>The size of the connected component of a vertex who is killed is zero.

players, i.e.,

$$\sum_{i \in V} u_i(\mathbf{s})$$

is called the social welfare of  $\mathbf{s}$ .

The probability of attack to a targeted region (therefore, the expected utility of the players) depends on the adversary's choice of attack, which is a distribution on  $\mathcal{T}$ . We could think of multiple different adversaries with different strategies, but in this work we will focus in the *maximum carnage* adversary:

**Definition 5.** *The maximum carnage adversary attacks the vulnerable region of maximum size. If there are more than one, she attacks one of them uniformly at random. Once the attacked vulnerable region is selected, the adversary chooses a vertex inside this region uniformly at random to start the attack.*

Therefore, with respect to the maximum carnage adversary, the targeted regions are the vulnerable regions of maximum size, and the distribution on  $\mathcal{T}$  is a uniform distribution. For example, in figure 3.1 (figure borrowed from [8, 9]), the targeted regions are  $\mathcal{V}_1$  and  $\mathcal{V}_3$ , because they are the vulnerable regions of maximum size. The probability of attack to both of them is 0.5. The probability of attack to  $\mathcal{V}_2$  is, therefore, 0:

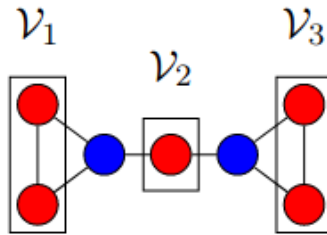


Figure 3.1: Vulnerable regions  $\mathcal{V}_1$ ,  $\mathcal{V}_2$  and  $\mathcal{V}_3$ . Blue nodes denote  $\mathcal{I}$  and red nodes denote  $\mathcal{U}$ .

Taking this into account, we can write the expected utility of  $i$  in the

strategy profile  $\mathbf{s}$  with respect to the maximum carnage adversary as

$$u_i(\mathbf{s}) = \frac{1}{|\mathcal{T}|} \left( \sum_{\mathcal{T}' \in \mathcal{T}} CC_i(\mathcal{T}') \right) - |x_i|C_E - y_i C_I$$

In [8, 9] they present two more adversaries:

- The *random attack* adversary attacks a vulnerable vertex at random. In figure 3.1, all vulnerable regions are targeted regions; the probability of attack to  $\mathcal{V}_1$ ,  $\mathcal{V}_2$  and  $\mathcal{V}_3$  is 0.4, 0.2 and 0.4, respectively.
- The *maximum disruption* adversary attacks the vulnerable region which minimizes the post-attack social welfare. If there are more than one, she attacks one of them uniformly at random. Once the attacked vulnerable region is selected, the adversary chooses a vertex inside this region uniformly at random to start the attack. In figure 3.1, the only targeted region is  $\mathcal{V}_2$ , with probability of being attacked 1. The probability of attack to  $\mathcal{V}_1$  and  $\mathcal{V}_3$  is, therefore, 0.

### 3.1.1 Swapstable Equilibrium

Remember that a strategy profile  $\mathbf{s}$  is a Nash equilibrium if no player  $i$  can strictly increase her expected utility by changing her strategy  $s_i = (x_i, y_i)$ , that is, by changing the set of players  $i$  buys an edge to,  $x_i$ , and/or her immunization choice,  $y_i$ .

We now introduce a new type of equilibrium, the *swapstable equilibrium* (first introduced in [10]). A strategy profile  $\mathbf{s}$  is a swapstable equilibrium if no player can strictly improve her expected utility under any of the following *swap deviations*:

1. Dropping any single purchased edge.
2. Purchasing any single unpurchased edge.
3. Dropping any single purchased edge and purchasing any single unpurchased edge (swapping edges).
4. Making any one of the deviations above and, in addition, changing the immunization status.

In [8, 9], Goyal et al. also work with the *linkstable equilibrium*, which is similar to the swapstable equilibrium, but with only the deviations (1), (2) and (4), that is, without swapping edges (see [5]). Swapstable networks, nevertheless, share properties with Nash networks that linkstable networks do not (see Section 3.2.1). This is why in the simulations in [9] Goyal et al. work with swapstable equilibria instead of with linkstable equilibria. We will do the same, so we can compare our results with theirs.

Note that every Nash equilibrium is a swapstable equilibrium and every swapstable equilibrium is a linkstable equilibrium, but the reverse of these statements are not true in our game.

### 3.1.2 Swapstable Best Response Dynamics

In this project we focus on the *swapstable best response dynamics* in our game.

We proceed in rounds. Each round consists in a *swapstable best response update* for each one of the players in some order. For each player  $i$ :

- We fix the strategies (edge and immunization purchases) of all the other players, that is,  $s_{-i}$ .
- We compute the set of swapstable best responses of player  $i$  to  $s_{-i}$ , which is formed by the strategies that we can reach from  $s_i$  applying one of the swap deviations mentioned in 3.1.1 that maximize the expected utility.
- We pick one of the strategies of the set of swapstable best responses, which we denote by  $s'_i$ .
- We change the strategy  $s_i$  by  $s'_i$ . Therefore, the strategy profile becomes  $(s_{-i}, s'_i)$ .

We proceed with the rounds until we eventually arrive to a swapstable equilibrium.

Note that, in order to define the dynamics, we need to define too a *turn policy*, that is, the order in which the players update their strategy, and a rule for breaking swapstable best response ties, that is, to choose  $s'_i$  from the set of swapstable best responses. In this work we always follow the same order for the rounds, following a Round-Robin policy. To break swapstable

best response ties, we always choose the first strategy from the set.

At this point, we want to remark that, as shown in [9], in this game the swapstable (and also Nash and linkstable) best response dynamics graph can have cycles. It seems this phenomenon is the result of a worst-case rule for breaking swapstable best response ties and/or a worst-case turn policy. Despite this possibility that the dynamics graph has cycles, we will see that in this work and in [9], with a fixed round policy and tie-breaking rule, the simulations always converge to an equilibrium.

Note that the swapstable best response dynamics is computable in polynomial time. In [9] Goyal et al. leave open the question *is Nash best response dynamics computable in polynomial time?* Later, in [7] Friedrich et al. positively resolve this question. In [11] Zhang provides an implementation of the Nash best response. However, in this work we consider only the swapstable best response dynamics, because at the moment of starting this project there was no accessible implementation of the Nash best response. Besides, we want to contrast our results with the ones provided in [9], and they work with the swapstable best response dynamics, so it is logic to do the same.

## 3.2 Summary of previous results

In this section we summarize the theoretical and experimental results of the previously mentioned work, [8, 9], so that we can later expand these results.

### 3.2.1 Diversity of Equilibria, Sparsity, Connectivity and Social Welfare (Theoretical results)

In [8, 9], Goyal et al. reach interesting theoretical conclusions about the diversity, the sparsity and the connectivity and social welfare of Nash, swapstable and linkstable equilibrium networks of our game. We expose now the most relevant ones.

**Diversity of Equilibrium Networks** They give some examples of Nash equilibrium networks with respect to the maximum carnage adversary:

- Empty graphs. They show that empty graphs with all immunized vertices and empty graphs with all vulnerable vertices can form in equilibria of our game.



- Trees. In the game with no attacks, [4], the only non-empty equilibria are trees. They show in this game we can find these equilibria too. They focus on the hub-spoke, which is interesting because the network is very efficient (it has the minimum number of edges to be a connected graph and has only one immunized node) and the social welfare is high.
- Forests. They find that, in our game, non-empty networks with multiple connected components and no immunized vertices can be formed at equilibria.
- Cycles. They also find that, unlike in the game with no attacks, [4], cycles can be formed in equilibria of our game.
- Flowers. They next show that multiple cycles can also form, and they illustrate this phenomenon with the flower equilibrium.
- Complete bipartite graphs. Finally, they show that specific complete bipartite graphs can also form in equilibria of our game.

We now show the examples they provide of the equilibrium networks enumerated above<sup>3</sup> (figure borrowed from [9]):

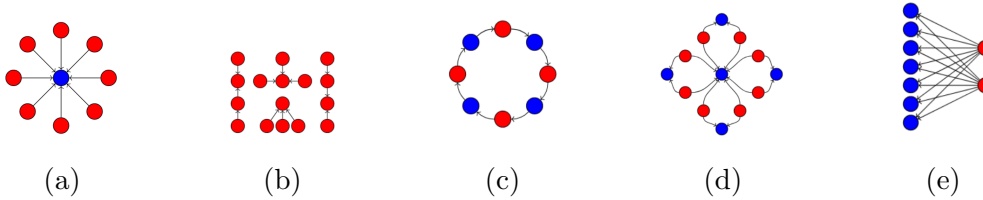


Figure 3.2: Examples of Nash equilibria with respect to the maximum carnage adversary. (3.2a) Hub-spoke equilibrium,  $C_E = 1$  and  $C_I = 1$ ; (3.2b) forest equilibrium,  $C_E = 1$  and  $C_I = 9$ ; (3.2c) cycle equilibrium,  $C_E = 1.5$  and  $C_I = 3$ ; (3.2d) 4-petal flower equilibrium,  $C_E = 0.1$  and  $C_I =$ ; (3.2e) complete bipartite equilibrium,  $C_E = 0.1$  and  $C_I = 4$ .

**Sparsity** Despite the existence of equilibria containing cycles (as shown before) they prove that, under a slight restriction on the adversary, Nash, swapstable and linkstable equilibrium networks are quite sparse.

<sup>3</sup>Throughout the work, we represent in blue immunized nodes and in red, vulnerable nodes. We treat our networks as undirected graphs (because, as we said in Section 3.1, the connectivity benefits and risks are bilateral), but nevertheless we use directed edges in our figures to represent which player purchased the edge.

First, they make a restriction on the adversary. To do so, they define equivalence classes for networks: two networks are equivalent if the connected component of all their vertices is the same in both networks for every possible choice of initial attack vertex. Then, they make the restriction: an adversary is *well-behaved* if on any pair of equivalent networks, the probability that a vertex is chosen for attack is the same.

Then, they show that in Nash, swapstable and linkstable equilibrium networks all vulnerable regions are trees if the adversary is well-behaved.

Finally, they show that Nash, swapstable and linkstable equilibrium networks with  $n \geq 4$  players have at most  $2n - 4$  edges for any well-behaved adversary. Also, this upper bound on the density of equilibria is tight, as they find equilibria with exactly  $(2n - 4)$  edges, e.g., the one in Figure 3.2e.

**Connectivity and Social Welfare in Equilibria** They focus on the maximum carnage adversary, provide some theoretical results about connectivity in equilibria and prove that, if  $C_E > 1$ , social welfare is  $\Omega(n^2)$ , all under the assumption that the equilibria are *non-trivial*, that is, they contain at least one edge and at least one immunized node. Note that this assumption is necessary, as for example the empty graph has a social welfare of only  $O(n)$ .

First, they show that, when  $C_E > 1$ , in any component with at least one immunized vertex and at least one edge of a non-trivial Nash or swapstable equilibrium network with respect to the maximum carnage adversary, if there exist targeted regions, they are singletons.

Then, they show that any non-trivial Nash, swapstable or linkstable equilibrium network with respect to the maximum carnage adversary is connected when  $C_E > 1$ .

Finally, they show that the welfare of any non-trivial Nash or swapstable equilibrium network with  $n$  players, with respect to the maximum carnage adversary, is  $n^2 - O(n^{5/3})$  if  $C_E$  and  $C_I$  are constants and  $C_E > 1$ .

### 3.2.2 Simulations: Dynamics (Experimental results)

We finish this section by presenting the experimental results provided in the previously mentioned work, [9]. In this, Goyal et al. investigate various properties of the swapstable best response dynamics with respect to the maximum

carnage adversary, which we summarize now.

They focus their experiments on studying the convergence speed of the dynamics, the topology of the equilibria, the evolution of the dynamics (the number of edges, number of immunizations and average welfare per player, per round) and the degree distribution on equilibrium.

**Topology** They show (Figure 3.3, borrowed from [9]) the topology of different equilibria reached for different  $C_E$  and  $C_I$  and a fixed number of players, starting with a very sparse and fragmented graph:

- With inexpensive edges and not very expensive immunization ( $C_E = 0.5$  and  $C_I = 2$ ) they often find equilibria with a long cycle formed by both vulnerable and immunized nodes, having some of these last ones some vulnerable children. This is because players want to buy edges even to isolated vertices as long as they do not make them more vulnerable to attack. (Figure 3.3a).
- With more expensive edges and the same immunization cost ( $C_E = 2$  and  $C_I = 2$ ) cycles are less common because of the higher  $C_E$ . They use to see trees formed by immunized nodes connected among them, each with some vulnerable children. (Figure 3.3b).
- With inexpensive edges and very expensive immunization ( $C_E = 0.5$  and  $C_I = 20$ ), they reach trivial equilibria: fragmented graphs with no immunizations. (Figure 3.3c).

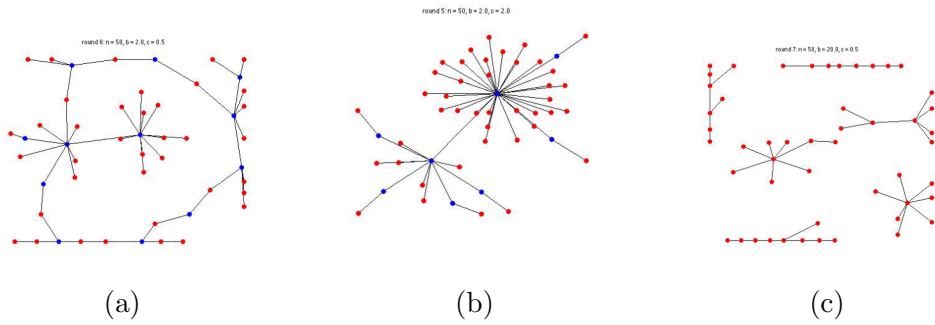


Figure 3.3: Swapstable equilibria for for  $n = 50$ . (3.3a)  $C_E = 0.5$  and  $C_I = 2$ ; (3.3b)  $C_E = 2$  and  $C_I = 2$ ; (3.3c)  $C_E = 0.5$  and  $C_I = 20$ .

**Convergence speed** They find that, even though theoretically the swapstable best response dynamics graph can have cycles and, therefore, the

dynamics algorithm may not converge (as we have said in Section 3.1.2), empirically it always converges pretty quickly.

**Evolution of the dynamics** They conclude that for high  $C_E$  and  $C_I$  players drop edges all the way to the empty graph and immunizations are never purchased; therefore, social welfare is low. For low  $C_E$  and  $C_I$  connectivity stabilizes near a threshold and immunizations grow at the beginning and then decline; social welfare grows a lot when the benefits of immunization are noticed.

**Degree distribution** Finally, they find that, for many parameters, the maximum degree of the nodes is a lot higher than the average degree.

## Chapter 4

# New Simulations for a Single Attack and Immunization

In this chapter we extend the experiments presented in [9]. We want to go deeper into the model presented in the former chapter, to fully understand its behavior. We think that in the paper in which we base our work, [9], there are few simulations, so we find the need to enlarge the experimental part to do a complete study of the model. We focus on the topology of the swap-stable equilibria: we study different equilibria with different edge costs,  $C_E$ , and immunization costs,  $C_I$ , and starting from different initial graphs. We carry out a more systematic study of the resulting configurations in equilibria. As we will see next, we try more different values for the parameters of the simulations and we even introduce a new parameter. Remember we focus on the maximum carnage adversary, as Goyal et al. did in [9].

We experiment with different initial configurations and different parameters. Specifically, our simulation allows the designation of the following parameters: number of players  $n$ ; initial number of edges  $|E|$ ; probability  $p$  of a player to be immunized at the beginning; edge cost  $C_E$ ; and immunization cost  $C_I$ . The first three parameters configure an initial graph, to which we apply the dynamics with different  $C_E$  and  $C_I$ . We could think that for the initial configuration we only need the number of players. Nevertheless, it is completely necessary to fix an initial edge density. Note that for any  $C_E \geq 1$ , the empty graph is a Nash equilibrium and, therefore, a swapstable equilibrium. Thus in order to study the dynamics, we need to start with some initial connectivity. Furthermore, in this work we introduce the parameter  $p$ , which is not in the original paper, because we find interesting to study if there exist variations in the results depending on the number of immunized players at the beginning.

We present in this chapter six initial graphs, with the following values for the parameters:

- The number of players,  $n$ , is always 30. We found the dynamics behaves in the same way with smaller and bigger graphs, so we fix this parameter in a way that it is big enough to present significant and realistic results but not too big so the simulations take too much time and resources.
- The initial number of edges is 10 or 30, to see the differences as we increase the number of initial edges.
- The probability of a player to be immunized at the beginning,  $p$ , is 0.25, 0.50 or 0.75. We choose a low, a medium and a high probability to see the differences among initial configurations with different quantity of immunized players.

Note that the number of initial configurations with the same values is very huge. Finally, we decided to present only one initial graph for each combination of values, because we found two different graphs with the same values behave the same way and provide very similar results.

As we have said, for each initial graph we arrive to swapstable equilibria with different  $C_E$  and  $C_I$ , which have the following values:

- We present results with edge costs 0.5, 2 and 10, to see the differences among low, medium and high  $C_E$ .
- For  $C_E = 0.5$  and  $C_E = 2$  we present results with immunization costs 0.5, 2, 20 and 50, to see the differences among low, medium, high and very high  $C_I$ . For  $C_E = 10$  we present results with immunization costs 0.05 and 0.5 because, as the edge cost is high, with low  $C_I$  the total expenditures are quite high and, therefore, with these values we see results regarding the connectivity of the graphs and the immunization of the nodes in equilibria.

To do each one of our simulations, we first generate the initial graph with the formerly described parameters. As we have indicated, there exist a huge number of graphs with the same values. We want to generate the graph in a way that all initial configurations are equally likely. With this purpose, we use the Erdős–Rényi model<sup>1</sup> to choose the initial connectivity. Also, we

---

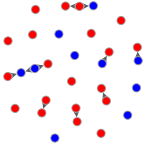
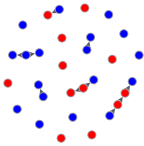
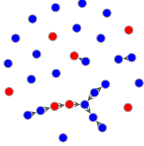
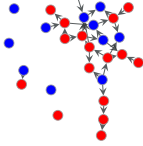
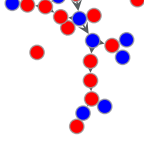
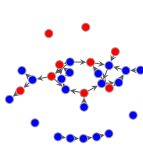
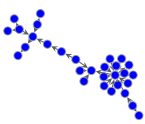
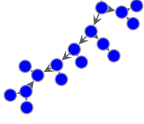
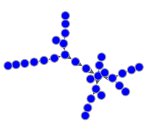
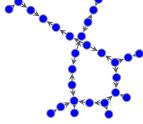
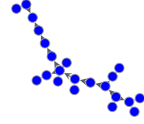
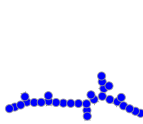
<sup>1</sup>In the Erdős–Rényi model, a graph is chosen uniformly at random from the collection of graphs which have  $n$  nodes and  $m$  edges.

choose whether to immunize each node with the indicated probability. Then, we start with the swapstable best response dynamics proceeding in the way we explained in Section 3.1.2.

To do the simulations, we have used the programming language C++, because it is a compiled language which compiler has many optimizations and, for what we want to do, it is faster than other languages. Besides, my programming skills in C++ are better than in other programming languages. To draw the graphs, we have used Python and its library *graph-tool*.

Before starting with the experiments, we want to reflect on the results we will find. We expect, on one hand, to corroborate the results Goyal et al. obtain in [9], that is, the relationship between  $C_E$ ,  $C_I$  and  $n$  determine whether the equilibrium is trivial (and, therefore the connectivity is low) or non-trivial (and, therefore, the connectivity is high). On the other hand, we expect to see certain differences among different initial configurations.

In the following table we present a collection of swapstable equilibria found via the explained simulations for  $n = 30$  and the six different initial configurations:<sup>2</sup>

|       |       | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$   | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-------|-------|---|---|---|---|---|---|
| $C_E$ | $C_I$ |  |  |  |  |  |  |
| 0.5   | 0.5   |  |  |  |  |  |  |

<sup>2</sup>Remember that blue and red represent immunized and vulnerable nodes, respectively, and that directed edges denote which player purchased the edge.

|     |     |  |  |  |  |  |  |
|-----|-----|--|--|--|--|--|--|
| 0.5 | 2   |  |  |  |  |  |  |
| 0.5 | 20  |  |  |  |  |  |  |
| 0.5 | 50  |  |  |  |  |  |  |
| 2   | 0.5 |  |  |  |  |  |  |
| 2   | 2   |  |  |  |  |  |  |



|    |      |  |  |  |  |  |  |
|----|------|--|--|--|--|--|--|
| 2  | 20   |  |  |  |  |  |  |
| 2  | 50   |  |  |  |  |  |  |
| 10 | 0.05 |  |  |  |  |  |  |
| 10 | 0.5  |  |  |  |  |  |  |

Table 4.1: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversary, for  $n = 30$ , with different initial configurations and different  $C_E$  and  $C_I$ .

In the first row,  $C_E = 0.5$  and  $C_I = 0.5$ , we see that whatever it is the initial configuration, the swapstable equilibrium consists in a connected graph, with no cycles, and all its nodes immunized, as we expected. All the nodes are immunized and the graph is connected because both the immunization cost and the edge cost are very low. There are no cycles because all the players will resist the attack and any additional edge would, therefore, be redundant. We see that in the table, it seems that the figure (\*) has a cycle. If we enlarge the figure, we see this cycle does not exist:

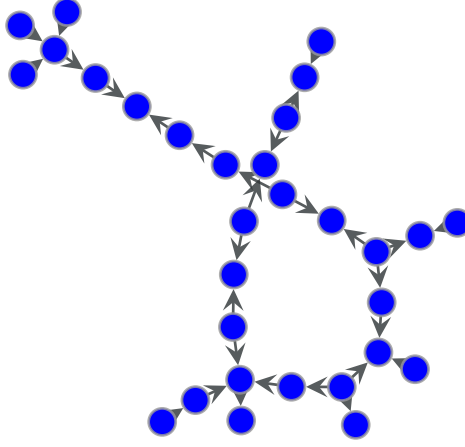


Figure 4.1: Swapstable equilibrium for  $n = 30$ , 30 initial edges,  $p = 0.25$ ,  $C_E = 0.5$  and  $C_I = 0.5$ .

In the second row,  $C_E = 0.5$  and  $C_I = 2$  we have that, for the first initial configurations, the swapstable equilibria are very similar, and for the last configuration (30 edges and a lot of immunized players) it is different. First, let us analyse the results for the first configurations. We take as example the first one, taking into account that the other ones are similar:

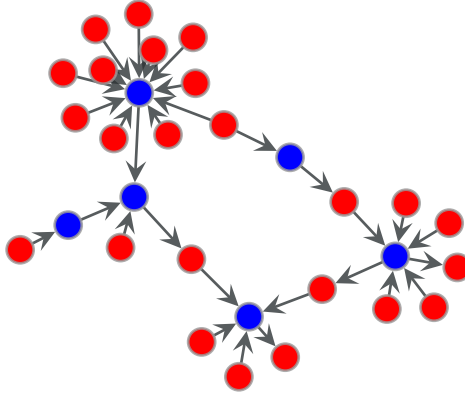


Figure 4.2: Swapstable equilibrium for  $n = 30$ , 10 initial edges,  $p = 0.25$ ,  $C_E = 0.5$  and  $C_I = 2$ .

In this swapstable equilibrium we see that this time we do have vulnerable nodes, because the immunization cost is higher. We have a cycle made up of both vulnerable and immunized nodes, with some of the latter having

some children. The cycle exists because the  $C_E$  is very low and the cycle contains targeted nodes, therefore it is beneficial to the nodes having two paths connecting them to the same immunized nodes, so the graph is still connected after the attack. The nodes outside the cycle are connected to immunized nodes because, obviously, a targeted node prefers to be connected to an immunized node than to a targeted one, otherwise the probability of dying increases.

It catches our attention that some immunized nodes have purchased edges to vulnerable nodes. We believe this happens because, as the  $C_E$  is very low, it is beneficial for the immunized nodes to form a cycle, as we have said before, so the graph is still connected after the attack. The immunized nodes are the ones to buy the edges probably because they are first in the round policy.

Now let us analyse the result for the last initial configuration:

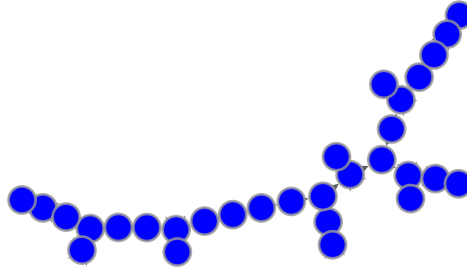


Figure 4.3: Swapstable equilibrium for  $n = 30$ , 30 initial edges,  $p = 0.75$ ,  $C_E = 0.5$  and  $C_I = 2$ .

This equilibrium is similar to the one with  $C_E = 0.5$  and  $C_I = 0.5$  and the same initial configuration. Therefore, we see that the number of immunized nodes in the initial graph has an impact in the final result. We believe that, as there are so few vulnerable nodes at the beginning, they have too many possibilities of dying, so they prefer to immunize, reaching at the end a state where all nodes are immunized and, as the  $C_E$  is still very low, similar to the ones of the previous row.

In the third row,  $C_E = 0.5$  and  $C_I = 20$  we have different results when the initial graph has 10 edges than when the graph has 30 edges. Let us first analyse the initial configuration with lower edge density. We take as an example the second one:

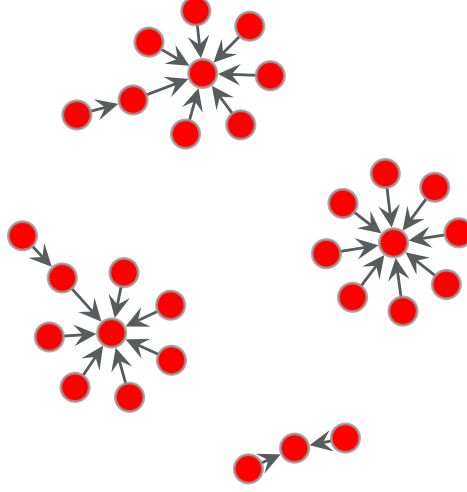


Figure 4.4: Swapstable equilibrium for  $n = 30$ , 10 initial edges,  $p = 0.50$ ,  $C_E = 0.5$  and  $C_I = 20$ .

In this case, the  $C_I$  is high, so it is no surprise there are no immunized nodes. We reach, therefore, a trivial equilibrium: we have a very fragmented graph, with several connected components because the more connected components, the more vulnerable regions, so the probability of dying for a vulnerable region is less. Nevertheless, the vulnerable regions are not singletons, because the  $C_E$  is very low and it is beneficial for the nodes to have a bigger connected component in case they do not die. We can also see that the sizes of almost all the connected components are the same. This happens because, in this way, there are more targeted regions, so the probability for each of them to die is less. Anyway, it catches our attention that, in this figure, there is a connected component of only three nodes, so why a node from another connected component does not swap and changes from her original connected component to this new one? We think this is because it is more beneficial for a node to be in a targeted region with a probability of  $\frac{1}{3}$  of dying, but with a connected component size post-attack of 9 (in the case of not dying) than in a non-targeted region with a connected component size of only 3. Let us now analyse the results for the higher initial edge density. In this case we have two different types of swapstable equilibria: when there are few immunized nodes at the beginning we have the same result as in the sparser initial graphs, and when there are more immunized nodes ( $p = 0.50$  or  $p = 0.75$ ) we have a swapstable equilibrium like this one:

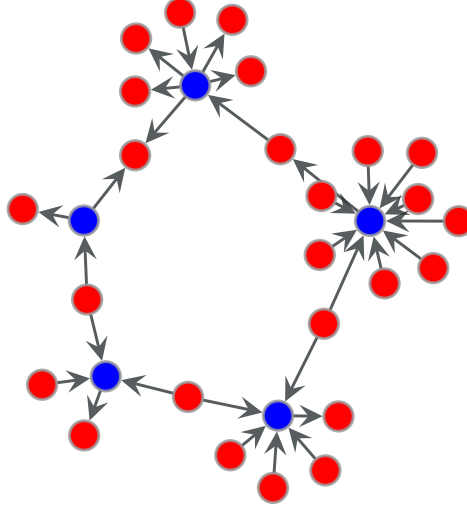


Figure 4.5: Swapstable equilibrium for  $n = 30$ , 30 initial edges,  $p = 0.50$ ,  $C_E = 0.5$  and  $C_I = 20$ .

We see this time there are immunized nodes and the graph is connected, with a cycle. We believe this is because, as at the beginning we have more edges and quite few vulnerable nodes, they do not have interest in being vulnerable (and therefore have a non-null probability of dying) and selling edges (and therefore have a smaller connected component post-attack). Again, we have seen the different initial edge density and number of immunized nodes make a difference in the final result.

In the fourth row,  $C_E = 0.5$  and  $C_I = 50$  we have similar swapstable equilibria no matter which is the initial configuration: several connected components of the same size, with only vulnerable nodes. In this case we see that even in the denser initial graph with a lot of immunized nodes, the immunization cost is so high that all players stop being immunized.

In the fifth row,  $C_E = 2$  and  $C_I = 0.5$  we see that all swapstable equilibria consist in connected graphs, with no cycles, and all their nodes immunized. It is exactly the same case than with  $C_E = 0.5$  and  $C_I = 0.5$ ; we see that even when the edge cost is higher, the players still want to remain connected to the rest:

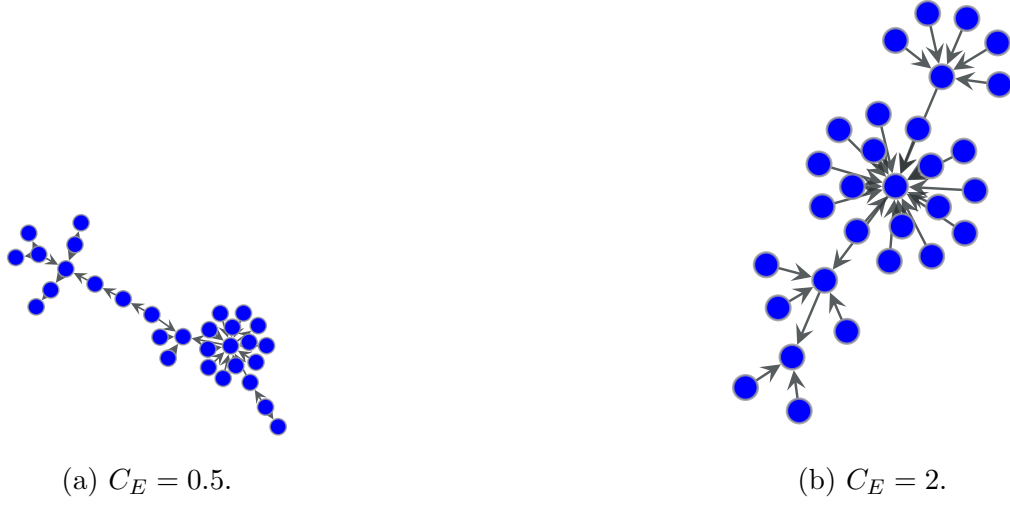


Figure 4.6: Swapstable equilibria for  $n = 30$ , 10 initial edges,  $p = 0.25$  and  $C_I = 0.5$ .

In the sixth row,  $C_E = 2$  and  $C_I = 2$ , we have a case similar to the second row,  $C_E = 0.5$  and  $C_I = 2$ . We see the swapstable equilibrium for the last initial configuration is very similar in both rows, for the same reasons. We will, then, analyse the differences of the other swapstable equilibria. We take the first one from both rows (sparser graph with few immunize nodes) as example:



Figure 4.7: Swapstable equilibria for  $n = 30$ , 10 initial edges,  $p = 0.25$  and  $C_I = 2$ .

We see that both graphs are very similar, with only one (but very important) difference: in this case,  $C_E = 2$ , the cycle disappears. This is not rare, as the edge cost is higher, so the swapstable equilibrium is a less dense graph (still connected, because the edge cost is not so high, but without cycles).

In the seventh row we have different equilibria. First, let us analyse the first and third columns, an initial graph with 10 edges, with  $p = 0.25$  and  $p = 0.75$ . We take the first equilibrium as example, and compare it to the equilibrium with  $C_E = 0.5$ :



Figure 4.8: Swapstable equilibria for  $n = 30$ , 10 initial edges,  $p = 0.25$  and  $C_I = 20$ .

We see that we have similar results, even if the edge cost has increased. This happens because each player buys, in the second case, at most one edge (in the first case, there are players who buy two edges, but, as we can see, we can arrive to the same configuration with each player buying at most one edge). Now let us see why, even with a higher  $C_E$ , no player wants to sell her edge:

- In the case of the connected component of size 3, if a player sold its edge, its connected component post-attack would decrease by 2 and its expenditures would also decrease by 2, so it has no incentive to sell her edge.
- In the other cases, the probability of a player  $i$  who has bought an edge of dying is  $\frac{1}{3}$ , so with probability  $\frac{2}{3}$  her connected component post-attack will be of size 9. Her expenditures are  $C_E = 2$ . Therefore her utility is  $u_i(\mathbf{s}) = \frac{2}{3} \cdot 9 - 2 = 4$ . If  $i$  sold its edge, her probability of dying would be 0, her connected component post-attack would be of size 1 and her expenditures would be 0. Therefore her utility would be  $u_i(\mathbf{s}) = 1$ . Thus  $i$  does not have an incentive to sell her edge.

We have seen that we have similar results because, although the edge cost is higher, it is not high enough to provide different results.

Now let us analyse the second column. We compare it to the equilibrium with  $C_E = 0.5$  and  $C_I = 20$  and to the one with  $C_E = 2$  and  $C_I = 2$ :

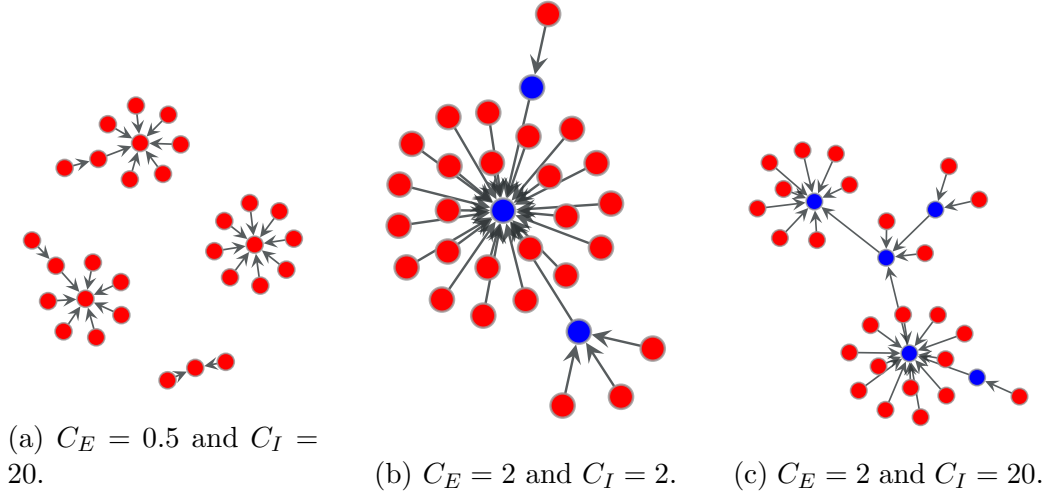


Figure 4.9: Swapstable equilibria for  $n = 30$ , 10 initial edges and  $p = 0.50$ .

We see the results with the same  $C_I$ , 20, are very different. In this case, (c), with  $C_E = 2$  and  $C_I = 20$ , we have a tree, where the immunized nodes have some vulnerable children (the leaves). We expected this equilibrium to be similar to the equilibrium (a), with  $C_E = 0.5$  and  $C_I = 20$ , but although the  $C_E$  is higher in this case we have a connected graph. We find it is similar to the equilibrium (b), with  $C_E = 2$  and  $C_I = 2$ , although the  $C_I$  is much higher (and we do not have a lot of immunized nodes at the beginning). We believe this happens because the targeted regions have size 1, so they all have the same probability of dying, which is low, and each node purchases at most one edge, so their expenditures are lower than the connectivity benefits. The immunization expenses are high, but the immunized nodes do not want to change their status because otherwise they would make their region a vulnerable region bigger than 1 and, therefore, the only targeted region. Now let us analyse the forth column. We compare it to the equilibrium with  $C_E = 0.5$ :





Figure 4.10: Swapstable equilibria for  $n = 30$ , 30 initial edges,  $p = 0.25$  and  $C_I = 20$ .

We see the results are very similar, with the only difference that in this case we have some singletons. This happens because the cost of buying an edge is 2 and the connected component post-attack of a singleton who bought an edge to another singleton would increase by only one, so she does not have an incentive to buy an edge. Note that no singleton would buy an edge to a node of a bigger connected component because otherwise this would be the only targeted region and, therefore, the player would die with probability 1. Now let us analyse the fifth column. We compare it to the equilibrium with  $C_E = 0.5$ :



Figure 4.11: Swapstable equilibria for  $n = 30$ , 30 initial edges,  $p = 0.50$  and  $C_I = 20$ .

We see that the equilibrium in this case is a hub-spoke with an immunized node at the center. It does not have the cycle the equilibrium (a) has, because the edge cost is higher and, therefore, the result is a less dense graph, without

cycles. Anyway, we still have a connected graph, unlike in the most cases with  $C_E = 2$  and  $C_I = 20$  we have already seen, because we start from a denser graph. We believe it is different to the previous case (Figure 4.10) because it starts with more immunized players, so it is more difficult to get all the nodes vulnerable.

Now let us analyse the last column. We compare it to the equilibrium with  $C_E = 0.5$ :

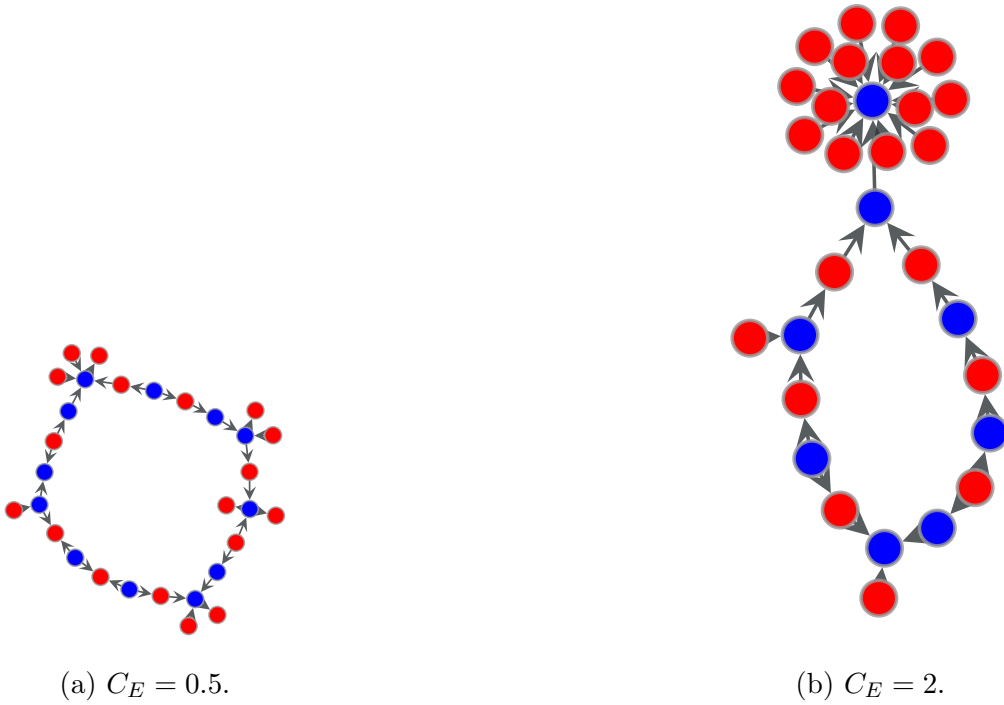


Figure 4.12: Swapstable equilibria for  $n = 30$ , 30 initial edges,  $p = 0.75$  and  $C_I = 20$ .

We see the results are very similar: a cycle with both vulnerable and immunized nodes, where some immunized nodes have some vulnerable children. We see that, in this case, even the edge cost has increased, unlike in the previous case, we still have a cycle. We believe this happens because we have more immunized nodes, so it is harder to change the strategy of most of them to become vulnerable.

In the eighth row,  $C_E = 2$  and  $C_I = 50$ , we have similar results for all initial configurations. We take as example the first one:

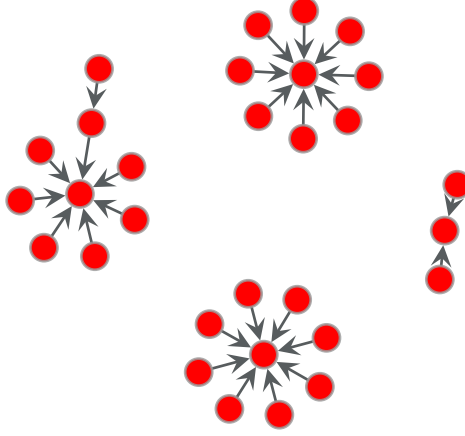


Figure 4.13: Swapstable equilibrium for  $n = 30$ , 10 initial edges,  $p = 0.25$ ,  $C_E = 2$  and  $C_I = 50$ .

In this case, the immunization cost is very high, so in all cases all the nodes are vulnerable. As we have seen in previous cases, as the edge cost is not very high, the result is a collection of connected components of the same size.

In the ninth row,  $C_E = 10$  and  $C_I = 0.05$ , we again find a difference between the different initial edge densities.

For sparser initial graphs, we have as swapstable equilibria empty graphs, because the edge cost is very high, so the players do not want to buy edges. When we start with few immunized nodes, the resulting graph has all its nodes vulnerable, while when we start with more immunized nodes, the resulting graph has all its nodes immunized, because the more immunized nodes we have at the beginning, the easier is to reach a state where all players are immunized. Note that, with an empty graph, all players must have the same immunization strategy: if all nodes are immunized, obviously no one will want to change her immunization status, because otherwise she would die with probability 1; if all nodes are vulnerable and one of them finds that she prefers to immunize (because the probability of dying, which is  $\frac{1}{n}$ , is higher than the  $C_I$ ), then the others will prefer to immunize too (because the probability of dying will be higher -because there will be less targeted nodes- and, therefore, will be higher than  $C_I$  too).

For denser initial graphs, we do not have empty graphs as a result, but connected graphs with all its nodes immunized. We believe this happens because the connected component post-attack for each player is  $n = 30$ , which is much

higher than the expenses of buying an edge ( $C_E = 10$ ) and the immunization ( $C_I = 0.05$ ). We can see that for a combination of  $C_E$  and  $C_I$  that add up more than  $n$ , this equilibrium would be impossible.

We note that, even though all these equilibria are swapstable equilibria with this values of  $C_E$  and  $C_I$ , when the graph is sparser it reaches a trivial equilibrium (the empty graph) and when it is denser it reaches a non-trivial equilibrium. It is obviously harder to reach an equilibrium where the graph is connected when we start from a configuration with less edges.

In the last row we find almost the same result than in the previous row, but when the graph is sparser, no matter how many immunized nodes we have at the beginning, we have an empty graph with all the nodes vulnerable. This happens because the immunization cost is higher, so immunized players prefer to change their immunization status. Anyway, it is not high enough so that, with some initial configurations (graphs with 30 initial edges), the swapstable equilibria are not connected graphs, as we have explained before.

As a final conclusion, we can say:

- Small values of  $C_I$  (0.5) combined with low and medium values of  $C_E$  (0.5 and 2) result in equilibria consisting in trees with all the nodes immunized.
- Small values of  $C_E$  (0.5) combined with medium values of  $C_I$  (2) usually lead to equilibria consisting in connected graphs with a cycle and both immunized and vulnerable nodes.
- Medium values of  $C_E$  (2) combined with medium values of  $C_I$  (2) usually result in trees with both immunized and vulnerable nodes.
- Large values of  $C_E$  compared to  $n$  ( $C_E = 10$ ) usually lead to trivial equilibria: empty graphs.
- Large values of  $C_I$  compared to  $n$  ( $C_I = 20$  and  $C_I = 50$ ) usually lead to another kind of trivial equilibria: high fragmented graphs with no immunized nodes.

We can also say that the initial configuration plays an important role in the swapstable equilibrium we arrive to. We have seen that different initial edge densities and different number of immunized nodes at the beginning make a difference when it comes to arrive to an equilibrium. As we increase the number of edges, we usually find denser equilibria. Also, for initial configurations with 30 edges, more immunized nodes at the beginning usually

mean more immunized nodes at equilibrium. Therefore, we conclude that the probability  $p$  of the nodes to be immunized at the beginning has an effect on the final configuration.

We also corroborate that, in all cases, the graphs are quite sparse, and all their vulnerable regions are trees, as the theory shows (see Section 3.2.1). Also, non-trivial equilibria with  $C_E > 1$  are connected graphs, and in all their connected components with at least one edge and at least one immunized node, the targeted regions are singletons, as has been theoretically proven (see Section 3.2.1).

# Chapter 5

## Extension of the model: Two attacks and immunization

In this chapter we extend the model defined in Section 3.1 by giving more power to the adversary: two different players can be attacked.

We want to study a more realistic model, in which more than one player can be attacked at a time. We find this interesting because it gets closer to real world. We want to study the topology of the swapstable equilibria, to see if the number of immunization grows, if the graphs in equilibria are denser..., so we can compare our results with the ones we obtain with the original model (see Chapter 4).

### 5.1 Definition

Let us first define the model. We introduce a new adversary, which acts in a similar way to the maximum carnage adversary but, instead of attacking one player, she attacks two players.

This adversary attacks two vulnerable regions such that the sum of their sizes is maximum, i.e.

$$\max_{\mathcal{T}'_1, \mathcal{T}'_2 \in \mathcal{T}} \{|\mathcal{T}'_1| + |\mathcal{T}'_2|\}$$

Remember  $\mathcal{T}$  is the set of vulnerable regions with non-zero probability of being attacked.

If there is only one targeted region, i.e.,  $|\mathcal{T}| = 1$  (which happens when there is only one vulnerable region), the adversary only attacks this region,

let it be  $\mathcal{T}'$ , and the utility of player  $i$  is the size of her connected component after the attack to  $\mathcal{T}'$ ,  $CC_i(\mathcal{T}')$ , minus her expenses.

When  $|\mathcal{T}| > 1$ , the expected utility of player  $i$  is the expected size of her connected component after the attack to the two targeted regions minus her expenses. Formally, let  $Pr[\{\mathcal{T}'_1, \mathcal{T}'_2\}]$  be the probability to attack to the targeted regions  $\mathcal{T}'_1$  and  $\mathcal{T}'_2$ , and  $CC_i\{\mathcal{T}'_1, \mathcal{T}'_2\}$  the size of player  $i$ 's connected component after the attack to both  $\mathcal{T}'_1$  and  $\mathcal{T}'_2$ . Then the expected utility of  $i$  in the strategy profile  $\mathbf{s}$ ,  $u_i(\mathbf{s})$ , is

$$u_i(\mathbf{s}) = \sum_{\mathcal{T}'_1, \mathcal{T}'_2 \in \mathcal{T}} \left( Pr[\{\mathcal{T}'_1, \mathcal{T}'_2\}] CC_i\{\mathcal{T}'_1, \mathcal{T}'_2\} \right) - |x_i|C_E - y_i C_I$$

When  $|\mathcal{T}| > 1$ , we have two cases, depending on the number of vulnerable regions of maximum size:

- There is only one vulnerable region of maximum size. Then, the targeted regions are this vulnerable region of maximum size, which we name  $\mathcal{T}'_1$ , and the set of all the vulnerable regions of the next maximum size, which we name  $\mathcal{T}'_2$ . Then,  $Pr[\{\mathcal{T}'_1, \mathcal{T}'_2\}] = Pr[\mathcal{T}'_2] = \frac{1}{|\mathcal{T}'_2|}$ . In this case, the expected utility of  $i$  in the strategy profile  $\mathbf{s}$  is:

$$u_i(\mathbf{s}) = \frac{1}{|\mathcal{T}'_2|} \left( \sum_{\mathcal{T}'_2 \in \mathcal{T}'_2} CC_i\{\mathcal{T}'_1, \mathcal{T}'_2\} \right) - |x_i|C_E - y_i C_I$$

- There are more than one vulnerable regions of maximum size. Then, the targeted regions are all the vulnerable regions of maximum size, and  $Pr[\{\mathcal{T}'_1, \mathcal{T}'_2\}] = \frac{2}{|\mathcal{T}|(|\mathcal{T}|-1)}$ . In this case,  $i$ 's expected utility in the strategy profile  $\mathbf{s}$  is:

$$u_i(\mathbf{s}) = \frac{2}{|\mathcal{T}|(|\mathcal{T}|-1)} \left( \sum_{\mathcal{T}'_1, \mathcal{T}'_2 \in \mathcal{T}} CC_i\{\mathcal{T}'_1, \mathcal{T}'_2\} \right) - |x_i|C_E - y_i C_I$$

## 5.2 Simulations: dynamics

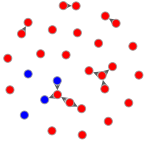
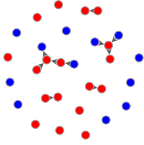
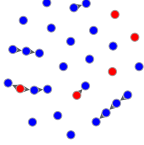
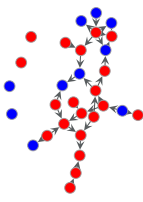
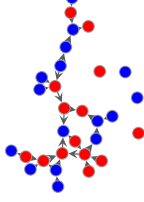
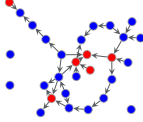
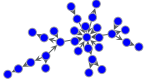
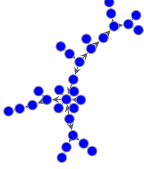
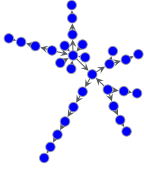
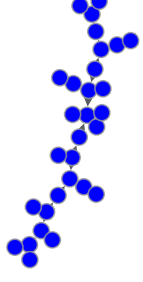
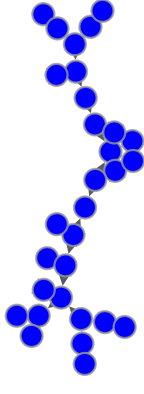
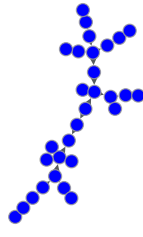
In this section we present the results of the simulations we have implemented to investigate the properties of swapstable best response dynamics with respect to the new adversary, the maximum carnage adversary which attacks two players. We again focus on the topology of the swapstable equilibria

found. We want to study the number of immunized players, the density of the resulting graphs, the connectivity... so, in the next section, we can compare the results we find with the results found in Chapter 4.

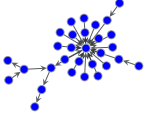
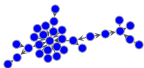
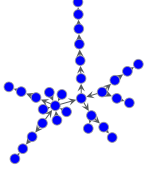
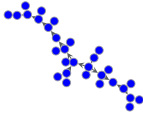
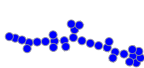
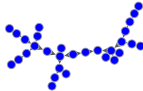
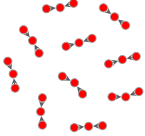
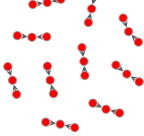
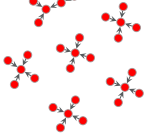
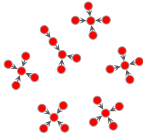
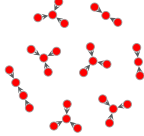
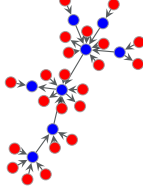
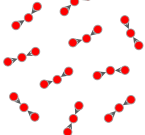
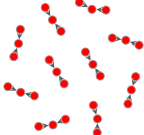
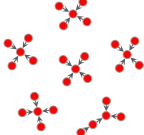
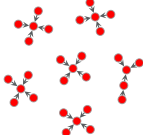
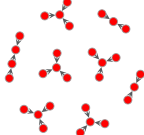
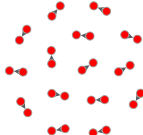
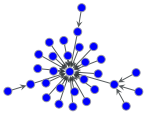
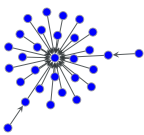
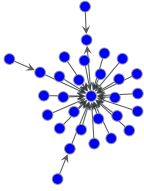
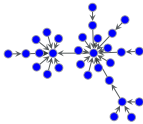
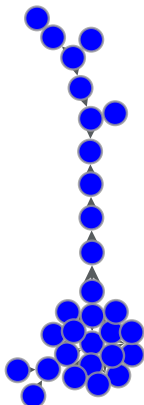
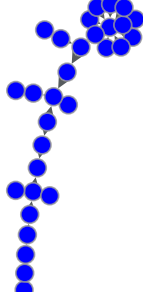
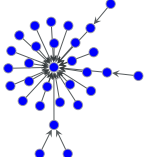
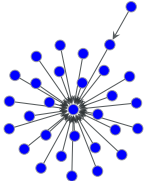
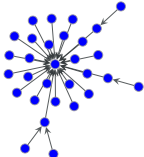
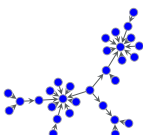
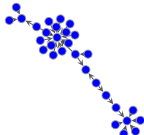
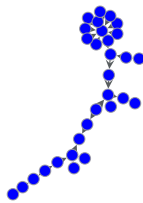
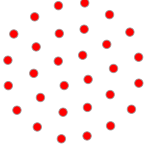
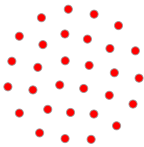
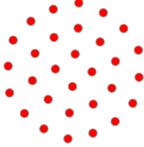
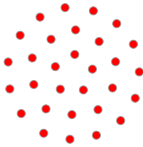
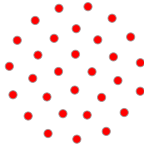
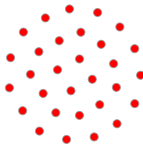
Again, we experiment with different initial configurations and different parameters. We choose the same values for the parameters as in Chapter 4 and we generate the initial graph in the same way.

We want to think, before starting with the experiments, about the results we will find. We expect to obtain equilibria with more immunized nodes than in Chapter 4, because the probability of dying for targeted nodes is higher. We also expect to find denser graphs, with maybe more than one cycle, so the graphs remain connected after the attack.

In the following table we present some equilibria we have obtained with the swapstable best response dynamics for  $n = 30$  and the six different initial configurations:

|       |       | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$   | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-------|-------|---|---|---|---|---|---|
| $C_E$ | $C_I$ |  |  |  |  |  |  |
| 0.5   | 0.5   |  |  |  |  |  |  |



|     |     |   |   |   |   |   |   |
|-----|-----|---|---|---|---|---|---|
| 0.5 | 2   |    |    |    |    |    |    |
| 0.5 | 20  |    |    |    |    |    |    |
| 0.5 | 50  |    |    |    |    |    |    |
| 2   | 0.5 |  |  |  |  |   |  |
| 2   | 2   |  |  |  |  |  |  |
| 2   | 20  |  |  |  |  |  |  |

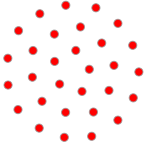
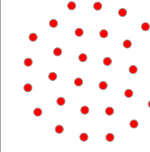
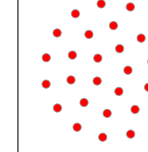
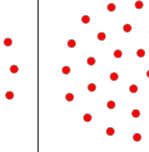
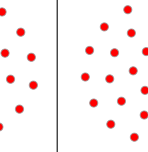
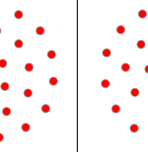
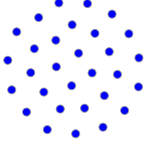
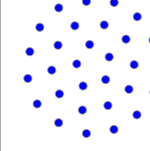
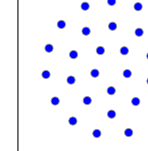
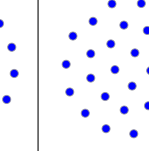
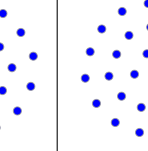
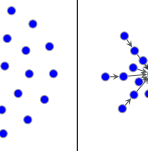
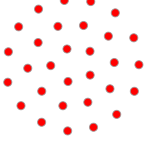
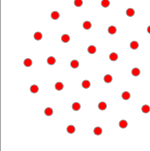
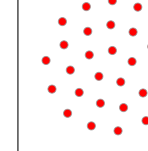
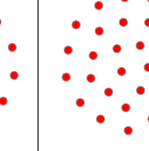
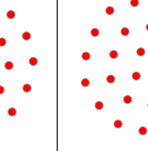
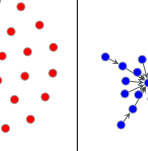
|    |      |   |   |   |  |   |   |
|----|------|---|---|---|--|---|---|
| 2  | 50   |  |  |  |  |  |  |
| 10 | 0.05 |  |  |  |  |  |  |
| 10 | 0.5  |  |  |  |  |  |  |

Table 5.1: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversary which attacks two players, for  $n = 30$ , with different initial configurations and different  $C_E$  and  $C_I$ .

We see that, for the combination of low and medium  $C_E$  and  $C_I$ , i.e.,  $C_E = 0.5$  or  $C_E = 2$  and  $C_I = 0.5$  or  $C_I = 2$ , all the swapstable equilibria are connected graphs with all their nodes immunized (and, obviously, no cycles), no matter the initial graph. For example, for  $C_E = 0.5$ ,  $C_I = 0.5$  and the first initial graph (10 initial edges and few immunized nodes at the beginning, i.e.,  $p = 0.25$ ) we arrive to the following swapstable equilibrium:

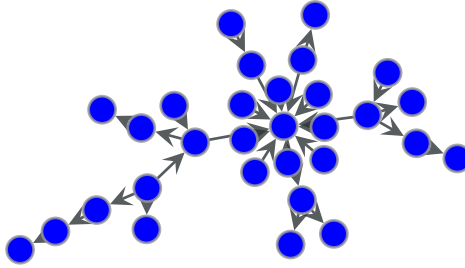


Figure 5.1: Swapstable equilibrium for  $n = 30$ , 10 initial edges,  $p = 0.25$ ,  $C_E = 0.5$  and  $C_I = 0.5$ , with respect to the maximum carnage adversary which attacks two nodes.

All the other equilibria in the corresponding rows are similar to the previous one. We see that, even when  $C_I = 2$ , all nodes decide to immunize,

because this time vulnerable nodes have more probability of dying.

In the third row,  $C_E = 0.5$  and  $C_I = 20$ , we have two different types of swapstable equilibria. Let us first analyse the equilibria we reach with the first five initial configurations. We take the first one as an example:

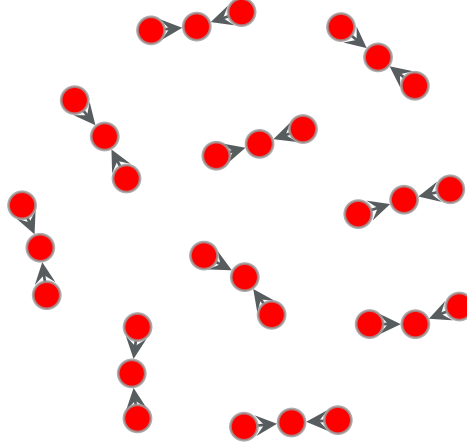


Figure 5.2: Swapstable equilibrium for  $n = 30$ , 10 initial edges,  $p = 0.25$ ,  $C_E = 0.5$  and  $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes.

In this case, we have a trivial equilibrium: a very fragmented graph, with no immunized nodes, because the immunization cost is high. We have several connected components and not the empty graph because the edge cost is very low and, therefore, players still have an incentive to purchase an edge. With the last configuration, the graph with 30 edges and a lot of immunized nodes, we get the following swapstable equilibrium:

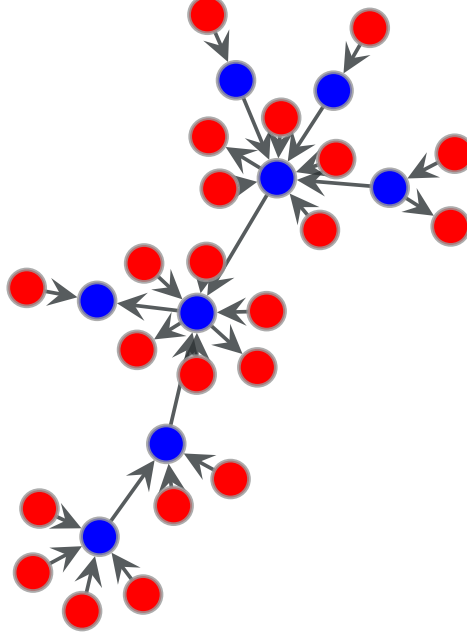
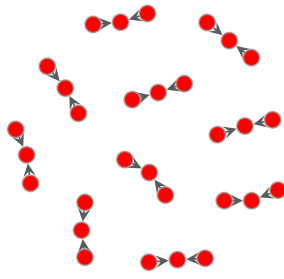


Figure 5.3: Swapstable equilibrium for  $n = 30$ , 30 initial edges,  $p = 0.75$ ,  $C_E = 0.5$  and  $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes.

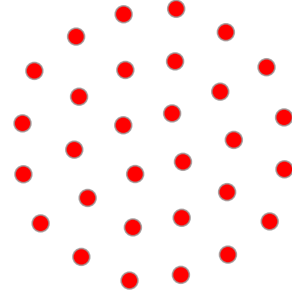
We have a tree, with some connected immunized nodes with some vulnerable children, which are the leaves. In this case we see that, as at the beginning we have more edges and immunized nodes, at equilibrium we still have a connected graph and, although the immunization cost is high, some nodes prefer to remain immunized, because the security of having a connected component post-attack of 28 (because two vulnerable nodes die), even if  $C_I = 20$ , is preferable to having a small connected component and having the possibility of dying.

In the fourth row, we have that the immunization cost is so high that all equilibria are trivial: several connected components of small size with no immunized nodes. In this case, with  $C_I = 50$ , players do not prefer having a connected component post attack equal to 28, because the  $C_I$  is even higher.

In the seventh and eighth row, with  $C_E = 2$  and  $C_I = 20$  or  $C_I = 50$ , the results are the same. We compare the equilibrium with  $C_E = 0.5$  and  $C_I = 20$  and the equilibrium with  $C_E = 2$  and the same  $C_I$ . We take as example the first initial configuration, i.e., the sparser graph with few immunized nodes:



(a)  $C_E = 0.5$ .

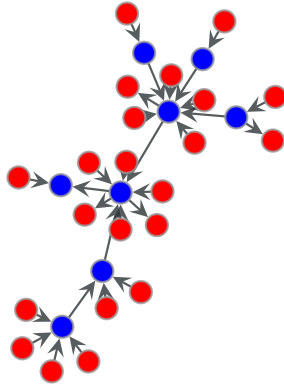


(b)  $C_E = 2$ .

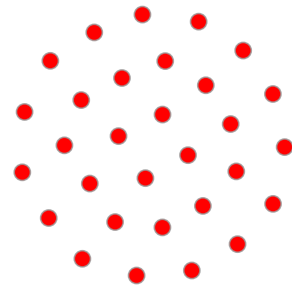
Figure 5.4: Swapstable equilibria for  $n = 30$ , 10 initial edges,  $p = 0.25$  and  $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes.

We see that, in this case, (b), as the edge cost is higher, players have no incentive to purchase edges, because their expected connected component post-attack would not increase as much as their expenditures. Therefore, the equilibrium is the empty graph, with all its nodes vulnerable (because the  $C_I$  is high).

We now compare the equilibria with the same  $C_E$  and  $C_I$  but starting from the last initial configuration, i.e., the graph with 30 edges and a lot of immunized nodes:



(a)  $C_E = 0.5$ .



(b)  $C_E = 2$ .

Figure 5.5: Swapstable equilibria for  $n = 30$ , 30 initial edges,  $p = 0.75$  and  $C_I = 20$ , with respect to the maximum carnage adversary which attacks two nodes.

In this case, (b), players do not choose to remain immunized and con-

nected, because the sum of  $C_E$  and  $C_I$  is too high. Therefore, they prefer to drop edges and become vulnerable until we get the empty graph with no immunized nodes.

In the last two rows,  $C_E = 10$ , we have different types of equilibria starting for the last initial configuration (graph with 30 edges and a lot of immunized nodes) to those we have starting from the others. Let us first analyse the the results we get with in the last column. We have the same equilibrium with  $C_I = 0.05$  than with  $C_I = 0.5$ . We take the first one as an example:

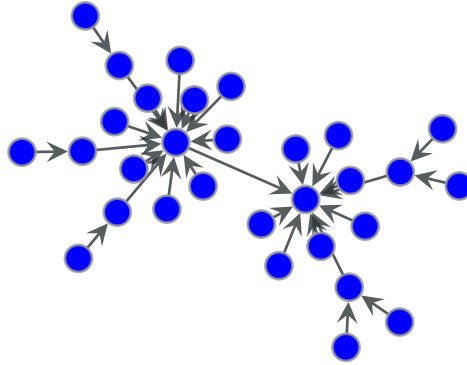


Figure 5.6: Swapstable equilibrium for  $n = 30$ , 30 initial edges,  $p = 0.75$ ,  $C_E = 10$  and  $C_I = 0.05$ , with respect to the maximum carnage adversary which attacks two nodes.

The equilibrium is again a tree with all its nodes immunized, because the  $C_I$  is low, so players decide to remain immunized and even the few vulnerable players choose to immunize, and although the  $C_E$  is high, players still have an incentive to keep an edge (or even purchase it, the few nodes who have not edges at the beginning) because they prefer to have a connected component post-attack of 30, even if they have to pay 10 for the edge.

Now let us analyse the equilibria we get starting for the other initial configurations. We always have empty graphs, when  $C_I = 0.05$  with all the nodes immunized, and when  $C_I = 0.5$  with all the nodes vulnerable. In the first case, all the players are immunized because the probability of dying is high compared to the immunization cost. In the second case, the  $C_I$  is higher, and players do not have an incentive to immunize. This equilibria are empty graphs, unlike in Figure 5.6, because now we do not have the combination of many edges and a lot of immunized nodes at the beginning, so players are not predisposed to stay connected and immunized.

As a final conclusion, we can say that:

- Small and medium values of  $C_E$  (0.5 and 2) combined with small and medium values of  $C_I$  (0.5 and 2) lead to trees with all the nodes immunized as equilibria.
- Large values of  $C_I$  (20 and 50) combined with low values of  $C_E$  (0.5) usually result in trivial equilibria: very fragmented graphs with no immunized nodes.
- Large values of  $C_I$  (20 and 50) combined with medium values of  $C_E$  (2) lead to another kind of trivial equilibria: empty graphs.
- Large values of  $C_E$  (10) usually result in empty graphs too.

We can also say that we get more non-trivial equilibria and, therefore, higher connectivity, starting for the last initial configuration, that is, a graph with 30 edges and many immunized nodes ( $p = 0.75$ ). Therefore, with two attacks the number of initial edges and the probability  $p$  of the nodes to be immunized at the beginning also have an impact on the result.

Finally, we want to note that in all cases, the graphs are quite sparse, and all their vulnerable regions are trees, because the adversary is well-behaved and, therefore, graphs must meet the properties stated in Section 3.2.1. In fact, the graphs are even sparser than in the case in which the adversary attacks a single player, because we have not found cycles.

We also see that all non-trivial equilibria are connected, and in their connected components with at least one edge and one immunized node, all the targeted regions are singletons.

### 5.3 Comparison with a single attack

In this section we compare the results we obtained with the adversary which makes a single attack to the results we obtained with the adversary which attacks two players.

We present several tables comparing the swapstable equilibria we found with the adversary which attacks a single player to the ones we found with the adversary which attacks two players, with the same values for the parameters of configuration of the initial graph and several choices of  $C_E$  and  $C_I$ .

Let us first compare the equilibria obtained with  $C_E = 0.5$  and  $C_I = 0.5$ :

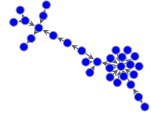
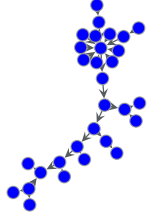
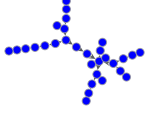
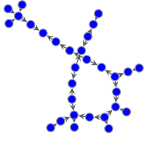
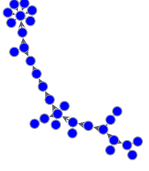
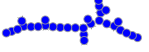
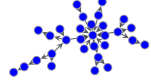
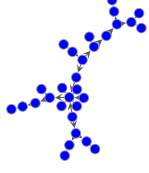
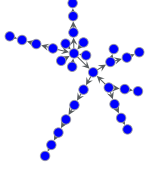
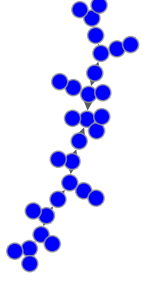
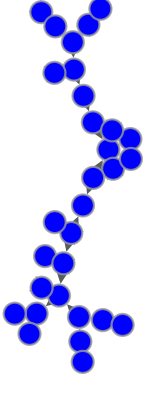
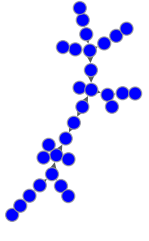
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$  | $ E  = 10,$<br>$p = 0.75$  | $ E  = 30,$<br>$p = 0.25$  | $ E  = 30,$<br>$p = 0.50$  | $ E  = 30,$<br>$p = 0.75$  |
|-----------|---|--|--|--|--|--|
| 1 attack  |    |   |   |   |   |   |
| 2 attacks |  |  |  |  |  |  |

Table 5.2: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 0.5$  and  $C_I = 0.5$ .

We see that, in this case, we obtain similar results, because in both cases the  $C_I$  is so low that all the players choose to immunize.

Let us now compare the equilibria found with  $C_E = 0.5$  and  $C_I = 2$ :



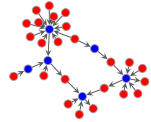
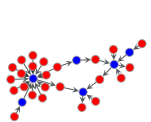
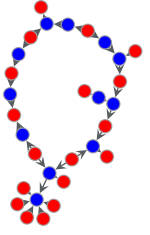
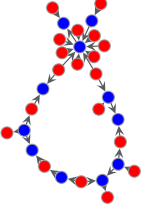
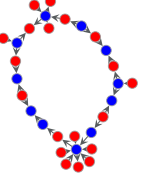
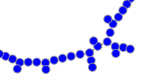
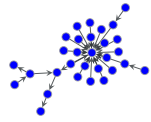
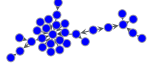
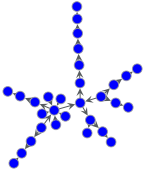
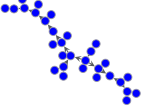

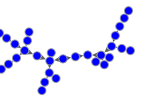
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$   | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|---|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.3: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 0.5$  and  $C_I = 2$ .

In this case, we obtain very different results. When the adversary attacks a single player, the resulting equilibria usually has a cycle and both immunized and vulnerable nodes, while when the adversary attacks two players, we have trees with only immunized nodes as equilibria. This happens because, with two attacks, the probability for vulnerable nodes of dying is higher and, as the cost of immunization is not high, they prefer to immunize.

Now let us compare the results obtained with  $C_E = 0.5$  and  $C_I = 20$ :

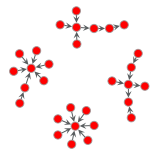
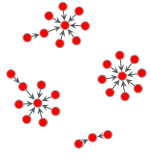
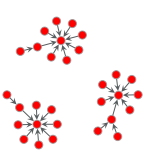
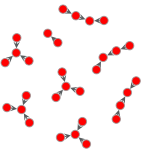
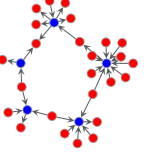
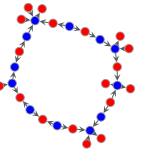
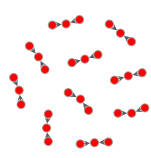
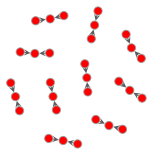
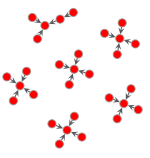
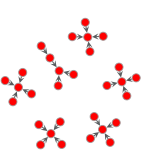
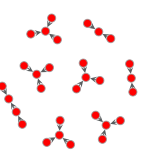
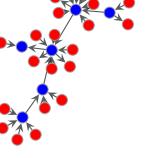
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$   | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|---|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.4: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 0.5$  and  $C_I = 20$ .

We find similar results, but with the adversary which makes two attacks we find more trivial equilibria. It is understandable that in this case the connectivity is lower, because the damage the adversary makes is higher.

Now, let us compare the equilibria found with  $C_E = 0.5$  and  $C_I = 50$ :

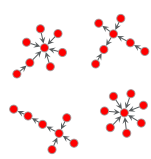
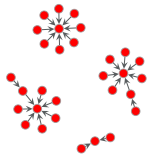
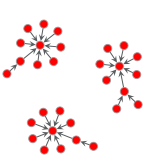
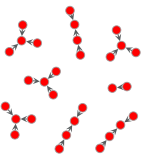
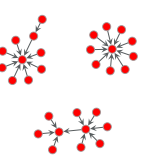
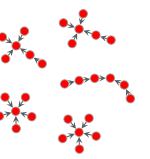
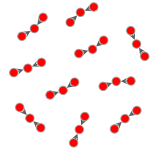

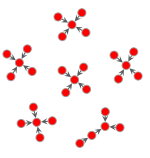
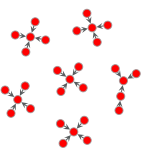
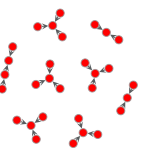
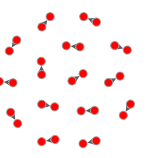
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$   | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|---|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.5: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 0.5$  and  $C_I = 50$ .

We find the same results: trivial equilibria with several connected compo-

nents and no immunized nodes, because the cost of immunization is very high.

Let us now compare the results found for  $C_E = 2$  and  $C_I = 0.5$ :

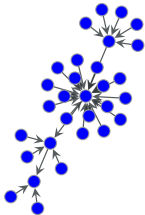
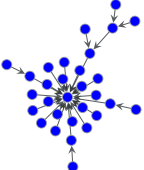
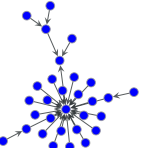
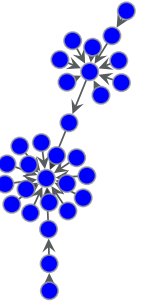
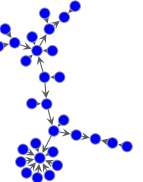
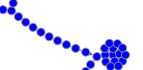
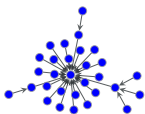
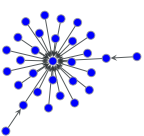
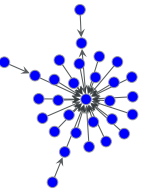
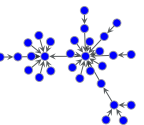
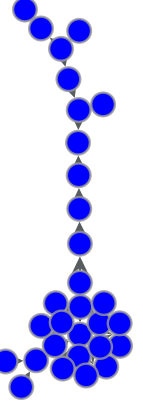
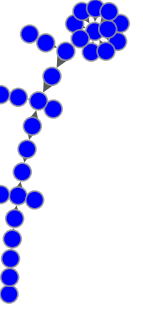
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$   | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$  | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|---|---|--|---|
| 1 attack  |    |    |    |    |   |    |
| 2 attacks |  |  |  |  |  |  |

Table 5.6: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 2$  and  $C_I = 0.5$ .

We find again the same results. In both cases all the players immunize because the immunization cost is very low.

Let us now compare the equilibria found with  $C_E = 2$  and  $C_I = 2$ :

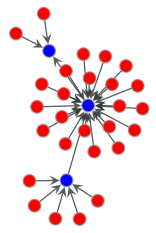
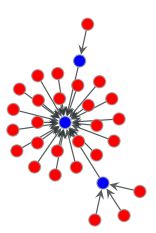
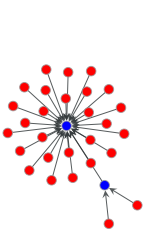
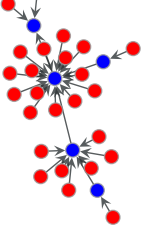
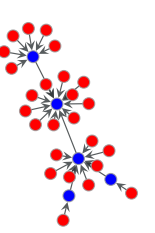
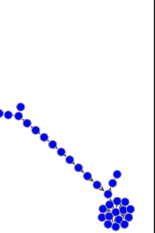
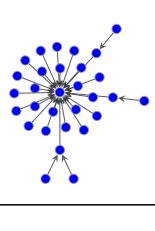
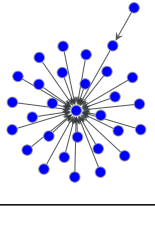
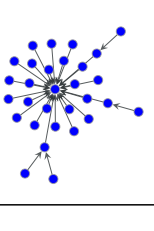
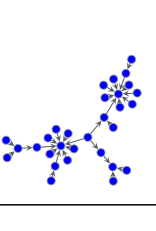
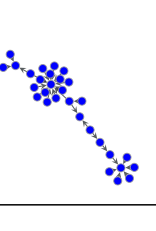
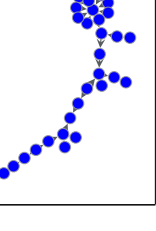
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$  | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|--|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.7: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 2$  and  $C_I = 2$ .

In this case, we obtain different results. In both cases we obtain trees as swapstable equilibria, but when the adversary attacks a single player we usually have vulnerable nodes as the leaves of the trees, and when the adversary attacks two players we only have immunized nodes. This happens, again, because the  $C_I$  is not high and in the case in which the adversary attacks two nodes the probability of dying is higher; therefore, in that case, players prefer to immunize.

Let us now compare the results obtained for  $C_E = 2$  and  $C_I = 20$ :

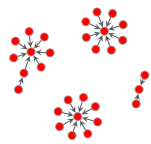
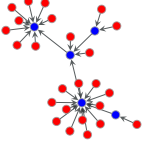
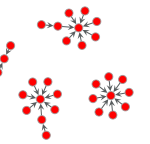
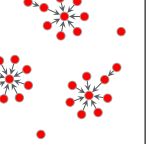
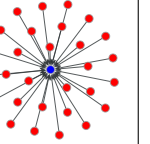
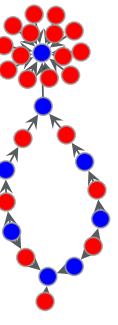
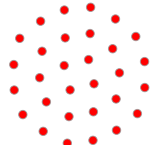
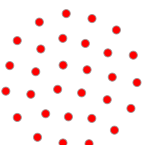
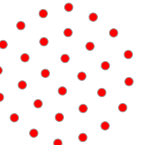
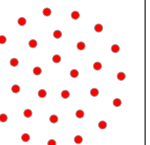
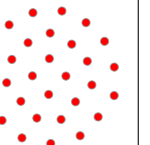

|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$  | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|--|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.8: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 2$  and  $C_I = 20$ .

In the case of a single attack, we find both fragmented graphs and connected graphs as swapstable equilibria. In the case of two attacks, we only find empty graphs. This happens because, in the second case, the probability of dying for the targeted regions is higher and, therefore, the size of the expected connected component post-attack of each player, with the same configuration, would be lower. Therefore, it does not compensate to the players to pay the edge cost.

Let us now compare the results obtained with  $C_E = 2$  and  $C_I = 50$ :

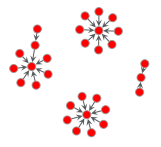
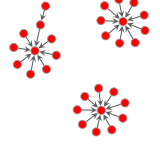
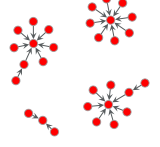
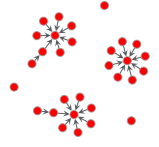
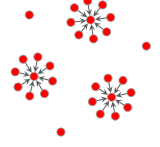
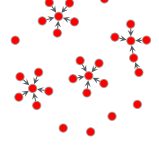
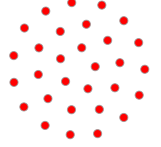
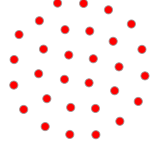
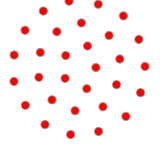
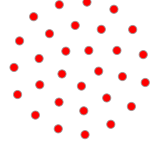
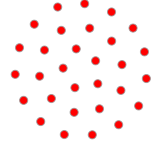
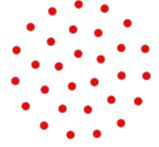
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$  | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|--|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.9: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 2$  and  $C_I = 50$ .

In both cases we obtain trivial equilibria, but when the adversary attacks a single player the resulting equilibria are several connected components and, when the adversary attacks two players, empty graphs. This happens for the exactly same reason than in the former case, with  $C_E = 2$  and  $C_I = 20$ .

Let us now compare the equilibria obtained with  $C_E = 10$  and  $C_I = 0.05$ :

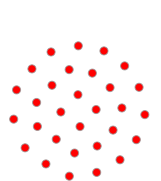
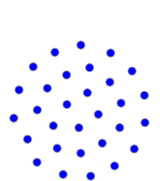
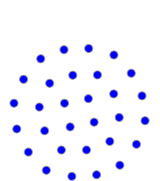
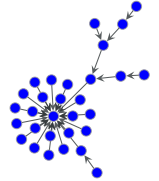
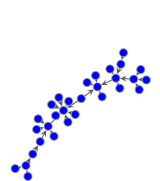
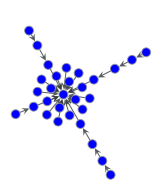
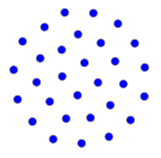
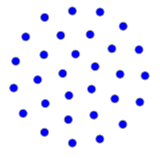
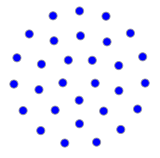
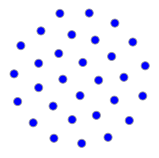
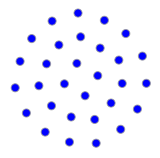
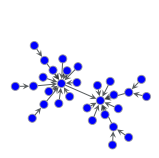
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$  | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|--|---|---|---|
| 1 attack  |  |  |  |  |  |  |
| 2 attacks |  |  |  |  |  |  |

Table 5.10: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 10$  and  $C_I = 0.05$ .

We see we obtain more trivial equilibria with the adversary which attacks

two players, because in this case players have less incentive to purchase edges, as in Table 5.8. We also see that for the initial configuration with 10 initial edges and few immunized players, when the adversary attacks a single player all the nodes are vulnerable at equilibrium and when the adversary attacks two players, they are immunized. This happens because in the second case, as the probability of dying for the targeted nodes is higher, they prefer to immunize.

Finally, let us compare the equilibria found with  $C_E = 10$  and  $C_I = 0.5$ :

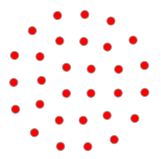
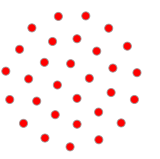
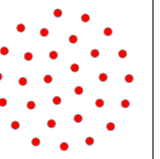
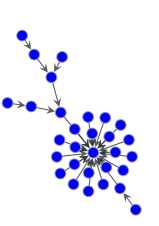
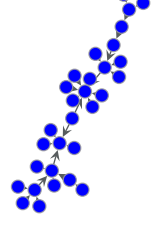
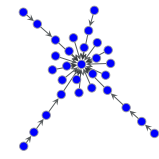
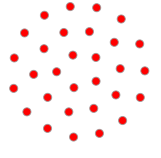
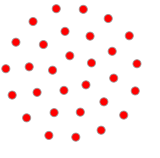
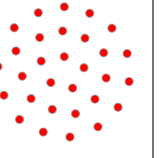
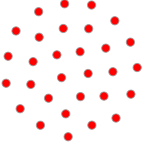
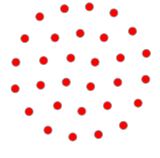
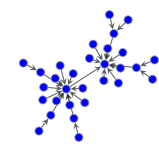
|           | $ E  = 10,$<br>$p = 0.25$   | $ E  = 10,$<br>$p = 0.50$   | $ E  = 10,$<br>$p = 0.75$  | $ E  = 30,$<br>$p = 0.25$   | $ E  = 30,$<br>$p = 0.50$   | $ E  = 30,$<br>$p = 0.75$   |
|-----------|---|---|--|---|---|---|
| 1 attack  |   |   |   |   |   |   |
| 2 attacks |  |  |  |  |  |  |

Table 5.11: Equilibria reached by swapstable best response dynamics with respect to the maximum carnage adversaries which attack one and two players, for  $n = 30$ , with different initial configurations and  $C_E = 10$  and  $C_I = 0.5$ .

We obtain similar results, but we again find more empty graphs at equilibria when the adversary attacks two players.

As a conclusion, we want to note that we have not found denser graphs with the adversary which attacks two players than with the adversary which attacks a single player. In fact, with two attacks we have obtained sparser graphs (at most, trees), without cycles.

Besides, with two attacks we have obtained more graphs with all the nodes immunized because, as the probability of dying for targeted players is higher, they have more incentive to immunize.

Also, we have obtained more trivial equilibria with two attacks than with a single attack. Therefore, we can conclude that with the new model we usually get lower connectivity than with only one attack.

Finally, we want to mention that we have checked that all the equilibria found with the adversary which attacks two players are equilibria with respect to the adversary which attacks a single player. We conjecture that *all* the swapstable equilibria with respect to the adversary which attacks two players are also equilibria with respect to the adversary which makes a single attack. We leave the demonstration of this conjecture for future work. Nevertheless, there are equilibria with respect to the adversary which attacks a single player that are not equilibria with respect to the adversary which attacks two players. For example, in Figure 5.7a we see that, when the adversary makes a single attack and  $C_E = 1.5$  and  $C_I = 3$  the cycle is a swapstable equilibrium. However, when the adversary makes two attacks, (Figure 5.7b) the cycle is not a swapstable equilibrium.



Figure 5.7: (a) The cycle is a swapstable equilibrium when  $C_E = 1.5$  and  $C_I = 3$  and the adversary makes a single attack. (b) Swapstable equilibrium when  $C_E = 1.5$  and  $C_I = 3$  and the adversary makes two attacks, starting from the cycle.



# Chapter 6

## Conclusions and Future Work

The main goal of this project was to study the topologies of the equilibria resulting of the swapstable best response dynamics in the Network Formation Game with attacks and immunization.

First, we have studied the model in which the adversary attacks a single player. We have found that large values of  $C_E$  lead to empty graphs. Small values of  $C_I$  result in trees with all the nodes immunized. Large values of  $C_I$  result in high fragmented graphs with no immunized nodes. Finally, medium values of  $C_I$  lead to connected graphs with both immunized and vulnerable nodes and, in this case, when  $C_E$  is small graphs usually have a cycle. We have also found that the initial configuration of the graphs (the initial number of edges and the probability  $p$  of the nodes to be immunized) makes a difference in the resulting equilibria: as we increase the number of edges in the initial graph, we usually find denser equilibria, and more immunized nodes in the initial graph usually mean more immunized nodes at equilibrium.

Then, we have experimented with the model in which the adversary attacks two players. We have found that large values of  $C_E$  have as a result empty graphs. Large values of  $C_I$  lead to trivial equilibria: empty graphs or very fragmented graphs with no immunized nodes. Finally, small and medium values of  $C_E$  and  $C_I$  lead to trees with all the nodes immunized. We have also found that, again, the initial configuration pays an important role in the equilibria found.

Finally, we have compared the results obtained with the two models. We have found that with two attacks we have obtained sparser graphs (without cycles) than with a single attack. Additionally, with two attacks we have obtained more equilibria with all the nodes immunized. Besides, with two at-

tacks we usually get lower connectivity than with one attack (we have found more trivial equilibria). We have also checked that all the equilibria found with two attacks are also equilibria with one attack, and we have shown an example in which a equilibrium found with a single attack is not an equilibrium in the model with two attacks.

We want to mention that this model, extended to any number of attacks, can be applied to the current situation with COVID-19: when everybody is vaccinated, everybody can be connected; when the  $C_I$  is very high (for example, we still do not have the vaccine) people are not immunized and are only connected to their coexistence bubble group. When we start having vaccines, some people are immunized and may be connected to other people.

For future research, it would be interesting to extend the model to any number of attacks. We could also study the way the random adversary and the maximum disruption adversary behave both when they attack a single player and when they attack more players. Finally, it would also be interesting to prove or disprove the conjecture that, with respect to the maximum carnage adversary, all swapstable equilibria with two attacks are also swapstable equilibria with a single attack.

As a personal experience, this project has allowed me to learn the basic concepts of Game Theory and, particularly, Network Creation Games. I have also learned to read scientific papers and summarize their main ideas. Besides, I have trained skills I acquired during the degree. Overall, the project has introduced me into the world of research, thing for which I am very grateful.

# Bibliography

- [1] Amazon. <https://www.amazon.es/>.
- [2] Payscale. <https://www.payscale.com/>.
- [3] United states environmental protection agency. <https://www.epa.gov/>.
- [4] Venkatesh Bala and Sanjeev Goyal. A noncooperative model of network formation. In *Econometrica*, volume 68, 5, pages 1181–1229. The Econometric Society, 2000.
- [5] Francis Bloch and Matthew Jackson. Definitions of equilibrium in network formation games. In *International Journal of Game Theory*, volume 34, 3, pages 305–318, 2006.
- [6] Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos Papadimitriou, and Scott Shenker. On a network creation game. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing*, pages 347–351, 2003.
- [7] Tobias Friedrich, Sven Ihde, Christoph Keßler, Pascal Lenzner, Stefan Neubert, and David Schumann. Efficient best-response computation for strategic network formation under attack. In *10th International Symposium on Algorithmic Game Theory*, pages 199–211, 2017.
- [8] Sanjeev Goyal, Shahin Jabbari, Michael Kearns, Sanjeev Khanna, and Jamie Morgenstern. Strategic network formation with attack and immunization. In *Proceedings of 12th International Conference on Web and Internet Economics*, pages 429–443, 2016.
- [9] Sanjeev Goyal, Shahin Jabbari, Michael Kearns, Sanjeev Khanna, and Jamie Morgenstern. Strategic network formation with attack and immunization. *arXiv:1511.05196 [cs.GT]*, 2016.

- [10] Pascal Lenzner. Greedy selfish network creation. In *Proceedings of 8th International Conference on Internet and Network Economics*, pages 142–155, 2012.
- [11] Niebo Zhang. Best response computation. efficient algorithm to compute the best response for strategic network formation with attack and immunization., 2021.

# Appendix A

## Project Management

In this chapter we present all the information about the project planning. It includes the temporal planning, the budget and the sustainability report.

### A.1 Temporal planning

In this section we summarize the temporal planning of the whole project. We explain the tasks carried out and the amount of time dedicated to each task.

#### A.1.1 Tasks description

In this subsection we explain all the tasks that conform the project.

##### **Project definition**

The first task is to decide what the project will be about. We decide to focus on Algorithmic Game Theory and, specifically, on the Strategic Network Formation with attack and immunization model.

##### **Project planning**

This task can be divided into three stages:

- Temporal planning. We define the time we will spend in each part of the project.
- Economical management. We plan the resources we will need to develop the project and their economical costs.

- Sustainability report. We analyze the environmental, economical and social impact of our project.

### **Control meetings**

This task consists of regular meetings to discuss the current state of the project and the obtained results.

### **Previous study of the model**

We study the previous work done about the Strategic Network Formation with attacks and immunization. We study the model presented in [8, 9] and the theoretical and experimental results they obtain.

### **Theoretical work**

We extend the model presented in [8, 9] to a new one in which the adversary has more power and attacks two players. We study this new model and try to get theoretical conclusions. Unfortunately, we have not been able to provide theoretical proofs to our conjectures, so this work is barely reflected in this document.

### **Experimental setup and implementation**

First, we decide which simulations we want to carry out. We decide we want to study the topology of the swapstable equilibria found with different parameters, for both the model in which the adversary attacks a single player and the model in which the adversary attacks two players. Then, we implement a program which creates an initial graph given the parameters and applies a swapstable dynamics to it, for both models.

### **Experimental execution**

This task consist of deciding the values of the parameters of our program and executing it.

### **Results study**

We analyse the obtained results and we arrive to conclusions. We also evaluate where these results can be applied.

## Documentation

This last tasks consists of documenting all the work done during the project. The documentation includes the project planning, the formal writing of the studied models, the explanation of the simulations done, the conclusions about the obtained results and the code used to run the experiments.

### A.1.2 Dedication to each task

In this subsection we provide a table showing the number of hours we spend on each task.

| #            | Task                                  | Hours      |
|--------------|---------------------------------------|------------|
| 1            | Project definition                    | 5          |
| 2            | Project planning                      | 75         |
| 3            | Control meetings                      | 20         |
| 4            | Previous study of the model           | 50         |
| 5            | Theoretical work                      | 60         |
| 6            | Experimental setup and implementation | 70         |
| 7            | Experimental execution                | 5          |
| 8            | Results study                         | 75         |
| 9            | Documentation                         | 90         |
| <b>Total</b> |                                       | <b>450</b> |

Table A.1: Number of hours spent on each task.

## A.2 Budget

In this section we analyze the economic impact of our project. We divide the costs into direct costs, indirect costs and unplanned costs. We then propose a budget.

### A.2.1 Direct costs

In order to analyze the costs of our project, we count the resources needed, including human, hardware and software resources.

#### Human resources

There are three roles that participate in the development of the project:

1. The **researcher**. She is the responsible of the theoretical part of this project: most part of the previous study of the model, the definition of the new model, the theoretical work and the study of the experimental results.
2. The **software developer**. She is the responsible of implementing the needed software. To do so, she needs to participate on the study of the models.
3. The **project manager**. She is the responsible of planning the project and coordinating the other roles so that the project finishes on time and according to the expected budget.

In the following table we indicate the division of tasks among the three roles:

| #            | Task                                  | Hours | Role 1 | Role 2 | Role 3 |
|--------------|---------------------------------------|-------|--------|--------|--------|
| 1            | Project definition                    | 5     | -      | -      | 5      |
| 2            | Project planning                      | 75    | -      | -      | 75     |
| 3            | Control meetings                      | 20    | 10     | 6      | 4      |
| 4            | Previous study of the model           | 50    | 35     | 15     | -      |
| 5            | Theoretical work                      | 60    | 60     | -      | -      |
| 6            | Experimental setup and implementation | 70    | 10     | 60     | -      |
| 7            | Experimental execution                | 5     | -      | 5      | -      |
| 8            | Results study                         | 75    | 75     | -      | -      |
| 9            | Documentation                         | 90    | 70     | 20     | -      |
| <b>Total</b> |                                       | 450   | 260    | 106    | 84     |

Table A.2: Number of hours spent on each task by each role.

In the following table we present the human resources needed for this project and their cost. The price per hour of each role is computed according to PayScale [2]:

| Role               | Hours | Price per hour | Total    |
|--------------------|-------|----------------|----------|
| Researcher         | 260   | 20 €           | 5,200 €  |
| Software developer | 106   | 25 €           | 2,650 €  |
| Project manager    | 84    | 45 €           | 3,780 €  |
| <b>Total</b>       | 450   |                | 11,630 € |

Table A.3: Cost of the human resources.



## Hardware resources

The only hardware resource needed for this project is the computer used for the experiments, consulting articles, having the control meetings and writing the documentation. The computer is an Acer Aspire V3-572G-781E. The cost is specified in the following table. The price is obtained from the Amazon webpage [1]. To calculate the amortization, we consider 4 months of work.

| Product      | Price | Useful life | Amortization |
|--------------|-------|-------------|--------------|
| Computer     | 750 € | 5 years     | 50 €         |
| <b>Total</b> |       |             | 50 €         |

Table A.4: Cost of hardware resources.

## Software resources

We have used the following software resources in the project:

- Operating system: Ubuntu.
- Programming languages: C++ for the algorithms and Python for the graphs visualization.
- Graphic visualization: graph-tool (Python library).
- Text editor: Sublime Text.
- Communication: Gmail and Google Meet.
- Documentation: L<sup>A</sup>T<sub>E</sub>X.

Most of the software used is Open Source, so there is no cost associated to it, and the rest is free.

| Product                         | Price |
|---------------------------------|-------|
| Ubuntu                          | 0€    |
| C++                             | 0€    |
| Python                          | 0€    |
| graph-tool                      | 0€    |
| Sublime Text                    | 0€    |
| Gmail                           | 0€    |
| Google meet                     | 0€    |
| L <sup>A</sup> T <sub>E</sub> X | 0€    |
| <b>Total</b>                    | 0€    |

Table A.5: Cost of software resources.

### A.2.2 Indirect costs

We take into account the cost of other resources not directly related to the project but necessary for its development. The most important are the electricity used, the Internet and the office supplies.

The average power we need is about  $90W$ , including the computer (it is a laptop that consumes approximately  $70W$ ) and the lighting (an energy saving light bulb which consumes  $20W$ ).

The total energy consumed is the average power multiplied by the total working time that, as we have said before, is 450 hours:  $energy_{total} = 0.09kW * 450h = 40.5kWh$ .

Regarding the office supplies we will need a packet of 500 sheets and about 2 pens.

| Product      | Price        | Units      | Cost     |
|--------------|--------------|------------|----------|
| Electricity  | 0.1 €/ $kWh$ | 40.5 $kWh$ | 4.05 €   |
| Internet     | 40 €/ month  | 4 months   | 160 €    |
| Paper        | 4 €          | 1 packet   | 4 €      |
| Pens         | 0.25 €       | 2          | 0.5 €    |
| <b>Total</b> |              |            | 168.55 € |

Table A.6: Indirect costs.

### A.2.3 Unplanned costs

These are the costs caused by unexpected events that might increase the price of the project.

The computer used may break, in that case we would need to pay the repair cost.

| Probability  | Price | Cost |
|--------------|-------|------|
| 10%          | 100 € | 10 € |
| <b>Total</b> |       | 10 € |

Table A.7: Unplanned costs.

### A.2.4 Final budget

After an exhaustive analysis of the costs the final budget is:

| Concept                      | Price       |
|------------------------------|-------------|
| Human resources              | 11,630 €    |
| Hardware resources           | 50 €        |
| Software resources           | 0 €         |
| <b>Total direct costs</b>    | 11,680 €    |
| Electricity                  | 4.05 €      |
| Internet                     | 160 €       |
| Paper                        | 4 €         |
| Pens                         | 0.5 €       |
| <b>Total indirect costs</b>  | 168.55 €    |
| <b>Total unplanned costs</b> | 10 €        |
| <b>Total</b>                 | 11,858.55 € |

Table A.8: Final budget of the project.

## A.3 Sustainability report

In this section we study the viability and sustainability of the project, using the questions in the Sustainability Matrix.

We divide this analysis into the environmental aspect, the economic aspect and the social aspect. For each one, we study the sustainability in the

project development phase and in the exploitation phase, as well as the risks (variables that may condition the success or failure of the project but are beyond our control) during both phases.

### A.3.1 Environmental sustainability

We have estimated in Section A.2.2 the amount of electricity used during the project development phase, about  $40.5kWh$ . In the United States Environmental Protection Agency (EPA) web page, [3], we find that the  $CO_2$  produced per  $kWh$  is  $0.7kgCO_2/kWh$ . Therefore, the emissions produced in our project are  $emissions = 0.7kgCO_2/kWh * 40.5kWh = 28,35kgCO_2$ , which is much smaller than the  $CO_2$  emissions produced per capita in Spain in 4 months (about 1.8 tons of  $CO_2$  per capita).

During the project development phase, it also has an impact on the environment the amount of paper sheets we use. The impact depends on its origin. We could have left a slightly smaller environmental footprint if we had used recycled paper. Even so, the project has very little impact on the environment.

Regarding the exploitation phase, there is no lifespan maintenance because the project does not produce a product, but some experimental results. The project will not improve the ecological footprint either, as it will not reduce the use of other resources. Therefore, during that phase, our project will have no impact on the environment.

There are eventualities that could cause the project to have a worse impact on the environment. For example, having to work longer hours during the development phase would increase the amount of electricity used. In addition, if the computer broke, we would have to take it to repair, and this reparation would have a cost in electricity consumed and maybe in hardware components used.

### A.3.2 Economic sustainability

In Section A.2 we have quantified the cost of the development phase of the project, about 11,858 €. It is considered viable, since the most part of the budget is assigned to human resources. All contemplated salaries are reasonable taking into account the tasks that have to be done. To reduce the cost, we use Open Source or at least free software.

During the exploitation phase, the project will not have costs because, as we have said before, it does not produce a product, so there is no lifespan maintenance.

The risk is that there is no guarantee to obtain economical benefit from the project, because at the beginning we are not sure to obtain relevant results.

### **A.3.3 Social sustainability**

On a personal level, this project has introduced me into research and into the Algorithmic Game Theory field, which has helped me decide on my professional future. The project has also helped me train some skills such as programming and interpreting results.

Regarding the exploitation phase, the results obtained can be applied to fields such as biology or computer networks, helping us understand how we must protect ourselves or our computers against biological or computer viruses. Therefore, it may improve people's quality of life.

# Appendix B

## Code for Simulations

### B.1 Graph Class

#### B.1.1 Headers file

```
/**
    Represents an undirected graph.
 */

#include <list>
#include <vector>
using namespace std;

typedef vector<list<int>> AdjacencyList;

class Graph {

private:
    ///For each node  $i$ , contains a list of the nodes
    /// $j$  such that exists  $(i, j)$ , or a  $-1$  if  $i$  has
    ///been deleted.
    AdjacencyList edges;

public:
    /**
        Creates an empty graph with  $n$  nodes.
    */
}
```

```

    @param n The number of nodes.
    */
    Graph(int n);

    /**
     Says whether there exists an edge (i,j).

     @param i, j The nodes of the edge.
     @returns True if there exists an edge (i,j).
     */
    bool existsEdge(int i, int j);

    /**
     Adds the edge (i,j) if it does not exist.

     @param i, j The nodes of the edge.
     */
    void addEdge(int i, int j);

    /**
     Erases the edge (i,j) if it exists.

     @param i, j The nodes of the edge.
     */
    void dropEdge(int i, int j);

    /**
     Returns the list of nodes connected to i.

     @param i A node.
     @returns The list of nodes j such that exists
     (i,j).
     */
    list<int> getEdges(int i);

```

```

    /**
     * Says whether the node i has been deleted from
     * the graph.

     * @param i A node.
     * @returns True if the node i has been deleted
     *          from the graph
     */
    bool isDeleted(int i);

    /**
     * Deletes the node i from the graph.

     * @param i A node.
     */
    void deleteNode(int i);

    /**
     * Deletes a list of nodes from the graph.

     * @param i A list of nodes.
     */
    void deleteNodes(list<int> l);
};

```

### B.1.2 Source file

```

#include "graph.h"

Graph::Graph(int n) {
    edges = AdjacencyList(n);
}

bool Graph::existsEdge(int i, int j) {
    list<int> e = edges[j];
    list<int>::iterator it;
    for (it = e.begin(); it != e.end(); ++it) {

```



```

        if (i == *it)
            return true;
    }
    return false;
}

void Graph::addEdge(int i, int j) {
    if (not existsEdge(i, j)) {
        edges[i].push_back(j);
        edges[j].push_back(i);
    }
}

void Graph::dropEdge(int i, int j) {
    edges[i].remove(j);
    edges[j].remove(i);
}

list<int> Graph::getEdges(int i) {
    return edges[i];
}

bool Graph::isDeleted(int i) {
    list<int> e = edges[i];
    return (e.begin() != e.end() and *(e.begin()) == -1);
}

void Graph::deleteNode(int i) {
    edges[i].clear();
    edges[i].push_back(-1);

    for (int j = 0; j < edges.size(); j++)
        edges[j].remove(i);
}

void Graph::deleteNodes(list<int> l) {
    list<int>::iterator it;
    for (it = l.begin(); it != l.end(); ++it)
        deleteNode(*it);
}

```

## B.2 Model Class

### B.2.1 Headers file

```
/**
    Represents the Network Formation with attacks
    and immunization model.
 */

#include <cstdlib>
#include <ctime>
#include <fstream>
#include <utility>
#include "graph.h"
using namespace std;

typedef list<int> VulnerableRegion;

struct strategy {
    list<int> bought;
    bool immunization;
};

class Model {

private:
    ///The current strategy profile:
    vector<strategy> s;

    ///The corresponding graph to s:
    Graph graph;

    ///Edge cost and immunization cost:
    double ce, ci;

    ///The adversary (true for two attacks):
    bool adv2attacks;
```

```

/**
    Immunizes with probability  $p$  the nodes
    of the current strategy profile  $s$  (deimmunizes
    with probability  $1-p$ ).

    @param  $p$  The probability that a node
                immunizes.
*/
void initImmunizations (double p);

/**
    Adds  $m$  random edges into the current
    strategy profile  $s$ .

    @param  $m$  The number of edges to add.
*/
void initEdges(int m);

/**
    Returns a swapstable best response  $s'_i$  for the
    player  $i$  to  $s_{-i}$ , as well as the
    corresponding graph to the strategy profile
     $(s_{-i}, s'_i)$ . If the current strategy of  $s$  is
    already a swapstable best response, returns
    the current strategy.

    @param  $i$  A player.
    @returns A swapstable best response  $s'_i$  and
             the corresponding graph to  $(s_{-i}, s'_i)$ .
*/
pair <strategy , Graph> swapstableBR(int i);

/**
    Tries all the deviations from  $s$  consisting of
     $i$  dropping an edge, both with and without
    changing  $i$ 's immunization status. If a strategy
     $s'_i$  such that  $i$  has a better utility in
     $(s_{-i}, s'_i)$  than  $bu$  is found, updates the

```

```

    best strategy  $bs$ , the corresponding utility
     $bu$  and the graph corresponding to  $(s_{-i}, bs)$ .

    @param[in] i A player.
    @param[out] bs The strategy  $s'_i$  such that  $i$  in
         $(s_{-i}, s'_i)$  has the best utility
        found, if this utility is better
        than  $bu$ .
    @param[out] bg The graph corresponding to
         $(s_{-i}, bs)$ , if a strategy  $s'_i$ 
        such that  $i$  has a better utility
        in  $(s_{-i}, s'_i)$  than  $bu$  is found.
    @param[in, out] bu In: the utility we compare to
        the utilities found. Out: the
        utility of  $i$  in the strategy
        profile  $(s_{-i}, bs)$ , if a
        strategy  $s'_i$  such that  $i$  has
        a better utility in
         $(s_{-i}, s'_i)$  than this
        one is found.

    */
void doDropEdgeDeviations(int i, strategy &bs,
    Graph &bg, double &bu);

/**
    Tries all the deviations from  $s$  consisting of
     $i$  purchasing an edge, both with and without
    changing  $i$ 's immunization status. If a strategy
     $s'_i$  such that  $i$  has a better utility in
     $(s_{-i}, s'_i)$  than  $bu$  is found, updates the
    best strategy  $bs$ , the corresponding utility
     $bu$  and the graph corresponding to  $(s_{-i}, bs)$ .

    @param[in] i A player.
    @param[out] bs The strategy  $s'_i$  such that  $i$  in
         $(s_{-i}, s'_i)$  has the best utility
        found, if this utility is better
        than  $bu$ .
    @param[out] bg The graph corresponding to
         $(s_{-i}, bs)$ , if a strategy  $s'_i$ 

```

```

        such that  $i$  has a better utility
        in  $(s_{-i}, s'_{-i})$  than  $bu$  is found.
    @param[in, out] bu In: the utility we compare to
        the utilities found. Out: the
        utility of  $i$  in the strategy
        profile  $(s_{-i}, bs)$ , if a
        strategy  $s'_{-i}$  such that  $i$  has
        a better utility in
         $(s_{-i}, s'_{-i})$  than this
        one is found.
*/
void doBuyEdgeDeviations(int i, strategy &bs,
    Graph &bg, double &bu);

/**
    Tries all the deviations from  $s$  consisting of
     $i$  swapping two edges, both with and without
    changing  $i$ 's immunization status. If a strategy
     $s'_{-i}$  such that  $i$  has a better utility in
     $(s_{-i}, s'_{-i})$  than  $bu$  is found, updates the
    best strategy  $bs$ , the corresponding utility
     $bu$  and the graph corresponding to  $(s_{-i}, bs)$ .

    @param[in] i A player.
    @param[out] bs The strategy  $s'_{-i}$  such that  $i$  in
         $(s_{-i}, s'_{-i})$  has the best utility
        found, if this utility is better
        than  $bu$ .
    @param[out] bg The graph corresponding to
         $(s_{-i}, bs)$ , if a strategy  $s'_{-i}$ 
        such that  $i$  has a better utility
        in  $(s_{-i}, s'_{-i})$  than  $bu$  is found.
    @param[in, out] bu In: the utility we compare to
        the utilities found. Out: the
        utility of  $i$  in the strategy
        profile  $(s_{-i}, bs)$ , if a
        strategy  $s'_{-i}$  such that  $i$  has
        a better utility in
         $(s_{-i}, s'_{-i})$  than this
        one is found.

```

```

*/
void doSwapEdgesDeviations(int i, strategy &bs,
                           Graph &bg, double &bu);

/**
    Tries the deviation from  $(s_{-i}, cs)$  consisting
    of  $i$  changing her immunization status. If  $i$  has
    a better utility in the new strategy profile
     $(s_{-i}, s'_{-i})$  than  $bu$ , updates the best strategy
     $bs$ , the corresponding utility  $bu$  and the graph
    corresponding to  $(s_{-i}, bs)$ .

    @param[in] i A player.
    @param[in] cs A strategy of  $i$ .
    @param[out] bs The strategy found after  $i$  changes
                    her immunization status,  $s'_{-i}$ ,
                    if  $i$  has a better utility in
                     $(s_{-i}, s'_{-i})$  than  $bu$ .
    @param[in] cg The graph corresponding to
                     $(s_{-i}, cs)$ .
    @param[out] bg The graph corresponding to
                     $(s_{-i}, bs)$ , if  $i$  has a better
                    utility than  $bu$  in the strategy
                    profile after she changes her
                    immunization status.
    @param[in, out] bu In: the utility we compare to
                       the utility found after  $i$ 
                       changes her immunization
                       status. Out: the utility of  $i$ 
                       in the strategy profile
                        $(s_{-i}, bs)$ , if  $i$  has a
                       better utility than
                        $bu$  in the strategy profile
                       after she changes her
                       immunization status.

*/
void changeImmunizationDeviation(int i, strategy cs,
                                  strategy &bs, Graph cg, Graph &bg, double &bu);

```

```

/**
    In strategy  $si$ ,  $i$  buys the edge  $(i, j)$ 
    if it does not exist.

    @param[in, out]  $si$  A strategy of  $i$ .
    @param[in, out]  $g$  The graph corresponding
                        to  $(s_{-i}, si)$ .
    @param[in]  $i$  The node that buys the edge.
    @param[in]  $j$  The node  $i$  buys an edge to.
*/
void buyEdge(strategy &si, Graph &g, int i, int j);

/**
    In strategy  $si$ ,  $i$  drops the edge  $(i, j)$ 
    if it exists.

    @param[in, out]  $si$  A strategy of  $i$ .
    @param[in, out]  $g$  The graph corresponding
                        to  $(s_{-i}, si)$ .
    @param[in]  $i$  The node that drops the edge.
    @param[in]  $j$  The node  $i$  drops the edge from.
*/
void dropEdge(strategy &si, Graph &g, int i, int j);

/**
    In strategy  $si$ ,  $i$  swaps the edge  $(i, j)$  to the
    edge  $(i, k)$  if  $(i, j)$  exists and  $(i, k)$  does not.

    @param[in, out]  $si$  A strategy of  $i$ .
    @param[in, out]  $g$  The graph corresponding
                        to  $(s_{-i}, si)$ .
    @param[in]  $i$  The node that swaps the edge.
    @param[in]  $j$  The node  $i$  drops the edge from.
    @param[in]  $k$  The node  $i$  buys an edge to.
*/
void swapEdges(strategy &si, Graph &g, int i,
               int j, int k);

```

```

/**
    If the utility of  $i$  in the strategy profile
     $(s_{-i}, cs)$  is better than  $bu$ , updates the
    best strategy  $bs$ , the corresponding utility  $bu$ 
    and the graph corresponding to  $(s_{-i}, bs)$ .

    @param[in]  $i$  A player.
    @param[in]  $cs$  A strategy of  $i$ .
    @param[out]  $bs$  The strategy  $cs$ , if  $i$  has a better
        utility in the strategy profile
         $(s_{-i}, cs)$  than  $bu$ .
    @param[in]  $cg$  The graph corresponding
        to  $(s_{-i}, cs)$ .
    @param[out]  $bg$  The graph  $cg$ , if  $i$  has a better
        utility in the strategy profile
         $(s_{-i}, cs)$  than  $bu$ .
    @param[in, out]  $bu$  In: the utility we compare to
        the utility of  $cs$ . Out:  $bs$ 's
        utility, if  $i$  has a better
        utility in the strategy profile
         $(s_{-i}, cs)$  than  $bu$ .
*/
void updateBestStrategy(int i, strategy cs,
    strategy &bs, Graph cg, Graph &bg, double &bu);

/**
    Returns the utility of  $i$  in the strategy profile
     $(s_{-i}, si)$ .

    @param  $i$  A player.
    @param  $si$  A strategy of  $i$ .
    @param  $g$  The graph corresponding to the strategy
        profile  $(s_{-i}, si)$ .
    @returns The utility of  $i$  in the strategy
        profile  $(s_{-i}, si)$ .
*/
double calculateUtility(int i, strategy si,
    Graph g);

```



```

/**
Returns the expected size of the connected
component of  $i$  in the graph  $g$ , with the list of
targeted regions  $tr$ , after the adversary makes
a single attack.

@param  $i$  A player.
@param  $g$  A graph.
@param  $tr$  A list of the targeted regions of the
strategy profile to which corresponds
the graph  $g$ , of at least size 1.
@returns The expected size of  $i$ 's connected
component in the graph  $g$  after the
adversary makes the attack.

*/
double calculateExpectedSzCC1attack(int  $i$ , Graph  $g$ ,
list<VulnerableRegion>  $tr$ );

/**
Returns the expected size of the connected
component of  $i$  in the graph  $g$ , corresponding
to the strategy profile  $(s_{-i}, si)$ , after the
adversary makes two attacks.

@param  $i$  A player.
@param  $si$  A strategy of player  $i$ .
@param  $g$  The graph corresponding to the strategy
profile  $(s_{-i}, si)$ .
@param  $vr$  A list of the vulnerable regions of
maximum size of the strategy profile
 $(s_{-i}, si)$ , of at least size 1.
@returns The expected size of  $i$ 's connected
component on the graph  $g$  after the
adversary makes the attacks.

*/
double calculateExpectedSzCC2attacks(int  $i$ ,
strategy  $si$ , Graph  $g$ ,
list<VulnerableRegion>  $vr$ );

```

```

/**
Returns the expected size of the connected
component of  $i$  in the graph  $g$ , corresponding
to the strategy profile  $(s_{-i}, s_i)$ , which only
has a vulnerable region of maximum size, after the
adversary makes two attacks.

@param  $i$  A player.
@param  $s_i$  A strategy of player  $i$ .
@param  $g$  The graph corresponding to the strategy
profile  $(s_{-i}, s_i)$ .
@param  $vr$  A list which only has an element, the
vulnerable region of maximum size of the
strategy profile  $(s_{-i}, s_i)$ .
@returns The expected size of  $i$ 's connected
component on the graph  $g$  after the
adversary makes the attacks.
*/
double calculateExpectedSzCC1VRmaxSz(int  $i$ ,
strategy  $s_i$ , Graph  $g$ ,
list<VulnerableRegion>  $vr$ );

/**
Returns the expected size of the connected
component of  $i$  in the graph  $g$ , which has more than
one vulnerable region of maximum size, after the
adversary makes two attacks.

@param  $i$  A player.
@param  $g$  The graph.
@param  $tr$  A list of the targeted regions of the
strategy profile to which corresponds
the graph  $g$ , of at least size 2.
@returns The expected size of  $i$ 's connected
component on the graph  $g$  after the
adversary makes the attacks.
*/
double calculateExpectedSzCCmoreVRmaxSz(int  $i$ ,
Graph  $g$ , list<VulnerableRegion>  $tr$ );

```

```

/**
Returns the list of vulnerable regions of
maximum size of the strategy profile
( $s_{-i}$ ,  $si$ ).

@param  $i$  A player.
@param  $si$  A strategy of player  $i$ .
@param  $g$  The graph corresponding to ( $s_{-i}$ ,  $si$ ).
@returns The list of vulnerable regions of
maximum size.
*/
list<VulnerableRegion> getVulnerableRegionsMaxSize
(int  $i$ , strategy  $si$ , Graph  $g$ );

/**
Returns the list of vulnerable regions of the
strategy profile ( $s_{-i}$ ,  $si$ ).

@param  $i$  A player.
@param  $si$  A strategy of player  $i$ .
@param  $g$  The graph corresponding to ( $s_{-i}$ ,  $si$ ).
@returns The list of vulnerable regions.
*/
list<VulnerableRegion> getVulnerableRegions(int  $i$ ,
strategy  $si$ , Graph  $g$ );

/**
Deletes from the graph corresponding to the
strategy profile ( $s_{-i}$ ,  $si$ ) the immunized
nodes of such strategy profile.

@param[in]  $i$  A player.
@param[in]  $si$  A strategy of player  $i$ .
@param[in, out]  $g$  In: the graph corresponding to
( $s_{-i}$ ,  $si$ ). Out: the same
graph without the immunized
nodes of such strategy profile.
*/

```

```

void deleteImmunizedNodes(int i, strategy si,
                          Graph &g);

/**
Returns the size of i's connected component in
the graph g.

@param i The node of the graph.
@param g The graph.
@returns The size of i's connected component in g.
*/
int getConnectedComponentSize(int i, Graph g);

/**
Calculates i's connected component in graph g.

@param[in] i The node of the graph.
@param[in, out] visited In: the already visited
                        nodes of the connected
                        component are true. Out:
                        all the nodes of the
                        connected component are
                        true.

@param[in, out] CC In: the connected component
                    of i that has already been
                    calculated. Out: the whole
                    connected component of i.

@param[in] g The graph.
*/
void getConnectedComponentUtil(int i,
                              vector<bool> &visited, list<int> &CC, Graph g);

public:

/**
Creates a random strategy profile and its
corresponding graph.

```

```

        @param n The number of players.
        @param m The number of edges.
        @param p The probability that a player
                    immunizes.
    */
    Model(int n, int m, double p);

    /**
     Exports the graph corresponding to the
     current strategy profile s as a csv file.
     First row is the number of nodes. Then, for
     each row, first column is the node i,
     second column i's immunization status, and the
     rest of columns the nodes j belonging to xi
     (the nodes i has bought an edge to). The row
     ends with a  $-1$ .

     @param nameFile The name of the file.
    */
    void exportGraph(string nameFile);

    /**
     Runs a swapstable best response dynamics,
     starting from the current strategy profile s.

     @param ce The cost of the edges.
     @param ci The immunization cost.
     @param adv2attacks The adversary (true for the
                        one that makes 2 attacks)
    */
    void dynamics(double ce, double ci,
                  bool adv2attacks);
};

```

## B.2.2 Source file

```
#include "model.h"
```

```

Model::Model(int n, int m, double p) : graph(n)
{
    s = vector<strategy>(n);

    initImmunizations(p);
    initEdges(m);
}

void Model::exportGraph(string nameFile) {
    ofstream myfile;
    myfile.open(nameFile);

    myfile << s.size() << endl;
    for (int i = 0; i < s.size(); ++i) {
        myfile << i << ", " ;

        myfile << s[i].immunization;

        list<int> bought = s[i].bought;
        list<int>::iterator it;
        for (it = bought.begin(); it != bought.end(); ++it){
            myfile << ", " << *it;
        }

        myfile << ", -1" << endl;
    }

    myfile.close();
}

void Model::dynamics(double ce, double ci,
                    bool adv2attacks) {
    this->ce = ce;
    this->ci = ci;
    this->adv2attacks = adv2attacks;

    bool equilibrium;
    do {
        equilibrium = true;

```

```

    for (int i = 0; i < s.size(); ++i) {
        strategy si = s[i];
        pair<strategy, Graph> swBR = swapstableBR(i);
        strategy sbr = swBR.first;

        bool sameStrategy = (si.bought == sbr.bought and
                               si.immunization == sbr.immunization);

        if (not sameStrategy) {
            equilibrium = false; //The strategy profile s
                                   //is not a swapstable
                                   //equilibrium

            s[i] = sbr;
            graph = swBR.second;
        }
    }
    while (not equilibrium);
}

void Model::initImmunizations(double p) {
    srand(time(NULL));

    for (int i = 0; i < s.size(); ++i) {
        if (rand() % 100 < p*100) //True with probability p
            s[i].immunization = true;
        else
            s[i].immunization = false;
    }
}

void Model::initEdges(int m) {
    srand(time(NULL));

    int n = s.size();

    for(int i = 0; i < m; ++i) {
        bool sameNodes;
        bool existsEdge;
    }
}

```

```

do {
    sameNodes = false;
    existsEdge = false;

    int x = rand() % n;
    int y = rand() % n;

    if (x == y)
        sameNodes = true;

    else if (graph.existsEdge(x, y))
        existsEdge = true;

    else {
        if ((rand() % 2) == 0)
            buyEdge(s[x], graph, x, y); //x buys the edge
        else
            buyEdge(s[y], graph, y, x); //y buys the edge
    }
}
while(sameNodes or existsEdge);
}

pair <strategy, Graph> Model::swapstableBR(int i) {
    strategy cs = s[i]; //Current strategy of i, s_i
    strategy bs = cs; //Best strategy of i found.
                        //Initialized to s_i

    Graph cg = graph; //The graph corresponding to s
    Graph bg = cg; //The graph corresponding to
                  //(s_{-i}, bs). Initialized to the
                  //graph corresponding to s

    double bu = calculateUtility(i, cs, cg);
    //bu is the utility of i in (s_{-i}, bs).
    //Initialized to i's utility in s

    changeImmunizationDeviation(i, cs, bs, cg, bg, bu);

    doDropEdgeDeviations(i, bs, bg, bu);
}

```



```

doBuyEdgeDeviations(i, bs, bg, bu);

doSwapEdgesDeviations(i, bs, bg, bu);

return make_pair(bs, bg);
}

void Model::doDropEdgeDeviations(int i, strategy &bs,
                                Graph &bg, double &bu){
    list<int> bought = s[i].bought;
    list<int>::iterator it;
    for (it = bought.begin(); it != bought.end(); ++it) {
        //For each edge i has bought

        strategy cs = s[i]; //Current strategy of i, s_i
        Graph cg = graph; //The graph corresponding to s

        dropEdge(cs, cg, i, *it);

        updateBestStrategy(i, cs, bs, cg, bg, bu);

        changeImmunizationDeviation(i, cs, bs, cg, bg, bu);
    }
}

void Model::doBuyEdgeDeviations(int i, strategy &bs,
                                Graph &bg, double &bu){
    for (int j = 0; j < s.size(); j++) {
        if (not graph.existsEdge(i, j)) {
            //For each edge i has not bought

            strategy cs = s[i]; //Current strategy of i, s_i
            Graph cg = graph; //The graph corresponding to s

            buyEdge(cs, cg, i, j);

            updateBestStrategy(i, cs, bs, cg, bg, bu);

            changeImmunizationDeviation(i, cs, bs, cg, bg, bu);
        }
    }
}

```

```

    }
}

void Model::doSwapEdgesDeviations(int i, strategy &bs,
                                   Graph &bg, double &bu){
    list<int> bought = s[i].bought;
    list<int>::iterator it;
    for (it = bought.begin(); it != bought.end(); ++it) {
        //For each edge i has bought

        for (int j = 0; j < s.size(); ++j) {
            if (not graph.existsEdge(i, j)) {
                //For each edge i has not bought

                strategy cs = s[i]; //Current strategy of i, s_i
                Graph cg = graph; //The graph corresponding
                               //to s

                swapEdges(cs, cg, i, *it, j);

                updateBestStrategy(i, cs, bs, cg, bg, bu);

                changeImmunizationDeviation(i, cs, bs, cg, bg, bu);
            }
        }
    }
}

void Model::changeImmunizationDeviation(int i,
                                         strategy cs, strategy &bs, Graph cg,
                                         Graph &bg, double &bu) {
    cs.immunization = not cs.immunization;
    updateBestStrategy(i, cs, bs, cg, bg, bu);
}

void Model::buyEdge(strategy &si, Graph &g,
                    int i, int j) {
    if (not g.existsEdge(i, j)) {
        si.bought.push_back(j);
        g.addEdge(i, j);
    }
}

```

```

}

void Model::dropEdge(strategy &si, Graph &g,
                    int i, int j) {
    si.bought.remove(j);
    g.dropEdge(i, j);
}

void Model::swapEdges(strategy &si, Graph &g,
                    int i, int j, int k) {
    if (g.existsEdge(i, j) and not g.existsEdge(i, k)) {
        dropEdge(si, g, i, j);
        buyEdge(si, g, i, k);
    }
}

void Model::updateBestStrategy(int i, strategy cs,
                              strategy &bs, Graph cg, Graph &bg,
                              double &bu) {
    double cu = calculateUtility(i, cs, cg);

    if (cu > bu) {
        bs = cs;
        bg = cg;
        bu = cu;
    }
}

double Model::calculateUtility(int i, strategy si,
                              Graph g) {
    double expsz; //The expected size of i's connected
                 //component after the attack.
    list<VulnerableRegion> vr =
        getVulnerableRegionsMaxSize(i, si, g);

    if (vr.size() == 0) //No vulnerable regions, so the
                       //adversary makes no attack
        expsz = getConnectedComponentSize(i, g);

    else if (not adv2attacks) //The adversary is the one
                             //that makes a single attack

```

```

        expsz = calculateExpectedSzCC1attack(i, g, vr);

    else //The adversary is the one that makes two attacks
        expsz = calculateExpectedSzCC2attacks(i, si, g, vr);

    list<int> xi = si.bought;
    bool yi = si.immunization;
    return expsz - (xi.size() * ce + yi * ci);
}

double Model::calculateExpectedSzCC1attack(int i,
        Graph g, list<VulnerableRegion> tr) {
    double expSz = 0; //The expected size of i's
        //connected component after the
        //attack
    double probT = 1.0/tr.size(); //The probability of
        //attack to a
        //targeted region

    list<VulnerableRegion>::iterator it;
    for (it = tr.begin(); it != tr.end(); ++it) {
        //For each targeted region

        VulnerableRegion t = *it;

        Graph aux = g;
        aux.deleteNodes(t); //Delete targeted region t

        expSz += probT * getConnectedComponentSize(i, aux);
        //The size of i's connected component
        //post-attack to t is the size of i's connected
        //component in the graph where we have deleted t
    }
    return expSz;
}

double Model::calculateExpectedSzCC2attacks(int i,
        strategy si, Graph g,
        list<VulnerableRegion> vr) {
    if (vr.size() == 1) { //If there is only one
        //vulnerable region of

```

```

        //maximum size
    return calculateExpectedSzCC1VRmaxSz(i, si, g, vr);
}

else { //If there are more than one vulnerable
        //regions of maximum size
    return calculateExpectedSzCCmoreVRmaxSz(i, g, vr);
}
}

double Model::calculateExpectedSzCC1VRmaxSz(int i,
        strategy si, Graph g,
        list<VulnerableRegion> vr) {
    VulnerableRegion t = *(vr.begin());
    //The vulnerable region of maximum size t

    g.deleteNodes(t); //Delete the vulnerable region t

    vr = getVulnerableRegionsMaxSize(i, si, g);
    //The vulnerable regions of the next maximum size

    if (vr.size() == 0) //The adversary only attacks a
        //vulnerable region, t
        return getConnectedComponentSize(i, g);
        //The size of i's connected component
        //post-attack to t is the size of i's connected
        //component in the graph where we have deleted t

    else //The adversary attacks the vulnerable region
        //t and one vulnerable region of vr
        return calculateExpectedSzCC1attack(i, g, vr);
        //The expected size of i's connected component
        //post-attack to t and a vulnerable region of vr
        //is the expected size of i's connected
        //component in the graph where we have deleted
        //t, with targeted regions vr
}

double Model::calculateExpectedSzCCmoreVRmaxSz(int i,
        Graph g, list<VulnerableRegion> tr) {
    double expSz = 0; //The expected size of i's

```

```

//connected component after the
//attack

double probT = 2.0/(tr.size()*(tr.size()-1));
//The probability of attack to two targeted regions

list<VulnerableRegion>::iterator it;
for (it = tr.begin(); it != tr.end(); ++it) {
    //For each targeted region

    VulnerableRegion t1 = *it;

    Graph aux = g;
    aux.deleteNodes(t1); //Delete targeted region t1

    list<VulnerableRegion>::iterator it2 = it;
    ++it2;
    for (; it2 != tr.end(); ++it2) {
        //For each targeted region after t1

        VulnerableRegion t2 = *it2;

        Graph aux2 = aux;
        aux2.deleteNodes(t2); //Delete targeted region t2

        int ccsz = getConnectedComponentSize(i, aux2);
        //The size of i's connected component
        //post-attack to t is the size of i's
        //connected component in the graph where we
        //have deleted t1 and t2

        expSz += probT * ccsz;
    }
}
return expSz;
}

list<VulnerableRegion> Model::
getVulnerableRegionsMaxSize(int i,
                             strategy si, Graph g) {

```

```

list<VulnerableRegion> vr = getVulnerableRegions(i, si,
                                                    g);

list<VulnerableRegion> tr;
int max = 0;

list<VulnerableRegion>::iterator it;
for (it = vr.begin(); it != vr.end(); ++it) {
    int size = (*it).size();
    if (size > max) {
        max = size;
        tr.clear();
        tr.push_back(*it);
    }
    else if (size == max)
        tr.push_back(*it);
}
return tr;
}

list<VulnerableRegion> Model::getVulnerableRegions
    (int i, strategy si,
     Graph g) {
    deleteImmunizedNodes(i, si, g);

    list<VulnerableRegion> vr;
    vector<bool> visited(s.size(), false);

    //Puts in vr all the connected components of the
    //graph g, where we have deleted the immunized nodes
    for (int j = 0; j < s.size(); ++j) {
        if (not visited[j] and not g.isDeleted(j)) {
            VulnerableRegion v;
            getConnectedComponentUtil(j, visited, v, g);
            vr.push_back(v);
        }
    }
    return vr;
}

void Model::deleteImmunizedNodes(int i, strategy si,
                                  Graph &g) {

```

```

for (int j = 0; j < s.size(); ++j) {
    bool imm; //i is immunized in (s_{-i}, si)
    if (j == i)
        imm = si.immunization;
    else
        imm = s[j].immunization;

    if (imm)
        g.deleteNode(j);
}
}

int Model::getConnectedComponentSize(int i, Graph g) {
    vector<bool> visited(s.size(), false);
    list<int> CC;

    if (not g.isDeleted(i)) //If i has been deleted,
                                //her connected component
                                //size is 0
        getConnectedComponentUtil(i, visited, CC, g);

    return CC.size();
}

void Model::getConnectedComponentUtil(int i,
                                       vector<bool> &visited, list<int> &CC,
                                       Graph g) {
    visited[i] = true;
    CC.push_back(i);

    list<int>::iterator it;
    list<int> edgesi = g.getEdges(i);
    for(it = edgesi.begin(); it != edgesi.end(); ++it) {
        if (not visited[*it])
            getConnectedComponentUtil(*it, visited, CC, g);
    }
}

```



## B.3 Main

```
#include <iostream>
#include <sstream>
#include "model.h"
#include "assert.h"
using namespace std;

int main() {
    int n, m;

    cin >> n;
    assert(n > 0);

    cin >> m;
    assert(m > 0 and m < ((n * (n-1)) / 2));

    double p;
    cin >> p;
    assert(p >= 0 and p <= 1);

    Model model(n, m, p);

    stringstream nameFileInitial;
    nameFileInitial << "initial_graph_n" << n << "_m"
                    << m << "_p" << p << ".csv";
    model.exportGraph(nameFileInitial.str());

    Model auxModel = model;

    int adversary;
    cin >> adversary;
    assert(adversary == 1 or adversary == 2);
    bool adv2attacks = (adversary == 2);

    double ce, ci;
    while (cin >> ce >> ci) {
        assert(ce >= 0 and ci >= 0);

        model = auxModel;
```

```

model.dynamics(ce, ci, adv2attacks);

stringstream nameFileFinal;
nameFileFinal << "final_graph_n" << n << "_m" << m
               << "_p" << p << "_ce" << ce << "_ci"
               << ci << "_" << adversary
               << "attacks.csv";
model.exportGraph(nameFileFinal.str());
}
}

```

## B.4 Graph viewer

```

# Draws a specified graph g in a specified file.
# Input parameters: the name of the csv file where
# g is stored, and the name of the file where we want
# to draw g.

from graph_tool.all import *
import csv
import sys

g = Graph() # Creates an empty directed graph.
vprop_color = g.new_vertex_property("string")

with open(sys.argv[1]) as csv_file:
    csv_reader = csv.reader(csv_file)
    first_row = True

    for row in csv_reader:
        if first_row:
            n = int(row[0]) # In the first row we have
                               # the number of nodes.
            g.add_vertex(n) # Adds n nodes to the graph g.
            first_row = False

        else:
            v = int(row[0]) # The node v.

```

```

if int(row[1]): # The immunization status.
    vprop_color[v] = "blue" # Blue for
                            # immunized nodes.
else:
    vprop_color[v] = "red" # Red for
                          # vulnerable nodes.

i = 2
while int(row[i]) != -1: # The node u.
    e = g.add_edge(v, int(row[i])) # Adds the edge
                                    # (v, u) to
                                    # the graph.

    i += 1

graph_draw(g, output=sys.argv[2],
           vertex_fill_color=vprop_color)

```