UNIVERSITAT POLITÈCNICA DE CATALUNYA

GRAU EN ENGINYERIA INFORMÀTICA

ESPECIALITAT EN COMPUTACIÓ

The Speed of Social Contagion

Author:
Pablo LLUECA MARTÍNEZ

Supervisor:
Dra. Carme ÀLVAREZ FAURA

Q2 Curs 2015-2016

Abstract

In this project we study the spread of the information through a social network. The well known K-CENTER problem consists in, given a graph $G = (V, E)$ and a integer k, to find a subset of nodes $S \subseteq V$, $|S| = k$ such that all nodes in $V \setminus S$ are as close as possible to a node in S. In this project we study a related problem in the context of social networks. Given a graph modelling a social network, we want to find a small subset of nodes such that targeting a product to the persons in the subset lead an adoption of the product in all the graph as fast as possible. We study the problem from an algorithmic game theoretical perspective.

We model the K-CENTER problem as a strategic game where each player chooses a node in the subset S. The set of Nash Equilibria of the game can be seen as a set of alternative approximated solutions to the optimal solution. The computation of the best response for a player and deciding if a strategy is a Nash Equilibrium are polynomial time computable. The computation of the Social Optimum is NP-hard. We show that the Price of Stability of the game is 1. We study the Price of Anarchy of the game for different graph topologies and number of players. We show that for paths and cycles graph the POA is a factor of the number of players. We perform experiments on other graph topologies and with graphs extracted from DBLP co-authoring database.

Resum

En aquest projecte estudiem la difusió de la informació a través de les xarxes socials. El conegut problema dels K-CENTER consisteix en, donat un graf $G = (V, E)$ i un enter k, trobar un subconjunt de nodes $S \subseteq V$, $|S| = k$ tal que tots els nodes de $V \setminus S$ estiguin tan propers com sigui possible a algun node de S. En aquest projecte estudiem un problema relacionat en el context de les xarxes socials. Donat un graf que representa una xarxa social, volem trobar un subconjunt d'individus a qui fer publicitat del nostre producte, de manera que facin que tota la xarxa acabi coneixent-lo el més ràpid possible. Estudiem aquest problema des del punt de vista de la teoria de jocs algorísmica.

Modelem el problema dels K-CENTER com un joc estratègic on cada jugador escull un node del subconjunt S. El conjunt dels *Nash Equilibria* del joc es poden interpretar com un conjunt de solucions aproximades a la solució òptima. La computació de la *best response* i decidir si una estratègia és un Nash Equilibrium són computables en temps polinòmic. La computació d'un *Social Optimum* és NP-hard. Demostrem que el *Price of Stability* del joc es 1. Estudiem el *Price of Anarchy* del joc amb diferents topologies de grafs i nombre de jugadors. Demostrem que per grafs camí i cicles el POA depèn del nombre de jugadors. Mostrem els resultats dels experiments realitzats en altres topologies de grafs així com amb grafs extrets de la base de dades de coautors de publicacions científiques DBLP.

Resumen

En este proyecto estudiamos la difusión de la información a través de las redes sociales. El problema de los K-CENTER consiste en, dado un grafo $G = (V, E)$ y un entero k, encontrar un subconjunto de nodos $S \subseteq V$, $|S| = k$ tal que todos los nodos en $V \setminus S$ estén lo más cerca posible de algún nodo de S. En este proyecto estudiamos un problema relacionado en el contexto de las redes sociales. Dado un grafo que representa una red social, queremos encontrar un grupo de individuos a los que hacer publicidad de nuestro producto, de manera que ellos consigan que toda la red acabe conociendo el producto lo más rápido posible. Estudiamos este problema desde el punto de vista de la teoría de juegos algorítmica.

Modelamos el problema de los K-CENTER como un juego estratégico donde cada jugador escoge un nodo del subconjunto S. El conjunto de los *Nash Equilibria* del juego se puede interpretar como un conjunto de soluciones aproximadas a la solución óptima. La computación de la *best response* de un jugador y decidir si una estrategia es un Nash Equilibrium son computables en tiempo polinómico. La computación de una estrategia que sea un *Social Optimum* es NP-hard. Mostramos que el *Price of Stability* del juego es 1. Estudiamos el *Price of Anarchy* del juego para diferentes topologías de grafos y número de jugadores. Demostramos que para grafos camino y ciclos el POA depende del número de jugadores. Mostramos los resultados de los experimentos realizados con otras topologías de grafos así como con grafos extraídos a partir de la base de datos de co-autoría de publicaciones científicas DBLP.

Contents

List of Figures

Glossary

Nash Equilibrium (NE):
Strategy profile where any player can not unilaterally improve his cost

Social Optimum (OPT):
Strategy profile with the minimum cost

Price of Stability (PoS):
Ratio between the minimum cost of a NE and the OPT

Price of Anarchy (PoA):
Ratio between the maximum cost of a NE and the OPT

Chapter 1

Introduction

This project is guided by two main goals: First, the culmination of the Degree in Informatics Engineering, and the wish to apply together many of the knowledge acquired during the degree, specially from some fields like computational complexity, algorithmic game theory, graph theory and analysis and design of algorithms. Second, to havethe possibility of participating in a work as an introduction to scientific research, to learn how these kind of research is performed, and to go deeper in the study of a concrete field, algorithmic game theory.

1.1 Motivation

On these days marketing based on the social networks and the word-by-mouth have proved to be very effective, with a very little cost a product can reach much more people than traditional marketing. The idea of viral marketing is that a consumer of your product may 'sell' it to his friends, family or co-workers sharing their impressions in social networks such as Twitter, Facebook, Instagram or Youtube. There exist studies about the fact known as 'emotional contagion'. For example in [15] it is suggested that users of social networks are likely to acquire opinions and emotions similar of what they see in their news feed.

Let us imagine that we have a manufacturing company of smart-phones and we build our latest product, the 'aPhone'. We want that our marketing campaign focus on social networks, such as Facebook, Twitter or Instagram. We will select a subset of users of these networks and give them a free 'aPhone', telling them to share their impressions about the product and the brand on social networks, for example posting a picture of it on Instagram or writing a Facebook post about

it. Contacts of these users will see the product and may buy it and share their impressions too, so after a while a large set of people will know the product. Our goal is to maximize the spread of this information, so we have to choose wisely the users that are given the 'aPhone' at the first instance.

Originally this problem was addressed with a centralized approach, as in [10] or in [13]. We want to propose a different approach where different selfish players have to choose which user, or nodes in a graph, should be activated first in order to made a product know by everybody as fast as possible. There are some studies about the problem of social diffusion that will be reviewed in Section 1.2. The relevant difference in our project is the study of the problem from a model of strategic game. We define the QUICK-BURNING GAME. We study the existence of Nash Equilibria in the game and its quality. The analysis of the quality of the equilibria give us an alternative way to attack the problem of finding approximate solutions efficiently. The quality of the solutions depends on how far are those equilibria from the optimal solution.

1.2 The K-CENTER problem and Influence Propagation

In this section we review the previous known results in the literature related with our project. We focus the review in the classical optimization problem of the K-CENTER of a graph as well in the related problem of the propagation of information in social networks.

The K-CENTER problem is a classical facility location problem that asks to find a subset of nodes of a given graph such that all the remaining nodes are "close" to a node in the subset. For instance, one may wish to know where are the best places in a city to place firefighter stations, Internet provider or whatever facility. Formally, the problem can be defined as follows:

K-CENTER PROBLEM: Given a graph $G = (V, E)$ and a integer k, find the smallest l such that exists a subset of nodes $S \subseteq V$ with $|S| \leq k$ such that such that $\max_{v \in V} \{\min_{s \in S} d(v, s)\} \leq l$.

The K-CENTER PROBLEM is computationally hard (see [11]). The MINIMUM DOMINATING SET can be reduced to the K-CENTER PROBLEM, and then the later is NP-hard.

Let us consider the decisional version of the K-CENTER PROBLEM and the decisional version of MINIMUM DOMINATING SET:

K-CENTER-DECISIONAL: Given a graph $G = (V, E)$, an integer k, and an integer l decide if there exists a set of nodes $S \subseteq V$ with $|S| \leq k$ such that $\max_{v \in V} \{\min_{s \in S} d(v, s)\} \leq l$, where $d(u, v)$ is the distance between nodes u and v.

DOMINATING SET: Given a graph $G = (V, E)$ and integer k decide if we there is a subset $S \subseteq V, |S| \leq k$ such that every node of G is in S or has a neighbor that is in S.

K-CENTER-DECISIONAL can be reduced from DOMINATING SET.

Theorem 1.2.1. K-CENTER DECISIONAL *is* NP*-Complete.*

Proof. It is not hard to see that K-CENTER-DECISIONAL $\in$NP. Given a graph $G = (V, E)$, a integer $1 \leq k \leq n$, a integer $l \geq 1$ and a solution $S \subseteq V$, $|S| = k$ the property that $\forall_{v \in V} \exists_{x \in S}$ such that $d(v, x) \leq l$ can be verified in polynomial time with respect to $|V|$.

Let us define a function f such that $< G, k > \in$ DOMINATING SET if and only if $f(< G, k >) \in$ K-CENTER-DECISIONAL defined as $f(< G, k >) =< G, k, 1 >$. By definition $< G, k > \in$ DOMINATING SET if and only if $\exists S \subseteq V |S| \leq k$ such that $\forall_{v \in V} (v \in s) \vee (\exists_{x \in V} (v, x) \in E)$. Then, $< G, k > \in$ DOMINATING SET if and only if $< G, k, 1 > \in$ K-CENTER-DECISIONAL. Hence, $f(< G, k >) =< G, k, 1 >$ reduces from DOMINATING SET to K-CENTER-DECISIONAL. □

Observe that the K-CENTER PROBLEM arises naturally in the context of social networks. For instance, in [17] Nehez et al. study the maximum distance between the nodes of a given set $S \subseteq V$ (called *seeds*) and any other node of the graph. They propose a greedy algorithm to estimate a lower bound of this distance. They also present a genetic algorithm that computes a solution with a quality close to the optimal solution.

We can see that the K-CENTER PROBLEM also have a close relationship with problems of viral marketing and influence propagation in social networks.

Domingos and Richardson study in [10] the problem of choosing in which potential customer should focus a company in order to increase their benefit, not only thinking on a customer as someone who may buy your product, but as someone that can spread your product among their relatives. They modeled

the 'market' of customers as a social network, where every customer has his contacts and influences and it is influenced by them.

In [13] Kempe et al. study the problem of maximizing the social spread of information. They explain the two main diffusion models studied in the literature: the *Linear Threshold Model*, where each node v have a threshold $\theta_v \in [0,1]$ that represents the fraction of neighbours that should be active for v to become active. Each edge between nodes v and u has also a weight $b_{v,u}$. Initially a set of nodes A_0 is activated. At every step t all nodes that were active in step $t-1$ are active, and every node such that the sum of the weights of the edges between its active neighbours is at least his threshold becomes activated:

$$\sum_{u \text{ active neighbour of } v} b_{u,v} \geq \theta_v$$

The process ends when all nodes are active or when no more nodes can become active. The other diffusion model is the *Independent Cascade Model*, which is a probabilistic model. The system has a parameter $p_{v,u}$, where v and u are neighbour nodes, that is the probability that u becomes active because of the influence of v. Like in the Linear Threshold Model a set of nodes A_0 is initially active. When a node v becomes active in step t it can to activate each of his neighbours u with probability $p_{v,u}$. If it succeeds, node u will become active in step $t+1$. If it fails, then node v can not convince u and v will not make any further attempt to convince him in subsequent steps. The process ends when all nodes are active or when no more nodes can become active. In both models the *influence* is represented by $\rho(A)$, the expected number of activated nodes at the end of the process if the initial set of active nodes A_0 is A. The INFLUENCE MAXIMIZATION problem asks, for a parameter k, to find a subset of nodes S of cardinality k such that $\rho(S)$ is maximized. The INFLUENCE MAXIMIZATION problem is NP-hard independently of the diffusion model. They show that the INFLUENCE MAXIMIZATION problem can be approximated with a polynomial time greedy algorithm with a factor of $(1-1/e-\epsilon)$, about a 63% of the quality of the optimum. They experimentally tested their algorithm with data from Arxiv [2] and proof that their algorithm outperforms other know heuristics for this problem. They also introduce a general framework for influence maximization problems, where different types of processes are studied, such the case where nodes can change from inactive to active and from active to inactive.

In [16] Lappas et al. introduce the k-EFFECTORS problem, that asks to find the k nodes of a graph that better explains the observed activation state in a probabilistic diffusion model. They prove that for the general case the k-EFFECTORS problem is NP-hard to solve optimally and also NP-hard to approximate. In some concrete graph topologies, such as the case of trees, the k-EFFECTORS

problem can be solved in polynomial time using a dynamic programming algorithm. They proposed an approximation algorithm that from any graph extracts what they call the *Influence Tree* and use their polynomial time algorithm to find the effectors on that tree. They tested their algorithms against the DBLP dataset, searching effectors in different scientific communities getting that their algorithm performs well enough, finding sets of authors that are really relevant for their communities.

In the majority of the previously commented works the goal is to maximize the influence of an idea of a product without having competence. In [18] Tzoumas et al. study how to maximize the spread of a product when there are other trying to maximize the spread of their products too. They propose a competitive game model where two or more firms try to convince as most of consumers as possible. Initially each firm chooses a number of nodes that will be initially 'infected' with their product and will infect its adjacent nodes. A node can only be infected with one product, and once it adopts it, it never changes. Their main results are that in their proposed game with 2 players is co-NP-hard to decide if there exists a NE in a given game. In the game with an arbitrary number of players, where it may exists NE strategies but its cost may be far from the optimum. They also discuss the specific parameters of the network in relation with the existence of Nash Equilibria. This parameters are the *diffusion depth* of a game (defined as the maximum possible duration of the diffusion process) the *ideal spread* (defined as the maximum possible spread that a strategy can achieve) and the *diffusion collision factor* (defined as a measure for comparing how two strategies of one player perform against a given strategy of another player).

In [8, 9] Bonato et al. introduce a new graph parameter, the *burning number* that measures how fast the nodes of a graph can be contagied. Its contagion model imitates the spread of a fire. Every node is either *burned* or *unburned*. There are discrete time-steps. At every time-step t an unburned node is selected and becomes burned and all the nodes that were unburned and had a burned neighbour at step $t-1$ become burned. The process ends when all nodes are burned. The burning number of a graph G, $b(G)$ is the minimum number of time-steps that needs the process to end. They prove that computing the burning number of a graph in NP-complete. They characterize the burning number via a decomposition of a graph in trees and they conclude that the burning number of an arbitrary graph is the minimum burning number among the spanning trees of the graphs. They also explore the bounds of the burning number for different graph topologies, such as Path graph where they proof that the burning number of a path is $b(\mathscr{P}_n = \lceil n^{1/2} \rceil)$, and similarly, for cycles the burning number is $b(\mathscr{C}_n = \lceil n^{1/2} \rceil)$. They also study the bounds of the burning

number in graph that have been proven to have a similar structure than social networks, such as Cartesian grids and local iterated transitivity model.

In [6, 7] Àlvarez et al. study the problem of the firefighters, which consists in given a graph where a fire is started in a node, choose where to place firefighters that would prevent the fire to be spread. At each round burning nodes spread the fire to their neighbours and a new firefighter is placed. The firefighter problem is NP-hard. In this paper the firefighter problem is modelled as a strategic game, where each player chooses where to place a firefighter only based on their own interest (a player is rewarded in function of the nodes that are saved by his action) instead of being globally coordinated. Their main results are that for the general case the Price of Anarchy of the game, the relation between the Nash Equilibrium with worst cost is lineal with respect the size of the input graph, but there are some topologies, such as trees where the Price of Anarchy is constant. If there can be coalitions between players then the quality of Equilibria increases, yielding that for the general case the Price of Anarchy is $\Theta(\frac{n}{k})$, where n is the size of the graph and k the coalition size.

1.3 Project contribution

We wanted to study the problem of information diffusion in social networks from a game theoretical perspective. The contribution of this project may be divided in two main parts:

- The introduction of a strategic game model where each player chooses one of the seeds in the initial set of active nodes in a graph, chooses a node as a active node. Each of theses selected nodes is called *seed*. The aim is to minimize the time until all nodes are activated. We study the existence of equilibria strategies in the game as a solution concept, as well as the associated algorithmic problems, such as deciding if an strategy is an equilibrium, computing the best response for a player and computing the social optimum.
- The study of the quality of the equilibrium strategies, or what is the same, the Price of Anarchy in different graph topologies. We compare the quality and efficiency of computing Nash Equilibria with other known approximation algorithms for the problem.

1.4 Project outline

The structure of this document goes as follows:

- In Chapter 2 we explain the planning of the project, the tasks subdivision and the planning of the cost and the social impact.
- In Chapter 3 we introduce the formal definitions and concepts related with Game Theory and Strategic games used along this project.
- In Chapter 4 we introduce the problem studied in this project, the QUICK-BURNING PROBLEM and we introduce a strategic game model that models the problem, the QUICK-BURNING GAME. We introduce related computational problems and analyse its complexity.
- In Chapter 5 we study the quality of the strategies that are Nash Equilibria in our game. We study different topologies and we explain the experiments carried out during the project.
- Finally, in Chapter 6 we summarize the obtained results and in Chapter 7 we consider new directions to attack the QUICK-BURNING PROBLEM.
- The project has an Appendix A where the implementation of the algorithms used in the experiments is described.

Chapter 2

Project Management

2.1 Temporal planning

In this section, we will sumarize the temporal planning as well as the resources needed for the development of the project.

2.1.1 Task description

In this subsection we will breakdown the task that conform the development of the project.

2.1.1.1 Project definition

The first task in the project consist in deciding how will the project be. In a initial stage a study on the field of Algorithmic Game Theory was performed as well as an study of the state of the art of the social diffusion problems. We will decide wich specific problem we will treat and how we will modelize it.

Project planning

This task consist in planning how much time we will spend on this project and in what parts, as well as the tasks with relation to the formal inscription and presentation of the project.

2.1.1.1.1 Project planning

This consist in the temporal planning of the project. It have as a result this document.

2.1.1.1.2 Control meetings

We will keep track of the evolution of the project by regular meeting to discuss the obtained results and the current situation at each stage of the project. It will help to prevent posible deviations and to manage them.

2.1.1.1.3 Bureaucratic tasks

This part is the bureaucratic part of the project, consist in inscribe the project in the university, the GEP course and submission and the final presentation of the project in the university.

Main theoretical development

This part will be the core of the project, a theoretical approach of a classical problem using algorithmic game theory. We will propose models and bound its parameters, what will make us able to know how good this models are.

2.1.1.1.4 Model proposal

We want to model the problem of burning a graph (select the nodes where a fire will start, trying to minimize the time spent) as a game, so we have to chose who will the players be, what will be their strategy set, what will be their cost (or payoff) function and how will the social cost (or welfare) computed.

As an example of model, the set of players may be the set of nodes of the graph, their set of actions the choose wheter it is initially burning or not and the cost function for each player may be the out-degree of nodes who are set on fire and 0 for the ones who are not initially burning, if there are at most k nodes initially burning, otherwise all costs will be ∞. The social cost may be the number of time-steps required to burn the graph.

2.1.1.1.5 Model study

Once we have a proper definition of how our game will behave we want to study its properties. This consist of computing the Price of Anarchy and the Price of Stability or, in case its hard to find exact values try to find lower and upper bounds, that may depend of the size of the input graph, the number of players, etc.

2.1.1.1.6 Refination of known properties

Some models may have different behaviour in different graph topologies or may be easier to find equilibriums with a fixed number of players, so we will search where our model performs the best and under wich conditons.

Experimentation

Once we have a good theoretical knowledge about the problem we will implement an algorithm based on our results and test how it behaves with real social networks data.

2.1.1.1.7 Setup

Before start to code we have to install all the needed tools. The programming part will be done using the Python programming language, so the only requirements are to install the interpreter and a text editor. We will also use git as a Version Control System.

2.1.1.1.8 Graph module implementation

This part is not only important for the experimental part, it will be an important helper in the theorical study of the problem. It will consist in implement a library that simulates the game we are studying and computes if given a strategy profile it is a Nash Equilibrium or how many time is needed to burn a graph with some given initial seeds.

2.1.1.1.9 Game theory based algorithm implementation

We will implement an algorithm based on the theoretical results obtained so we will be able to know how optimally we can solve the given problem with selfish actors, and how good is it compared with centralized algorithms.

2.1.1.1.10 Data gather

As the main idea of the project is to study how the ideas propagate through an online social network we will experiment with real world data. There are graph repositories with data from social networks as Facebook or Twitter data, from scientific literature citations an coauthoring or other online comunities. We will execute our algorithm with this data and compare with some centralized algorithm. When we get the data we have to implement a parser to use it as input for our algorithm.

2.1.1.1.11 Results study

After experimenting with real data we will have a good vision about how good is our algorithm in practice, so we will be able to evaluate where can it be applied, for example in marketing.

Documentation

This will consist in the documentation of all the work done during the project. It compromises the project planification documentation, the formal writing of the study we perform and the formal proof of the results we give. The code will also be documented, as well of the results and some conclusions.

2.2 Budget

In this section we will estimate the economic impact of the project. The project does not plan to generate any revenue, so there are no studied profit.

2.2.1 Direct costs

Human resources

The different roles that will participate in the development of the project are the following ones:

1. **Researcher** Responsible of the theoretical part of the project, consisting in propose and study a model for a given problem.
2. **Software developer** Responsible of the implementation of the discussed models and the empirical evaluation of the software.
3. **Project manager** Responsible of planning the project and coordinate the ressources and the actors in order to finish it as expected with the given timespan and budget.

In the following table is especified the division of the tasks described in the temporal planning document. The price per hour of each actor is computed according to PayScale [5].

#	Task	Hours	Role 1	Role 2	Role 3
1	Project definition	5	-	-	5
2	Project planning	70	-	-	70
3	Model Proposal	40	40	-	-
4	Model Study	50	50	-	-
5	Refination of known properties	40	40	-	-
6	Experimental Setup	5	-	5	-
7	Graph module development	30	-	30	-
8	Algorithm implementation	60	30	30	-
9	Data gathering	20	-	20	-
10	Algorithm test and execution	40	-	40	-
11	Results study and discussion	40	30	10	-
12	Control meetings	20	7	7	4
13	Documentation	40	20	20	-
	Total	460	207	162	79

Role	Hours	Price per hour	Total
Researcher	207	30 €	6.210 €
Software developer	162	30 €	4.860 €
Project manager	79	45 €	3.555€
Total	460		14.625 €

Hardware resources

The only hardware resource needed for this project is the computer we will use for developing it. The price is obtained from Apple webpage. The cost is especified in the following table:

Product	Price
Computer	1.449 €
Total	1.449 €

Software resources

Most of the software that will be used to carry out this project is Open Source, so there is no cost associated with it. The Operative System used is not Open Source, but it is licensed with the computer, so it has no aditional cost.

Product	Price
python	0€
vim	0€
LaTeX	0€
git	0€
OSX	0 €
Total	0€

2.2.2 Indirect costs

Other resources not directly related with the project will also be needed for its development, so they must be also considered in the budget planing. The more important are the electricity used and the paper.

The power consumption of a Macbook Pro is 74.9 W per hour, according with Apple [4]. As we said in the planning the project will take about 460 hours, so the total consumed power will be $energy_{total} = 460kW * 74.9h = 34,454kWh$ We will use a lot of paper, about a pack of 500 sheets. It costs around 5€.

Product	Price	Units	Cost
Electricity	0.12 €/ kWh	34,454 kWh	4,13 €
Paper	5 €/ pack	500	5 €
Total			9,13 €

2.2.3 Unplanned costs

This section covers the costs caused by unexpected events that might increase the price of the project. Ideally, none of them will happen. We contemplate two main scenarios where we will need more money to continue with the development of the project:

1 Computer repairs The computer used to develop the project may have hardware failures, in that case we will have to pay a repairment or to replace it.

2 Lack of computer power The computer that we will use is surely enough powerful for the development, but maybe not for the execution of simulations. If the computational resources aviable to use are not enough for executing simulations with big inputs we may have to use an Amazon AWS instance in order to execute there our algorithms.

According to the Amazon price calculator [1] a AWS instansce that fits our requirements will cost 272,2€per month. Suposing that there is an 80% of probability that we need it we compute its cost as $cost_{AWS} = 272,2 * 0,8 = 21,7$ €. An estimate cost of a computer repair is 200€. We dont think it will crash, so we assing a probability of 5% of needing a repair thus the cost will be $cost_{repair} = 200 * 0,05 = 10$ €.

Event	Probability	Price	Cost
1	5 %	200 €	10 €
2	80 %	272.75 €/ Month	21,7€
Total			31,7 €

2.2.4 Total budget

After an exhaustive analysis of the different costs of the project we are able to estimate the total cost of the project.

Concept	Price
Human resources	14.625 €
Hardware resources	1.449 €
Software resources	0 €
Total direct costs	16.074 €
Paper	5 €
Electricity	4,13 €
Total indirect costs	9,13 €
Total unplanned costs	31,7 €
Total	16114.83 €

2.3 Sustainability

2.3.1 Economic sustainability

The price of the project is considered viable since most of the cost come by humans resources, and a big part of the budget will be assigned to them. All contempled salaries are reasonable given the high complexity of the task, both the research and the implementation.

2.3.2 Social sustainability

Our studies may help us on understanding how we behave in social networks and how are we affected by the other individuals in that scenario. It may be applied to make comunication between people faster, what will improve its life quality.

2.3.3 Environmental sustainability

The hardware resources needed to develop the project are described in 2.2.1. From them we cant estimate the impact studying how much electricity does

them used, in total about $34kWh$, and the emisions produced in the production of that energy are about 13Kg of CO2, wich is significantly smaller thant the average energy consumption per capita. In general this project have a very small environmental footprint, because most of the work will be purely theoretical.

Chapter 3

Preliminaries: Strategic Games

Game theory is the branch of mathematics and economics that models situations where multiple individuals try to get as much benefit as they can, but their outcome is affected by other individuals choices.

A *strategic game* is the mathematical representation of a situation where a set of individuals, called *players*, have to decide what they do. Each player can only do a limited number of things, the *actions* that he can play. Players recibe a *payoff* or pay a *cost* that depends on the actions of the whole set of players. All players play simoultaneously and they are unware of what the other players do. The game consist in only one turn, where every player picks an action. Players are selfish, so they only seek their own benefit (they try to maximice they payoff or minimize his cost).

Definition 3.0.1. Formally a strategic game is defined by a tuple

$$\Gamma = (N, (A_i)_{i \in N}, (c_i)_{i \in N})$$

where:

- $N = \{1, \ldots, n\}$ is the set of players.
- A_i is the set of actions that are available to player $i \in N$. A strategy of player i consists in selecting an action $s_i \in A_i$. Let $\mathbb{S} = A_1 \times \ldots \times A_n$ be the set of all possible strategy profiles.
- For each player $i \in N$, $c_i : \mathbb{S} \to \mathbb{R}$ is de cost function of player i.

Strategies. Each player chooses his action $s_i \in A_i$ once, completely uninformed from the decisions of other players. A *strategy profile* $s = (s_1, \ldots, s_n) \in \mathbb{S}$ of Γ is a n-tuple composed of the strategies s_i selected by each player i. Given s

and given a strategy $s_i' \in A_i$, let (s_{-i}, s_i') denote the strategy profile that is the outcome of replacing s_i by s_i', in s, $(s_{-i}, s_i') = (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$

Costs and Payoffs. The preferences of player i are usually represented with his *cost* (or *payoff*) *function* c_i. The aim of each player is to select an strategy that minimizes (or maximize, if the preference is defined as a payoff) his cost.

Social cost. The *social cost (payoff) function* $C : \mathbb{S} \to \mathbb{R}$ ($U : \mathbb{S} \to \mathbb{R}$ if it is a payoff function) defines how bad (good) is a given strategy profile not for one particular individual, but for all the society.

Nash Equilibrium

A *Nash Equilibrium* is a strategy profile where no player, knowing his cost with this strategy, wants to change his action, because in any case he would not improve his cost.

Definition 3.0.2. Given a strategic game $\Gamma = (N, (A_i)_{i \in N}, (c_i)_{i \in N})$ we say that a strategy profile $s^* = (s_i^*, \ldots, s_n^*)$ is a Nash Equilibrium (NE) of Γ if for every player i and for every $s_i' \in A_i$ it holds that $c_i(s^*) \leq c_i(s_{-i}^*, s_i')$. Let $\mathrm{NE}(\Gamma)$ denote the set of all NE strategies.

Best Response. Given a strategy profile s, the *best response* of player i to s_{-i} is defined by the set of actions of player i that minimize his cost function suposing that other players will stick to their actions. Formally, given any strategy profile s and given any player i,

$$\mathrm{BR}(\Gamma, s, i) = \{s_i \in A_i \mid c_i(s_{-i}, s_i) = \min_{s_i' \in A_i}(c_i(s_{-i}, s_i'))\}$$

Notice that $s^* = (s_1^*, \ldots, s_n^*)$ is a NE if and only if for each player i it holds that $s_i \in \mathrm{BR}(s^*, i)$.

Social Optimum

A strategy profile s of $\Gamma = (N, (A_i)_{i \in N}, (c_i)_{i \in N})$ is a *social optimum* (OPT) if for all $s' \in \mathbb{S}$, $C(s) \leq C(s')$, i.e. s minimizes the social cost C. Let $\mathrm{OPT}(\Gamma)$ be the set of all OPT of Γ. Note that a social optimum does not necessarily have to be a Nash Equilibrium.

Price of Anarchy

The Price of Anarchy (PoA) was first defined by Koutsoupias and Papadimitriou in [14]. It is a parameter that measures how selfish behaviour of players affect the social outcome of the game. In other words, it quantifies how much the social cost grows if players are not coordinated. If we think of NE as the positions to where players will naturally evolve, a good way of quantificate the degradation is comparing the social costs of worst NE and the OPT.

Definition 3.0.3. Let $\Gamma = (N, (A_i)_{i\in N}, (c_i)_{i\in N})$ be a strategic game and $s^* \in \text{OPT}(\Gamma)$. The Price of Anarchy of Γ is defined as follows:

$$\text{PoA}(\Gamma) = \frac{\max_{s\in \text{NE}(\Gamma)} C(s)}{C(s^*)}$$

Price of Stability

The Price of Stability (PoS) is the ratio between the lowest NE cost and the cost of any OPT. It quantifies how good can selfish players do even if they act not coordinated. In a game with $\text{PoS} > 1$ the social cost of equilibrium strategies will be always higher than the social optimum.

Definition 3.0.4. Let $\Gamma = (N, (A_i)_{\forall i\in N}, (c_i)_{\forall i\in N})$ be a strategic game and $s^* \in \text{OPT}(\Gamma)$. The Price of Stability of Γ is defined as follows:

$$\text{PoS}(\Gamma) = \frac{\min_{s\in \text{NE}(\Gamma)} C(s)}{C(s^*)}$$

Examples

For better understanding all these concepts we give a pair of examples of strategic games with significant differences.

The *Prisoners Dilemma*. There are two criminals that are caught by the police and being judged simultaneously. The police have evidences of both of them committing a minor crime and they suspect but do not have evidences of any of them committing a major crime unless either of them betrays his collaborator. If they remain silent, then they will spend 1 year in prison for his minor crime. If only one of them betrays his partner, the traitor will be free as a reward and the betrayed will be convicted for the major crime, so will spend 4 years in prison.

If both of them fink and betray their partner, both will be convicted for a major crime, but will be sightly rewarded for cooperation, so they will spend 3 years in prison. The game is represented by $\Gamma = (N = \{1,2\}, (A_1, A_2), (c_1, c_2))$ where:

- The players are the two criminals
- $A_1 = A_2 = \{\text{Quiet}, \text{Fink}\}$
- $\mathbb{S} = A_1 \times A_2 = \{(\text{Quiet}, \text{Quiet}), (\text{Quiet}, \text{Fink}), (\text{Fink}, \text{Quiet}), (\text{Fink}, \text{Fink})\}$
- We can represent the cost of each player as the years that would have to spend in prison. For player 1 we have that c_1(Fink, Quiet) = 0, c_1(Quiet, Quiet) = 1, c_1(Fink, Fink) = 3, and c_1(Quiet, Fink) = 4. For player 2 we have that c_2(Fink, Quiet) = 4, c_2(Quiet, Quiet) = 1, c_2(Fink, Fink) = 3, and c_2(Quiet, Fink) = 0.

The costs of both players with respect to the strategies are shown in the table below:

	Quiet	Fink
Quiet	1,1	4,0
Fink	0,4	3,3

The Social Cost of the game is defined as the sum of the years that both of them spend in prision, so $C(s) = c_1(s) + c_2(s)$, as shown in the table below:

	Quiet	Fink
Quiet	2	4
Fink	4	6

In the Prisoners Dilemma there only exists one NE, when both prisoners fink. If both prisoners fink any of them does not have incentive of changing from fink to stay quiet, because his cost will increase. As we see the social cost of the equilibrium strategy is greater than the social optimum, so in this case the Price of Anarchy and the Price of Stability are 3.

Altought the Prisioners Dilemma have 2 NE strategies, there exists strategic games with no NE. One of them if the *Matching Pennies* game. There are two players, each one with a coin. They have to show a face of the coin, so they have two strategies, Head and Tail. The first player wins if both shown faces are equal, and the second player wins if they are different. The payoff for each player are shown in the table below, with 1 representing a win and -1 a loss.

	Head	Tail
Head	$1,-1$	$-1,1$
Tail	$-1,1$	$1,-1$

If the strategy is (Head, Head) or (Tail, Tail) player 1 cannot improve his cost, but player 2 can change his strategy and win the game, improving his cost. Similarly, if strategy is (Tail, Head) or (Head, Tail) player 1 can change his action and improve his cost, hence there do not exists any strategy where players are in equilibrium.

Chapter 4

Quick-Burning Game

In this chapter we introduce a strategic game that models the social contagion. In a graph that represent a social network. We let each of the players choose a node of the graph where the information will star. Thinking in terms of viral marketing each player chooses a customer that will be given a free product or a good bargain and will talk about it to their contacts.

4.1 QUICK-BURNING PROBLEM

We define the *burning time* of a graph $G = (V, E)$ and a set of nodes $S \subseteq V$, $b(G, S)$, called SEEDS, as the number of steps untill all nodes are activated if initially the only active nodes are the ones in S. Every step all the unactive neighbors of an active node became activated. We can also define the burning time as:

$$b(G, S) = \max_{v \in V} \{\min_{s \in S} \{d(v, s)\}\}$$

Thinking in a Social Network, if we want to publicy a new product we can show it to a few users that will share the information in the social network and expect other users to get to know the product for subsequents shares. The burning time is the time untill all users are know the product.

We define the QUICK-BURNING problem as follows: Given a graph $G = (V, E)$ and an integer k, we want to find a subset of nodes $S \subseteq V$ such that the burning time $b(G, S)$ is minimized. The QUICK-BURNING problem is equivalent to the optimization version of K-CENTER problem.

4.2 Definition of the QUICK-BURNING GAME

The main subject of this project is to develop and study a non centralised algorithm for solving the K-CENTER problem. We present a strategic game model where each player chooses a node to place a seed independently of other players. We study how are the social optimums of the game and the properties of its equilibria.

We introduce a new model of strategic game named QUICK-BURNING GAME. An instance of QUICK-BURNING GAME consist in a graph and a number of players. Each player choose a node of the graph where he place a *seed* that will activate that node. The aim of every player is to activate all of the nodes as fast as possible.

Formally an instance of the QUICK-BURNING GAME is defined by a tuple $\Gamma = (G, k)$, where:

- $G = (V, E)$ is a undirected graph. From now on, let $n = |V|$.
- k is the number of players, $k \leq n$. We assume that the set of players is $N = \{1, \ldots, k\}$.

Each player chooses a node in the graph that will be activated at the initial step (seed node). Hence, for each $i \in N$ the set of actions of player i is defined by $A_i = V$. A strategy profile of Γ is a k-tuple $s = (s_1, \ldots, s_k) \in \overbrace{(V \times V \times \cdots \times V)}^{k}$ where s_i is the node selected by player i.

The goal of the problem is to minimize the burning time, so players are rewarded as the burning number decreases. All players have the same cost function, so for any strategy the cost of every player will be the same. For each player i the cost function is defined as the burning time of G given the set of seeds $S = \{s_i | 1 \leq i \leq k\}$ or ∞ cost when his action is the same than some other player action. With this we force that the equilibriums have no repeated seeds, which would make the cost of the equilibrium strategies increase. More formally,

$$c_i(s) = \begin{cases} \infty & \exists j \neq i, s_i = s_j \\ b(G, S) & \text{otherwise} \end{cases}$$

Figure 4.1: Cost function for players in QUICK-BURNING GAME

Initially we tough about a cost function that rewards every player for their individual contribution, computing the cost of every player as

$$c_i'(s) = \begin{cases} \infty & \exists j \neq i, s_i = s_j \\ b(G, S - \{s_i\}) - b(G, S) & \text{otherwise} \end{cases}$$

Figure 4.2: Alternative cost function for players

We saw that the equilibrium strategies are exactly the same with both cost functions. If a strategy is a NE using the first cost function it means that $b(G,S)$ cannot improve unilateraly, so any player can make the graph burn faster, then using the second cost function no player can improve his cost, because it cannot burn the graph faster. Similarly, if a strategy is a NE using the second cost function then any player can not contribute more to his cost. If any player can not improve his cost, then $b(G,s)$ can not decrease by an unilateral change. Then the strategy is a NE using the first cost function. As the NE are the same with both cost functions we kept the simpler one.

The social cost of a strategy profile $s = (s_1, \ldots, s_k)$ is the burning time of G given $S = \{s_i | 1 \leq i \leq k\}$, $C(s) = b(G,S)$. The social cost of a strategy is the same that the cost of that strategy for each player.

4.3 Social Optimum and Nash Equilibria

In this section we study the existence and properties of Nash Equilibria in the QUICK BURNING GAME and the relation with the Social Optimum.

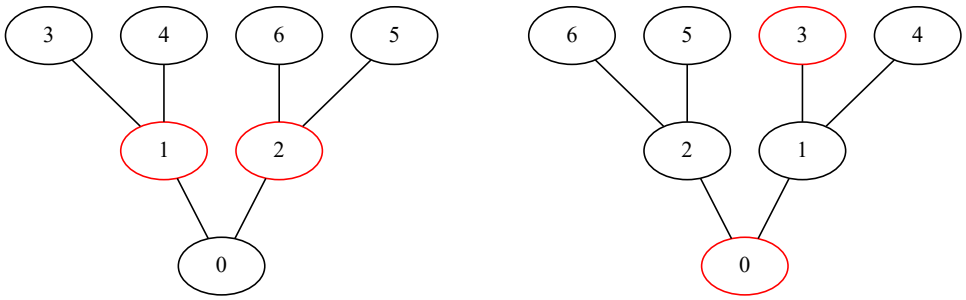
Proposition 4.3.1. *Any social optimum of a* QUICK-BURNING GAME Γ *is a NE. Then,* $\text{POS}(\Gamma) = 1$.

Proof. Let us suppose that $s^* = (s_1^*, \ldots, s_k^*) \in \text{OPT}(\Gamma)$ and $s^* \notin \text{NE}(\Gamma)$. Then, there exists a player i that can improve his cost by changing his strategy s_i^* by another strategy s_i. Since $C(s_{-i}^*, s_i) = c_i(s_{-i}^*, s_i)$ and $C(s^*) = c_i(s^*)$ then □

As a corollary of Proposition 4.3.1 we can conclude that for any instance of QUICK BURNING GAME Γ there exists at least one NE, and that there exist a strategy s such that $s \in \text{NE}$ and $s \in \text{OPT}(\Gamma)$, so the Price of Stability of the QUICK BURNING GAME is 1.

Proposition 4.3.2. *There exists a strategy* $s^* \in \text{NE}$ *such that* $s^* \notin \text{OPT}(\Gamma)$

Proof. Let $G = (V, E)$ be a complete binary tree of depth 2. Let $\Gamma = (G, 2)$. The social optimum strategy, illustrated in Figure 4.3a, is the one which places seeds in the first level, so the root and all leaves are at distance 1 from a seed so the social cost is 1., where the seeds are marked in red. In Figure 4.3b we can see another NE strategy for Γ, but this one have cost 2, because nodes 5 and 6 are at distance 2 from node 0. Any player can change to improve the cost, so the strategy in Figure 4.3b is a NE but it is not the Social Optimum. □



(a) Social Optimum and NE

(b) NE but not Social Optimum

Figure 4.3: Example of NE with worst cost than Social Optimum

4.4 Computational Complexity

In this section we refer a strategy as s (a tuple) and S the set such that $S = \{s_i | 1 \leq 1 \leq k\}$.

In this section we analyse the complexity of some problems related to the QUICK-BURNING GAME such as the computation of the Best Response, finding a Nash Equilibrium or computing the Social Optimum of a game.

First we study how can we compute the best change that a player can unilaterally do, the BEST RESPONSE:

BEST RESPONSE Given Quick-burning game $\Gamma = (G, k)$, a strategy $s = (s_1, \ldots, s_k)$ and an integer$i$ $1 \leq i \leq k$ compute a strategy s'_i that for all s''_i $c_i(s_{-i}, s'_i) \leq c_i(s_{-i}, s''_i)$.

The best response of a player may consist in changing the seed that he choose. In order to compute the best response one may iterate over all possible actions for a player, keeping only the one with the lowest cost.

Proposition 4.4.1. *The* BEST RESPONSE *problem is polynomial time computable.*

Proof. Algorithm 1 computes the best response testing all possible alternative moves for player i and then return one with minimum cost. As the alternative moves are the n nodes of G the for loop will be executed n times. □

Input: $\Gamma = (G, k)$, $s = (s_1, \ldots, s_k)$, i integer
Output: s'_i such that $s'_i \in \mathrm{BR}(s, i)$
$c \leftarrow b(G, s)$
$s_r \leftarrow s_i$
for $s'_i \in (V - S)$ **do**
    $c' \leftarrow b(G, s')$ where $s' = (s_{-i}, s'_i)$
    if $c' < c$ **then**
        $s_r \leftarrow s'_i$
        $c \leftarrow c'$
    end
end
return s_r

Algorithm 1: `Best_response`

Another important problem is how hard is to check if a given strategy is a Nash Equilibrium or not:

IS NE Given a Quick Burning Game $\Gamma = (G, k)$ and a strategy $s = (s_1, \ldots, s_k)$ decide if $s \in \mathrm{NE}(\Gamma)$, where $\Gamma = (G, k)$.

Since the strategy of any player in a NE profile is a best response for that player, then it is not hard to see that deciding whether a given strategy is a NE is polynomial time computable.

Proposition 4.4.2. *The* IS NE *problem is polynomial time computable.*

Proof. Algorithm 2 determines if a given strategy profile s is a NE. For each player i computes its best response so if the action of that player is not in its best response then s is not a NE because player i could change his action to one in his best response and his cost decrease. The for loop will be executed at most k times, so knowing that the best response can be computed in $\Theta(n)$ time the temporal complexity of the algorithm is $\Theta(nk)$.

□

Input: $\Gamma = (G, k)$, $s = (s_1, \ldots, s_k)$
Output: True if $s \in \text{NE}(\Gamma)$, False otherwise
for $i \leftarrow \{1, \ldots, k\}$ **do**
    $s_i^* \leftarrow \text{BR}(\Gamma, s, i)$
    $c* \leftarrow c_i((s_{i-}, s_i*))$
    $c \leftarrow c_i(s)$
    if $c > c*$ **then**
        return False
    end
end
return True

Algorithm 2: `is_NE`

Now we can proof that, given a game, finding a strategy s such that $s \in \text{NE}$ and that the social cost of s is the lowest possible in the game is an intractable problem.

OPT STRATEGY Given a Quick-burning Game $\Gamma = (G, k)$ compute a strategy profile $s \in \text{OPT}(\Gamma)$

Proposition 4.4.3. *The* OPT STRATEGY *problem is* NP*-hard.*

Proof. Let us shown that OPT STRATEGY is in FP then K-CENTER-DECISIONAL is in P. If OPT STRATEGY is in FP, given an input for K-CENTER-DECISIONAL, $< G, k, l >$, we can represent the game $\Gamma = (G, k)$ and compute an optimum strategy $s^* \in \text{OPT}(\Gamma)$, the one with minor cost possible.

$$< G, k, l > \in \text{K-CENTER-DECISIONAL} \iff l \geq C(s^*), s^* \in \text{OPT}(\Gamma)$$

If $C(s^*) \leq l$ then the the subset of vertices $S = \{v | v \in s^*\}$ holds that for every vertex $v \in V$ there exists a vertex $u \in S$ such that $d(u, v) \leq l$, then $< G, k, l >$ is in K-CENTER-DECISIONAL. If the cost of the optimum is bigger than l then does not exists such subset, so $< G, k, l >$ is not in K-CENTER-DECISIONAL.

Then,

□

4.5 Best Response Dynamics

In this section we study a Dynamics process for the Quick-burning game model based on the computation of the best response.

Our dynamics start can start at any given strategy profile and at each round a player computes his best response and changes his action to that. It is described by the following algorithm:

Input: Γ Game, $s = (s_1, \ldots, s_k)$ strategy profile
Output: strategy s such that $s' \in \text{NE}((\Gamma))$
while $s \notin \text{NE}(\Gamma)$ **do**
  foreach *player* i **do**
    Compute $s'_i \in \text{BR}_i(s)$
    if $c_i(s_{-i}, s'_i) < c(s)$ **then**
      $s = (s_{-i}, s'_i)$
    end
  end
end
return s

Algorithm 3: `Best_Response_Dynamics`

Proposition 4.5.1. *Best Response Dynamics on* QUICK-BURNING GAME *converges to a* NE *in a polynomial time.*

Proof. As the social cost and the cost for every player are the same, whenever a player switches his strategy during the best response dynamics the social cost decreases. The maximum social cost possible is the diameter of G ($diam(G)$), which is clearly smaller than the number of nodes, so there may be at most $diam(G)$ rounds. □

A δ-approximation algorithm for a NP-hard problem is a polynomial time algorithm that computes a solution to the problem whose cost (the parameter that we want to optimize) is at most δ times the optimum. A δ-approximation algorithm for the K-CENTER PROBLEM is is NP-hard for any $\delta < 2$, but there exists a polynomial time 2-approximation proposed by Hochbaum and Shmoys in [12]. The proposed algorithm consists in finding an Independent Set I in the square graph G^2 of the input graph G such that $|I|$ is smaller than the size of the minimum dominating set in G. The square graph of G is a graph $G^2 = (V', E')$ where the edge $(u, v) \in E'$ if there is a path of length at most 2 between u and v in G.

It seems natural to think that if the Dynamics algorithm always converge in a polynomial number of steps then the PoA will ≥ 2, so the strategies computed will have a cost far from the social optimum. If from any starting strategy profile s the social cost $C(\texttt{best\_response\_dynamics}(\Gamma, S)) < 2\text{OPT}(\Gamma)$ then P = NP because as stated in [12] by Hochbaum et al. the best possible approximation for the K-CENTER problem is a 2-approximation. Then we think that the cost of the strategies computed by `best_response_dynamics` can not give an approximation guarantee of at most two times the cost of the social optimum. P = NP.

In the next chapter we study the PoA for particular graph topologies.

Chapter 5

Quality of Equilibria

Given a game is interesting to know how good are the equilibria, what is the cost of the best and worst equilibrium with respect to the optimum. If the worst equilibrium have a bad cost with respect to the optimum that means that anarchic players can end in a very bad strategy but none of them can do anything to improve it.

We have observed that with a Best Response dynamics process the cost of the NE outputed is it likely to be at least twice of the cost of OPT. In this section we study the quality of the equilibrium strategies with respect to the topology of the graph and the number of players that induced the game.

5.1 Nash Equilibria in path graphs

In this section we study the quality of the equilibria in the class of path graphs. A path graph is a connected graph where each node has degree 2 except two of them, the endpoints, that have degree 1. Let $\mathscr{P}_n$ denote a path graph of n nodes. From now on, we refer the nodes of $\mathscr{P}_n$ as follows: The endpoints are labeled by 1 and n, respectively. A node at a distance i from node 1 is labeled by $i+1$, for $1 < i < n$ (see an example in Figure 5.1).

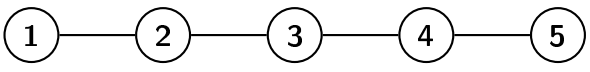


Figure 5.1: Example of path graph with 5 nodes

2 Players game

First we characterize the quality of equilibria in the QUICK-BURNING GAME when the number of players is fixed to 2.

Given any strategy profile (s_1, s_2) we can assume without loss of generality that $s_1 \leq s_2$. Note that the burning distance in a path graph $\mathscr{P}_n$ is upper bounded by the distances between seeds and between seeds and endpoints. Formally,

$$b(\mathscr{P}_n, \{s_1, s_2\}) = \max\{d(s_1, 1), \left\lceil \frac{d(s_1, s_2)}{2} \right\rceil, d(s_2, n)\}$$

Proposition 5.1.1. *Let* $\Gamma = (\mathscr{P}_n, 2)$ *be a* QUICK-BURNING GAME *and let* opt $\in$ OPT(Γ). *Then,* $C(\text{opt}) = \lceil \frac{n-2}{4} \rceil$

Proof. Let us consider the graph $\mathscr{P}_n$ (see Figure 5.2) and strategy $s = (\lceil \frac{n-2}{4} \rceil + 1, n - \lceil \frac{n-2}{4} \rceil)$. The burning time of this strategy is $b(\mathscr{P}_n, s) = \lceil \frac{n-2}{4} \rceil$.

If player 1 decides to change s_1 to $s_1' \in \{1, \ldots, \lceil \frac{n-2}{4} \rceil\}$ then $b(G, (s_{-1}, s_1')) = d(s_1, s_2) \geq \lceil \frac{n-2}{4} \rceil$. If player 1 decides to change s_1 to $s_1' \in \{\frac{n-2}{4}+2, \ldots, s_2-1\}$ then $b(G, (s_{-1}, s_1')) = d(1, s_1) \geq \lceil \frac{n-2}{4} \rceil$. Finally, if $s_1' \in \{s_2+1 \ldots, n\}$ then $b(G, (s_{-1}, s_1')) = d(1, s_2) > \lceil \frac{n-2}{4} \rceil$.

In a similar way we can also prove that neither player 2 has incentive to change his strategy individually nor both of them at the same time.Therefore, any strategy $s' \neq s$ satisfies $C(s') \geq C(s)$. Then, s is a social optimum. □

- $a = \frac{n-2}{4}$
- $s_1 = \lceil \frac{n-2}{4} \rceil + 1$
- $b = \frac{n-2}{4} + 2$
- $c = \lceil \frac{3n-2}{4} \rceil - 1$
- $s_2 = \lceil \frac{3n-2}{4} \rceil$

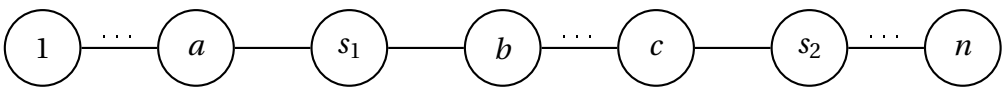


Figure 5.2: Social optimum in line graph with 2 players

In a path graph it is convenient that the seeds divide the path in parts of similar length. Then the burning time is optimum. If they are very close one to the other at the center of the path, then they can not improve individually the

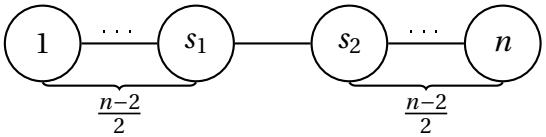


Figure 5.3: Worst Nash Equilibrium in path graphs with n even, $s_1 = \frac{n}{2}$ and $s_2 = \frac{n}{2} + 1$ the distance between them is 1

burning time. This is the case of a NE with a high social cost (see Figures 5.3 and 5.4).

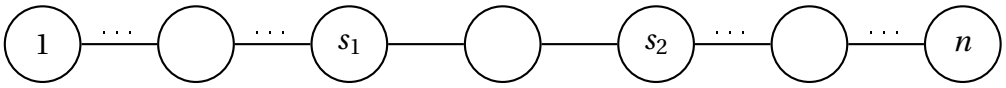


Figure 5.4: Worst Nash Equilibrium in path graphs with n odd $s_1 =$ and $s_2 =$ the central node of the graph is between them, so they are at distance 2:

Proposition 5.1.2. *Let* $\Gamma = (\mathscr{P}_n, 2)$ *be a* QUICK-BURNING GAME. *There exists* $s \in \mathrm{NE}(\Gamma)$ *such that for all* $s' \in \mathrm{NE}(\Gamma)$, $C(s') \leq C(s) = \frac{n-2}{2}$.

Proof. The strategy shown in Figure 5.3 is $s_{\text{even}} = (\frac{n}{2}, \frac{n}{2}+1)$ and the one in Figure 5.4 is $s_{\text{odd}} = (\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 2)$.

There cannot be a strategy profile with worst cost than s_{even} or s_{odd} and is a Nash Equilibrium. If player 1 decides to change s_1 to s_1' we can have two different cases:

- $s_1' \in \{1, \ldots \lfloor \frac{n}{2} \rfloor\}$: The distance between s_2 and n will not be modified, so $b(G, (s_{-1}, s_i')) = \frac{d(s_2,n)}{2} = b(G, s)$.
- $s_1' \in \{\frac{n}{2}+2, \ldots n\}$: In this case there is not any seed between 1 and s_2, so the burning time is the maximum between $d(1, s_2)$, $d(s_1', s_2)/2$ and $d(s_1', n)$. The burning time $b(G, (s_{-1}, s_1') > b(G, s)$ because $d(s_2, 1) > d(s_1, 1)$.

Similarly we have that player 2 cannot change his strategy and improve his cost. □

Corollary 5.1.1. *Let* $\Gamma = (\mathscr{P}_n, 2)$ *be a* QUICK-BURNING GAME. *Then,* $\mathrm{POA}(\Gamma) = 2$

k players game

Lemma 5.1.1. *Let* $\Gamma = (\mathscr{P}_n, k)$ *be a* QUICK-BURNING GAME. *Then,* $\mathrm{OPT}(\Gamma) = \lfloor \frac{n}{2k} \rfloor$.

Proof. Let l be an integer such that $2(l-1)k+k < n \leq 2lk+k$.

Let $s^* = (l+1+i(2l+1))_{\forall i \in \{0,k-1\}}$, with cost $C(s^*) = \frac{n-k}{2k}$ be a strategy for $\Gamma = (\mathscr{P}_n, k)$. The strategy s^* is represented in Figure 5.5.

In s the distance between two consecutive seeds s_i, s_{i+1} is at most $2l$ and at least $2l-1$. Also $l-1 \leq (s_1, 0) \leq l$ and $l-1 \leq d(s_k, n) \leq l$. The cost of s^* is $C(s^*) = \max\left\{d(s_1,1), d(s_k,n), \left(\frac{d(s_i,s_{i+1})}{2}\right)_{\forall 1 \leq i \leq k}\right\}$ then $C(s^*) = l = \frac{n-k}{2k}$.

$\forall_{s' \in \mathbb{S}} C(s') \geq C(s^*) \leftrightarrow s^* \in \text{OPT}(\Gamma)$ Let us suppose s' a strategy such that $C(s') = l-1 < C(S^*)$. If $C(S') = l-1$ then for each node $v \in V$ there is a seed in $x \in s'$ such that $d(u,x) \leq l-1$, then the maximum number of nodes that can have a path graph with cost $l-1$ is $n' = 2(l-1)k+k$, so we reach a contradiction because $n > n'$.

□

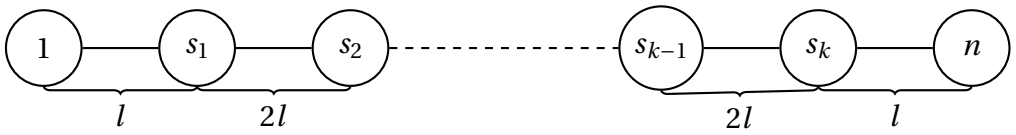


Figure 5.5: Best Nash Equilibrium in path graphs

Proposition 5.1.3. *Let $\Gamma = (\mathscr{P}_n, k)$, for any $n \geq k$. There exists a Nash Equilibrium strategy s such that $C(s) = \frac{n-k}{2}$ and for every other Nash Equilibrium strategy s' it holds that $C(s') \leq C(s)$*

Proof. Let $s = (\frac{n-k}{2}+1, \ldots \frac{n-k}{2}+k)$ be a strategy profile for Γ.

$C(s) = \frac{n-k}{2}$. The burning distance $b(G,s) = \max\{d(1,s_1), d(s_k,n)\} = \frac{n-k}{2}$.

$s \in$ NE. If any player i wants to change his strategy s_i to s_i' he have two options:

- $s_i' \in \left\{1, \ldots \frac{n-k}{2}\right\}$ The distance $d(s_k,n)$ does not change, so $b(G,(s_{-i}, s_i')) = b(G,s)$.
- $s_i' \in \left\{\frac{n-k}{2}+k+1, \ldots, n\right\}$ The distance $d(s_1,1)$ does not change, so $b(G,(s_{-i}, s_i')) = b(G,s)$.

All other possible moves are already ocuped by another seed, so player i does not improve his cost changing to there (the cost would be ∞). Then any player has incentive to change his strategy, therefor $s \in$ NE.

$\forall_{s' \in \mathbb{S}} C(s') \leq C(s)$. Let us suppose a strategy s' such that $s' \in$ NE and $C(s') > C(s)$. Having a cost bigger than s means that all the seeds in s' are in $\{1, \ldots, \frac{n}{2}-1\}$ or in

$\{\frac{n}{2}+1,\ldots,n\}$ because if there is at least one seed in the center of the graph and the others are distributed in both sides the maximum cost is $C(s)$. If all seeds are in one side of the graph then it cannot be a NEbecause any player could change to the node $\frac{n}{2}$ and improve the cost. □

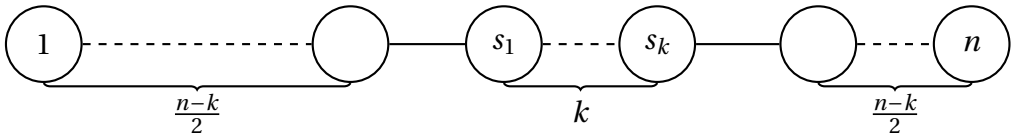


Figure 5.6: Worst Nash Equilibrium in path graphs

Corollary 5.1.2. *Let* $\Gamma = (\mathscr{P}_n, k)$. *Then* $\text{PoA}(\Gamma) = O(k)$

Proof. The PoA is the ratio between the Social Optimum and the Equilibrium with worst cost:

$$\text{PoA}(\Gamma) = \frac{\max_{s\in \text{NE}} C(s),}{\min_{s\in\mathbb{S}} C(s)}$$

$$\text{PoA}(\Gamma) = \frac{\frac{n-k}{2}}{\frac{n}{2k}}$$

$$\text{PoA}(\Gamma) = \left(1-\frac{k}{n}\right)k$$

□

As a conclusion of the study of the QUICK-BURNING game in path graphs we can say that the Price of Anarchy is at most $n/4$. Let us supose a path graph $\mathscr{P}_n$ for a arbitrary big n and the game $\Gamma(\mathscr{P}_n, \frac{n}{2})$. With $\frac{n}{2}$ we can place all seeds starting from one endpoint of the graph each two nodes, $s = (1,3,\ldots,n-2,n)$ then $C(s) = 1$. The strategy profile s is a social optimum and a NE. In this case the worst NE is $s_w = (\frac{n}{4},\ldots,n-\frac{n}{4})$, where all seeds are ocuping the middle of the graph. The cost of s_w is $C(s_w) = \frac{n}{4}$ then $\text{PoA} = \frac{\frac{n}{4}}{1} = \frac{n}{4}$.

The worst case equilibrium possible in Path graphs is when $k = n/2$, so there is a seed for every two nodes of the graph. The social optimum in this case is when all odd (or even) labeled nodes of the graph have a seed on them, then the Social Cost is clearly 1.

But, if we place all nodes in the middle of the path, the social cost is $n/4$, then $\text{PoA} = n/4$

5.2 Nash Equilibria in cycle graphs

In this section we study the quality of the equilibria in cycle graphs. A cycle graph (from now on denoted by $\mathscr{C}_n$, where n is the number of nodes in the graph) is a connected graph where each node have degree 2.

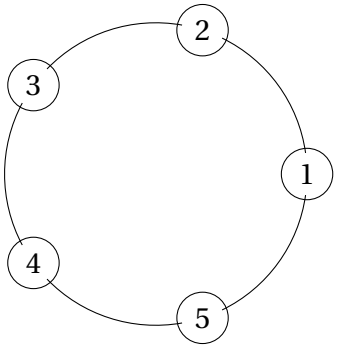


Figure 5.7: Cycle graph $\mathscr{C}_5$

2 players game

Lemma 5.2.1. *Let* $\Gamma = (\mathscr{C}_n, 2)$. *If* s *is a Nash Equilibrium of* Γ *then* $C(s) = \left\lceil \frac{n-2}{4} \right\rceil$

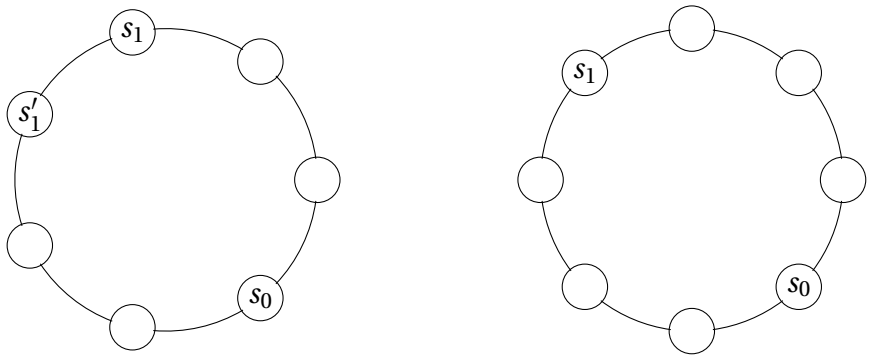
Proof. This proof is divided in two cases, depending on the parity of n.

n even For each node v in $\mathscr{C}_n$ there exists a node v_o, that we will call the *opposite* of v in $\mathscr{C}_n$, such that the distance from v to v_o is $\frac{n-2}{2}$. There are two paths from v to v_o that have this length. The burning time of the strategy $s = (v, v_o), \forall v \in \mathscr{C}_n$ is $\frac{n-2}{2}/2 = \frac{n-2}{4}$ and it is a Nash Equilibrium because no player can improve his cost. If $s_1 = v$ and $s_2 \neq v_o$ the burning time will be greater than $\frac{n-2}{4}$ because one of the paths from s_1 to s_2 will be longer that $\frac{n-2}{2}$, then s_2 will always have incentive to change to v_o because it will improve his cost. This case is illustrated in Figure 5.8a

n odd For each node v in $\mathscr{C}_n$ there exists two opposite nodes v_{o1} and v_{o2} that are at maximum distance from v, the distance from v to v_{o1} and v_{o2} is $\frac{n-3}{2}$. A strategy consisting in $s = (v, v_{o1})$ or $s = (v, v_{o2})$, $\forall v \in \mathscr{C}_n$ have cost $C(s) = \frac{n-3}{4}$ and it is a Nash Equilibrium because none of the players can improve his cost. Given a strategy profile $s = (s_1, s_2)$ if $s_1 = v$ and $s_2 \notin \{v_{o1}, v_{o2}\}$ the burning time will be greater than $\frac{n-3}{4}$, so it holds that

$C(s_{-2}, v_{o1}) < C(s)$ then s cannot be a Nash Equilibrium.This case is illustrated in Figure 5.8b

□



(a) (s_0, s_1) and $(s_0, s_1') \in$ NE (b) NE with n odd

Figure 5.8: NE in $\mathscr{C}_n$ with $k = 2$

Corollary 5.2.1. *Let* $\Gamma = (\mathscr{C}_n, k)$. *Then* $\text{PoA}(\Gamma) = 1$

Proof. The PoA is the relation of the cost of OPT and the worst NE. By Proposition 4.3.1 every NE is an optimum then in the game with 2 players PoA = 1. □

k players game

In the k players game with cycle graph, given a strategy s we can define the burning cost as:

$$b(G, s) = \max\left\{\frac{d(s_1, s_2)}{2}, \frac{d(s_2, s_3)}{2}, \ldots, \frac{d(s_{k-1}, s_k)}{2}, \frac{d(s_k, s_1)}{2}\right\}$$

supposing that for each $1 \leq i \leq k$ there is not any seed in the path between s_i and s_{i+1} ($s_{k+1} = s_1, s_{1-1} = s_k$).

Lemma 5.2.2. *Let* $\Gamma = (\mathscr{C}_n, k)$, *for any* $k \leq n$. *Then* $\text{OPT}(\Gamma) = \left\lceil \frac{n-k}{2k} \right\rceil$

Proof. Let l be a integer such that $(l-1)k + k < n \leq lk + k$. Let

$$s^* = ((l+1)\, i)_{\forall\, i \in \{0,\ldots,k-1\}}$$

be a strategy for Γ. In s^* it holds that $\exists_{i \in \{1,\ldots,k\}}$ such that $d(s_i, s_{i+1}) = l$ and that $\forall\, i \in \{1, \ldots, k\}$ then $d(s_i, s_{i+1}) \leq l$. An example of this strategy is show in Figure 5.9.

$C(s^*) = l$. The maximum distance between two consecutive seeds is l, then $b(G, s^*) = \frac{l}{2}$.

$s^* \in \text{OPT}(\Gamma)$. Let us suppose the strategy s' such that $C(s') < C(s^*) = l$. In s' the maximum distance between two consecutive seeds it at most $l-1$, if l is odd and $l-2$ is l is even. Then, the number of nodes would be $n' \leq k(l-1)+k$, so we reach a contradiction because $n' < n$. □

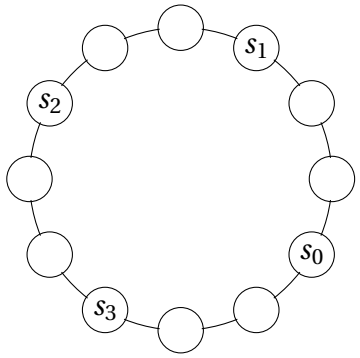


Figure 5.9: OPT in $\mathscr{C}_{12}$ with $k = 4$

Lemma 5.2.3. *Let $\Gamma = (\mathscr{C}_n, k)$, for any $n \geq k$. There exists a Nash Equilibrium strategy s such that $C(s) = \left\lceil \frac{n-k-1}{4} \right\rceil$ and for every other Nash Equilibrium strategy s' it holds that $C(s') \leq C(s)$*

Proof. Let $r = \frac{n-k}{2}$. Let $s = (i)_{\forall i \in \{0,\ldots,(k-1)/2\}}, (k+r+1)_{\forall_{i \in \{1,\ldots,/k-1)/2\}}}$. There is an example of s in Figure 5.10a with $n = 12$ and $k = 6$.

$C(s) = r/2$. In s the seeds are grouped in two opposite sides of the cycle. The maximum distance between two consecutive seeds is the maximum between $d(s_1, s_k)$ and $d(s_{\frac{k}{2}}, s_{\frac{k}{2}+1})$ that is $r/2$.

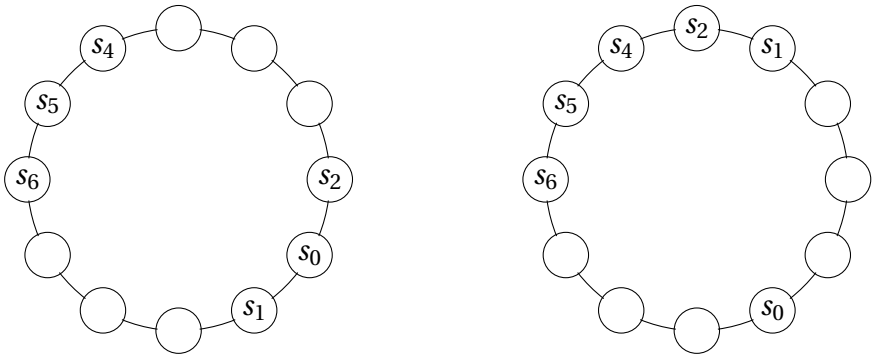
$s \in$ NE. Any player cannot change his action and improve his cost. Suppose a player i want to change from s_i to s_i'. Since both sides of the cycle are full of seeds s_i' can only be in the seeds between s_1 and s_k or between $s_{\frac{k}{2}}$ and $s_{\frac{k}{2}+1}$.

- In the seeds between s_1 and s_k there wont be any new seed between $s_{\frac{k}{2}}$ and $s_{\frac{k}{2}+1}$, so $b(G, (s_{-i}, s') = b(G, s)$.
- In the seeds between $s_{\frac{k}{2}}$ and $s_{\frac{k}{2}+1}$ there wont be any new seed between $s_{\frac{k}{2}}$ and $s_{\frac{k}{2}+1}$, so $b(G, (s_{-i}, s') = b(G, s)$.

Therefore any player cant improve his cost changing his strategy.

□

There can be other strategies that are in NE and have cost $r/2$ such the one in Figure 5.10a.



(a) (b)

Figure 5.10: Worst NE in $\mathscr{C}_{12}$ with $k = 6$

Corollary 5.2.2. *Let* $\Gamma = (\mathscr{C}_n, k)$. *The* PoA$(\Gamma) = O(k)$.

Proof.

$$\text{PoA}(\Gamma) = \frac{\max_{s \in \text{NE}} C(s),}{\min_{s \in \mathbb{S}} C(s)}$$

$$\text{PoA}(\Gamma) = \frac{\frac{n-k-1}{4}}{\frac{n-k}{2k}} = \frac{2k(n-k-1)}{4(n-k)}$$

$$\text{PoA}(\Gamma) = \frac{k}{2}\left(1 - \frac{1}{n-k}\right)$$

□

As we can see the PoA is sightly smaller in cycles than it is in paths but it is also dependent on the number of players.

5.3 Other topologies: An experimental study

The experiments with different types of graphs gave us intuition to obtain the theoretical results of the previous section. We implemented a experimental setup where we could simulate the QUICK-BURNING GAME with different graphs and number of players.

5.3.1 Algorithms

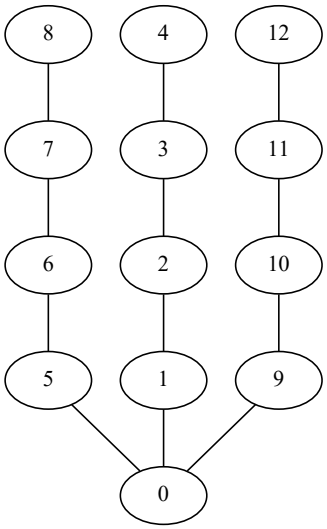
We implemented first a Python module for representing and constructing graphs. We can construct graph with a list of vertices and edges, from a concrete topology, such as a path graph, cycle graph, different types of trees or a random graph with concrete features, such as a connected graph with a given number of nodes and edges, a random tree of a given number of nodes or a bipartite graph. The implementation details can be found at Appendix A.1. Given a graph we can compute the matrix distance between its nodes and compute the burning time of a graph and a subset of its nodes.

For each graph topology we generate a dataset of graphs with different sizes and features and, for every graph, we compute the cost of the best and the worst NE with different number of players.

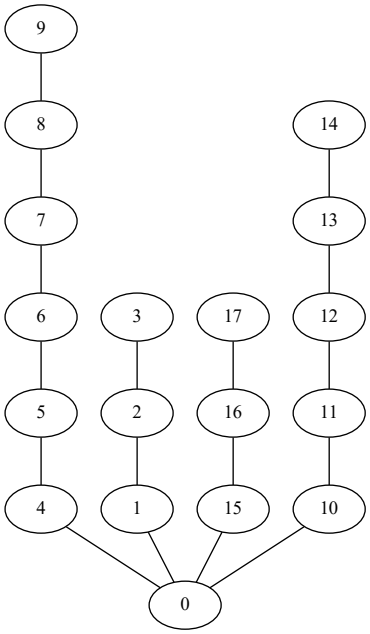
5.3.2 Topologies

Path-Star Graphs

A Path-Star Graph is a tree such that only one node, called root, have unbounded degree and the remaining nodes have degree at most 2. The paths between the root and the leaves are called *legs*. In Figure 5.11a we can see a star graph with 3 legs of length 4. Note that legs don't necessarily have the same length, such as the one in Figure 5.11b.



(a) Star Graph with 3 legs of length 4



(b) Star Graph with 4 legs of different length

We generate star-path graph with 3, 4 and 5 legs, all with same length, from 5 to 10. The implementation of the star-path graph is described in Appendix A.1. For each graph, we compute all the NE with from 3 to 6 players.

The relevant results on star-path graph are summarized in table 5.12. The parameters of the experiment are the following:

- # legs: Degree of the root of the graph
- length: Distance from the root to the leaves
- k: Number of players
- $C_{\min}$: Minimum cost of a NE
- $C_{\max}$: Maximum cost of a NE
- POA: Price of the Anarchy

# legs	length	k	$C_{\min}$	$C_{\max}$	PoA
4	5	2	5	5	1.0
4	5	3	5	5	1.0
4	5	4	3	5	1.6
4	5	5	2	5	2.5
4	5	6	2	5	2.5
4	10	2	10	10	1.0
4	10	3	10	10	1.0
4	10	4	5	10	2.0
4	10	5	3	10	3.33
5	5	2	5	5	1.0
5	5	3	5	5	1.0
5	5	4	5	5	1.0
5	5	5	3	5	1.6
5	5	6	2	5	2.5

Figure 5.12: Quality of Equilibria in star-path graph

If $k <$ #legs, then the PoA is 1, because the cost of all equilibrium strategies is the length of the longest leg. As there are not enough seeds to place one in every leg then all equilibrium strategies should place one seed in the root. If there is not a seed in the root, the cost of the strategy is the distance from the closest seed to the root plus the length of the longest leg without a seed. But, if $k \geq$ #legs, then the PoA increases because there exists equilibrium strategies that split the seeds among the legs, so the nodes in every leg have a close seed, then the cost is smaller than the leg length, but there are equilibrium strategies with worst cost, for example any strategy that places a seed at the root and leaves two or more legs without any seed.

The worst NE are strategies where all seeds are close to the root and there are at least two legs without any seed. The best NE are strategies where every leg have nodes dividing it as much as possible. The PoA increases with the number of players and the length of the legs.

Binary Trees

A complete binary tree is a binary tree in which all nodes but the root and leaves have degree 3. The distance between the root and the leaves is called the *depth* of the tree. In Figure 5.13 we can see a complete binary tree of depth 4.

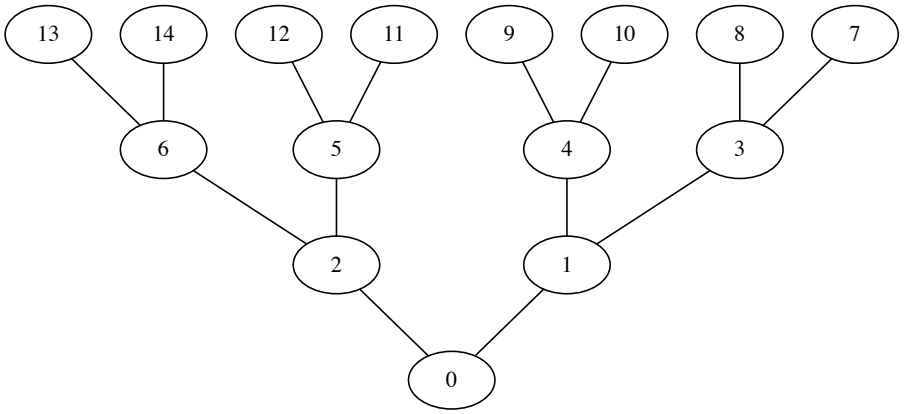


Figure 5.13: Complete binary tree of depth 4

We generate complete binary trees with depth from 2 to 6. With each graph we compute all the NE with from 2 to 7 players. The implementation of the binary tree is described in Appendix A.1.

The relevant results on complete binary trees are summarize in Table 5.14. The parameters of the experiment are the following:

- depth: Distance from the root to the leaves
- k: Number of players
- $C_{\min}$: Minimum cost of a NE
- $C_{\max}$: Maximum cost of a NE
- PoA: Price of the Anarchy

depth	k	$C_{\min}$	$C_{\max}$	PoA
4	2	2	3	1.5
4	3	2	3	1.5
4	4	2	3	1.5
4	5	1	3	3
4	6	1	3	3
4	7	1	3	3
5	2	3	4	1.33
5	3	3	4	1.33
5	4	2	4	2
5	5	2	4	2
5	6	2	4	2
5	7	2	4	2
6	2	4	5	1.25
6	3	4	5	1.25
6	4	3	5	1.66
6	5	3	5	1.66
6	6	3	5	1.66

Figure 5.14: Results on Binary Tree

We observe from the experiments in binary trees the worst NE are when the seeds are close to the root, so with only one change at least one of the paths between the root and the leaves will be uncovered, then the cost is close to the depth of the tree. The best case NE are when all paths from the leaves to the root have seeds distributed over them. Than is possible when we have enough seeds to cover a whole level of the tree, then all paths have nodes in it. The cost in this case depends on how deep is the level that we can cover, the cost is better as the level is more equidistant from the root and the leaves, and on how many levels can we cover. The PoA increases with the number of players.

n-ary Trees

A n-ary tree is a tree such that every node have at most n children. A n-ary tree is called complete if all interior nodes (those that are neither the root or a leaf) have exactly n children and all the leaves are at the same distance from the root. The distance from a leaf to the root is called the depth of the tree. In Figure 5.15 we can see a 3-ary tree with depth 3.

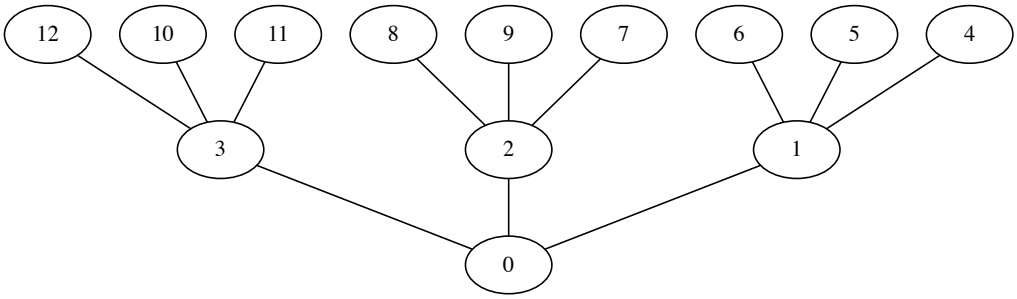


Figure 5.15: 3-ary tree of depth 3

We generate n-ary trees with from 3 to 5 branches and depth from 3 to 5. With each graph we compute all the NE with from 2 to 5 players. The implementation of the n-ary tree is described in Appendix A.1

The relevant results on complete n-ary trees are summarized in Table 5.16. The parameters of the experiment are the following:

- # branches: Maximum children that a node can have
- depth: Distance from the root to the leaves
- k: Number of players
- $C_{\min}$: Minimum cost of a NE
- $C_{\max}$: Maximum cost of a NE
- POA: Price of the Anarchy

# branches	depth	k	$C_{\min}$	$C_{\max}$	POA
3	4	2	3	3	1.0
3	4	3	2	3	1.5
3	4	4	2	3	1.5
3	4	5	2	3	1.5
4	4	2	3	3	1.0
4	4	3	3	3	1.0
4	4	4	2	3	1.5
4	4	5	2	3	1.5

Figure 5.16: Results on n-ary trees

As in the case of binary trees, the worst case NE are strategies where all seeds are close to the root. The best NE are when some levels as far from the leaves and to the root are completely covered with seeds. The PoA depends on the number of players.

Series parallel graphs

We define simple series parallel graph as the subclass of series parallel graphs where each graph consists in two terminal nodes s and t and one or more disjoint paths between s and t. In Figure 5.17 is shown a Simple Series Parallel graph with 4 paths of length 4.

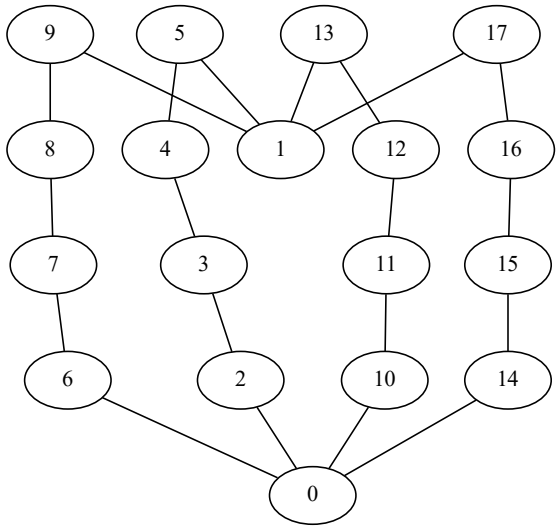


Figure 5.17: Simple series parallel graph with 4 paths of length 4

We generate simple series parallel graphs with 3 and 4 paths of length between 3 and 8. The implementation of simple series parallel graphs is described in Appendix A.1. With each graph we compute all the NE with from 2 to 8 players.

The relevant results on simple series parallel graphs are summarized in Table 5.18. The parameters of the experiment are the following:

- # paths: Number of disjoint paths between terminal nodes

- length: Length of the paths between terminal nodes
- k: Number of players
- $C_{\min}$: Minimum cost of a NE
- $C_{\max}$: Maximum cost of a NE
- PoA: Price of the Anarchy

# paths	length	k	$C_{\min}$	$C_{\max}$	PoA
3	8	2	4	5	1.25
3	8	3	4	5	1.25
3	8	4	3	5	1.66
3	8	5	2	5	2.5
3	8	6	2	5	2.5
3	8	7	2	5	2.5
3	8	8	1	5	5

Figure 5.18: Results on simple series parallel graphs

As we have been observing in different types of graphs all the diametral paths should be covered in order to have good strategies. The worst NE are strategies where all seeds are close to the terminals s and t, then the cost is closer to the length of the paths between s and t. The best NE are strategies where seeds are distributed along all the paths between s and t. If all of the paths between the terminals cannot be covered with at least one seed because k is too small, then the PoA is small, but when we have enough seeds it rapidly increases when we increase the number of players.

5.3.3 Real data: DBLP dataset

After studing simple graphs we study how the game behaves with real data from an online social network. We tested the game model using a graph extracted from DBLP dataset. We used a snapshot of the DBLP dataset from February 2, 2016, downloaded from [3]. The DBLP dataset consist un a database of scientific papers, with attributes such as the topic, the publication date, the author (or authors), etc.

From this data we constructed the graph $G_{\text{dblp}} = (V, E)$ as follows: Nodes in V are authors of papers. There exists an edge (u, v') in E if the authors u and v have co-authored at least one paper.

Since the whole graph was excessively big we extract subgraphs from it and evaluate how the equilibrium strategies are. We select different subgraphs by diameter. We generated graphs with sizes from 20 to 1000, with diameters from 1 to 12. The computation of all the NE of a graph than more than 100 nodes last for long time, so we only study the equilibria of small subgraphs of DBLP (the whole graph have about 300000 nodes). Wich each graph we computed all the NE in games with from 2 to 6 players. In Figures 5.20 and 5.19 we can see different DBLP subgraphs.

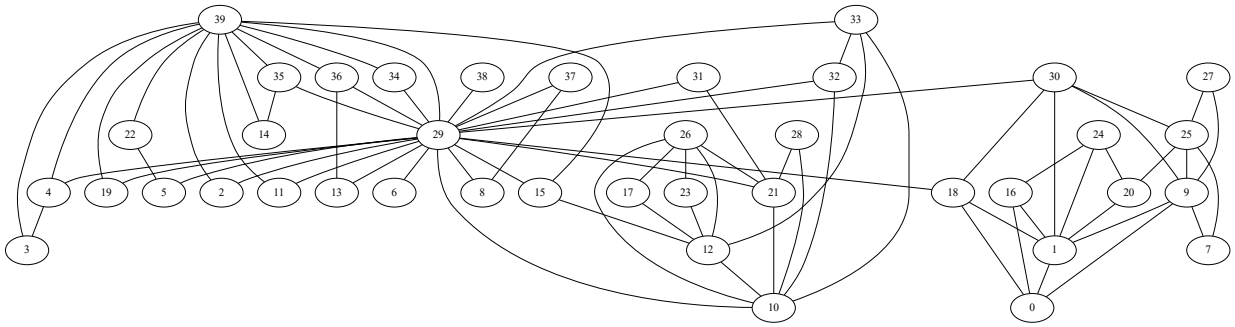


Figure 5.19: Example subgraph of DBLP colaboration with $n = 40$ and diameter= 6

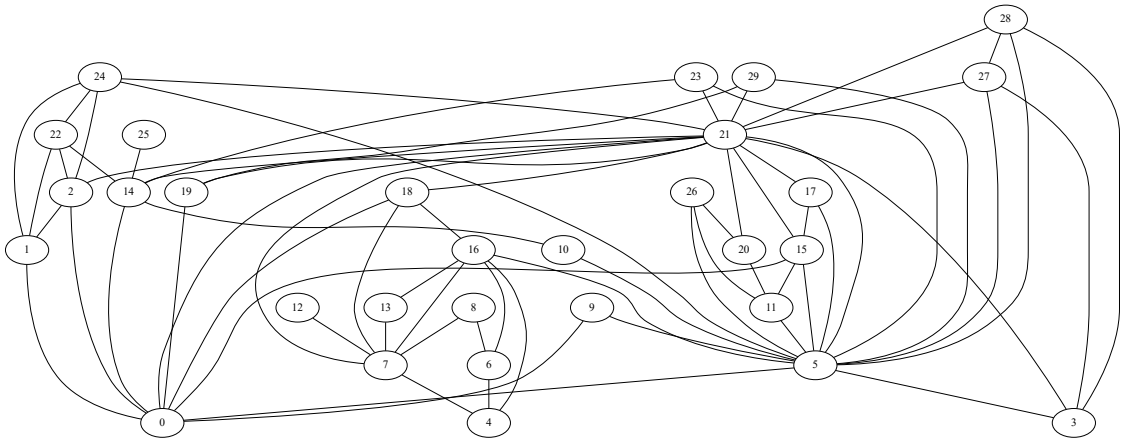


Figure 5.20: Example subgraph of DBLP colaboration with $n = 30$ and diameter= 5

Some of the results with graphs exatracted from DBLP dataset are summarized

in Table 5.21. The parameters of the experiment are the following:

- n: Number of nodes of the graph
- Diameter: Length of the longest shortest path between any two nodes in the graph
- k: Number of players
- $C_{\min}$: Minimum cost of a NE
- $C_{\max}$: Maximum cost of a NE
- PoA: Price of the Anarchy

n	Diameter	k	Min C	Max C	PoA
25	5	2	2	3	1.5
25	5	3	2	3	1.5
25	5	4	2	3	1.5
25	5	5	1	3	3.0
25	5	6	1	3	3.0
30	5	2	2	3	1.5
30	5	3	2	3	1.5
30	5	4	2	3	1.5
30	5	5	2	3	1.5
40	6	2	3	3	1.0
40	6	3	2	3	1.5
40	6	4	2	3	1.5
40	6	5	2	3	1.5
40	6	6	1	3	3.0

Figure 5.21: Results on DBLP

In the subgraphs of DBLP we observe that, as well in other studied topologies, as the number of players increase the PoA increases too. Graphs with higher diamter have higher PoA too, because the cost of the worst case equilibria is bounded by the diameter of the graph.

Chapter 6

Conclusions

The initial goals of the project were to develop a game theoretical model for the optimization problem of the K-CENTER, where players would selfishly choose the subset of seeds. We wanted to study the existence of NE in the game and the computational complexity of different problems associated with it, like deciding if an strategy is a NE, computing the best response for a player or computing an optimum strategy. Our aim was to study the quality of the NE of the game. We wanted to compare the quality of the NE of the game with the results of known approximation algorithms for the K-CENTER PROBLEM.

We have defined the QUICK-BURNING GAME, a game theoretical model of the K-CENTER problem. In our model each player chooses where to place a seed in a graph expecting that all the nodes of the graph become active as fast as possible. We model the diffusion process similarly to the Linear Threshold Model described in [13], where all nodes have threshold $\theta_v = 1$ and where for every two nodes u, v that are connected in the graph the weight is $b_{u,v} = 1$, so it its only needed one activate neighbour to activate any node. The cost of a player in our model is the same for all players and it is defined as what we called the Burning Time of a graph $G = (V,E)$ and a set of nodes $S \subseteq V$, the number of time-steps all nodes are infected. This is equivalent to the maximum distance from any node to its closest node in S. The Social Cost of the game is the same that he cost for any player.

Our main results for general graphs are the following: All instances of the QUICK-BURNING GAME have a NE because the OPT is always a NE. As the OPT is always a NE then the Price of Stability is 1. In a QUICK-BURNING GAME there may exist NE with a social cost that are not optimal strategies.

We studied different computational problems associated with the game. The

computation of the best response for a player can be computed in polynomial time. Deciding if a given strategy is a NE can be done in polynomial time . Unless P = NP computing a OPT strategy cannot be done in polynomial time, because it is NP-hard.

We experimented with a Best Response dynamics process where, starting from a random strategy, players are given the opportunity in rounds to change his action. It seems that such process always converge in an equilibrium strategy in a reasonable amount of rounds. It would be interesting to study the quality of the resulting equilibrium strategies.

We focus the study of the behaviour of our game model in different graph topologies. For each different topology we were interested in see how are the best and the worst equilibrium strategies and how is the cost of those strategies. In the case of paths graphs we show that:

- The OPT strategies are the ones that distribute the seeds along the path, where the maximum distance between two consecutive seeds is similar the double of the distance of the first seed and the start of the path and the last seed and the end of the graph.
- All OPT strategies are also NE, so the PoS is path graphs is 1.
- The worst case NE are strategies where all seeds are accumulated in the middle of the graph, no permitting that any unilateral change of the action of a player improves the cost, because there are two paths that are not covered with any seed and a unilateral change may only improve one. The PoA in path graphs is PoA = $O(k)$, where k is the number of players.

In the case of cycle graph we show that:

- In games with 2 players we show that all NE strategies are OPTs, then both the PoA and the PoS are 1.
- In games with more than 2 players the OPT strategies are the ones where seeds are distributing along the cycle, with distances between consecutive seeds as close as possible. In this case the cost is $\frac{n-k}{2k}$. The PoS in games with more than 2 players is 1.
- The worst case NE are strategies where seeds are grouped in two opposite sites of the graph, so the nodes between the two groups of seeds are far from their closest seeds. In those strategies there are to paths between the groups of seeds so any unilateral change would not be able to cover both paths, then there is no way that a single player improves that strategies. The cost of this strategies is $\frac{n-k-1}{4}$. The PoA in games with cycle graphs

and more than 2 players is PoA = $O(k)$.

With other topologies we do not have proved theoretical results, but we carried out experiments that give us some ideas.

With star-path graphs we observed that the equilibrium strategies have a very similar cost when there are no seeds enough to place at least one in each of the legs that the worst and best cost are the same, but if there are enough seeds to place at least one in each leg then there are equilibrium strategies with a variety of costs, some of them far from the optimum. A worst case equilibria is a strategy where all seeds are close to the root and there are at least two or more legs without seeds distributed along them (there may be any seed or all of them can be very close to the root, so the cost is close to the length of the legs). The best NE strategies in this case are strategies where each leg have seeds distributed in it, as much divided as possible. The PoA increases with the number of players and the length of the legs.

With n-ary (including binary) complete trees we observed that, similarly as happens in star-path graph, if we do not have enough seeds to fill the intermediate levels of the tree then the worst and best cost are the same, but if we can fill whole levels then the cost of the equilibrium strategies varies. Best NE in trees places are strategies that fill as much as possible levels in the tree such that the seeds are dividing the paths between the root and the leaves, as much equidistant from both as possible. The worst NE are strategies where seeds are in the top levels of the tree, close to the root, making not able to cover all paths to the leafs in one move. In complete n-ary trees the PoA increase with the number of players and the depth of the tree.

In simple series parallel graphs the best NE are strategies that distribute the seeds along all the paths between terminal nodes. The worst NE are strategies where the seeds are close to terminal nodes, so the cost is close to the half of the distance between terminal nodes. As in star-path graphs and in n-ary trees at least one seed should be placed in each of the diametral paths of the graph in order to have a equilibrium strategy with good cost.

Experimenting with real world data from the DBLP co-authoring database we observed that, as seen in the studied topologies, that the PoA increases with the number of players. Subgraphs of DBLP with larger diameter had also a higher PoA.

The theoretical and empirical results suggest that the PoA of a game is in relation with the number of players and with the diameter of the graph. Graph with large diameter must have seeds placed wisely, spread on the diametral paths of the graph in order to achieve a good cost. But if all the seeds are placed close to

the nodes with maximum eccentricity the cost will be close to the diameter of the graph, which may be times worse than the optimum.

Chapter 7

Future work

There remain some open problems related to the proposed strategic game and its variations.

We though about a different cost function that penalizes players for placing seeds not only in the same node as another player, but in a node that is close to another seed. That cost function would lead the player to spread the seeds along the graph, avoiding some of the worst case equilibria described in this project, that occurred when seeds where very close between them.

Another interesting modification of our game would be to assign a 'budget' of seeds to each player, so the actions for every player are a set of nodes of the graph, not only one.

It may be interesting to study the cost of the NE computed by Best Response dynamics process compared with the approximation algorithm in [12] and study how are the subsets of nodes computed by this and other known approximations, if they can be mapped to an equilibrium strategy and how good is it quality. We expect them to have good costs, because the approximation ratio of those algorithm is a constant factor of 2, usually smaller than the number of players.

Appendix A

Appendix: Code

A.1 Graph module

The graph is implemented as a dictionary of nodes, where the key is the label of a node (a string) and the value is a list of the labels of its neighbours.

```
class Graph:
        def __init__(self, V=[], E=[]):
                self.graph = dict()
                self.mat=None
                self.actives = set()
                for v in V:
                        self.add_node(v)
                for e in E:
                        fr, to = e
                        self.add_edge(fr,to)

        def get_nodes(self):
                return list(self.graph.keys())

        def add_node(self, node_name):
                if node_name not in self.graph:
                        n = Node(node_name)
                        self.graph[node_name] = n

        def add_edge(self, node_from, node_to):
                if isinstance(node_from, int):
```

```
            node_from = str(node_from)
        if isinstance(node_to, int):
            node_to = str(node_to)
        if  node_from in self.graph and \
            node_to in self.graph   and \
            node_from != node_to:
            self.graph[node_from].add_neighbor(node_to)
            self.graph[node_to].add_neighbor(node_from)
```

We implemented some predefined graph, such as paths, cycles or trees. The labels of the generated graphs are $'1',\ldots,'n'$ where n is the number of nodes in the graph.

```
def line_graph(n):
    V = [str(x) for x in range(n)]
    g = Graph(V)
    for x in range(n-1):
        g.add_edge(str(x), str(x+1))
    return g

def cycle_graph(n):
    g = line_graph(n)
    g.add_edge('0', str(n-1))
    return g

def binary_tree(depth):
    V = [str(x) for x in range(2 ** depth - 1)]
    E = []
    g = Graph(V,E)

    if depth>1:
        g.add_edge('0', '1')
        g.add_edge('0', '2')

    for level in range(1,depth):
        first_node = 2 ** level-1
        last_node = 2 ** (level+1) -2
        n = first_node
        while n <= last_node:
            g.add_edge(str(n), str(2*n + 1))
            g.add_edge(str(n), str(2*n + 2))
```

```
                        n += 1
        return g

def n_ary_tree(n, depth):
        V = ['0']
        E = []
        last_lvl = ['0']
        last_node = 0
        for lvl in range(1, depth):
                new_lvl = []
                for act in last_lvl:
                        for chld in range(n):
                                last_node += 1
                                new_lvl.append(str(last_node))
                                E.append((str(act), str(last_node)))
                                V.append(str(last_node))
                last_lvl = new_lvl
        return Graph(V,E)


def star_path_graph(n_legs, leg_length):
        n_nodes = 0
        V = ['0']
        E = []
        if isinstance(leg_length, int):
                leg_length = [leg_length for _ in range(n_legs)]
        for leg in leg_length:
                V.append(str(n_nodes+1))
                E.append(('0', str(n_nodes+1)))
                for v in range(n_nodes+2, n_nodes+1+leg):
                        V.append(str(v))
                        E.append((str(v-1), str(v)))
                n_nodes += leg
        g = Graph(V, [])
        for e in E:
                g.add_edge(*e)
        return g
```

A.2 QuickBurningGame module

We implemented some computational problems of the QUICK-BURNING GAME.

Cost Functions

The cost of the player is described in Section 4.2 as

$$c_i(s) = \begin{cases} \infty & \exists j : s_i = s_j \\ b(G,S) & \text{otherwise} \end{cases}$$

```
def cost_player(graph, strategy, i):
        strategy = list(strategy)
        si = strategy.pop(i)
        for sj in strategy:
                if graph.distance(si,sj) < 1:
                        return float('inf')
        strategy.insert(i,si)
        graph.clear_actived()
        for s in strategy:
                graph.activate_node(s)
        cost = graph.burn_graph()
        return cost
```

The Social Cost of a game is described in Section 4.2 as $C(s) = b(G,S)$

```
def social_cost(graph, seeds):
        if len(set(seeds)) < len(seeds):
                cost = float('inf')
        else:
                graph.activate_nodes(seeds)
                cost = graph.burn_graph()
                graph.clear_actived()
        return cost
```

Nash Equilibria

Checks if a strategy is a Nash Equilibrium. The parameters are a graph and the strategy. The number of players is the length of the strategy vector. For each

player it test all the alternative actions and returns `False` if by changing to any of them the cost improve, otherwise return `True`.

```
def is_nash(graph, strategy_profile):
        initial_cost = social_cost(graph, strategy_profile)
        for i in range(len(strategy_profile)):
                new_strategy = list(strategy_profile)
                current_move = new_strategy.pop(i)
                alternative_moves = set(graph.get_nodes()) - \
                        set(new_strategy)
                for a_ii in alternative_moves:
                        aux_strategy = list(new_strategy)
                        aux_strategy.insert(i, a_ii)
                        new_cost = social_cost(graph, aux_strategy)
                        if new_cost < initial_cost:
                                return False
        return True
```

The function below computes all NE in a game and returns a tuple containing the cost of the worst NE, a NE that have the cost of the worst NE, the cost of the best NE and a NE that have the cost of the best NE.

```
def analize_equilibrium(g,k):
        worst_strategy = best_strategy = None
        worst_cost = -1
        best_cost = float('inf')
        for strategy in combinations(g.get_nodes(), k):
                if is_nash(g, strategy):
                        cost = social_cost(g,strategy)
                        if cost > worst_cost:
                                worst_cost = cost
                                worst_strategy = strategy
                        if cost < best_cost:
                                best_cost = cost
                                best_strategy = strategy
        return worst_cost, worst_strategy, best_cost, best_strategy
```

Best Response

The best response for a player given a strategy are those actions that give less cost for that player if he changes his action to one of them. The function below

computes an action that is in the best response for that player given a strategy.

```
def BR(g,s,i):
        i = int(i)
        act_cost = cost_player(g,s,i)
        act_a = s[i]
        s_act = list(s)
        for n in g.get_nodes():
                s_act[i] = n
                new_cost = cost_player(g,s_act,i)
                if new_cost < act_cost:
                        act_cost = new_cost
                        act_a = n
        return act_a
```

Best Response Dynamics function

We implemented the Best Response dynamics process described in Section 4.5. The implementation can be found below. Given graph and a number of players, it generated a random strategy and, in a round-robin style each player is given the opportunity to change his action to one in his best response. The process end when in a full round any player changes his action.

```
def dynamics(g, k):
        s = random_strategy(g,k)
        initial_cost = social_cost(g, s)
        act_player = 0
        changes = 0
        changes_r= 0
        rounds = 0
        s0 = list(s)
        while 1:
                a_i = BR(g,s,act_player)
                sp = list(s)
                sp[act_player] = a_i
                if cost_player(g,sp,act_player)<cost_player(g,s,act_player):
                        s = list(sp)
                        changes_r += 1
                        changes += 1

                act_player += 1
                if act_player >= k:
                        rounds += 1
                        act_player = 0
                        if changes_r == 0:
                                break
                        changes_r = 0
        final_cost = social_cost(g, s)
        return (
                final_cost,
                initial_cost,
                changes,
                rounds,
                s
        )
```

Bibliography

[1] Amazon web services monthly cost calculator. https://calculator.s3.amazonaws.com/index.html.

[2] Arxiv. http://arxiv.org/.

[3] DBLP dataset. http://dblp.uni-trier.de/xml/. Accessed: 01-02-2016.

[4] Macbook pro and environment. http://www.apple.com/macbook-pro/environment/,.

[5] Payscale. http://www.payscale.com/.

[6] Carme Àlvarez, Maria J Blesa, and Hendrik Molter. Firefighting as a game. In *Algorithms and Models for the Web Graph*, pages 108–119. Springer, 2014.

[7] Carme Àlvarez, Maria J Blesa, and Hendrik Molter. Firefighting as a strategic game. *Internet Mathematics*, 12(1-2):101–120, 2016.

[8] Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. Burning a graph as a model of social contagion. In Anthony Bonato, Fan Chung Graham, and Pawe Praat, editors, *Algorithms and Models for the Web Graph*, volume 8882 of *Lecture Notes in Computer Science*, pages 13–22. Springer International Publishing, 2014.

[9] Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. Burning a graph is hard. *arXiv preprint arXiv:1511.06774*, 2015.

[10] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[12] Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

[13] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[14] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *STACS 99*, pages 404–413. Springer, 1999.

[15] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[16] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1068. ACM, 2010.

[17] Martin Nehéz, Marek Lelovskỳ, Jakub Bendžala, and Tomáš Blaho. On the k-center problem in social networks.

[18] Vasileios Tzoumas, Christos Amanatidis, and Evangelos Markakis. A game-theoretic analysis of a competitive diffusion process over social networks. In *Internet and Network Economics*, pages 1–14. Springer, 2012.