Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Màster en Computació

Tesi de Màster

The Complexity of Pure Nash Equilibria in Max-Congestion Games

Estudiant: Guillem Francès Medina
Director: Carme Àlvarez Faura
Ponent: -

Data: 8 de Setembre de 2008

Abstract

We study *Network Max-Congestion Games* (NMC games, for short), a class of network games where each player tries to minimize the *most congested* edge along the path he uses as strategy. We focus our study on the complexity of computing a pure Nash equilibria in this kind of games. We show that, for single-commodity games with non-decreasing delay functions, this problem is in `P` when either all the paths from the source to the target node are disjoint or all the delay functions are equal. For the general case, we prove that the computation of a PNE belongs to the complexity class `PLS` through a new technique based on generalized ordinal potential functions and a slightly modified definition of the usual local search neighborhood. We further apply this technique to a different class of games (which we call *Pareto-efficient*) with restricted cost functions. Finally, we also prove some `PLS`-hardness results, showing that computing a PNE for Pareto-efficient NMC games is indeed a `PLS`-complete problem.

Contents

Chapter 1

Introduction

1.1 Game Theory

Broadly speaking, game theory is a branch of mathematics focused on the modelling and study of situations where a number of entities or individuals (usually called *players*) interact. Every player is supposed to have some preferences with respect to his context (i.e. to the context modeled, be it a market where a number of companies try to maximize their benefits or an ecologic system where different species strive for survival) as well as a certain degree of rationality (i.e. he is expected to act in accordance to his preferences). Therefore, we can see as a game any situation where a set of players modify their environment through their actions in order to adapt it to their preferences. In non-cooperative games, which is the kind of games on which we will focus our study, players are not required to cooperate, that is they are selfish and respond only to their own interests. Often, it is also assumed that they can not even communicate.

Given a number of players and assuming that the actions they may perform and their preferences are both known, the main objective of game theory is to study the behaviour of the players, usually with the purpose to predict the outcome of the game, that is, the state of the environment after the interaction of the players, if such a definitive state does exist. To be more precise:

every player knows the actions that the other players are adopting, so he can adapt his own actions to them. Thus, it is possible that the situation enters a state of equilibrium where everyone is satisfied with his own acting, but it could also happen that such a situation is never reached because the game enters a dynamics where players keep acting and reacting endlessly. In this state, it would be very interesting to be able to determine whether a given game can enter one of this equilibrium situations, for instance, or this is not possible. Naturally, it would be even more interesting to actually find the equilibrium. This kind of issues are the ones that game theory deals with and the ones that motivate this thesis.

We said that game theory is a branch of mathematics. To be more precise, we should say that it was born as such. It is generally accepted that the study of game theory started in the forties with the publication of the book *Theory of Games and Economic Behaviour* by John von Neumann and Oskar Morgenstern. Initially, game theory developed as a tool of economic analysis, but in the following years the applications of the theory to other areas such as evolutive biology[1] or philosophy of language[2] began to be explored. Nowadays, game theory is applied also to a number of other subjects related with human behaviour such as psychology, sociology, anthropology or neurology. With respect to computer science, one simple example should suffice to show one of the applications of game theory to it: the Internet. Game theory is a perfect tool to model a large number of situations related to the use and functioning of the Internet, given that this very large network can be seen as a field where the (usually selfish) interests of a vast number of entities converge: users of the network, ISP, large enterprises, etc. Thus, game theory seems an adequate tool to study different aspects related to the

[1]See, for instance, the paper [Lewontin, 1961]. Quite interestingly, it is usually acknowledged that the first (and probably unnoticed) application of game theory to the field of evolutive biology dates back to the 19th century, when Darwin gave sort of a game-theoretic justification of the fact that the natural proportion (i.e. the situation of equilibrium) between the number of male and female subjects of any species was 1:1.

[2]See, for instance, [Lewis, 2002], first published in 1969

behaviour of this entities on the Internet. Quoting professor Papadimitriou [Papadimitriou, 2001],

> *The Internet has arguably surpassed the von Neumann computer as the most complex computational artifact (if you can call it that) of our time. Of all the formidable characteristics of the Internet [...], I believe that the most novel and defining one is its socio-economic complexity: The Internet is unique among all computer systems in that it is built, operated, and used by a multitude of diverse economic interests, in varying relationships of collaboration and competition with each other. This suggests that the mathematical tools and insights most appropriate for understanding the Internet may come from a fusion of algorithmic ideas with concepts and techniques from Mathematical Economics and Game Theory.*

This thesis will be focused on an specific area of game theory, which we shall try to describe informally now. As we mentioned before, we will study non-cooperative games, that is, games where players do not have means to cooperate among them – although a player seeking to satisfy his own interest might indirectly benefit other players. There are plenty of different types of non-cooperative games, but we will only consider a kind of games where players compete for the use of a number of given resources, and the costs they have to assume depends on how congested are the resources they use. This genre of games are a first step towards the modelling of the Internet as a game, since communication links can be understood as the resources for which the users of the Internet compete, and it is rather natural to think that their interest may be to use the least congested links available.

1.2 Aim of this Thesis

Generally speaking, this thesis is aimed at doing a theoretical research on game theory, a field which is related to a number of other disciplines, being sort of a point where they all coincide (which is, in my opinion, one of the facts that makes it such an appealing field). Therefore, it is one of the objectives of this thesis to get introduced to the above-mentioned field, i.e. to understand the significance and the technical details of the different models of games and main theorems about them. After this first stage of the research has been concluded, the second objective of the thesis will consist of actively trying to prove new results about the new model of maximum congestion games.

1.3 Outline of the document

The remainder of this document is organized as follows. Chapter 2 contains a general introduction to those aspects of game theory which are somehow related to this thesis, including a description of classical models of games such as congestion games and a description of the model of maximum congestion games upon which we will base our work. Chapters 3 and 4 contain the results of our research. The first of them contains the results about existence of pure Nash equilibria, whereas the second is focused on the complexity of computing these equilibria. Chapter 5 summarizes our results and includes a discussion about their originality and relevance. Finally, Appendix A contains some additional information about the experiments made in Section 4.1.

Chapter 2

Background

In this chapter we will introduce a number of concepts related to game theory which we believe indispensable to the successful understanding of the rest of the document. We will directly provide the formal definition of the different concepts and game models involved in the thesis.

2.1 Game Theory

2.1.1 Strategic Games

The most general kind of game we will be dealing with here is the so-called *Strategic Game*, which can be defined with the following elements: a set of players, a set of actions for every player and some information about the preferences of the players.

Definition 2.1.1. *A* Strategic game *(or* game*, to abbreviate) can be formally defined as a tuple* $\Gamma = (N, (P_i)_{i\in N}, (c_i)_{i\in N})$ *where*

- N is a set of $n \geq 2$ *players.*
- For each player $i \in N$, P_i is a set of *actions* or *strategies* available to him. In any given game, every player has to choose as strategy one of the actions $p \in P_i$. Every element $\pi \in P_1 \times \cdots \times P_n$, known

as a *pure strategy profile* of the game, precisely describes a possible outcome of the game, that is the strategy followed by all players in the game. Given a profile $\pi = (p_1, \ldots, p_n)$, it is understood that p_i is the strategy followed by the player $i \in N$. In addition, Π denotes the set $P_1 \times \cdots \times P_n$ of all the possible profiles for the game.

- $u_i : \Pi \to \mathbb{R}$ is the *payoff function* (or *utility function*) of each one of the players $i \in N$. In any game Γ, $u_i(\pi)$ is the payoff received by player $i \in N$ when the strategies of all the players are as described in the strategy profile π. Thus, this payoff does not depend solely on the strategy adopted by the player, but also on the strategies of the other players. This payoff function can also be seen as an indication of the preferences of the player.

Pure and Mixed Profiles Sometimes it is useful to consider that players, instead of deterministically choosing their strategy among all their available actions, are able to randomize their choice. In that case, every player $i \in N$ can adopt as strategy a probability distribution over his action set P_i. Let R_i denote the (infinite) set of all possible probability distributions over the set of actions of player i

$$R_i = \left\{ r : P_i \to [0,1] \;\middle|\; \sum_{p \in P_i} r(p) = 1 \right\}$$

Then, every element $\rho \in R_1 \times \cdots \times R_n$ is known as a *mixed strategy profile* and, for any given mixed profile $\rho \in R_1 \times \cdots \times R_n$, $r_i(p)$ is the probability that player i chooses the action $p \in P_i$[1]. When using mixed profiles, we usually consider *expected payoffs*. The expected payoff of any player $i \in N$ is then formally defined as

$$u_i(\rho) = \sum_{(p_1,\ldots,p_n)\in\Pi} \left(\left(\prod_{j=1}^{n} r_j(p_j) \right) \cdot u_i((p_1, \ldots, p_n)) \right)$$

[1]Notice, however, that we will mainly study pure strategy profiles. Thus, unless explicitly stated, we will always be referring to pure profiles.

Payoffs and Costs Although the functions u_i have been traditionally interpreted as utility functions, as we described above, it is also quite common - especially in the research area of this document - to use the alternative interpretation which considers them as cost functions - i.e. the players have to assume a cost for the choices they make. In this case, a game is usually denoted by a tuple $\Gamma = (N, (P_i)_{i \in N}, (c_i)_{i \in N})$ where each function $c_i : \Pi \to \mathbb{R}$ is the cost function of player $i \in N$ and $c_i(\pi)$ is the cost assumed by player i under the pure strategy profile π. Naturally, players try to *minimize* the cost of the strategy that they adopt. When mixed strategy profiles are allowed, then the concept of *expected cost* is defined analogously to that of expected payoff: the expected cost for player $i \in N$ under the mixed profile $\rho = (r_1, \ldots, r_n)$ is

$$c_i(\rho) = \sum_{(p_1,\ldots,p_n) \in \Pi} \left(\left(\prod_{j=1}^{n} r_j(p_j) \right) \cdot c_i((p_1, \ldots, p_n)) \right)$$

This cost terminology is the one that we will use for the rest of this document.

A Simple Example One of the classical examples in game theory is the *Prisoner's Dilemma*. It is a game involving two players, two suspects of having committed a crime. Each of them, being interrogated in isolation by the police, is offered the same deal. He can either testify against the other suspect or remain silent. The set of allowed actions is therefore equal for both players, $P_1 = P_2 = \{\text{Denounce (D)}, \text{Silence (S)}\}$. The deal goes like this: if one suspect denounces the other but the other remains silent, then the first one is freed and the second one receives a ten-year sentence (i.e. $c_1((D,S)) = 0$ and $c_2((D,S)) = 10$ and, conversely, $c_2((S,D)) = 0$ and $c_1((S,D)) = 10$). If both suspects denounce the other, both are convicted for five years (i.e. $c_1((D,D)) = c_2((D,D)) = 5$). Finally, if they both remain silent, then both are convicted for only one year (i.e. $c_1((S,S)) = c_2((S,S)) = 1$). Assuming that both suspects act rationally in defense of their own interests, their choice should be clear. Both of them should reason that no matter what the other suspect does, the best option is always to denounce him. Indeed, if any of them thinks that the other one will denounce him, he would better denounce too in order to get a five-year sentence instead of a ten-year one. On the other hand, if he thinks that the other will remain silent, he would better denounce him too, since this would allow him to go with no charges. Therefore, the strategy profile (D, D) seems to be the natural outcome of the game. However, it can be easily seen that from a global point of view both suspects would be better off if they remained silent. This illustrates the conflicts that usually arise between the interests of individuals and those of the society.

2.1.2 Nash Equilibria

This notion of "natural outcome" that we have just seen in the previous section is of great interest. One of the applications of game theory consists of studying and predicting the future behaviour of any set of entities or individuals competing in a common market. In the case of the Prisoner's dilemma, we have already seen that the strategy profile (D, D) is the one that should arise as the predictable outcome of the interaction of the two suspects. This profile can be thought of as a state of *equilibrium*, a stable state in which no player of the game is willing to move, to change his strategy, since any movement would damage his own interests. Notice that in the Prisoner's dilemma this profile (D, D) is the only one of the four possible pure profiles (D, D), (D, S), (S, D), (S, S) which is an equilibrium.

This conceptualization of equilibrium has been known for years under the name of *Nash equilibrium* in honour of the American mathematician John F. Nash, and is arguably one of the most popular, at least in the area of non-cooperative games. Before giving a formal definition, let us provide some extra notation which is quite common in game theory. Given a tuple $\pi = (p_1, \dots, p_n)$, it is usual to denote with (π_{-i}, p) (where $1 \leq i \leq n$) the tuple that we obtain if we substitute the i-th element of π for the element p. Therefore $(\pi_{-i}, p) = (p_1, \dots, p_{i-1}, p, p_{i+1}, \dots, p_n)$.

Definition 2.1.2 (Pure Nash Equilibrium). *Let* $\Gamma = (N, (P_i)_{i \in N}, (c_i)_{i \in N})$

be a strategic game. The pure strategy profile $\pi = (p_1, \ldots, p_n) \in \Pi$ *is said to be a* Pure Nash Equilibrium *(PNE, for short) if no player is interested in changing his strategy, that is, if no unilateral modification of his strategy would decrease the cost that he is assuming. Formally,*

$$\forall i \in N \ \forall p \in P_i \ c_i((\pi_{-i}, p)) \geq c_i(\pi)$$

If the profile π is a mixed strategy profile, we then have a *Mixed Nash Equilibrium.* In this document, however, we will mainly study Pure equilibria. Therefore, any mention to a Nash Equilibria will refer to a Pure Nash Equilibria unless explicitly stated.

Observe that we have no guarantee that any of the pure strategy profiles of a given game is a Nash equilibrium, nor we have any guarantee that there is no more than one Nash equilibrium. This fact goes against the usefulness of the concept of Nash Equilibria as a tool for predicting the behaviour of the players of a given game, and is one of the most widely criticized points of this concept: how can we predict the final equilibria situation of a market if there are many pure Nash equilibria, how can we tell which of them is going to be the one where the game ends up? However this criticism, to the present date no one has been able to come up with a better notion of equilibrium.

2.1.3 Congestion Games

In his seminal paper [Rosenthal, 1973], Rosenthal describes a new class of strategic games that have since then proved useful to model a wide array of situations. All games of this class, called *Congestion Games*, share a fundamental property: they all possess at least one Nash Equilibrium. The proof of this property can be seen in Section 2.3, here we will formally describe the class of games. Any congestion game is a strategic game with the special feature that the n players of the game compete for the resources of a given set of resources. Specifically, every player has to choose a subset of this set of resources among a particular number of subsets. The cost that he will have to pay is given by the *congestion* of the resources he chooses, that is, by the number of other players that choose the same resources as him.

Definition 2.1.3. *An* (unweighted) Congestion Game *is defined as a tuple* $\Gamma = (N, E, (P_i)_{i \in N}, (d_e)_{e \in E})$ *where*

- $N = \{1, \dots, n\}$ is the *set of players*.
- E is the *set of resources*, which is finite.
- $P_i \subseteq \mathcal{P}(E)$ is the *set of allowed actions* for every of the layers $i \in N$.

  As it happened with general strategic games, each player $i \in N$ chooses as strategy one of the actions in P_i, and every element $\pi \in P_1 \times \cdots \times P_n$ is a pure strategy profile of Γ that describes the strategy followed by all players. Now, for any resource $e \in E$, $\Lambda_e(\pi)$ denotes the *set of users* of e under the profile $\pi = (p_1, \dots, p_n)$, that is, the set of players for which this resource is part of their strategy: $\Lambda_e(\pi) = \{i \in N \mid e \in p_i\}$. We finally say that the *congestion* $l_e(\pi)$ of the resource e under π is the number of users of e under π, $l_e(\pi) = |\Lambda_e(\pi)|$.

- $d_e : \mathbb{N} \to \mathbb{R}$ is the *delay function* of each resource $e \in E$, which we assume to be polynomial-time computable. This function models the delay $d_e(k)$ provoked by the resource e under a congestion $k \in \{1, \dots, n\}$ (in other words, the cost of using this particular resource under a given congestion). The cost assumed by each player is then defined as the sum of costs induced by each of the resources he uses. That is, the cost function $c_i : \Pi \to \mathbb{R}$ of every player $i \in N$ under the strategy profile $\pi = (p_1, \dots, p_n)$ is

$$c_i(\pi) = \sum_{e \in p_i} d_e(l_e(\pi))$$

Weighted Congestion Games Sometimes it is useful to model situations where the different players have a different degree of influence on the congestion of the resources they use. In other words, the players can be seen as having an associated *weight* or demand that determines the extent to which they congest the resources. This requires a small modification of the definition that we have already given, but has important consequences, as we will see in section 2.3. The games of this new kind are commonly called *weighted congestion games*, so we will call the games of the previous kind *unweighted* congestion games (or, simply, congestion games, given that the original definition does not consider weights). In his paper [Milchtaich, 1996], Milchtaich defines this kind of games as follows.

Definition 2.1.4. *A* weighted congestion game *is defined as a tuple* $\Gamma = (N, E, (P_i)_{i \in N}, (d_e)_{e \in E}, (w_i)_{i \in N})$. *The set of players N, the set of resources E, the sets P_i of available actions, the delay functions d_e as well as the concepts of strategy profile and set of users of a resource are defined in the same manner as in unweighted congestion games (see definition 2.1.3). In contrast, now every player $i \in N$ has a positive integer weight $w_i \in \mathbb{N}_+$ that influences the congestion of the resources used by player i as follows. The* congestion *of resource $e \in E$ is now the sum of the weights of its users*

$$l_e(\pi) = \sum_{i \in \Lambda_e(\pi)} w_i$$

Finally, the cost function $c_i : \Pi \longrightarrow \mathbb{R}$ associated to every player i has the same definition as before:

$$c_i((p_1, \ldots, p_n)) = \sum_{e \in p_i} d_e(l_e((p_1, \ldots, p_n)))$$

It is immediate to see that the class of weighted games is a generalisation of the class of unweighted games, given that every unweighted game can be seen as a weighted game where all players have weight one. Furthermore, both weighted and unweighted games are clearly a specific kind of strategic games.

2.1.4 Network Congestion Games

In some cases it can be useful to consider more succinct computational representations of a congestion game, given that the naive representation would require, in addition of the listing of all resources, that all available subsets of resources to each players be also listed, and that could potentially require an amount of space exponentially larger with respect to the number of resources. One of the most used implicit representations of the subsets of resources available to each player is given by means of the structure of a graph in the so-called *network congestion games*. In a game of this kind, the set of resources is considered to be the set of edges of a given graph, whereas the set of available actions of any player consists of the set of paths in the graph between two particular nodes.

Definition 2.1.5. *A* network congestion game *is defined as a tuple* $\Gamma = (N, G, ((s_i, t_i))_{i \in N}, (d_e)_{e \in E(G)})$*, where*

- $N = \{1, \ldots, n\}$ is the *set of players.*
- $G = (V, E)$ is a directed graph.
- $(s_i, t_i) \in V \times V$ is the pair of *origin and destination nodes* (or *source and target nodes*) of every player $i \in N$.
- $d_e : \mathbb{N} \to \mathbb{R}$ is the *delay function* of every edge $e \in E$, which we assume to be polynomial-time computable.

  As we said before, the set P_i of actions available to each player i is now defined implicitly by this pair of nodes as the set of all $(s_i - t_i)$ paths[2]. This set is finite, since we do not consider paths containing repeated edges. We will sometimes use P to denote the union of the sets of actions of all players, $P = \bigcup_{i \in N} P_i$. The definition of the rest of elements of the game is the same that we gave in the previous section for general congestion games.

Weighted NCGs The extension of network congestion games to allow players to have different weights is completely identical to that of general

[2]We will use the shorthand $(s - t)$ *path* to refer to a path between the nodes s and t.

congestion games. We therefore refer the reader to section 2.1.3 for details, and we will just give here a formal definition.

Definition 2.1.6. *A* weighted network congestion game *is defined as a tuple* $\Gamma = (N, G, ((s_i, t_i))_{i \in N}, (d_e)_{e \in E(G)}, (w_i)_{i \in N})$. *The set of players* N*, the directed graph* G*, the pair of origin and destination nodes* (s_i, t_i) *of each player, the set* P_i *of actions available to him, the delay function* d_e *of each edge in* G *and the cost function* $c_i : \Pi \to \mathbb{R}$ *associated to every player* i *are defined as in the model of unweighted congestion games (see definition 2.1.5). On the other hand, now every player has a positive integer weight* $w_i \in \mathbb{N}_+$*, and the* congestion *of any edge* $e \in E$ *is defined as*

$$l_e(\pi) = \sum_{i \in \Lambda_e(\pi)} w_i$$

Again, all unweighted NCG can be seen as a weighted NCG where all players have weight one, and they can be easily seen to be subclasses of unweighted CG and weighted CG, respectively.

Single-commodity Games If the all the origin nodes of all players in a NCG Γ are the same node, and so are all the destination nodes, then Γ is said to be a *single-commodity* game. If that does not happen, Γ is said to be a *multi-commodity* game. Notice that for single-commodity games, all players share the same set of actions $P_i = P$. For the sake of simplicity, we will refer to single-commodity games as tuples

$$\Gamma = (N, G, (s, t), (d_e)_{e \in E(G)})$$

Analogously, we will refer to *weighted single-commodity network congestion games* as tuples

$$\Gamma = (N, G, (s, t), (d_e)_{e \in E(G)}, (w_i)_{i \in N})$$

.

2.1.5 Network Maximum Congestion Games

We will now describe the game model which we plan to carry on our research on. It is a class of strategic game which highly resembles the class of network congestion games that we have just described, since all players have to choose a path in a graph and the cost assumed by them is still based on the congestion of the edges of the path they choose. In the games of this new model, which we call *network maximum congestion games* (or network max-congestion games, NMC games, for short), the exact cost is not given by the sum of the congestion of the edges, but instead by the maximum congestion among them.

Before we go into a formal definition, let us give a brief justification for the study of this model. Consider the next situation: we have a number of users of a communication network, each of them sending packets between different network endpoints. It may happen that the users want to route their information through a path with maximum bandwidth, that is, a path such that the congestion of the most congested link is minimized. In this case, the traditional model of congestion games is not useful anymore to study the needs and behaviour of the users. However, we believe that the model that we present here is a first and interesting step towards being able to successfully analyse situations like this one[3].

Let us now give a more formal definition of this new class of games.

Definition 2.1.7. *A* network max-congestion game *is defined by a tuple* $\Gamma = (N, G, ((s_i, t_i))_{i \in N}, (d_e)_{e \in E(G)})$ *where*

- $N = \{1, \dots, n\}$ is the *set of players.*
- $G = (V, E)$ is a directed graph.
- $(s_i, t_i) \in V \times V$ is the pair of *origin and destination nodes* (or *source and target nodes*) of every player $i \in N$.

[3]Other examples from the telecommunications field which justify the study of our model can be found in [Banner and Orda, 2006].

- $d_e : \mathbb{N} \to \mathbb{R}$ is the *delay function* of every edge $e \in E$, which we assume to be polynomial-time computable.

The definition of the different elements of the game mirrors that of the network congestion games model (see definition 2.1.5). The set P_i of actions available for player i is the set of $(s_i - t_i)$ paths in the graph G. Every pure strategy profile, thus, is an element π in the set $P_1 \times \cdots \times P_n$. For every edge $e \in E$, $\Lambda_e(\pi) = \{i \in N \mid e \in p_i\}$ is the *set of users* of e in the profile π.

The congestion $l_e(\pi)$ of the edge e in the strategy profile π is again defined as the number of players of the game that use e in the strategy that they play in π, $l_e(\pi) = |\Lambda_e(\pi)|$. Finally, and this is where the novelty of this model lies, the cost assumed by each player $i \in N$ in the profile $\pi = (p_1, \ldots, p_n)$ is now defined as the maximum delay of the edges of its path:

$$c_i(\pi) = \max_{e \in p_i} d_e(l_e(\pi))$$

Weighted Network Max-Congestion Games As it happened with congestion games, we will also consider NMC games where players are allowed to have different weights.

Definition 2.1.8. *A weighted network max-congestion game is defined as a tuple $\Gamma = (N, G, ((s_i, t_i))_{i \in N}, (w_i)_{i \in N}, (d_e)_{e \in E(G)})$. The set N of players, the directed graph G, the origin and destination nodes (s_i, t_i) and the delay functions d_e are defined as we did with unweighted games (see definition 2.1.7), and so are the set P_i of actions available to every player, the set Λ_e of users of every edge e and the cost function $c_i : \Pi \to \mathbb{R}$ associated to every player i. However, every player i now has a positive integer weight $w_i \in \mathbb{N}_+$, and the* congestion *of any edge $e \in E$ is defined as*

$$l_e(\pi) = \sum_{i \in \Lambda_e(\pi)} w_i$$

Single-commodity Games The definition of single-commodity NMC games is also analogous to the one that we gave for congestion games (see the previous section). A (possibly weighted) NMC game is said to be single-commodity if all players share the same endpoints (s_i, t_i); if that is not the case, it is said to be multi-commodity. Therefore, we will denote unweighted and weighted single-commodity games with tuples $\Gamma = (N, G, (s,t), (d_e)_{e \in E(G)})$ and $\Gamma = (N, G, (s,t), (w_i)_{i \in N}, (d_e)_{e \in E(G)})$, respectively.

2.1.6 Potential Functions and Games

Potential functions were systematically studied for the first time in [Monderer and Shapley, 1996] as a mechanism to establish certain properties of strategic games, most notably the existence of pure Nash equilibria, as we will see in Section 2.3. Let us here give some definitions related to this technique. The main idea is to find a *global* measure of potential that decreases only along with any decrease of the cost assumed by any player, but there are different kinds of potential function that follow this idea in different grades.

Definition 2.1.9. *The function $\phi : \Pi \to \mathbb{R}$ is said to be an* exact potential *for a strategic game $\Gamma = (N, (P_i)_{i \in N}, (c_i)_{i \in N})$ if for every $\pi \in \Pi$, $i \in N$, $p \in P_i$, it holds that $c_i(\pi_{-i}, p) - c_i(\pi) = \phi(\pi_{-i}, p) - \phi(\pi)$. Any strategic game possessing an exact potential function is called an* exact potential game.

Definition 2.1.10. *The function $\phi : \Pi \to \mathbb{R}$ is said to be an* ordinal potential *for a strategic game $\Gamma = (N, (P_i)_{i \in N}, (c_i)_{i \in N})$ if for every $\pi \in \Pi$, $i \in N$, $p \in P_i$, it holds that $c_i(\pi_{-i}, p) < c_i(\pi) \Leftrightarrow \phi(\pi_{-i}, p) < \phi(\pi)$. Any strategic game possessing an ordinal potential function is called an* ordinal potential game.

Definition 2.1.11. *The function $\phi : \Pi \to \mathbb{R}$ is said to be a* generalized ordinal potential *for a strategic game $\Gamma = (N, (P_i)_{i \in N}, (c_i)_{i \in N})$ if for every $\pi \in \Pi$, $i \in N$, $p \in P_i$, it holds that $c_i(\pi_{-i}, p) < c_i(\pi) \Rightarrow \phi(\pi_{-i}, p) < \phi(\pi)$. Any strategic game possessing an ordinal potential function is called a* generalized ordinal potential game.

Notice that every exact potential game is an ordinal potential game and every ordinal potential game is a generalized ordinal potential game.

2.1.7 Pareto-efficient games.

We will here briefly describe a simple class of games whose definition arises from what in economics is known as *Pareto improvement* (a concept named after V. Pareto, an Italian economist). A Pareto improvement is performed when a player changes his strategy to reduce the cost he is paying while, at the same time, the cost of no other player is increased by this change.

Definition 2.1.12. *We say that a strategic game is* Pareto-efficient *if the cost functions of the game are defined in such a way that all the improving movements that a player may do provoke no harm to the other players, that is, if for any profile π, player i and strategy $p \in P_i$, it holds that*

$$c_i(\pi_{-i}, p) < c_i(\pi) \Rightarrow \forall j \in N \;\; c_j(\pi_{-i}, p) \leq c_j(\pi)$$

2.2 Computational Complexity

We will here give some computational complexity definitions which are related to our thesis; specifically to the study of the complexity of finding Nash equilibria for certain classes of games, as we will point out in the next section. Before we give a formal definition, we shall discuss a little bit about functional extensions of the basic complexity classes `P` and `NP`.

2.2.1 Search Problems, `FP` and `FNP`

When dealing with computational problems, sometimes we do not only want to know if an object with some desired properties exists (e.g. if there exists any truth assignment that satisfies a given Boolean formula), but to obtain one of the objects with such properties, if any of them exists at all (e.g. we want to obtain one of the satisfying assignments). This kind of problems are usually called *search problems* (see [Johnson, 2007] for a good discussion about them). The class `FNP` of search problems (which stands for Functional `NP`; see, for instance, [Megiddo and Papadimitriou, 1991]) is the analogous version of the decision problems complexity class `NP`.

Definition 2.2.1. *A `FNP` search problem* Π *consists of*

i) A polynomial-time recognizable set of instances I_Π.

ii) A polynomial-time recognizable relation

$$R_\Pi = \{(x, y) \mid x \in I_\Pi \wedge |y| \leq p(|x|)\}$$

where p *is a polynomial. The goal of the problem is, given any instance* $x \in I_\Pi$, *to find a* y *such that* $(x, y) \in R_\Pi$.

We will also consider the class of problems `TFNP` (for Total Functional `NP`) for which the existence of the object that we are looking for is guaranteed.

Definition 2.2.2. *A search problem* Π *is in `TFNP` if it is in `FNP` and for each* $x \in I_\Pi$ *there is at least one* y *such that* $(x, y) \in R_\Pi$.

Finally, let us give the definition of the class `FP` (for Functional `P`) for the sake of completeness:

Definition 2.2.3. *A search problem* Π *is in `FP` if it is in `TFNP` and there exists a polynomial-time algorithm such that for any* $x \in I_\Pi$ *outputs a* y *such that* $(x, y) \in R_\Pi$.

We clearly have that `FP` $\subseteq$ `TFNP` $\subseteq$ `FNP`, but it is currently unknown whether any of the inclusions is strict or not.

Now, a *combinatorial optimization problem* is a search problem where we do not only want to find an object with a given set of properties, but we want to find the object with these properties that optimizes a given function. Formally, we say that a combinatorial optimization problem Π is defined as a tuple (I_Π, R_Π, c_Π), where (I_Π, R_Π) is a search problem and c_Π is an associated *cost function*. For ease (and slightly abusing of the notation) let us denote by $R_\Pi(x)$ the (possibly empty) set of solutions of the instance $x \in I_\Pi$, that is, the set

$$R_\Pi(x) = \{y \mid (x, y) \in R_\Pi\}$$

Then, we say that any solution $y \in R_\Pi(x)$ of a given instance $x \in I_\Pi$ has a cost $c_\Pi(x, y)$. As we said, given any instance x of a combinatorial problem Π, we want to find a solution $y \in R_\Pi(x)$ that globally minimizes (or maximizes, depending on what kind of optimization we are dealing with) the function c_Π, i.e., a solution such that $c_\Pi(x, y)$ is an optimum value. Consider, as an illustration, the MAX-SAT problem. The set of instances $I_{\text{MAX-SAT}}$ of the problem contains all possible accepted encodings of a weighted boolean formula in conjunctive normal form, that is, a formula where all the clauses have an associated weight. For any given weighted formula $x \in I_{\text{MAX-SAT}}$, the set of feasible solutions $R_{\text{MAX-SAT}}(x)$ contains all possible truth assignments for formula x. Finally, the cost $c_{\text{MAX-SAT}}(x, y)$ of a solution $y \in R_{\text{MAX-SAT}}(x)$ is the sum of the weights of the clauses of x that are satisfied by the truth assignment y.

Observe that any combinatorial optimization problem $\Pi = (I_\Pi, R_\Pi, c_\Pi)$ can be seen as a simple search problem just by adding the condition of local optimality to the properties of the object that we want to find. That is, the combinatorial optimization problem $\Pi = (I_\Pi, R_\Pi, c_\Pi)$ is equivalent to the search problem $\Pi' = (I_{\Pi'}, R_{\Pi'})$, where $I_{\Pi'} = I_\Pi$ and

$$R_{\Pi'} = \{(x, y) \mid x \in I_\Pi \wedge y \in R_\Pi(x) \wedge \forall\, z \in R_\Pi(x)\; c_\Pi(x, z) \geq c_\Pi(x, y)\}$$

(in the case of a minimization problem). This allows us to be able to classify optimization problems into the classes `FNP`, `TFNP` and `FP` that we defined above.

There is a large number of combinatorial optimization problems for which there is no known polynomial-time algorithm able to solve them. In some cases, the algorithmic technique of *local search* [Aarts and Lenstra, 2003] has been proved useful for finding good (but not always globally optimal) solutions for this problems. The idea is to conduct a search guided by a *neighborhood structure* that, in some sense, relates good solutions to similar solutions that may possibly be as good, or even better. Formally, a local search problem is defined as a tuple $\Pi = (I_\Pi, R_\Pi, c_\Pi, N_\Pi)$ which is the conjunction of a combinatorial optimization problem (I_Π, R_Π, c_Π) with a neighborhood structure N_Π which relates solutions of the problem to other solutions. Thus, any solution $y \in R_\Pi(x)$ has a (possibly empty) set of neighboring solutions $N_\Pi(x, y) \subseteq R_\Pi(x)$. Given a pair (Π, N_Π), the local search problem is to find a solution $y \in R_\Pi(x)$ which is a *local* optimum, i.e. such that all his neighbors do not have a strictly lower cost (for the case of minimization problems). To follow with our previous example, a common neighborhood structure used to tackle the MAX-SAT problem with a local search algorithm is known as the *flip* algorithm, for it relates any given truth assignation to all other assignations where only a variable has a different truth value.

Note that again any local search problem $\Pi = (I_\Pi, R_\Pi, c_\Pi, N_\Pi)$ can be seen as a simple search problem $\Pi' = (I_{\Pi'}, R_{\Pi'})$ where $I_{\Pi'} = I_\Pi$ and

$$R_{\Pi'} = \{(x, y) \mid x \in I_\Pi \wedge y \in R_\Pi(x) \wedge \forall\, z \in N_\Pi(x, y)\; c_\Pi(x, z) \geq c_\Pi(x, y)\}$$

2.2.2 The Class PLS

The class PLS (for *Polynomial-time Local Search*), first introduced in [Johnson et al., 1988; Schäffer and Yannakakis, 1991], captures the complexity of those local search problems where a move from one solution to a better solution in the neighborhood requires only polynomial time.

Definition 2.2.4. *A local search problem $\Pi = (I_\Pi, R_\Pi, c_\Pi, N_\Pi)$ belongs to the complexity class PLS if there exist the following three polynomial-time algorithms:*

- *A polynomial-time algorithm A_Π that, for any string x is able to determine if $x \in I_\Pi$ and, if that is the case, to produce a feasible* initial *solution $y \in R_\Pi(x)$.*
- *A polynomial-time algorithm C_Π that, for any instance $x \in I_\Pi$ and any string y decides if y is a feasible solution for x (i.e. if $(x, y) \in R_\Pi$) and, if that is the case, computes the cost of the solution $c_\Pi(x, y)$.*
- *A polynomial-time algorithm F_Π that, given a pair $(x, y) \in R_\Pi$ determines if the solution y is locally optimal with respect to the neigh-*

borhood N_Π and, if not, returns a solution $z \in N_\Pi(x, y)$ such that $c_\Pi(z, y) < c_\Pi(x, y)$ (for a minimization problem).

Any PLS problem can be seen to belong the class FNP. In addition, since, by definition, any instance of a PLS problem is guaranteed to have at least one feasible solution (the initial one), and assuming finiteness of the search space, any instance of the problem will have at least one local optimum. Thus, we have that $\texttt{PLS} \subseteq \texttt{TFNP}$. On the other hand, any search problem Π in FP can be formulated as a PLS problem (see [Johnson et al., 1988] for the details. Thus, we have that

$$\texttt{FP} \subseteq \texttt{PLS} \subseteq \texttt{TFNP}$$

2.2.3 PLS-reductions

It is useful to define a suitable notion of reduction among search problems in order to identify problems that are complete, that is, problems that are the most difficult among those of their class. We will directly give the formal definition of the kind of reductions which are usually used to relate PLS problems, which are known as PLS-reductions.

Definition 2.2.5. *A local search problem Π_1 is PLS-reducible to another local search problem Π_2 if there exist the following two polynomial-time computable functions:*

i) A function $f : I_{\Pi_1} \to I_{\Pi_2}$ that maps every instance of the first problem to an instance of the second problem.

ii) A function $g : \{(x, y) \mid x \in I_{\Pi_1} \wedge y \in R_{\Pi_2}(f(x))\} \to R_{\Pi_1}(x)$ such that for all instances $x \in \Pi_1$ we have that if y is a local optimum for the instance $f(x)$, then $g(x, y)$ is a local optimum for x.

Intuitively, the function f maps instances x of the problem Π_1 to instances $f(x)$ of the problem Π_2. Once we have found a solution $y \in R_{\Pi_2}(f(x))$ for the instance $f(x)$ of Π_2, the function g maps this solution "back" to a solution of the first instance x. The only requirement that we place on this process of mapping and remapping back is that if we find a local optimum for the instance of Π_2, then the instance of Π_1 that we get when we map the local optimum back must also be a local optimum.

With this definition of a PLS-reduction, we now can define the usual notion of completeness:

Definition 2.2.6. *A local search problem* Π *is PLS-complete if*

i) Π *belongs to PLS*

ii) Any other problem in PLS can be reduced to Π *through a PLS-reduction.*

The first problem which was identified as PLS-complete (through a construction analogous to the NP generic reduction to CIRCUIT-SAT, see [Johnson et al., 1988]) was the CIRCUIT-FLIP problem, defined as follows. The set of instances contains all possible acyclic boolean circuits with *and*, *or* and *not* gates, while the feasible solutions are all those possible binary inputs (i.e. binary strings) to the circuit. The cost of any solution x is then the output of the circuit (seen as a binary number) when the input x is applied to the circuit. For any given solution x, the set of its neighbors contains all those binary strings obtained by flipping a single bit of the string x.

Since then, other interesting problems such as the WEIGHTED MAX-CUT with a flip neighborhood [Schäffer and Yannakakis, 1991] structure or the TSP with the Lin-Kernighan neighborhood structure [Papadimitriou, 1992] have been shown to belong to the class PLS.

2.3 Relevant Results

In this section we will describe some relevant results of game theory which are related to this thesis.

2.3.1 Existence of Nash equilibria

We will first present those results which refer to the existence of Nash equilibria for the games of the classes that we described above. As we mentioned, the potential nonexistence of pure Nash equilibria has often been signaled as one of the weak points of the concept of Nash equilibrium itself, since it places a limit on the usefulness of the concept as a tool of prediction. The next theorem, proved by Nash some fifty years ago in his famous paper [Nash, 1950], shows that if we allow the players to play mixed strategy profiles, then the existence of at least one Nash equilibrium is guaranteed for any strategic game.

Theorem 2.3.1 (Nash, 1950)**.** *Every strategic game has at least one mixed Nash equilibrium profile.*

If we want to be able to make such as a strong statement with pure profiles, however, we need to consider restricted classes of games. This is the case of congestion games

Theorem 2.3.2 (Rosenthal, 1973)**.** *Every unweighted congestion game has at least one pure Nash equilibrium profile.*

R.W. Rosenthal was the first to prove this interesting property in his paper [Rosenthal, 1973]. His proof essentially formulated the problem of finding a PNE in a congestion game as a binary integer programming optimization problem whose solution was a PNE for the game. Given that all binary integer programming problem can be solved (even though not always efficiently), all congestion game must have a PNE. From another point of view, what Rosenthal did was to provide an exact potential function and (implicitly) base his proof on the following theorem by Monderer and Shapley:

Theorem 2.3.3 (Rosenthal, 1973)**.** *Every unweighted congestion game has an exact potential function.*

Theorem 2.3.4 (Monderer and Shapley, 1996)**.** *Any strategic game possessing a generalized ordinal potential function has at least one PNE profile.*

Proof. This simple proof is implicit in [Monderer and Shapley, 1996]. Let ϕ be an ordinal potential function for the strategic game Γ. Since the strategy space Π of the game is finite, there must exist at least one strategy profile π^* that minimizes ϕ. This profile, by the definition of a generalized ordinal potential function, is a PNE. □

The next important theorem further clarifies the relation between exact potential games and congestion games, essentially stating that (up to isomorphism) they are the same class of games.

Theorem 2.3.5 (Monderer and Shapley, 1996)**.** *Every (finite) exact potential game is isomorphic to an unweighted congestion game.*

However, if we switch to games with weights things change quite significantly. On one hand, not all weighted congestion games have an exact potential function, in contrast with Theorem 2.3.3:

Theorem 2.3.6 (Fotakis et al., 2005)**.** *There exist weighted CG (even with identity delay functions) which possess no exact potential function.*

On the other hand, not only weighted games are not guaranteed to possess exact potential functions, but they are neither guaranteed to possess PNE profiles. Even for the most restricted class of weighted games, i.e. weighted single-commodity network congestion games, it is known that some games may not have pure Nash equilibria:

Theorem 2.3.7 (Fotakis et al., 2005)**.** *There are weighted single-commodity network congestion games which have no PNE profile.*

In fact, deciding if a given weighted network congestion game with non-decreasing delay functions possesses a PNE is a hard problem:

Theorem 2.3.8 (Dunkel and Schulz, 2006)**.** *The problem of determining if a given weighted network congestion game with non-decreasing delay functions has any PNE profile is strongly `NP`-complete for single-commodity instances as well as for multi-commodity instances when the number of players is fixed.*

The proof of Theorem 2.3.7 is through a game with piecewise delay functions. If we put an additional restriction, however, and consider only those games with linear delay functions, this is not true anymore, even for multi-commodity games:

Theorem 2.3.9 (Fotakis et al., 2006)**.** *Every weighted multi-commodity network congestion game with linear delay functions has at least one PNE profile.*

2.3.2 Complexity of Nash Equilibria Computation.

The proofs of the two main above theorems about PNE existence (Nash, Rosenthal) are both non-constructive. Thus, they leave unresolved an issue of remarkable importance from the point of view of the computer science: what is the computational complexity of finding a Nash equilibrium? Is it a task which can be solved efficiently? For the class of general strategic games, this questions were recently answered in a series of papers which were published in a short period of time, each of them extending the results of the previous one. Daskalakis, Papadimitriou and Goldberg started the series proving the next theorem in October 2005:

Theorem 2.3.10 (Daskalakis et al., 2005)**.** *The problem of computing a (possibly mixed) Nash equilibrium for a given strategic game with at least four players is* `PPAD`*-complete*[4].

One month later, Chen and Deng, in parallel with Daskalakis and Papadimitriou, proved that the theorem also holds for games with three players.

[4]The complexity class `PPAD` (for *Polynomial Parity Argument, Directed version*) was originally described in [Papadimitriou, 1994], and is another subclass of the `TFNP` class, which we described in Section 2.2. As it happens with the class `PLS`, the other subclass of `TFNP` that we have seen, we have that

$$\texttt{FP} \subseteq \texttt{PPAD} \subseteq \texttt{TFNP}$$

Note, in addition, that the relation between the classes `PPAD` and `PLS` is currently not known.

Theorem 2.3.11 (Chen and Deng, 2005b; Daskalakis and Papadimitriou, 2005)**.** *The problem of computing a (possibly mixed) Nash equilibrium for a given strategic game with at least three players is* `PPAD`*-complete.*

Finally, one month later, in December 2005, Chen and Deng resolved complexity of the problem for two-player games.

Theorem 2.3.12 (Chen and Deng, 2005a)**.** *The problem of computing a (possibly mixed) Nash equilibrium for any given strategic game is* `PPAD`*-complete.*

On the other hand, if we consider the class of unweighted congestion games (for which the existence of pure Nash equilibria is guaranteed, as we stated before), the complexity of finding a PNE was settled by Fabrikant, Papadimitriou and Talwar in a 2004 article.

Theorem 2.3.13 (Fabrikant et al., 2004)**.** *The problem of computing a pure Nash equilibrium is* `PLS`*-complete for the following classes of games:*

i) Unweighted congestion games.

ii) Symmetric unweighted congestion games.

iii) Unweighted multi-commodity network congestion games.

However, if we consider only unweighted single-commodity network congestion games, then the computation of a PNE can be done efficiently, as the same authors prove through a reduction to the min-flow problem.

Theorem 2.3.14 (Fabrikant et al., 2004)**.** *The computation of a PNE for unweighted single-commodity network congestion games can be done in polynomial time.*

If we allow players to have weights, we have that in network congestion games with linear delay functions (for which the existence of a PNE is assured, see Theorem 2.3.9), the computation of a PNE can be carried out in pseudo-polynomial time.

	Single-com.	Multi-com.
Unweight.	`FP`	`PLS`-complete
Weight.	stepwise delay functions: a PNE May not exist linear delay functions: pseudopolynomial time	

Figure 2.1: Existence and complexity of computing a PNE in network congestion games. If not stated otherwise, the games have non-decreasing delay functions.

Theorem 2.3.15 (Fotakis et al., 2005)**.** *The computation of a PNE for weighted multi-commodity network congestion games with linear delay functions can be carried out in pseudo-polynomial time.*

We have outlined the main results about existence and computability of PNE in network congestion games in Table 2.1. As it can be seen in the whole section, three are the components of the game which play a major role in this kind of results: the game being weighted or unweighted, the game being single or multi-commodity and, finally, the kind of delay function of the game.

2.3.3 Results on Max-congestion Games

As we said on the introduction, there are some papers which we have discovered during the development of this thesis that also present some results for the model of max-congestion games. We will here just mention the three papers that we are referring to, and we will leave a detailed comparison of the results for the conclusions chapter, once we have presented our results. The three papers are

i) [Caragiannis et al., 2005], presented at the *International Symposium on Algorithms and Computation* (ISAAC) of December 2005.

ii) [Busch and Magdon-Ismail, 2006], presented at the *International Symposium on Theoretical Aspects of Computer Science* (STACS) of February 2006.

iii) [Banner and Orda, 2006], presented at the INFOCOM conference of April 2006.

Chapter 3

Existence of Pure Nash Equilibria

In this first chapter or results we study the existence of pure Nash equilibrium profiles for different kinds of NMC games. Specifically, we will see how changes in the restrictions we place upon the main components of the game –i.e. weights, delay functions, graph topology (single- / multi-commodity games)– affect the existence of PNE.

Allowing all kind of delay functions, we can see that even some (weighted) single-commodity games have no PNE profile. Consider, as an example, a simple NMC game with only two parallel links, two players with weights $w_1 = 1$ and $w_2 = 2$ and delay functions as defined in Fig. 3.1. It can be seen (a quick check suffices, since there are only four possible strategy profiles) that no matter what strategies the players choose, at least one of them will always have a better alternative. If both choose the first link, player two has

$$d_1(w) = \begin{cases} 4, & \text{if } w = 1 \\ 1, & \text{if } w = 2 \\ 4, & \text{if } w = 3 \end{cases} \quad d_2(w) = \begin{cases} 5, & \text{if } w = 1 \\ 3, & \text{if } w = 2 \\ 2, & \text{if } w = 3 \end{cases}$$

Figure 3.1: Delay Functions of a NMC game with no PNE profiles.

an incentive to move to the second one (where his cost would be 3 instead of 4). If both choose the second link, player two again has an incentive to change, since moving to link 1 would report him a cost of 1 instead of two. If player one goes to the first link and player two to the second one, player one will likely want to move to the second link, where he would have to pay a cost of 2 units instead of 4. Finally, if player one chooses the second link and player two chooses the first one, player one will find out that a move to the first link would decrease the cost he is paying from 5 to 4. Actually, it is interesting to see that the problem of determining whether a given game of this class has any PNE is, NP-complete, as we show in the next theorem.

Theorem 3.0.1. *The problem of deciding if a weighted single-commodity NMC game with arbitrary delay functions has a PNE profile is NP-complete.*

Proof. The membership in NP is quite straightforward. On one hand, every strategy profile has size polynomial with respect to the size of the game, since it contains one path for each player (and, since the game is weighted, the representation of the input game contains at least one bit per player). On the other hand, checking if a given strategy profile is a PNE is a task that can be done in polynomial time: for each of the players, we just check if there is a $(s-t)$ path with lower cost. This can be done by fixing the strategies of all other players and applying a shortest-path-like algorithm to the graph.

With respect to the hardness, the proof is based on a reduction from the satisfiability problem. Starting with a boolean formula F with n different boolean variables $x_1, x_2, \ldots, x_n$, the reduction is as follows. There are $n+2$ players $p_1, \ldots, p_n, p_{n+1}, p_{n+2}$, each of them having a weight of $n+2$ bits which uniquely identifies the player (e.g. with all bits but the i-th set to 0). Note that in this manner the definition of the delay functions of any edge can be dependant on the specific players that are using the edge (in other words, the value of the delay of an edge can depend upon if a given player is using that edge). The graph of the game consists only on three parallel links e, e_T and e_F, with respective delay functions d, d_T and d_F defined in Fig. 3.2. Here, M is a very large integer, and σ_F and σ_T are boolean assignments defined as

$$d(w) = \begin{cases} 2, & \text{if only } p_{n+2} \text{ uses } e \\ M, & \text{otherwise} \end{cases} \qquad d_F(w) = \begin{cases} 0, & \text{if } \sigma_F(F) \\ 4, & \text{otherwise} \end{cases}$$

$$d_T(w) = \begin{cases} 0, & \text{if } \sigma_T(F) \\ 3, & \text{if } \neg\sigma_T(F) \wedge p_{n+1} \text{ and } p_{n+2} \text{ use } e_T \\ 5, & \text{if } \neg\sigma_T(F) \wedge p_{n+1} \text{ uses } e_T \wedge p_{n+2} \text{ does not} \\ 1, & \text{if } \neg\sigma_T(F) \wedge p_{n+2} \text{ uses } e_T \wedge p_{n+1} \text{ does not} \end{cases}$$

Figure 3.2: Delay Functions for the game of Theorem 3.0.1

follows:

$$\begin{aligned} \sigma_F(x_i) = \text{False} \quad &\Leftrightarrow \text{player } p_i \text{ uses edge } e_F \\ \sigma_T(x_i) = \text{True} \quad &\Leftrightarrow \text{player } p_i \text{ uses edge } e_T \end{aligned}$$

Now, suppose that F is satisfied by at least one assignment σ^*, and let π^* be a strategy profile such that every player p_i (with $1 \leq i \leq n$) uses edge e_T if $\sigma^*(x_i)$ is true and uses edge e_F otherwise. Both boolean assignments σ_T and σ_F are thus equivalent to σ^*, so (as long as players p_{n+1} and p_{n+2} choose either edge e_T or edge e_F) all players have to assume a cost of 0, which implies that π^* is a PNE.

On the other hand, suppose that F is unsatisfiable, and let π be any strategy profile. Since the formula is unsatisfiable, both boolean assignments σ_T and σ_F are always such that $\sigma_T(F) = \sigma_F(F) = \text{False}$. In this situation, no matter what the other players choose, it can be seen (e.g. by examining the 9 different possibilities) that either player p_{n+1} or player p_{n+2} always have an incentive to change their strategy. □

Now, it would be interesting to consider a very natural restriction on the delay functions of the games, namely that they be non-decreasing. If we examine first unweighted games, we have to note that, unlike what happens with (traditional) unweighted congestion games, not all unweighted non-decreasing delay NMC games have an exact potential function:

Lemma 3.0.2 (Caragiannis et al., 2005)**.** *There exist unweighted non-decreasing delay NMC games which possess no exact potential function.*

Therefore, the existence of PNE profiles cannot be proved directly by means of the usual potential function technique, as Rosenthal did. In early stages of our research, we managed to independently prove the existence of these PNE profiles by means of constructive proof, but our algorithm did only work for certain subclasses of unweighted non-decreasing delay NMC games. Further research led us to discover that a more general proof can be obtained which is actually made up of two results of previous papers, [Monderer and Shapley, 1996] and [Banner and Orda, 2006]. Indeed, even though the games of this class can not always be shown to have an exact potential function, even for weighted NMC games we can show that there always exist a generalized ordinal potential function. This implies, as we have seen in Theorem 2.3.4, that they possess at least one PNE profile.

Let us carefully go through the proof of this statement. First Let $\Gamma = (N, G, ((s_i, t_i))_{i \in N}, (d_e)_{e \in E(G)}, (w_i)_{i \in N})$ be a weighted network max-congestion game. For any possible strategy profile π of the game, let $A(\pi)$ be a tuple $A(\pi) = (c_{i_1}(\pi), c_{i_2}(\pi), \ldots, c_{i_n}(\pi))$ such that $c_{i_1}(\pi) \geq c_{i_2}(\pi) \geq \cdots \geq c_{i_n}(\pi)$ are the costs assumed by the players ordered in non-increasing order. Let $<_l \colon \mathbb{N}^n \to \mathbb{N}^n$ be the usual (total) lexicographical order defined over all possible pairs of tuples $A(\pi)$, $A(\pi')$. Now, consider two possible strategy profiles π and π' such that only one player has changed his strategy between them, i.e. such that $\pi = (\pi_{-i}, p)$ and $\pi' = (\pi_{-i}, p')$ for some strategies $p \neq p'$ of player i.

Claim 3.0.3 (Banner and Orda, 2006, Theorem 3)**.**

$$c_i(\pi') < c_i(\pi) \Rightarrow A(\pi') <_l A(\pi)$$

Proof. The proof of the statement is an adaptation of the proof of Theorem 3 of in [Banner and Orda, 2006] which can be directly generalized to the case of arbitrary non-decreasing delay functions.

Recall that player all players but player i have the same strategy both in π and π'. Thus, between π and π' only player i changes his strategy, and by doing this he obtains a lower cost. Assume that the value of the cost payed by player i in π is the j-th element of the vector $A(\pi)$, i.e.

$$A(\pi) = (c_{i_1}(\pi), c_{i_2}(\pi), \dots, c_{i_{j-1}}(\pi), c_i(\pi), c_{i_{j+1}}(\pi), \dots, c_{i_n}(\pi))$$

Assume also that the first m elements of $A(\pi)$ have values greater or equal than $c_i(\pi)$ (thus $m \geq j$).

Now, suppose that $A(\pi') \geq_l A(\pi)$. By definition of the ordering, this means that either the first m elements of the vector have the same value in both $A(\pi)$ and $A(\pi')$ or at least one of them has increased its value. Since the value $c_i(\pi)$ that was occupying the j-th position has decreased, in both cases another element of the vector must have increased its value to at least $c_i(\pi)$. The only way for player i to make a resource increase its delay is to actually use it (for the delay functions are non-decreasing), so we have that in π' player i is using a resource with a value at least $c_i(\pi)$. That contradicts the fact that $c_i(\pi') < c_i(\pi)$. Therefore, we have that $A(\pi') <_l A(\pi)$. $\square$

Now we are able to prove the following theorem:

Theorem 3.0.4. *Any weighted NMC game with non-decreasing delay functions has a generalized ordinal potential.*

Proof. The function γ that maps each profile π to the value of $A(\pi)$ seen as a n-digit number in base $D+1$ (that is, $\gamma(\pi) = \sum_{0 \leq j \leq n} A(\pi)_j \cdot (D+1)^j$), where $D = \max_{e \in E} d_e(W)$ and $W = \sum_{i \in N} w_i$, is therefore a generalized potential function. $\square$

Corollary 3.0.5 (Banner and Orda, 2006)**.** *Any weighted NMC game with non-decreasing delay functions game has a PNE.*

Thanks to the result by Monderer and Shapley that we mentioned in Theorem 2.3.4, the proof of the previous corollary is slightly simpler than the one in [Banner and Orda, 2006]. However, note that it is essential for the proof of Claim 3.0.3 that the delay functions of the game be non-decreasing; we have not been able to extend the idea of the proof to games with arbitrary delay functions.

Finally, let us briefly consider NMC games that are Pareto-efficient, i.e. where the cost functions are so that all improving movements that a player may do provoke no harm to the other players. It is easy to see that the function $\sigma(\pi) = \sum_{i \in N} c_i(\pi)$ is a generalized ordinal potential function (but not an exact potential function) for these games (as well as for the more general class of Pareto-efficient strategic games).

Proposition 3.0.6. *Let* $\Gamma = (N, (P_i)_{i \in N}, (c_i)_{i \in N})$ *be a Pareto-efficient game. Then,* $\sigma(\pi) = \sum_{i \in N} c_i(\pi)$ *is a generalized potential function for* Γ.

This is a sufficient condition for the following theorem.

Theorem 3.0.7. *Any Pareto-efficient weighted NMC game has a PNE profile.*

So far we have proved the existence of pure Nash equilibria profiles for a number of subclasses of network max-congestion games. In the following chapter we will deal with the computational complexity of finding these equilibria.

Chapter 4

Computation of Pure Nash Equilibria

In the previous chapter, we analysed what kind of network max-congestion games do possess pure Nash equilibrium profiles. We saw that as long as we enforce the delay functions of the games to be non-decreasing, we can ensure that all kinds of NMC games, be them single or multi-commodity, weighted or unweighted, have PNE. In this chapter, we tackle the issue from a more constructive point of view and study what is the computational complexity of computing those PNE. First, we study some classes of NMC games with non-decreasing delay functions for which the computation of a PNE can be done in polynomial time. Later, we move to some other classes for which we have not been able to prove that the computation of a PNE belongs to `P`, and we show that, at least, it does belong to the class `PLS`. We also prove the `PLS`-completeness of this same problem for a particular class of games.

4.1 Single-commodity Games

For single-commodity games, we were able to design an algorithm that computes PNE profiles for some kind of *weighted single-commodity non-decreasing delay NMC games*. The algorithm BDP (for *Best Disjoint Path*) is essen-

Algorithm 1: Computation of a PNE (BDP algorithm)

```
input  : A weighted single-com. game
         $\Gamma = (N, G, (s,t), (d_e)_{e \in E(G)}, (w_i)_{i \in N})$ with non-decreasing
         delay functions
output: A strategy profile $\pi$ for $\Gamma$
begin
    Let $M$ be a *maximal* set of $(s - t)$ disjoint paths;
    $N' := N$;
    while $N' \neq \emptyset$ do
        $i := \operatorname{argmax}_{i \in N'} w_i$;
        $p := \operatorname{argmin}_{p \in M} \max_{e \in p} d_e(l_e(\pi) + w_i)$;
        Allocate player $i$ to path $c$;
        $N' := N' \setminus \{i\}$;
end
```

tially an adaptation of the *round-robin* philosophy. It proceeds in two phases. First, it computes a maximal set M of $(s - t)$ disjoint paths, and then it assigns players to paths in M step by step: at each step, the player i with maximum weight (from the set of players still unassigned) is assigned to the path $p \in M$ that yields him the minimum cost, that is, the path that minimizes the value $\max_{e \in p} d_e(l_e(\pi) + w_i)$, where π is the (partial) strategy profile we are constructing.

Before we could find a formal proof of the correctness, we ran some experiments which empirically tried to confirm or refute the hypothesis that the algorithm actually computed a PNE profile. The experiments consisted of the following steps:

- We generated random unweighted single-commodity NMC games with identity delay functions and with a maximum of 500 players and with a graph of at most 100 nodes.
- We applied the BDP algorithm over the random game.

- We checked that the obtained strategy profile was indeed a PNE profile.

The randomly generated games were simpler than the weighted NMC games that we are considering in the section, but this does not invalidate the utility of the experiment. We ran more than 100.000 times the abovementioned steps to find that all computed profiles were PNE profiles. This clearly supported the idea that, at least for this simpler games, the algorithm could be proved to be correct. Some more details about the process of generation of random games can be found in Appendix A.

Let us now present a formal analysis of the properties of the profiles computed by the algorithm.

Lemma 4.1.1. *After each iteration of the BDP algorithm, no player already assigned to some path in the (partial) profile π has an incentive to change his strategy for another path from M.*

Proof. The claim obviously holds after the first iteration. By induction, suppose that it also holds for the (partial) profile π^k obtained after the k-th iteration, let i be the player assigned to path p on the iteration $k+1$ and let us consider the profile $\pi^{k+1} = (\pi^k_{-i}, p)$ computed after this $(k+1)$-th iteration. By construction, player i is satisfied with his strategy in π^{k+1}. Let us now suppose that for some player j already assigned to p in π^k there is an incentive to change his strategy to another path $p' \in M$. This means that $\max_{e \in p'} d_e(l_e(\pi^k)+w_j) < \max_{e \in p} d_e(l_e(\pi^k)+w_i)$. Since $w_i \leq w_j$ and the delay functions are non-decreasing, $\max_{e \in p'} d_e(l_e(\pi^k) + w_i) \leq \max_{e \in p'} d_e(l_e(\pi^k) + w_j)$, but this implies that $\max_{e \in p'} d_e(l_e(\pi^k) + w_i) < \max_{e \in p} d_e(l_e(\pi^k) + w_i)$, which contradicts the fact that p has been chosen to minimize this last value. Thus, no player assigned to p has incentive to change his strategy.

Now, let us suppose that for some player j assigned to a path $p' \neq p \in M$ there is an incentive to change his strategy to another path in M. The cost induced by paths different than p has clearly remained unchanged from π^k to π^{k+1} (since we are considering only disjoint paths), while the cost induced by path p cannot be lower in π^{k+1} than in π^k, for the delay functions are non-decreasing. Thus, this contradicts our inductive hypothesis that all players in π^k were satisfied with their path. Hence, all players in π^{k+1} are satisfied. □

For graphs where all the $(s-t)$ paths are disjoint, the set M computed by the BDP algorithm coincides with the set of all the strategies available to players. Hence, the computed profile can be immediately seen to be a PNE.

Corollary 4.1.2. *For single-commodity NMC games where the graph consists only of a number of $(s-t)$ disjoint paths, the profile computed by the BDP algorithm is a PNE.*

On the other hand, if we consider general graph topologies but we restrict the delay functions of the edges to be identical, we have that, although there may be strategies other than the disjoint paths, none of them will yield a better cost.

Corollary 4.1.3. *For single-commodity NMC games where all the delay functions are identical, the profile computed by the BDP algorithm is a PNE.*

Proof. Notice that the profiles π computed by the algorithm have the property that, for any path $p \in M$, all edges $e \in p$ have the same congestion $l_e(\pi)$. Thus, we now have that the delay induced by all edges of p is equal. Since M is maximal, any $(s-t)$ path p' shares at least one edge e with some path $p \in M$, and this edge e induces a delay equal to the delay induced by the whole path p. Hence, a player choosing p' would have to assume a cost at least as large as if he chose p, which (by the previous lemma) is no better than the cost of the path he has been assigned to in π. □

Given that a maximal set of disjoint paths (even a maximum set, see for instance [Kleinberg and Tardos, 2006]) can be obtained in polynomial time and that our algorithm runs also in polynomial time, we can then state the following theorem.

Theorem 4.1.4. *Computing a PNE can be done in polynomial time for both (a)Weighted single-commodity NMC games with non-decreasing delay* *functions where all the $(s{-}t)$ paths of the graph are disjoint. and (b) Weighted single-commodity NMC games with identical non-decreasing delay functions.*

We have not been able to extend the idea of this algorithm to general weighted single-commodity games, nor to prove completeness of the problem for some complexity class harder than `FP`. The complexity of the problem for general weighted single-commodity games, thus, remains open, but we believe this contribution to be interesting enough, both for the importance of the class of games with identical non-decreasing delay functions and for the simplicity of the algorithm.

4.2 Multi-commodity Games

We will here study the complexity of the computation of equilibria in multi-commodity NMC games. The technique of distributing players among a set of disjoint paths cannot be applied to this kind of games anymore, since edge-disjoint paths among different commodities may not exist (and even if they do, a simple distribution of players among paths may not lead to a PNE). The majority of our attempts to find an algorithm to do this task for general multi-commodity games have been unsuccessful. So far, we have only been able to solve the problem for a particular subclass of this games, but this, in our opinion, should be seen as an important first step.

4.2.1 l-common-edges Games

The particular subclass of games that we just mentioned, which we call *l-common-edges* games, is made up of unweighted NMC games with identity delay functions where the graph of the game has a restricted topology. Let us first describe the graph topology of these games. Let $M = \{(s_i, t_i) \mid i \in N\}$ be the set of the m different commodities of a NMC game (note that $m \leq |N|$). A *l-common-edges* game is a game whose graph has the following topology:

- There is a set E^* of l *common* or *public* edges such that for each edge $e \in E^*$ and each commodity $i \in M$ there is exactly one (s_i-t_i) path p_i^e using e.

- For each commodity $i \in M$ there are two different sets of (s_i-t_i) paths:

  - A set L_i of l_i *private* paths that are edge-disjoint with respect to all other paths in the graph.

  - A set $L'_i = \{p_i^e \mid e \in E^*\}$ of l *public* paths that are edge-disjoint one with respect to the other (exactly one path for every common edge).

Note that, given this l-common edges topology, the congestion of any path p_i^e is always equal to the congestion of the edge $e \in E^*$.

We will also use the following notation. For each edge $e \in E^*$, let $P_e = \bigcup_{i \in M} \{p_i^e\}$ be the set of m paths using the common edge e. For any commodity i, we denote by N_i the set of players choosing (s_i-t_i) paths and let $n_i = |N_i|$. For any subset $S \subseteq M$, we denote by N_S the set of players that choose paths between the two nodes of any commodity in S and let $n_S = |N_S|$. Let also $L_S = \bigcup_{i \in S} L_i$ and $L'_S = \bigcup_{i \in S} L'_i$ be, respectively, the sets of all private and public paths of commodities in S, and let $l_S = |L_S|$ and $l'_S = |L'_S|$. Finally, let k_S the quotient $\frac{n_S}{l+l_S}$.

4.2.2 An Algorithm

We now present an algorithm for the computation of PNE in the previously described class of games (see Algorithm 2). In the subsequent description, we will denote by k_S the quotient $\frac{n_S}{l+l_S}$. Given an unweighted multi-commodity NMC game, the algorithm starts (lines 2-5) by searching a subset of commodities which may be able to *dominate* all the set of common edges, that is, to be the only commodities whose players use those edges in such a way that players from other commodities can be satisfied even if they only use their private paths.

Algorithm 2: PNE computation in multi-commodity NMC games.

```
input  : An unweight. $l$-common-edges game $\Gamma = (N, G, ((s_i, t_i))_{i \in N})$
output: A profile $\pi$ for the given game
1  begin
       // Selection of a proper subset of commodities:
2      foreach $t := 1, \dots, m$ do
3          foreach $S \subseteq M$ s.t. $|S| = t$ and while $S$ is not proper do
4              if $\forall i \in M \setminus S \ \left(\lfloor k_S \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1\right)$ then
5                  $S$ is a proper subset
       // Distribution of players among paths:
6      foreach $i \in S$ do
7          foreach $p \in L_i$ do
8              Allocate $\lfloor k_S \rfloor$ players from $N_i$ to path $p$
9      foreach $e \in E^*$ do
10         Allocate $\lfloor k_S \rfloor$ players from $N_S$ to the paths in $P_e$
11     foreach $i \in S$ do
12         foreach $p \in L_i$ do
13             if there remain unallocated players in $N_i$ then
14                 Allocate one player from $N_i$ to path $p$
15     foreach $e \in E^*$ do
16         if there remain unallocated players in $N_S$ then
17             Allocate one player from $N_S$ to a suitable path $p \in P_e$
18     Let $N'_S$ be the set of players from $N_S$ still not allocated
19     while $N'_S \neq \emptyset$ do
20         Let $j$ be a player from $N'_S$
21         Let $i \in S$ be a commodity s.t.
22          1) $\exists$ a path $p \in L_i$ with only $\lfloor k_S \rfloor$ allocated players, and
23          2) $\exists$ a player $j' \in N_i$ already allocated to a path $p' \in L'_i$
24         Allocate player $j'$ to path $p$ and player $j$ to path $p'$
25         $N'_S := N'_S \setminus \{j\}$
26     foreach $i \in M \setminus S$ do
27         Allocate players in $N_i$ to paths in $L_i$ in a Round-Robin fashion
28 end
```

Lemma 4.2.1. *After the execution of lines 2-5 of Algorithm 2, the following statements hold:*

i) $\forall i \in M \setminus S \ \left(\lfloor k_S \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1\right)$

ii) $\forall S' \subseteq M \left(|S'| < |S| \Rightarrow \exists i \in M \setminus S' \left(\lfloor k_S \rfloor < \left\lceil \frac{n_i}{l_i} \right\rceil - 1\right)\right)$

iii) $\forall i \in S \left(\frac{n_i}{l_i} \geq k_S\right)$

Proof. The first proposition is exactly the condition in line 4. Notice that this condition will evaluate to true sooner or later, since it always holds when $S = M$. The second proposition is directly implied by the order in which the algorithm considers the subsets of M.

Finally, let us prove the last proposition. Suppose that there exists a commodity $i \in S$ for which the proposition does not hold, i.e. $\frac{n_i}{l_i} < k_S$ or, equivalently, $n_i < k_S \cdot l_i$. Thus,

$$\begin{aligned} k_{S\setminus\{i\}} &= \frac{n_{S\setminus\{i\}}}{l + l_{S\setminus\{i\}}} = \frac{n_S - n_i}{l + l_S - l_i} > \frac{n_S - k_S \cdot l_i}{l + l_S - l_i} = \frac{n_S - (\frac{n_S}{l+l_S}) \cdot l_i}{l + l_S - l_i} \\ &= \frac{n_S(l + l_S - l_i)}{(l + l_S)(l + l_S - l_i)} = \frac{n_S}{l + l_S} = k_S \end{aligned}$$

Therefore, it holds that

$$k_{S\setminus\{i\}} > k_S > \frac{n_i}{l_i} \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1$$

Since $\left\lceil \frac{n_i}{l_i} \right\rceil - 1$ is an integer, $\lfloor k_{S\setminus\{i\}} \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1$. On the other hand, thanks to the first proposition of the lemma we can state that

$$\forall i' \in M \setminus S \left(\lfloor k_{S\setminus\{i\}} \rfloor \geq \lfloor k_S \rfloor \geq \left\lceil \frac{n_{i'}}{l_{i'}} \right\rceil - 1\right)$$

Combining the two last propositions, we have that

$$\forall i' \in M \setminus (S \setminus \{i\}) \left(\lfloor k_{S\setminus\{i\}} \rfloor \geq \left\lceil \frac{n_{i'}}{l_{i'}} \right\rceil - 1\right)$$

but, since $|S \setminus \{i\}| < |S|$, that contradicts what we have shown in the second proposition of the lemma. Consequently, the third proposition must hold. □

Once the algorithm has determined which subset $S \subseteq M$ dominates the set E^* of common edges, it distributes players among paths in a particular manner. In the first place (l. 6-25), it distributes all the players from the commodities in S. The loop in lines 6-8 allocates $\lfloor k_S \rfloor$ players to every path of every commodity $i \in S$. Since, as we have seen, $n_i \geq l_i \cdot k_S$, enough players exist to allow this allocation. After the execution of the loop, $l_S \cdot \lfloor k_S \rfloor$ players have been allocated. Thus, there remain at least $k_S \cdot l$ unallocated players. This fact ensures the correct execution of the second loop (lines 9-10), given that it allocates exactly $\lfloor k_S \rfloor \cdot l$ players to paths in L'_S. After the execution of these loops, every path in L_S, as well as every edge in E^*, have a congestion of exactly $\lfloor k_S \rfloor$ units. Next (l. 11-14), the algorithm iterates again through all the private paths of every commodity in S, allocating –when possible– one more player to each of them. Similarly, the loop of lines 15-17 allocates to each group of paths P_e (for all $e \in E^*$) at most one more player from N_S. Therefore, after the execution of these two loops every path in L_S and every edge in E^* have congestion either $\lfloor k_S \rfloor$ or $\lfloor k_S \rfloor + 1$.

Let us now analyze the loop of lines 19-25:

Lemma 4.2.2. *At each iteration of the loop of lines 19-25 of Algorithm 2, the following propositions hold:*

i) There exists a commodity $i \in S$ such that

(a) There exists a path $p \in L_i$ with only $\lfloor k_S \rfloor$ allocated players, and

(b) There exists a player $j' \in N_i$ already allocated to a path $p' \in L'_i$

ii) Every path in L_S and every edge in E^* (and thus every path in L'_S have congestion either $\lfloor k_S \rfloor$ or $\lfloor k_S \rfloor + 1$

Proof. i) Consider the set $S^* \subseteq S$ of commodities $i \in S$ with at least one path in L_i having congestion $\lfloor k_S \rfloor$. Let us first see that, at any iteration of the loop, this set S^* is not empty. Since we have entered the loop, there remain some unallocated players in $N'_S \subseteq N_S$. That implies that all paths in L'_S have congestion $\lfloor k_S \rfloor + 1$. If there were a path in L'_S

with congestion $\lfloor k_S \rfloor$, then there would be a path with that congestion in each L'_i, $i \in S$, due to the nature of our graph topology. But that is not possible, otherwise the loop of lines 15-17 would have allocated to these paths the unallocated players from N'_S. It is then impossible that all commodities $i \in S$ have congestion $\lfloor k_S \rfloor + 1$, otherwise the total number of players allocated would be $(l_S + l) \cdot (\lfloor k_S \rfloor + 1)$, which is strictly greater than n_S, the actual number of players. Thus, we have that S^* contains at least one commodity.

In turn, that fact implies that no player of N_{S^*} is in N'_S, the set of unallocated players; otherwise, the loop of lines 11-14 would have allocated him to one of the paths of his commodity with congestion $\lfloor k_S \rfloor$. All the players in N'_S, then, come from commodities in $S \setminus S^*$ (and that also proves that $S^* \subsetneq S$, that is, that $S \setminus S^*$ is not empty).

Let us now suppose that all players in N_{S^*} are allocated to paths in L_{S^*}. One one hand, that means that all shared paths in L'_S are allocated only to players in $N_{S \setminus S^*}$. Since, by definition of S^*, all paths in $L_{S \setminus S^*}$ have congestion $\lfloor k_S \rfloor + 1$, the number of players in $N_{S \setminus S^*}$ already allocated equals $(\lfloor k_S \rfloor + 1) \cdot (l + l_{S \setminus S^*})$. Furthermore, there is at least one more (unallocated) player in $N_{S \setminus S^*}$, otherwise we would have not entered the loop. Therefore, the total number of players in $N_{S \setminus S^*}$ is $n_{S \setminus S^*} \geq (\lfloor k_S \rfloor + 1) \cdot (l + l_{S \setminus S^*}) + 1 > (\lfloor k_S \rfloor + 1) \cdot (l + l_{S \setminus S^*})$, so we have that $\frac{n_{S \setminus S^*}}{l + l_{S \setminus S^*}} > \lfloor k_S \rfloor + 1$.

On the other hand, that means that the number of players in each commodity $i^* \in S^*$ is $n_{i^*} < (\lfloor k_S \rfloor + 1) \cdot l_{i^*}$. Therefore, $\frac{n_{i^*}}{l_{i^*}} < \lfloor k_S \rfloor + 1 < \frac{n_{S \setminus S^*}}{l + l_{S \setminus S^*}} = k_{S \setminus S^*}$, using the previous inequality. In turn, the total number of players in N_{S^*} is $n_{S^*} < (\lfloor k_S \rfloor + 1) \cdot l_{S^*}$, so we have that $\frac{n_{S^*}}{l_{S^*}} < (\lfloor k_S \rfloor + 1) < k_{S \setminus S^*}$, using the same inequality as before. Therefore,

$$k_S = \frac{n_S}{l + l_S} = \frac{n_{S^*} + n_{S \setminus S^*}}{l + l_{S^*} + l_{S \setminus S^*}} < \frac{n_{S \setminus S^*}}{l + l_{S \setminus S^*}} = k_{S \setminus S^*}$$

which makes use of the general fact that $\frac{c}{d} < \frac{a}{b} \Rightarrow \frac{a+c}{b+d} < \frac{a}{b}$.

Now, we can apply lemma 4.2.1 to state that

$$\forall i \in M \setminus S \ \left(k_{S\setminus S^*} > k_S \geq \lfloor k_S \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1 \right)$$

while, as we have just seen, $\forall i^* \in S^* \ \left(k_{S\setminus S^*} > \frac{n_{i^*}}{l_{i^*}} > \left\lceil \frac{n_{i^*}}{l_{i^*}} \right\rceil \right)$. Merging both assertions, we have that

$$\forall i \in M \setminus (S \setminus S^*) \ \left(k_{S\setminus S^*} > \left\lceil \frac{n_i}{l_i} \right\rceil - 1 \right)$$

which implies that

$$\forall i \in M \setminus (S \setminus S^*) \ \left(\lfloor k_{S\setminus S^*} \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1 \right)$$

since $\left\lceil \frac{n_i}{l_i} \right\rceil - 1$ is an integer.

This last assertion, however, contradicts the second proposition of lemma 4.2.1. For that reason, we have to conclude that there exists at least one player in N_{S^*} which is allocated to a path in L'_S, and the proposition is proved.

ii) The second proposition is not difficult to prove, since the algorithm first reallocates a player from a path with congestion $\lfloor k_S \rfloor + 1$ to a path with congestion $\lfloor k_S \rfloor$ and then allocates a player to the first path.

□

At the end of the loop of lines 19-25, then, we have that for any commodity $i \in S$, any player $j \in N_i$ is allocated to a path with congestion at most $\lfloor k_S \rfloor + 1$. Since all the other paths available to j are in $L_S \cup L'_S$ and thus have congestion at least $\lfloor k_S \rfloor$, player j has no incentive to change his strategy.

With respect to the players of commodities in $M \setminus S$, the algorithm (l. 26-27) simply allocates them to paths in L_i in a Round-Robin fashion. We then have that, if n_i is divisible by l_i, all paths in L_i will have congestion $\frac{n_i}{l_i}$. Thus, for any commodity $i \in M \setminus S$ and any player $j \in N_i$, player j will have no incentive to change his strategy, since all the other paths in L_i have the same congestion $\frac{n_i}{l_i}$, whereas all the paths in L'_i have an edge in E^* with congestion at least $\lfloor k_S \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1 = \frac{n_i}{l_i} - 1$ (recall lemmas 4.2.1 and 4.2.2).

On the other hand, if n_i is not divisible by l_i, all paths in L_i will have congestion either $\left\lfloor \frac{n_i}{l_i} \right\rfloor$ or $\left\lfloor \frac{n_i}{l_i} \right\rfloor + 1$. Thus, for any commodity $i \in M \setminus S$ and any player $j \in N_i$, player j will have no incentive to change his strategy, since the paths in L_i will have congestion at least $\left\lfloor \frac{n_i}{l_i} \right\rfloor$, and –as we have already seen– the paths in L_i' have congestion at least $\lfloor k_S \rfloor \geq \left\lceil \frac{n_i}{l_i} \right\rceil - 1 = \left\lfloor \frac{n_i}{l_i} \right\rfloor$. As no player has an incentive to change his strategy, we can conclude that the profile computed by our algorithm is a PNE. Furthermore, if we consider that the number of commodities is fixed, the running time of the algorithm can be proved to be polynomial. Therefore, the following theorem is proved.

Theorem 4.2.3. *Let* Γ *be an unweighted* l-common-edges *game. Algorithm 2 computes a PNE for* Γ*; furthermore, if we consider the number of commodities to be a fixed constant* k*, the algorithm solves the problem of computing a PNE for unweighted* k*-commodities* l-common-edges *games in polynomial time.*

4.3 Towards PLS

The absence of an exact potential function for NMC games with non decreasing delay functions prevents us from being able to prove (in a manner completely analogous to the way that this can be done for congestion games, see [Fabrikant et al., 2004]) that the computation of a PNE for general NMC games is in PLS. However, we will now see that the generalized ordinal potential function γ introduced in the proof of theorem 3.0.4, along with a special definition of the neighborhood structure of the search problem, suffice for the proof of the aforementioned membership in PLS.

Theorem 4.3.1. *The problem of computing a PNE for weighted multi-commodity non-decreasing delay NMC belongs to* PLS*.*

Proof. Consider the search problem of finding a local optimum of γ, where the set of feasible solutions contains all valid strategy profiles of our game and the neighborhood $N(\pi, \Gamma)$ of a solution π is the set of all profiles π' where exactly one player has changed his strategy for a better one:

$$N(\pi, \Gamma) = \{\pi' \in \Pi \mid \exists i \in N \ \exists p \in P_i \ \pi' = (\pi_{-i}, p) \wedge c_i(\pi') < c_i(\pi)\}$$

Notice that this last condition $c_i(\pi') < c_i(\pi)$ implies that (i) the neighborhood of π is empty if and only if π is a PNE and (ii) For any neighbor $\pi' \in N(\pi, \Gamma)$ of a given profile π, it holds that $\gamma(\pi') < \gamma(\pi)$ (by definition of γ and Theorem 3.0.4). Thus, finding a PNE is equivalent to finding a local minimum of the function γ with respect to the neighborhood defined above.

This search problem belongs to PLS, since (a) an initial solution can be produced in polynomial time by assigning to every player an arbitrary strategy, (b) The cost $\gamma(\pi)$ of any profile π can be computed in polynomial time. (c) deciding whether $N(\pi, \Gamma) = \emptyset$ (i.e. π is a local optimum) or, if this is not the case, computing a strategy profile $\pi' \in N(\pi, \Gamma)$ s.t. $\gamma(\pi') < \gamma(\pi)$ can be done in polynomial time using a modification of the Dijkstra algorithm where the shortest path computation is done considering that the length of an edge e is $d_e(l_e(\pi))$ and the length of a path p is $\max_{e \in p} d_e(l_e(\pi))$. $\square$

Notice that the previous proof can be immediately generalized to the case of general max-congestion games. We only have to see that, since the representation of the game explicitly contains the set of actions for each player, deciding if a given profile is a local optimum with respect to the neighborhood can be done in polynomial time by computing the cost of all neighbors of the profile.

Also observe that, as we mentioned before, the neighborhood used in the proof is not the neighborhood that one may initially think of when considering the nature of games (see, for instance, [Fabrikant et al., 2004; Nisan et al., 2007]), where the neighbors of a given profile π are *all* profiles $\pi' = (\pi_{-i}, p)$ (for some $i \in N$ and $p \in P_i$). Given that max-congestion games are not exact potential games and that we only have at our disposal a *generalized ordinal potential* function, we have to restrict the neighbors to those profiles where the deviating player improves his cost in order to get a search problem whose local optima coincide with the PNE profiles of the game.

Finally, let us say that the requirement of the delay functions being non-decreasing is essential to the proof of Claim 3.0.3, so the above technique can not be trivially extended to games with arbitrary delay functions. However, there is another particular class of games with delay functions that are not restricted to be non-decreasing which does actually possess a generalized ordinal potential function. An analogous proof can be used to show that the problem of computing a PNE for any Pareto-efficient strategic is equivalent to the `PLS` problem of finding a local minimum of the function σ that allowed us to prove theorem 3.0.7) with respect to the neighborhood defined in the proof of theorem 4.3.1.

Theorem 4.3.2. *Computing a PNE for Pareto-efficient strategic games is in `PLS`.*

Proof. This problem is equivalent to the problem of finding a local minimum of the function σ with respect to the neighborhood defined in the proof of theorem 4.3.1. □

Corollary 4.3.3. *Computing a PNE for Pareto-efficient NMC is in `PLS`.*

The Pareto-efficiency may seem too strong a restriction, i.e. one may think that the class of NMC games which are Pareto-efficient is a very simple class and the the previous `PLS` upper bound is too loose. However, allowing unrestricted delay functions makes the problem complex to the point that it can be proved to be `PLS`-hard: we next prove that the computation of a PNE for Pareto-efficient weighted parallel links[1] NMC games is a `PLS`-hard problem. The proof is based on a `PLS`-reduction from the MAX-CUT problem with the flip neighborhood, for which finding a local optimum is known to be `PLS`-hard [Schäffer and Yannakakis, 1991]. In this problem, we are given an undirected graph with weights on the edges and we have to find a partition

[1]A parallel links game is a restricted version of a single-commodity game where all the edges of the graph are edges between s and t (i.e. the graph is actually a multi-graph).

of the nodes into two disjoint sets A and B such that the cut of the partition (i.e. the sum of weights of edges between nodes assigned to different sets) cannot be increased by changing one single node from A to B or vice versa.

Theorem 4.3.4. *The problem of computing a PNE for weighted Pareto-efficient parallel links NMC games is* `PLS`*-complete.*

Proof. Let us define W_T as the sum of weights of all edges and W_A (and analogously, W_B) as the sum of weights of edges with both endpoints in A (B). Then, the value of the cut is $W_T - (W_A + W_B)$, and maximizing it is equivalent to minimizing $W_A + W_B$, since W_T is fixed.

Now, given a MAX-CUT instance, we can define a weighted parallel links max-congestion game in the following way. There's one player for every node of the original graph, and the weight of each player is used as a means of identifying that player. Thus, the weight w_i of player i is a binary number of length n where the i-th digit is 1 and the rest of the digits are 0. The graph G of the game is a simple graph with two nodes, s and t, and two parallel links e_1 and e_2 from s to t. Finally, the delay function of both edges is defined as

$$d(l) = \sum_{\substack{1 \leq i < j \leq n \\ \text{s.t. } l_i = l_j = 1}} w_{i,j} + \sum_{\substack{1 \leq i < j \leq n \\ \text{s.t. } l_i = l_j = 0}} w_{i,j}$$

where l_i is the i-th digit of l and $w_{i,j}$ is the weight of edge (i, j). Intuitively, player i choosing edge e_1 can be thought of as node i being assigned to set A (equivalently for edge e_2 and set B). The delay of both edges is then $W_A + W_B$. Thus, in any PNE the value $W_A + W_B$ cannot be decreased by one player moving from one edge to the other, so the partition of nodes of the MAX-CUT problem induces a maximum cut with respect to the flip neighborhood. Besides, the game is clearly Pareto-efficient, so the proof is complete. $\square$

Note that the previous reduction implies the `PLS`-hardness of the computation of a PNE for Pareto-efficient weighted network congestion games (since there is no distinction between parallel links NMC games and parallel links network congestion games) and general Pareto-efficient strategic games.

Chapter 5

Conclusions

In this final chapter we will quickly summarize the results so far presented in this thesis. Afterwards, we will point out the main open problems and suggest some future research lines.

5.1 Summary and Discussion of the Results

We will here discuss, as a recapitulation, about the development of this thesis and the obtained results and their relevance when compared to the results obtained in the other papers that deal with the model of max-congestion games, which were mentioned in Section 2.3.3. Unfortunately for the originality of the results that we present here, it was not until a medium stage of our research that we found these other papers, some of whose results subsumed part of ours. The cause of our ignorance, our lack of skill apart, could be attributed to the fact that two of the mentioned papers were published once we had already started our research. Also, the novelty of this research area and the high number of publications provoke, on one hand, a somewhat misleading lack of uniformity in denominations and notation and, on the other hand, a difficulty to keep up to date with new publications. A proof of this can be seen in the fact that each of the three mentioned papers gives a different name to what we call *max-congestion game* (*Bottleneck routing* *games*, *Network load games*, *Routing games on maximum congestion*), not to mention that the results of the three papers are overlapping, which probably shows that the authors of each of the papers were not aware of the other publications.

For this reason, we believe that the value of this thesis is also to bee looked for in the documenting effort, the unification of results mostly written under different notation paradigms and the clarification of obscure points.

Chronologically, the first model of max-congestion games that we studied was the simplest one: unweighted and single-commodity network games with identity delay functions. A preliminary study of the properties of this games led us to the development of the polynomial-time BDP algorithm and the analysis of its properties and correctness. The idea behind the algorithm is quite simple: given that the edges have identity delay functions, the multiples choices that players have (as many as $(s - t)$ paths are in the graph) can actually be reduced to a smaller set of disjoint paths. This guarantees some nice properties which can be exploited in order to obtain an equilibrium profile just by evenly distributing players among paths. Once we could prove the correctness of this idea for games with identity functions, we tried to extend it to the general class of single-commodity games. We could prove that the algorithm was still correct if we allowed non-decreasing delay functions, but only as long as we enforce the different edges to have all the same delay function. This is not a minor restriction; however, we believe that the model of games where all edges have identical delay functions is still quite broad and useful. Consider, for instance, that we want to model a situation where different users of a corporate network are routing their packets solely within the network. In this case, it seems acceptable to think that all the links in the network will probably be equal and thus have equal delay functions.

We also managed to prove the correctness of the quite straight-forward extension of the algorithm to single-commodity games with non-decreasing delay functions where the graph of the game was made up only of disjoint paths, but we could not further extend the algorithm to the general case nor find another way of dealing with this case.

Later, we tackled the extension of the algorithm to unweighted multi-commodity games. There was no obvious way of dealing with the fact that the idea of considering only disjoint paths is not appropriate for multi-commodity games, since it may be the case that there exist less disjoint paths than commodities. Thus, we started to consider restricted graph topologies and ended up designing a rather involved algorithm for games with what we called *l-common-edges* graphs (see Section 4.2). This is quite a restrictive graph topology, for the interactions among players are confined to a set of l common disjoint edges, but we believe it to be a first step for the study of these games.

After this first algorithmic approach, we moved on to a more theoretical approach. We discovered the existence proof in [Banner and Orda, 2006], which applied to general network max-congestion games, as long as the delay functions were restricted to be non-decreasing. We reworked, clarified and slightly simplified this proof by means of some ideas which were present in [Monderer and Shapley, 1996]. Afterwards we proved that the computation of a PNE for network max-congestion with non-decreasing delay functions belongs to the class PLS. Our proof is based in the classical congestion games proof, but introduces a couple of novelties that overcome the absence of exact potential functions for our games. Namely, we show that the existence of a generalized ordinal potential function is sufficient, as long as we use a particular neighborhood structure, to prove the membership in PLS.

Once this result was accomplished, we tried to prove PLS-hardness. We found a rather interesting reduction from a well-known PLS-complete problem, the MAX-CUT problem with the flip neighborhood, to the problem of computing a PNE in max-congestion games (see 4.3.4). Unfortunately, the reduction used some delay functions which were not non-decreasing, so the PLS-hardness could only be proved for games with arbitrary delay functions, which so far we do not know to belong to PLS. This means that it is unlikely to find an efficient algorithm to compute PNE profiles for NMC games with arbitrary delay functions, but we still do not know if it happens the same if we enforce the delay functions to be non-decreasing. With the goal of proving the PLS-completeness of our problem for at least one subclass of network max-congestion games, we moved on to the consideration of Pareto-efficient games, an apparently very restricted class of games for which we had easily showed that the computation of a PNE belongs to PLS. However, it turned out that our previous PLS-reduction used a Pareto-efficient game, so the PLS-completeness was proved.

5.2 Novelty of the Results

Let us here discuss the relevance of our results. As we have already said, the existence of PNE for general network max-congestion games with non-decreasing delay functions had already been proved in [Banner and Orda, 2006]. However, we think that our proof, which uses part of theirs, is slightly clearer. It also uses a previous result of the field that the authors of the paper seem to ignore, and we believe this to be interesting, especially considering the high degree of fragmentation of this research field.

With respect to the constructive part of the thesis, that is, the two algorithms that we have developed to compute pure Nash equilibria, they are also mainly original results. [Caragiannis et al., 2005] states that there is a PTAS[1] that computes a strategy profile with optimum social cost[2] for weighted single-commodity NMC games with identical delay functions on the links. This means that if we are just interested in the complexity of a PNE without taking into account its social cost, this PTAS can compute it in

[1]A PTAS (for *Polynomial-Time Approximation Scheme*) is an approximation algorithm that, given an instance of an optimization problem and a value $\varepsilon > 0$, outputs a solution of the instance with cost at most $(1+\varepsilon)$ times worse than the optimal cost. The computation time of the PTAS is polynomial with respect to the size of the instance, but not necessarily with respect to the value $1/\varepsilon$. Indeed, if it is also polynomial with respect to this value, it is called a FPTAS (*Fully Polynomial-Time Approximation Scheme*)

[2]The social cost of a profile is some kind of *global* measure of its cost.

polynomial time. However, the mentioned paper does not provide any proof of their statement, so we still believe that our BDP algorithm is interesting for its conceptual clarity. On the other hand, we do not know any attempt to algorithmically solve the problem in the case of multi-commodity NMC games, so our second algorithm, however valid only for a restricted class of games, is an original contribution.

Finally, concerning all our results about membership in PLS and PLS-completeness, as far as we know they are all completely original, since no other paper relates NMC games to the PLS class.

5.3 Future Research

Let us here provide some insight on possible future lines of research which have came up during the development of the thesis.

- First, it would be very interesting to settle the computational complexity of PNE computation for weighted single-commodity NMC games with non-decreasing delay functions and no other restriction, either through a novel algorithmic approach or through a PLS-reduction.
- Another major result would be the analog for multi-commodity games, i.e. to determine if general NMC games with non-decreasing functions (for which we only know that belong to PLS) are PLS-complete or not.
- Another possibility would be to consider the so-called splittable games, that is weighted games where the players can route their weight through different paths, and see to which extent our results apply to this model
- Other interesting problems related to this model which we have not dealt with here include a number of decisional problems such as the following: What is the complexity of what is known as *nashifying* a given strategy profile, that is, transforming a given profile into an equilibrium profile without increasing the cost paid by any player? How

many selfish movements do we need from one given strategy profile to reach a situation of equilibrium or, in other words, how far from the equilibrium is a given situation? It would be very interesting if we could solve any of them, since this would give us more insight into the model.

Bibliography

S. Aaronson. Complexity zoo. 2006. URL `http://www.complexityzoo.com`.

E.H.L. Aarts and JK Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003.

H. Ackermann, H. Roglin, and B. Vocking. On the Impact of Combinatorial Structure on Congestion Games. *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)-Volume 00*, pages 613–622, 2006.

E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 295–304, 2004.

B. Awerbuch, Y. Azar, and A. Epstein. The price of routing unsplittable flow. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 2005. ACM Press.

R. Banner and A. Orda. Bottleneck Routing Games in Communication Networks. *Proceedings of the 25th INFOCOM Conference*, 2006.

B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.

C. Busch and M. Magdon-Ismail. Atomic Routing Games on Maximum Congestion. *Lecture notes in computer science*, 4041:79–91, 2006.

I. Caragiannis, C. Galdi, and C. Kaklamanis. Network load games. *LNCS*, 3827: 809–818, 2005.

X. Chen and X. Deng. Settling the Complexity of 2-Player Nash-Equilibrium. Technical Report TR05-140, Electronic Colloquium on Computational Complexity, Dec. 2005a.

X. Chen and X. Deng. 3–Nash is PPAD–Complete. Technical Report TR05-134, Electronic Colloquium on Computational Complexity, Nov. 2005b.

G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *STOC'05*, pages 67–73, New York, 2005. ACM Press.

G. Christodoulou, E. Koutsoupias, and Nanavati A. Coordination mechanisms. *Lecture Notes in Computer Science*, 3142:345–357, 2004.

A. Czumaj and B. Vcking. Tight bounds for worst-case equilibria. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 413–420. Society for Industrial and Applied Mathematics, 2002.

C. Daskalakis and C.H. Papadimitriou. Three-Player Games Are Hard. Technical Report TR05-139, Electronic Colloquium on Computational Complexity, Nov. 2005.

C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a Nash equilibrium. Technical Report TR05-115, Electronic Colloquium on Computational Complexity, Oct. 2005.

C. Daskalakis, A. Fabrikant, and C. Papadimitriou. The game world is flat: The complexity of nash equilibria in succinct games. In *Lecture Notes in Computer Science*, volume 4051, pages 513–524. Springer Berlin / Heidelberg, 2006.

J. Dunkel and A.S. Schulz. On the Complexity of Pure-Strategy Nash Equilibria in Congestion and Local-Effect Games. *LNCS*, 4286:62, 2006.

A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 604–612, 2004.

R. Feldmann, M. Gairing, T. Lcking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. *Lecture Notes in Computer Science*, 2719:514–526, 2003.

D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, volume 2380. Springer, 2002.

D. Fotakis, S. Kontogiannis, and P. Spirakis. Selfish unsplittable flows. *Theor. Comput. Sci.*, 348(2-3):226–239, 2005.

D. Fotakis, S. Kontogiannis, and P. Spirakis. Symmetry in network congestion games: Pure equilibria and anarchy cost. volume 3879, pages 161–175. Springer, 2006.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* WH Freeman & Co. New York, NY, USA, 1979.

P. Goldberg and C. Papadimitriou. Reducibility among equilibrium problems. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 61–70, New York, NY, USA, 2006. ACM Press.

R.L. Graham. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

D.S. Johnson. The NP-completeness column: Finding needles in haystacks. 2007.

D.S. Johnson, C.H. Papadimtriou, and M. Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

J. Kleinberg and E. Tardos. *Algorithm Design.* Addison-Wesley, 2006.

S. Kontogiannis and P. Spirakis. Atomic selfish routing in networks: A survey. In *Proceedings of the First International Workshop on the Internet and Network Economics (WINE 2005)*, pages 989–1002. Springer, 2005.

E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *STACS'99*, volume 1563, pages 404–413. Springer, 1999.

D.K. Lewis. *Convention: A Philosophical Study.* Blackwell Publishers, 2002.

RC Lewontin. Evolution and the theory of games. *J Theor Biol*, 1:382–403, 1961.

L. Libman and A. Orda. Atomic Resource Sharing in Noncooperative Networks. *Telecommunication Systems*, 17(4):385–409, 2001.

V. Mazalov, B. Monien, F. Schoppmann, and K. Tiemann. Wardrop Equilibria and Prize of Stability for Bottleneck Games with Splittable Traffic. *Proc. of the 2nd International Workshop on Internet and Network Economics (WINE 06)*, 2006.

N. Megiddo and C.H. Papadimitriou. On total functions, existence theorems and computational complexity. *complexity (Note)*, 81:317–324, 1991.

I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13(1):111–124, 1996.

D. Monderer and L.S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

J.F. Nash. Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic Game Theory.* Cambridge University Press New York, NY, USA, 2007.

M.J. Osborne. *An introduction to game theory.* Oxford University Press, 2004.

P.N. Panagopoulou and P. Spirakis. Efficient convergence to pure nash equilibria in weighted network congestion games. *Lecture Notes in Computer Science*, 3503:203–215, 2005.

C. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

C. Papadimitriou. Algorithms, games, and the internet. In *STOC'01*, pages 749–753. ACM Press, 2001.

C.H. Papadimitriou. The Complexity of the Lin–Kernighan Heuristic for the Traveling Salesman Problem. *SIAM Journal on Computing*, 21:450, 1992.

R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

D. Ross. Game theory. In E.N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2006. URL `http://plato.stanford.edu/archives/spr2006/entries/game-theory/`.

A.A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

R. Sedgewick. *Algorithms in C++ Part 5: Graph Algorithms*. Addison-Wesley, 2002.

M.J. Serna, C. Àlvarez, R. Cases, and A. Lozano. *Els Límits de la computació:: indecidibilitat i NP-completesa*. Edicions UPC, 2001.

B. Stroustrup. *El Lenguaje de programacin C++*. Addison-Wesley, Madrid, 2002.

J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press Princeceton, 1953.

M. Yannakakis. Computational complexity. *Local Search in Combinatorial Optimization*, pages 19–55, 1997.

Appendix A

Experimentation

In this appendix we present the algorithms that we used for the generation of random graphs and random NMC games. These algorithms have been used to experiment with our BDP algorithm, as we mentioned in Section 4.1. The algorithms were implemented in `C++` with the aid of the Boost libraries. We will here just describe the design of the algorithms.

First, Figure A.1 shows the algorithm used to randomly generate acyclic graphs. The algorithm takes as input an integer *n_nodes* which determines the number of nodes of the generated graph and a real number p_{edge} which determines the probability that two given nodes of the graph have an edge between them. The graph generation is based on the well-known fact that every directed acyclic graph has at least one topological ordering. Our algorithm follows then the next procedure: it numbers the nodes of the graph, puts them in order and, for every pair v_1 and v_2 such that $v_1 < v_2$ in the ordering, it builds a directed edge (v_1, v_2) with probability p_{edge}. This way the graph is guaranteed not to have cycles.

Second, Figure A.2 shows the algorithm used to randomly generate the unweighted NMC games with identity delay functions. This algorithm takes as its input a pair of integer numbers *max_nodes* i *max_players* which determine the maximum number of nodes in the graph and players in the game, respectively. The functioning of the algorithm is next described. First, it

```
Algorithm DAGGeneration
  input  : A positive integer number $n\_nodes$
  input  : A positive real number $p_{\text{edge}} \in [0,1]$
  output: A directed acyclic graph $G$
1 begin
2     $G :=$ emptyGraph($n\_nodes$)
3     for all $i \in \{1, \ldots, n\_nodes\}$ do
4         for all $j \in \{1, \ldots, n\_nodes\}$ do
5             if randomRealNumber(0,1) $< p_{edge}$ then
6                 $G :=$ addEdge($G$, $i$, $j$)
7     return $G$
8 end
```

Figure A.1: Directed acyclic graph generation.

randomly chooses a number $n\_nodes \leq max\_nodes$ of nodes and a number $n < max\_players$ of players. Second, it generates a random directed acyclic graph and chooses randomly the pair of origin and destination nodes (s, t) of all players. If there is at least one $(s - t)$ path, then the generated game is returned. Otherwise, the process goes back to the point where the graph is generated.

Algorithm `GameGeneration`

```
input  : A positive integer number $max\_nodes$
input  : A positive integer number $max\_players$
output: An unweighted single-commodity NMC game with identity
        delay functions $\Gamma = (N, G, (s,t), (id)_{e \in E(G)})$
1  begin
2      $n\_nodes$ := randomIntegerNumber(3, $max\_nodes$)
3      $n$ := randomIntegerNumber(2, $max\_players$)
4      $p$ := randomRealNumber(0,1)
5      $graph\_ok$ := fals
6      do
7          $G$ := DAGGeneration($n\_nodes$, $p$)
8          if checkConnectivity($G$, 0, $n\_nodes - 1$) then
9              $graph\_ok$ := True
10     while $\neg graph\_ok$
11     $\Gamma$ := createGame($n$, $G$, (0, $n\_nodes - 1$), $(id)_{e \in E(G)}$)
12     return $\Gamma$
13 end
```

Figure A.2: Generation of an unweighted single-commodity NMC game with identity delay functions.