ÉCOLE
POLYTECHNIQUE
ParisTech

LIX

PhD Thesis

# Pure Equilibria: Existence and Inefficiency & Online Auction

NGUYEN KIM Thang

# PURE EQUILIBRIA: EXISTENCE AND INEFFICIENCY & ONLINE AUCTION

Nguyen Kim Thang

Ecole Polytechnique, 2009.

Every atom in universe exists in equilibrium of different physical forces. Everyone in society lives in equilibrium of their work and family, of their enthusiasm and reality. Equilibrium is one of the main notions in Game Theory — the domain studies behaviors of entities according to their own interests. The conflicting interests, the lack of coordination and regulation may not lead a game to a pure equilibrium; even if pure equilibria exist, the result of local optimization of rational players in general does not have any type of global property.

Our contributions in equilibria are twofold: we study the existence of a pure Nash equilibrium and analyze the inefficiency of equilibria in games. To prove the existence of equilibria, we use extensively the potential argument in which we figure out potential functions according to different dynamics in games. Moreover, for games which does not necessarily admit a pure equilibrium, we present an useful technique in settling the complexity of deciding whether the games possess an equilibrium.

To quantify the loss caused by selfish behaviors regarding to a social objective function of a game, we study the two well-known measures: the price of anarchy which is defined as the worst-case ratio between the social objective value of an equilibrium and the optimum; the price of stability which is defined as the worst-case ratio between the best social objective value among all equilibria and the optimum. Moreover, we introduce and analyze the social cost discrepancy — the worst-case ratio between the social objective values of two equilibria — which compares the quality of different outcomes and measures the degree of choice in a game knowing that the optimum is not necessarily an outcome of the game. We provide upper and lower bounds for these measures in different games.

As long as games does not generally yield outcome as expected, it is necessary to design a mechanism that interacts with players so that their self-interested behaviors lead to a desirable outcome. We are interested in coordination mechanisms and online mechanism designs in which the underlying data is unknown. Moreover, in online mechanism design, the decision must be made without knowledge of future information in the sense of online algorithms. We give an optimal coordination mechanism for scheduling games and optimal online mechanisms for an online auction problem where the auctioneer needs to efficiently allocates perishable items and compute the payment for single-minded bidders.

# Bibliographical Sketch

Nguyen Kim Thang was born on September 20th, 1982 in Haiphong, Vietnam. He received an Engineer Diploma and a Master in Computer Science from Ecole Polytechnique in June 2006 and expects to receive a Ph.D. in Computer Science from Ecole Polytechnique in June 2009.

# Acknowledgement

First and foremost, I am deeply indebted to my parents and my whole family for their love, their understanding, their explicit and implicit encouragement and support at any step of my life.

Besides many abilities, psychology plays crucial role in doing research. I have been very lucky to learn and work with my advisor Christoph Dürr. I am very indebted to him; his insights and guidance have been invaluable to me. I have learned a lot from his advices on technical details, on talks to his computer skills. Psychologically, we understand and work well together. Working with Christoph is interesting with enjoyable discussions.

I thank people in my group Algorithms and Optimization, especially Claus Gwiggner, Mathilde Hurand, Florian Richoux, Julien Robert with whom I have shared the same office. Their humors, their characters have kept lively our lunch discussions by some never ending topics.

I would like to thank to Ecole Polytechnique with all staff members. I have spent 7 years, including my undergraduate, in Polytechnique with a lot of beautiful memories. My background and skills are mainly due to this school.

Finally, I am grateful to my friends in Polytechnique, especially vietnamese ones. I had a great environment and many funny stories with them.

# Contents

# Chapter 1

# Introduction

What route do you choose to go to work every morning? The one which takes as small amount of time as possible. Which service of telephone mobile are you looking for? The one with high quality and a reasonable price. Where does an entrepreneur decide to open a new facility in competition with others? The place attracts as many potential clients as possible. In fact, people and entities in society, each with its own information and interests, behave in *rational* manners. *Game theory* is by far the most developed theory studying such interactions. Game theory has influenced many fields, including economics, political science, biology and many others. Recently, the presence of game theory in theoretical computer science, especially in algorithms, is very remarkable.

*Theoretical computer science* studies optimization problems and seeks to optimal solutions, efficient computing, impossibility results, lower and upper bounds and so on. Inspired by the Internet as a source of concrete and immense applications, theoretical computer science would find, formulate novel problems, design and analyze algorithms for novel goals. Game theory, as a deep theory studying interactions of rational entities, plays an important role in the process of formulating problems and goals as well as providing tools and giving a fresh look at different issues.

*Algorithmic game theory* is a research field on the interface between game theory and theoretical computer science. The area has greatly exploded over the past decade and now contains many branches with different research trends [59], from computing Nash equilibrium in games and equilibrium of markets to quantifying the inefficiency of equilibria, from algorithmic mechanism design to sponsored search, prediction market, name for a few. In this thesis, we are mainly interested in the existence of pure equilibria in finite games and their inefficiency as well as in online mechanism design, particular in online auction.

## 1.1 Equilibrium and Existence

### 1.1.1 Finite Games and Equilibrium

A finite game consists of a triplet $\langle \mathcal{N}, \mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \mathcal{S}_n, u = (u_1, \ldots, u_n) : \mathcal{S} \rightarrow \mathbb{R}^n \rangle$ where

- $\mathcal{N}$ is the set of $n$ *players* $\{1, 2, \ldots, n\}$.

- $\mathcal{S}_i$ is the *set of finite pure strategies* of player $i$ for all $i \in \mathcal{N}$.

- A *pure strategy profile* of the game is a vector $s = (s_1, s_2, \ldots, s_n) \in \mathcal{S}$ where $s_i \in \mathcal{S}_i$ for $1 \leq i \leq n$.
- $u_i$ is the *utility* for player $i$ for every pure strategy profiles used by all players, i.e. $u_i : \mathcal{S} \to \mathbb{R}$.

To play the game, each player $i$ may choose a pure strategy $s_i \in \mathcal{S}_i$ or a *mixed* strategy – a probability distribution over pure strategies. A mixed strategy will be denoted $\sigma_i$ and the set of all possible mixed strategies for player $i$ will be denoted by $\Sigma_i$.

A *mixed strategy profile* is a vector $\sigma \in \Sigma = \prod_{i=1}^{n} \Sigma_i$. For each vector $s \in \mathcal{S}$ ($\sigma \in \Sigma$), we denote $s_{-i}$ ($\sigma_{-i}$) the $(n-1)$-dimensional vector of the strategies played by all other players. A pure strategy profile $s$ can be denoted as $(s_i, s_{-i})$ and similarly, $\sigma = (\sigma_i, \sigma_{-i})$. Suppose player $i$ uses mixed strategy $\sigma_i$ which specifies playing pure strategy $s_i \in \mathcal{S}_i$ with probability $p_i(s_i)$. Then the utility of player $i$ in strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ is:

$$u_i(\sigma_i, \sigma_{-i}) = \sum_{s_1 \in \mathcal{S}_1} \sum_{s_2 \in \mathcal{S}_2} \ldots \sum_{s_n \in \mathcal{S}_n} p_1(s_1) p_2(s_2) \ldots p_n(s_n) \cdot u_i(s_1, s_2 \ldots, s_n)$$

**Definition 1.1.** A pure strategy profile $s^*$ is said to be *pure Nash equilibrium* if for all players $i$ and each alternative $s_i \in \mathcal{S}_i$, we have:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$$

Similarly, a mixed strategy profile $\sigma^*$ is a *mixed Nash equilibrium* if for all players $i$

$$\sigma_i^* \in \arg\max_{\sigma_i \in \Sigma_i} u_i(\sigma_i, \sigma_{-i}^*)$$

In other words, a Nash equilibrium is a strategy profile in which no player has an incentive to unilaterally change her strategy in order to get a better utility – all players are *happy* in an equilibrium. Nash [58] proved in 1951 that every game with a finite number of players, each having a finite set of strategies, has a mixed Nash equilibrium.

**Theorem 1.2** ([58]). *Any game with a finite set of players and finite set of strategies has a mixed Nash equilibrium.*

However, a finite game does not necessarily possess a pure Nash equilibrium. This is illustrated by the following simple example.

**Example 1.3 (Matching Pennies).** Two players each place a penny on a table, either "head up" (strategy $H$) or "tail up" (strategy $T$). If the pennies match, player 1 wins (the pennies); if the pennies differ, the player 2 wins (the pennies). The game is modeled in the following table where the winner has utility 1 and the loser has utility -1.

Player 1

| | | $H$ | $T$ |
|---|---|---|---|
| Player 2 | $H$ | 1, -1 | -1, 1 |
| | $T$ | -1, 1 | 1, -1 |

In this game, there exists a mixed equilibrium in which two players play the same strategy $(1/2, 1/2)$, meaning choose strategies $H$ and $T$ each with equal probability. However, the game does not possess a pure equilibrium.

### 1.1.2 Cooperative Games and Strong Equilibrium

The games we described so far are non-cooperative games — in these games, not only players act selfishly but also they do not coordinate themselves in groups. *Cooperative games* model situations where groups of players coordinate their strategies. In a cooperative game we assume that some group $A$ of players can change their strategy jointly in such a way that every players in $A$ benefit. We are interested in games with *nontransferable utility*: no player with increased utility, by collaborating in a group, has the ability to compensate some other player with decreased utility in her cooperation. Hence, the condition for a coalition is that the utility of each player in the coalition is increasing. Aumann [6] introduced the concept of equilibrium which is appropriate in the context of cooperation game theory.

**Definition 1.4** ([6])**.** A pure strategy profile $s^*$ is a *pure strong Nash equilibrium* if for all subset $A$ of players, for any joint strategy $s_A$ of players in A ($s_A \in \prod_{i \in A} \mathcal{S}_i$), there exists at least one player $i \in A$ such that

$$u_i(s_A, s^*_{-A}) \leq u_i(s^*_A, s^*_{-A})$$

where $s^*_{-A}$ denotes the strategy profile $s^*$ restricted to players not in $A$.

A mixed strong Nash equilibrium can be defined similarly. In the thesis, we are only interested in pure strong Nash equilibrium, therefore from now we omit the word "pure" and use the term "strong equilibrium" instead of "pure strong Nash equilibrium".

The notion of strong equilibrium is stronger then the one of pure Nash equilibrium. Intuitively, a strong equilibrium is a strategy profile in which no group of players can change jointly their strategies in order to (strictly) increase all their utilities. In the other words, a strong equilibrium is resilient not only to deviation of individual player but also resilient to deviation of any group of players in the game. In a game, the set of strong equilibrium is a subset of pure Nash equilibrium and the well-known Prisoners' Dilemma shows that this relation is strict.

**Example 1.5** (Prisoners' Dilemma)**.** Two crooks are being questioned by the police in connection with a crime. They are held in separate cells and cannot communicate to each other. Without a confession, the police only have enough evidence to convict the two crooks on a smaller charge. The police makes the following offer to both prisoners: if one confesses (strategy $C$) that both committed the crime, then the confessor will be set free and the other will spend 5 years in jail (4 for the crime and 1 for obstructing justice); if both confess, then they will each get the 4-year sentence; if neither confess (stay silence – strategy $S$), then they will each spend 2 years in jail for the minor offense. The game is modeled as the following table.

|  |  | Prisoner 1 |  |
|---|---|---|---|
|  |  | $S$ | $C$ |
| Prisoner 2 | $S$ | -2, -2 | -5, 0 |
|  | $C$ | 0, -5 | -4, -4 |

The game has a single pure Nash equilibrium that both confess. However, this is not a strong equilibrium since two players can cooperate by staying silence and each gets the 2-year sentence. But, the latter is not an equilibrium – one of them can change the strategy to be free.

### 1.1.3 Potentials and Pure Equilibrium Existence

As discussed earlier, existence of pure Nash equilibria is not an universal property of games; some games always admit a pure equilibrium, others do not. A natural question is that given a game, whether it possesses pure equilibria. This kind of question, in case of positive answer, is mostly settled by using *potential argument.*

Consider a dynamical system with fixed points. To find a fixed point, while only local views of the whole picture are available, the most natural way in general is to choose some dynamical flow of the system to follow. The potential argument is based on this idea: to prove the existence of an equilibrium (a fixed point), follow a chosen dynamic (flow) which guarantees by a potential function that the process will end at an equilibrium (a fixed point). The main issue is that what the right dynamic to follow is.

Consider a strategy profile $s$ and a player $i$ using the strategy $s_i$ with utility $u_i(s)$. Changing the strategy $s_i$ to some other strategy $s'_i \in \mathcal{S}_i$, the player can change her utility to $u_i(s'_i, s_{-i})$, assuming that all other players stick to their strategies $s_{-i}$. We say that a change from strategy $s_i$ to $s'_i$ is a *better response* for player $i$ if $u_i(s'_i, s_{-i}) > u_i(s)$ and *best response* if $s'_i$ maximizes the utility of player $i$, i.e. $s'_i = \arg\max_{t \in \mathcal{S}_i} u_i(t, s_{-i})$. A player is *happy* if she is playing her best response, otherwise she is *unhappy.*

The *best-response dynamic (better-response dynamic)* is the process of repeatedly choosing an arbitrary unhappy player, and change it to an arbitrary best (better) response. Playing a game by repeatedly allowing some player to make a better or a best response move is perhaps the most natural game play.

**Definition 1.6.** Consider a domain $\Omega$ with a total order $\succ$. A game $G = (\mathcal{N}, \mathcal{S}, u : \mathcal{S} \to \mathbb{R}^n)$ is a *potential* game if there exists a function $\Phi : \mathcal{S} \to \Omega$ such that for any strategy profile $s \in \mathcal{S}$, for any player $i \in \mathcal{N}$ and for any strategy $s'_i \in \mathcal{S}_i$, we have:

$$\Phi(s_i, s_{-i}) \succ \Phi(s'_i, s_{-i}) \qquad \text{if and only if} \qquad u_i(s_i, s_{-i}) < u_i(s'_i, s_{-i}).$$

Informally, a potential game admits a function which strictly decreases when some player takes a better response. A finite potential game always admit an equilibrium. It is straightforward since no player has an incentive to change the strategy in the strategy profile where $\Phi$ is minimized ($\Phi$ can be minimized since $\mathcal{S}$ is finite, so is $\Phi(\mathcal{S})$). The class of potential games, introduced by Monderer and Shapley [56], is one of most important classes of games in which equilibria are guaranteed by potential functions.

The existence of pure Nash equilibrium is typically shown by using potential argument, in particular with the better-response or best-response dynamic. Intuitively, in potential game, one can obtain an equilibrium by following any flow of the dynamical system while in best-response potential game, one reaches an equilibrium by following some specific flow. The main difficulty is to find the right dynamic and the right potential function – the process of looking for a global invariant with only local views. However, the existence of potential function is not the sufficient condition for the existence of equilibrium. If a game possesses a potential function then it admits pure Nash equilibria. Nevertheless, the existence of equilibrium does not necessarily induce potential function.

## 1.2 Inefficiency of Equilibria

How good is an equilibrium? Consider the Prisonners' Dilemma (Example 1.5). Both players suffer a 4-year sentence in the unique Nash equilibrium of the game, while they both could get

2-year sentence with coordination. An observation is that the equilibrium is strictly Pareto inefficient, in the sense that there exists another strategy profile in which all players benefit.

In order to measure the quality of equilibria, first we need to posit an *objective function*, defined on the strategy profiles of the games, that quantitatively express the *social good* or *social cost* of a profile. The quality of an equilibrium of a game is quantified with respect to a choice of objective function and a choice of equilibrium concept of the game. Two prominent objective functions are the *utilitarian* and *egalitarian* functions, defined as the sum of the players' costs and the maximum player cost, respectively. The equilibrium in Prisonners' Dilemma is not optimum in neither of these objective functions.

The most popular measures of the inefficiency of equilibria are the *price of anarchy (PoA)* and the *price of stability (PoS)*. The *price of anarchy* is defined as the ratio between the *worst* objective value of an equilibrium of the game and that of an optimal outcome. The concept of the price of anarchy is inspired by the worst-case approach in theoretical computer science. A game with price of anarchy close to 1 means all equilibria in the game are good approximations of an optimal outcome; moreover, the game behaves smoothly as well as in case of (central or dictatorial) control over players' actions.

The game with high price of anarchy does not mean that all equilibria of the game are inefficient. The *price of stability* is a measure of inefficiency which differentiate games in which *all* equilibria are inefficient and games in which *some* equilibrium is inefficient. Precisely, the price of stability is defined as the ratio between the *best* objective value of an equilibrium of the game and that of an optimal outcome. The game with price of stability close to 1 means there is at least a "good" optimum. This simple argument has an important interpretation in many games, such as network games: if we envision the outcome as being initially designed by a central authority for subsequent use by selfish players, then the best equilibrium, which is close to the optimum, is an obvious solution to propose.

## 1.3 Online Mechanism Design

### 1.3.1 Online Algorithm

Online problems are optimization problems dealing with dynamic setting, in which the input is received in an online manner — the input is in form of sequence of requests that are revealed little by little — and in which the output must be made irrevocably with partial or without knowledge about the future. For example, problems such as paging, telephone circuit switching, investment planning and so on are intrinsically online. The issue in online algorithms is that each online output — decisions made with limited information — influences the cost of the overall solution.

Given an input $I$, an algorithm $\mathsf{ALG}$ for an online problem computes a feasible output with cost $\mathsf{ALG}(I)$. An *(offline) optimal algorithm* $\mathsf{OPT}$ is such that it knows the input $I$ in advance and the cost induced by $\mathsf{OPT}$ is optimum among costs induced by all other algorithms. An algorithm for an online problem is typically quantified by its *competitive ratio*.

**Definition 1.7.** For an optimization problem $\mathcal{P}$, an online algorithm $\mathsf{ALG}$ is *c-competitive* where $c \geq 1$ if there is a constant $\alpha$ such that for all finite input sequence $I$,

$$\begin{aligned} \mathsf{ALG}(I) &\leq c \cdot \mathsf{OPT}(I) + \alpha && \text{if } \mathcal{P} \text{ is a minimization problem,} \\ c \cdot \mathsf{ALG}(I) + \alpha &\geq \mathsf{OPT}(I) && \text{if } \mathcal{P} \text{ is a maximization problem.} \end{aligned}$$

The most convenient way to view the problem of analyzing online algorithms and their competitive ratio is to view the problem as the game between an *online player* and a malicious *adversary*. The online player runs an online algorithm on an input that is created by the adversary. The adversary, based on the knowledge of the algorithm used by the online player, constructs the worst possible input so as to maximize the competitive ratio. The adversary seeks to make the input costly to the online player but inexpensive for the optimal offline player. In the thesis, we are interested in *deterministic* online algorithms which concern to *deterministic* adversary. The deterministic adversary may choose the next request based on the online algorithm's action so far, but he pays the optimal offline cost to serve the whole request sequence. The models of *randomized* adversary can be found [13]

### 1.3.2 Algorithmic Mechanism Design

The goal of algorithmic mechanism design is to design an *efficient* mechanism, in which the underlying data is unknown, that interacts with rational participants such that *self-interested behavior of participants yields a desirable outcome.*

Generally, consider a feasible region $\Omega$, $n$ participants each with a private objective function $v_i : \Omega \to \mathbb{R}$ and a designer with an objective function $f : w \mapsto f(v_1(w), v_2(w), \ldots, v_n(w))$ where $w \in \Omega$. In optimization, the goal is that given all $v_i(\cdot)$, optimize the objective function $f$ over $\Omega$. The purpose of algorithmic mechanism design is harder: simultaneously *determine the private $v_i$'s* and optimize function $f$.

*Auction* design is the most obvious motivation for the field Mechanism Design. We confine our discussion to auction design. Informally, there are some available goods in an auction. The auctioneer first collects all bids from bidders and then decide who receive the goods and how much the winning bidders have to pay. Formally, an auction consists of two algorithms: an *allocation algorithm* and a *payment algorithm* to determine which customers receive a service and how much each one has to pay. An auction $M$ first asks participant $i$, now called *bidder* $i$, for a bid function $b_i(\cdot)$, desirably identical to the private objective function $v_i(\cdot)$; $M$ then invokes an allocation algorithm $x(b_1, \ldots, b_n)$ and an payment algorithm $\pi(b_1, \ldots, b_n)$ to determine an outcome (the set of winners) $w$ and payments $p_1, \ldots, p_n$, respectively. The *utility* of player $i$ is $u_i = v_i(w) - p_i$. By their self-interests, bidders may misreport their bid function in order to improve their utility.

**Definition 1.8.** An auction is *truthful (incentive-compatible)* if : for every bidder $i$ and every set of bids by the other players, player $i$ maximizes her utility $u_i$ by submitting her true private objective function: $b_i \equiv v_i$.

**Example 1.9** (Vickrey Auction)**.** Consider an auction for selling a single good. Each bidder $i$ has a private willingness-to-pay $v_i$ and submits a bid $b_i$ to the auctioneer. In Vickrey auction, the allocation algorithm picks the highest bidder as the winner; and the payment algorithm charges a payment the second-highest bid for the winner and 0 for the other bidders.

This auction is truthful. Moreover, the Vickrey auction maximize the *social welfare*, in the sense that the good is allocated to the one with the highest value $v_i$.

**Single-Parameter Mechanism Design** A mechanism design problem is *single-parameter* if all outcomes are real $n$-vectors, i.e. $w = (w_1, \ldots, w_n)$ or in the other words $\Omega = \mathbb{R}^n$, and the private objective function of each player $i$ has form $v_i(w) = t_i w_i$ for a private number $t_i$. Intuitively, $w_i$ and $t_i$ may be thought of as the quantity received and the value-per-unit of a good to player $i$, respectively. Single-parameter problems is an important special class of mechanism design and in the thesis, we only concentrate on this class.

**Definition 1.10.** An algorithm for a single-parameter problem is *monotone* if increasing the value of $t_i$ can only increase the value $w_i$ in the outcome $w$ or equivalently, a winner still wins if she increases her bid.

**Definition 1.11.** Given the bids of all bidders $b = (b_1, \ldots, b_n)$, the *critical value* $v_i^c$ of player $i$ is defined as:

$$v_i^c = \begin{cases} \min b'_i & \text{such that } i \text{ still wins with } b' = (b_1, \ldots, b_{i-1}, b'_i, b_{i+1}, \ldots, b_n), \\ \infty & \text{if no such } b'_i \text{ exists} \end{cases}$$

Intuitively, the critical value of $i$ is the smallest possible value of bid that $i$ can win by submitting this bid. Truthful mechanism for single-parameter domain is well characterized by monotonicity.

**Proposition 1.12** ([57])**.** *In the single-parameter domain, a deterministic mechanism is truthful if and only if its allocation algorithm is monotone and the payment algorithm collects a charge equal to the critical value from each winner.*

This proposition is very useful in designing truthful mechanism for single-parameter problems. Note that, Vickrey auction follows exactly the scheme in the proposition. An analogous characterization applies to randomized algorithms [4].

### 1.3.3 Online Mechanism Design

Online mechanism design extends the field of mechanism design to dynamic settings. Decisions in these settings must be made without knowledge of future information in the sense of online algorithms. Online auctions, such as Adwords of Google, Yahoo!, . . . or airline tickets markets are the most obvious and interesting motivations for this field.

Online mechanism design and mechanism design share the same difficulty in eliciting unknown private objective functions from self-interested participants. However, the dynamics of agent arrivals and departures, coupled with uncertainty about the set of feasible decisions in the future, makes the problem of online mechanism design fundamentally different from that of standard mechanism design. For example, the fact that a participant may misreport his departure time would influence the decision of designer, arises more problematic issues in learning participants' private objectives and results in worse outcome.

We are interested in the single-parameter online mechanism design in which there is no early-arrival and no late departure misreport. Now, a mechanism (an auction) asks not only the bid of participants but a more general type $\theta_i = (r_i, d_i, b_i)$ where $r_i$ and $d_i$ are the arrival and departure times, respectively.

We define a partial order $\preceq$ on types:

$$\theta_1 \preceq \theta_2 \Longleftrightarrow (r_1 \geq r_2) \wedge (d_1 \leq d_2) \wedge (b_1 \leq b_2)$$

The definitions of monotonicity and critical value are analogous.

**Definition 1.13.** An algorithm for a single-parameter problem is *monotone* a winner who reports type $\theta_i$ still wins if she report a type $\theta'_i \succeq \theta_i$.

**Definition 1.14.** Given the participants' types $\theta = (\theta_1, \dots, \theta_n)$, the *critical value* $v_i^c$ of player $i$ is defined as:

$$v_i^c = \begin{cases} \min b'_i & \text{such that } i \text{ still wins with } \theta' = (\theta_1, \dots, \theta_{i-1}, \theta'_i, \theta_{i+1}, \dots, \theta_n) \\ & \text{where } \theta'_i = (r_i, d_i, b'_i), \\ \infty & \text{if no such } b'_i \text{ exists.} \end{cases}$$

In this class, the characterization of truthful mechanism still holds.

**Proposition 1.15** ([45])**.** *In the single-parameter domain with no early-arrival and no late departure misreport, a deterministic online mechanism is truthful if and only if its allocation algorithm is monotone and the payment algorithm collects a charge equal to the critical value from each winner.*

Hence, in order to design a truthful mechanism for the problem, we only need to design a monotone efficient allocation algorithm and verify whether the critical value for any participant can be efficiently computed.

## 1.4 Our contributions

### 1.4.1 Existence of pure Nash equilibrium

We concentrate on the issue of the existence of a pure equilibrium in a game. In Section 1.1.3, we mainly summarize a technique used to answer positively this question. However, it is well-known that the existence of equilibrium is not an universal property for every games.

First, we are interested in the complexity of deciding whether a game possesses a pure Nash equilibrium. We realize in Chapter 2 that this problem is $\mathcal{NP}$-hard in many games and prove it with called *negated* gadgets. The technique is as follows. Suppose we need to show the $\mathcal{NP}$-hardness of some property of a game. First, we construct a gadget without the desired property – a negated gadget – and then embed it into a larger game which encodes a $\mathcal{NP}$-hard problem in order to prove the complexity of the desired property in a game. This technique is very efficient in proving $\mathcal{NP}$-hardness for deciding the existence of Nash equilibria, as illustrated in [29, 62]. Moreover, the technique is applied successfully not only in deciding the existence of Nash equilibria but also in other property relevant to Nash equilibria, for example, the existence of cost-sharing protocol with small price of stability in Connection Games [62].

In the positive angle, the existence of equilibrium is typically proved by using a potential argument with respect to the *best-response dynamic* — repeatedly among players who can increase their payoff, let an arbitrary player take the best strategy. In our work [29, 30], we also follow this scheme in proving the existence of equilibrium. Nevertheless, a game in which the best-response dynamic does not converge does not imply that there is no equilibrium in the game. In Chapter 4, we refine the best-response dynamic and construct a *novel* potential function with respect to this dynamic in order to prove the equilibrium existence. Informally, in a potential game, one can get an equilibrium by following an arbitrary orbit. In our proof, the dynamic follows a particular orbit and converges to an equilibrium. We believe that the dynamic together with the new potential function are useful in proving the existence of pure equilibrium for other games as well and are of independent interest.

### 1.4.2 Inefficiency of Equilibria

We study the inefficiency of equilibria in our work. Specifically, we study the price of anarchy in Scheduling Games [30] in Chapter 4. In this game, each player has a job and chooses a machine to schedule her own job such that the completion time of her job is as small as possible. Each machine, without communicating to others machines, decides a *policy* how to schedule the set of jobs assigned to it in such a way that the *social cost*, which is the makespan in the game, is as small as possible under the incentive of players. We introduce a new policy that, without any knowledge about jobs' characteristic, induces the price of anarchy as good as that of the best policy in case these policies have additional information on jobs' processing time.

Moreover, we introduce and study a new measure, called *social discrepancy* [29] which is the ratio between the worst and the best pure equilibrium. The idea is that a small social cost discrepancy guarantees that the social costs of Nash equilibria do not differ too much, and measures a degree of choice in the game. Additionally, in some settings it may be unfair to compare the cost of a Nash equilibrium with the optimal cost, which may not be attained by selfish players.

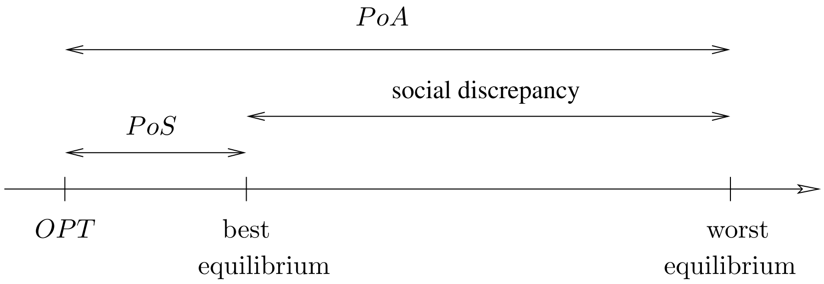

Figure 1.1: Illustration of different measures of inefficiency.

Note that the social discrepancy is not the ratio between the price of anarchy and the price of stability since each of these measures may be attained by different instances of a game. The social discrepancy is studied for Voronoi Games in Chapter 3.

### 1.4.3 Online Auction

Consider a production site that produces at the regular rate of one item per time unit. The items have to be immediately delivered to some customer, think for example at electricity which can hardly be stored. In this online scenario, every *single-minded* customer $i$ arrives at some release time $r_i \in \mathbb{N}$, and announces that he will pay $w_i \in \mathbb{R}$ euros if he gets $p_i \in \mathbb{N}^+$ items before the deadline $d_i \in \mathbb{N}$ ($i$ is *satisfied*), otherwise he pays nothing. The objective of the optimization problem is to maximize the *social welfare*, which is the total value $\sum_i w_i$ over all satisfied customers $i$.

In Chapter 5, we design truthful online mechanisms for this problem. Precisely, if all demands $p_i$ are bounded by a constant $k$, we give a truthful deterministic mechanism which is $\Theta(k/\log k)$-competitive and this a the best competitive ratio for deterministic mechanism. In particular, if all the demands are the same ($\forall i : p_i = k$) then there is a truthful constant-competitive deterministic mechanism.

In fact, we primarily study online algorithms for the optimization problem. As our algorithms are monotone, the corresponding mechanisms automatically turn out to be truthful using the framework described in Section 1.3.3.

## 1.5 Bibliographic Notes

Most of the work reported in this thesis has appeared previously in research papers [29, 62, 30, 31]. Chapter 2 appeared in [62]. Chapter 3 and Chapter 4 are joint work with Christoph Dürr and appeared in [29, 30], respectively. The results of Chapter 5 are drawn in [31].

# Chapter 2

# $\mathcal{NP}$-hardness of pure Nash equilibrium using negated gadgets

## 2.1 Introduction

Equilibrium is a key concept in Game Theory. As optimization problems seek to optimal solutions, a game looks for an equilibrium. In atomic view, pure equilibria play crucial role. For instance, on a daily day, one need to choose deterministically a route to go to work instead of choosing a route with a probability and the others with other probabilities because she cannot go, say, 1/2 on a route and 1/2 on the other route. However, in contrast to mixed equilibrium, the existence of the pure ones is not an universal property of finite games. Hence, it is important for the game designer as well as for game players to know that whether a given game admits a pure equilibrium.

In this chapter, we are interested in the complexity of some properties of pure Nash equilibria. Until now there are two methods, in general, to prove the $\mathcal{NP}$-hardness of a problem: using gadgets or using the PCP Theorem. We represent a technique based on the former, specifically on the gadgets called *negated* and polynomial-time reductions. The technique is the following. First, find a *negated* gadget which does not have the desired property. (In fact, a negated gadget is a counter-example of the property.) Next, construct a family of games which encode a $\mathcal{NP}$-hard problem, and embed the gadget into. We argue that the game has the desired property if and only if there is a solution for the instance of $\mathcal{NP}$-hard problem by using the gadget to enforce players' behaviors in such a way that the game possesses the desired property.

In this chapter, we present the technique as a framework and illustrate its application in different contexts. This technique is successfully applied in settling the complexity on the existence of pure Nash equilibrium in many games, such as the Matrix Scheduling Games, the Weighted Connection Games (in this chapter), the Voronoi Games (in the next chapter), etc. Interestingly, this technique is not only applied to the existence of equilibrium but also to other properties such as good cost-sharing mechanism in Connection Games.

**Related work** Rosenthal [60], Monderer and Shapley [56] introduced potential games which always possess a pure Nash equilibrium, for example: Congestion Games [56], Connection Games [59, chapter 19]. In these games, the existence of pure Nash equilibrium is proved by using a potential-function argument. The complexity of finding a pure equilibrium of

Congestion Games is settled in [35]. Dunkel and Schulz [28] prove that it is $\mathcal{NP}$-hard to decide if the weighted congestion games admits a Nash equilibirium.

**Organization** In Section 2.2, we introduce the Matrix Scheduling Games and prove the complexity of the existence of pure Nash equilibrium in this game. In Section 2.3, we prove the complexity of the existence of pure equilibrium in Weighted Connection Games – that answers a question in [18]. Moreover, in Connection Games we show the intractability of finding a fair cost-sharing mechanism which induces an efficient equilibrium.

## 2.2 Matrix Scheduling Games

In scheduling or planning problems, there are a set of jobs (tasks) and one need to schedule jobs in order to optimize an objective function such that the schedule satisfies different constraints of jobs, for example: the constraints on the release times and deadlines, the precedence constraints and so on. Particularly, jobs need to be scheduled and completed by using different set of resources, machines. For instance, in order to make an product, a company need to use different machines that specifically produce sub-items for the final product.

We consider the game version called Matrix Scheduling Games. In the game, there are $m$ machines, $n$ players and a load matrix $(p_{ij})_{n\times m}$ where $p_{ij} \geq 0\ \forall i,j$. Each player has a set of jobs that need to be executed. Players can complete their jobs by choosing a subset of machines on which their jobs will be executed, i.e., the strategy set $\mathcal{S}_i$ of player $i$ is a collection of subsets of machines. $p_{ij}$ represents the load contribution of player $i$ to machine $j$ if she uses this machine. Given a strategy profile $s = (s_1, s_2, \ldots, s_n) \in (\mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_n)$, the *load* $\ell_j$ of a machine $j$ depends on the set of jobs executed on the machine and is defined as:

$$\ell_j := \sum_{i:j\in s_i} p_{ij}.$$

The *cost* of a player is the sum of all loads of machines that the player uses:

$$c_i := \sum_{j\in S_i} \ell_j.$$

Players are selfish and they choose a strategy which minimize their cost. Remark that, without loss of generality, the strategy set of a player is *inclusion-free*, i.e., no player $i$ possesses two strategies $s_i$ and $s'_i$ such that $s_i \subset s'_i$ since otherwise the player always prefer use $s$ to $s'_i$ in order to get a smaller cost.

If players' strategies are restricted to singleton machines then the game becomes the well-studied Load Balancing Games. There always exists pure Nash equilibrium on that game [34].

**Proposition 2.1** ([34]). *The game always admits a pure Nash equilibrium if all players' strategies are singleton machines.*

*Proof.* We give the proof here for completeness. In case all players' strategies are singleton, each player has single job. Given a strategy profile $s$, the cost of player $i$ is the load of machine $s_i$ on which $i$ execute her job, i.e. $c_i = \ell_j$ where $j = s_i$. Consider the dominant potential function:

$$\Phi(s) := (\ell_1, n_1, \ell_2, n_2, \ldots, \ell_m, n_m)$$

where we rename machines in such that their loads are in decreasing order, i.e. $\ell_1 \geq \ell_2 \geq \ldots \geq \ell_m$ and $n_j$ is the number of machines that have the same load as machine $j$ in strategy profile $s$.

If a player $i$ has an incentive to change from machine $j$ to machine $j'$, meaning $\ell_j > \ell_{j'}+p_{ij}$, then $j' > j$ by definition of the indices after renaming. The new potential function after the move of $i$ is:

$$\Phi(s') := (\ell'_1, n'_1, \ell'_2, n'_2, \ldots, \ell'_m, n'_m)$$

where $\ell_t = \ell'_t$ and $n_t = n'_t$ for all $t < j$ since there is no change on these machines.

We have that $\ell'_j$ is bounded above by $\max\{\ell_{j+1}, \ell_{j'} + p_{ij}, \ell_j - p_{ij}\}$. If $\ell_j > \ell_{j+1}$ then $\ell'_j < \ell_j$. Otherwise, we have $\ell_j = \ell'_j$ but $n'_j = n_j - 1 < n_j$. Hence, the potential function strictly lexicographically decreases, hence follows the claim. □

However, without assumption on players' strategy sets, the game does not necessarily possess an equilibrium.

**Fact 2.2.** *There exists a matrix scheduling game in which there is no Nash equilibrium.*

*Proof.* We describe the game with no Nash equilibrium. There are 4 machines and 3 players, each one has two strategies. The strategy sets of players 1, 2 and 3 are $\mathcal{S}_1 = \{s_1^1 = \{1,3\}; s_1^2 = \{4\}\}, \mathcal{S}_2 = \{s_2^1 = \{1\}; s_2^2 = \{2\}\}$ and $\mathcal{S}_3 = \{s_3^1 = \{2\}; s_3^2 = \{3\}\}$, respectively. The load matrix is given as in Figure 2.1.

| | machine 1 | machine 2 | machine 3 | machine 4 |
|---|---|---|---|---|
| player 1 | 2 | $\infty$ | 10 | 15 |
| player 2 | 5 | 4 | $\infty$ | $\infty$ |
| player 3 | $\infty$ | 4 | 1 | $\infty$ |

Figure 2.1: A matrix scheduling game with no Nash equilibrium.

We prove that there is no Nash equilibrium in the game by verifying all $2^3$ strategy profils. In Figure 2.2, the first three columns represent the strategies chosen by the players. The last column shows which player is unhappy and how she can decrease her cost. For example, the first row represents a strategy profile in which all players choose their first-strategy, player 1 has cost 17 and she has an incentive to change to her second-strategy, which induces a cost 15.

| player 1 | player 2 | player 3 | Best reponse |
|---|---|---|---|
| $s_1^1$ | $s_2^1$ | $s_3^1$ | $1 : s_1^1 \rightarrow s_1^2\ (17 \rightarrow 15)$ |
| $s_1^1$ | $s_2^1$ | $s_3^2$ | $3 : s_3^2 \rightarrow s_3^1\ (11 \rightarrow 4)$ |
| $s_1^1$ | $s_2^2$ | $s_3^1$ | $2 : s_2^2 \rightarrow s_2^1\ (8 \rightarrow 7)$ |
| $s_1^1$ | $s_2^2$ | $s_3^2$ | $3 : s_3^2 \rightarrow s_3^1\ (11 \rightarrow 8)$ |
| $s_1^2$ | $s_2^1$ | $s_3^1$ | $3 : s_3^1 \rightarrow s_3^2\ (4 \rightarrow 1)$ |
| $s_1^2$ | $s_2^1$ | $s_3^2$ | $2 : s_2^1 \rightarrow s_2^2\ (5 \rightarrow 4)$ |
| $s_1^2$ | $s_2^2$ | $s_3^1$ | $3 : s_3^1 \rightarrow s_3^2\ (8 \rightarrow 1)$ |
| $s_1^2$ | $s_2^2$ | $s_3^2$ | $1 : s_1^2 \rightarrow s_1^1\ (15 \rightarrow 13)$ |

Figure 2.2: There is no Nash equilibrium.

□

Using the game from previous proof as a gadget, we prove the following theorem.

**Theorem 2.3.** *Deciding whether there exists a Nash equilibrium for a given matrix scheduling game is strongly (unary) $\mathcal{NP}$-hard.*

*Proof.* We prove binary $\mathcal{NP}$-hardness of the decision problem with a constant number of machines by a reduction from Partition [42]. To show unary $\mathcal{NP}$-hardness with an arbitrary number of machines, a reduction from 3-Partion is used. Since the proofs are basically the same, in the following, we present the reduction of the former.

In Partition, we are given $n$ positive integers $a_1, \ldots, a_n$ and the question is whether there exists a partition of these $n$ numbers into two subsets $(P_1, P_2)$ such that the sums of elements in these subsets are equal.

We construct a matrix scheduling game in which the existence of a Nash equilibrium is equivalent to the existence of a solution for an instance of Partition. Given $n$ integers $a_1, \ldots, a_n$, let $B$ be an integer such that $\sum_{t=1}^{n} a_t = 2B$. The reduction game consists of $n+6$ players. The first $n$ players encode the Partition problem, the last three players encode the gadget of Fact 2.2 and the remaining three players acts as a connection between these two groups. Each of the first $n$ players has two strategies: machine $\{1\}$ and machine $\{2\}$, the loads of job of player $i$ $(1 \leq i \leq n)$ on machines 1 and 2 are the same and equal to $a_i$. Player $n+1$ has two strategies: machine $\{2\}$ and machine $\{3\}$ with loads 0 and $B/2+\epsilon$, respectively; player $n+2$ has two strategies: machine $\{1\}$ and machine $\{3\}$ with loads 0 and $B/2+\epsilon$, respectively. Player $n+3$ has two strategies: machines $\{3,4\}$ and machine $\{8\}$ with loads as shown in Table 2.2. The last three players represent the gadget of Fact 2.2. Note that the load contribution of a player to a machine is infinity if it is not explicitly given.

| player \ machine | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | $a_1$ | $a_1$ | | | | | | |
| 2 | $a_2$ | $a_2$ | | | | | | |
| $\vdots$ | $\vdots$ | $\vdots$ | | | | | | |
| $n-1$ | $a_{n-1}$ | $a_{n-1}$ | | | | | | |
| $n$ | $a_n$ | $a_n$ | | | | | | |
| $n+1$ | | 0 | $B/2+\epsilon$ | | | | | |
| $n+2$ | 0 | | $B/2+\epsilon$ | | | | | |
| $n+3$ | | | $B/2$ | 3 | | | | $B/2+4$ |
| $n+4$ | | | | 2 | | 10 | 15 | |
| $n+5$ | | | | 5 | 4 | | | |
| $n+6$ | | | | | 4 | 1 | | |

Table 2.1: The load matrix of the reduction game. ($p_{ij} = \infty$ if it is not explicitly given for all $i, j$.)

Fix an instance of Partition, we show that if there is a solution $(P_1, P_2)$ for the instance then there is a Nash equilibrium for this game. Consider a strategy profile in which player $i$ $(1 \leq i \leq n)$ uses machine $b$ for $b \in \{1,2\}$ such that $i \in P_b$, player $n+1$ uses machine 2, player $n+2$ uses machine $m1$, player $n+3$ uses machines $\{3,4\}$ and three others play their second-strategy (defined in Fact 2.2). It is straightforward that to verify that no one has an incentive to change her current strategy, hence this strategy profile is a Nash equilibrium. In this equilibrium, all $n$ first players' cost are $B$.

Inversely, suppose that there is a Nash equilibrium for this game. In this equilibrium, player $n+3$ must use machines $\{m_3, m_4\}$ since otherwise, by Fact 2.2, the strategy profile is not an equilibrium as it contains an unstable sub-game by the last three players. Hence, both players $n+1$ and $n+2$ play their first strategy since otherwise player $n+3$ will move. Therefore the costs of $n+1$ and $n+2$ are at most $B$. Moreover, $\sum_{t=1}^{n} a_t = 2B$ thus their costs are exactly $B$. In other words, all $n$ first players form a partition such that the sum in two subsets are the same and therefore we obtain a solution for the PARTITION instance. □

## 2.3 Connection Games

Consider transportation companies which will open new service to go from a city to another. Theses cities may be different for different companies. They need to build a new roads on a given plan of possible roads that can be constructed in the country. The goal of each company is to open its service and minimize its cost of the construction. Similarly, network providers need to buy or rent existing optic fibers to satisfy the clients' demands of communication between two locations. They do it together but each one takes a decision to get a cost as small as possible.

Connection Game models such situation. In the game, we are given a directed graph $G = (V, E)$ with nonnegative edge costs $c_e$ for all edges $e \in E$. There are $n$ players, each player $i$ has a specified source node $s_i$ and sink $t_i$. Player $i$'s goal is to build a network together with other players in order to connect her terminals $s_i$ and $t_i$ while paying as little as possible to do so. A strategy of player $i$ is a path $P_i$ from $s_i$ to $t_i$ in $G$. Given a strategy $P_i$ for player $i$, the constructed network is $\cup_i P_i$, which induces the *social cost* $\sum_{e \in \cup_i P_i} c_e$ that is fully paid by players in the game.

A cost-sharing mechanism can be viewed as the underlying mechanism that determines how much a network serving several participants will cost to each of them. Specifically, if player $i$ chooses a path $P_i$ then it will be charge a cost $c_i$ as a function of all the paths $P_1, P_2, \ldots, P_n$. The goal of a cost-sharing mechanism is to lead the game to efficient equilibria. Although there are in principle many possible cost-sharing mechanisms, research in this area has converged on a few mechanisms with good theoretical and empirical behavior in which, the most natural one is the *Shapley cost-sharing mechanism.*

Shapley cost-sharing mechanism splits the cost of an edge evenly among all players using this edge. Formally, given a strategy profile $S$, if $n_e$ denotes the number of players whose path contains edge $e$ then $e$ assigns a cost $c_e/n_e$ to each player using $e$. Thus, the total cost of player $i$ in strategy profile $S$ is:

$$c_i(S) := \sum_{e \in P_i} c_e/n_e.$$

The Connection Game using the Shapley cost-sharing mechanism is well studied in [2] (see also [59, chapter 19]). This game always possesses a Nash equilibrium. The inefficiency of the constructed network is measured by the price of anarchy (PoA) and the price of stability (PoS).

### 2.3.1 Weighted Connection Games

The Weighted Connection Game is a generalization of the Connection Games. In the former, each player $i$ has additionally a weight $w_i$ and she needs to carry the weight from her source to her sink. Consider the *weighted Shapley cost-sharing mechanism* that splits the cost of an edge proportionally to the players' weight. Formally, given a strategy profile $S$, let $W_e$ be the total weight of players whose path contains $e$, i.e., $W_e = \sum_{i:e \in P_i} w_i$, the cost of player $i$ on edge $e$ is $c_e \cdot w_i / W_e$. The total cost of player $i$ in strategy profile $S$ is:

$$c_i(S) = \sum_{e:e \in P_i} c_e \cdot w_i / W_e.$$

As showed in [20], there does not necessarily exist an equilibrium for Weighted Connection Games. We use the counterexample from [20] as the gadget to prove the complexity of the existence of Nash equilibria in the game. For completeness, we present here the gadget.

**Lemma 2.4** ([20])**.** *There is a 3-player weighted connection game using the weighted Shapley cost-sharing mechanism that admits no Nash equilibrium.*

*Proof.* Let $w > 1$ be an arbitrary number and $\epsilon > 0$ be much smaller than $1/w^3$. Consider the network in Figure 2.3 with players 1,2 and 3 which have weight $w^2, 1$ and $w$, respectively. The costs of edges are given in Figure 2.4 and are chosen in such a way that they satisfy the following inequalities.

$$c_5 \cdot \frac{w^2}{w^2+1} + c_9 \cdot \frac{w^2}{w^2+1} > c_7 > c_5 + c_9 \cdot \frac{w^2}{w^2+w+1} \tag{2.1}$$

$$c_6 + c_9 \cdot \frac{w}{w^2+w+1} > c_8 > c_6 \cdot \frac{w}{w+1} + c_9 \cdot \frac{w}{w+1} \tag{2.2}$$

If the second player uses path $e_2 \rightarrow e_5 \rightarrow e_9$ then the third player will use path $e_8$ (by the first half of the inequality (2.2)) and the first player strategy will be $e_7$ (by the first half of the inequality (2.1)). Hence, the second player will switch to the path $e_3 \rightarrow e_6 \rightarrow e_9$ to decrease her cost.

If the second player uses path $e_3 \rightarrow e_6 \rightarrow e_9$ then the third player will use path $e_4 \rightarrow e_6 \rightarrow e_9$ (by the second half of the inequality (2.2)) and so the first player strategy will be $e_1 \rightarrow e_5 \rightarrow e_9$ (by the second half of the inequality (2.1)). But in this case, the second player will switch to the path $e_2 \rightarrow e_5 \rightarrow e_9$ and get better off.

The two cases conclude the lemma.

□

We use the network in the previous lemma as the gadget to prove the $\mathcal{NP}$-hardness of Nash equilibrium. We construct a larger weighted connection game based on a $\mathcal{NP}$-hard problem and then embed the gadget into such that the existence of a solution for an instance of the $\mathcal{NP}$-hard problem is equivalent to the existence of Nash equilibrium in the game. Part of our constructed network is inspired by the construction of Dunkel and Schulz [28] where they proved $\mathcal{NP}$-hardness of the existence of equilibria in weighted congestion game.

**Theorem 2.5.** *It is $\mathcal{NP}$-complete to decide whether a given weighted Shapley connection game admits a Nash equilibrium.*

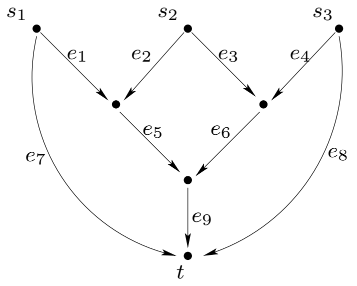

Figure 2.3: A 3-players weighted Shapley connection game with no Nash equilibrium.

| Edge | Cost |
|---|---|
| $e_1$ | $0$ |
| $e_2$ | $3\epsilon$ |
| $e_3$ | $0$ |
| $e_4$ | $0$ |
| $e_5$ | $w^3/(w^2+w+1)-\epsilon$ |
| $e_6$ | $w^3/(w^2+w+1)+\epsilon$ |
| $e_7$ | $(w^3+w^2)/(w^2+w+1)-$ $-\ \epsilon(2w^2+1)/(2w^2+2)$ |
| $e_8$ | $(w^3+w^2)/(w^2+w+1)-$ $-\ \epsilon(2w+1)/(2w+2)$ |
| $e_9$ | $1$ |

Figure 2.4: The cost of edges in Lemma 2.4

*Proof.* First, we show how to verify whether a strategy profile is an equilibrium of the game can be done in polynomial time. Consider a player $i$ while the strategies of the others are fixed. For each edge $e$ in graph $G(V, E)$, we can compute the cost $c'_e$ that player $i$ has to pay on this edge if she uses it. Then, we can construct a graph $G'(V, E)$ which is the same as $G$ except that now the cost of each edge $e$ is $c'_e$. On graph $G'$, we can compute the shortest path from $s_i$ to $t_i$ in polynomial time and verify if player $i$ is happy in the given strategy profile in graph $G$. Hence, applying the procedure for all players, we can answer the question in polynomial time.

In the following, we prove the $\mathcal{NP}$-hardness by a reduction from MONOTONE3SAT[1] [42]. Consider an instance of MONOTONE3SAT with variable set $X = \{x_1, \ldots, x_n\}$ and clause set $C = \{c_1, \ldots, c_m\}$. Each clause contains at most three literals and either all literals in a clause are negated or all are unnegated. Deciding whether there is a satisfying assignment for this instance is $\mathcal{NP}$-hard.

We construct a game such that for each literal $x \in X$, there is a *literal player* $p_x$ with weight 1, source $x$ and sink $\bar{x}$. Moreover, each clause $c \in C$ gives rise to a *clause player* $p_c$ with weight 1, source $c$ and sink $\bar{c}$. Besides, we have three additional players $p_1, p_2, p_3$ of weight $w^2, 1, w$ and terminals $(s_1, t), (s_2, t), (s_3, t)$, respectively. These three players will play the role of the gadget from Lemma 2.4. One additional player $p_4$ has weight 1 and terminals $(s_4, t_4)$. Remark that in the reduction network, all players have weight 1 except players $p_1$ and $p_3$. In our construction, $\epsilon$ is positive and arbitrarily small.

We first describe the sub-network for all players $p_x$ and $p_c$, $x \in X, c \in C$. Part of this sub-network is illustrated in Figure 2.5. For player $p_x$, there are two paths $P_x^0, P_x^1$ from $x$ to $\bar{x}$. Let $n_x := |\{c \in C | x \in c\}|$ and $n_{\bar{x}} := |\{c \in C | \bar{x} \in c\}|$. Path $P_x^1$ consists of $(2n_x + 2)$ edges and path $P_x^1$ consists of $(2n_{\bar{x}} + 2)$ edges. On each path, the cost of all odd$^{th}$ edges is 0 and that of all even$^{th}$ edges is 2 except for the last edge. If $n_x > n_{\bar{x}}$ then the cost of the last edge on path $P_x^0$ is $(n_x - n_{\bar{x}})$ and that on path $P_x^1$ is 0. Otherwise, the cost of the last edge on path $P_x^0$ is 0 and that on path $P_x^1$ is $(n_{\bar{x}} - n_x)$. Each player $p_c$ also has two paths $P_c^0, P_c^1$ from $c$ to $\bar{c}$. Path $P_c^0$ consists of two edges with cost $9/2 + \epsilon$ and 1. Path $P_c^1$ consists of seven edges and is constructed for $c = c_j$ in the order $j = 1, \ldots, m$ as follows. For a positive clause

[1] The choice of MONOTONE3SAT is driven by the simplicity in drawing the network. We can make reduction from 3SAT with exactly the same arguments.

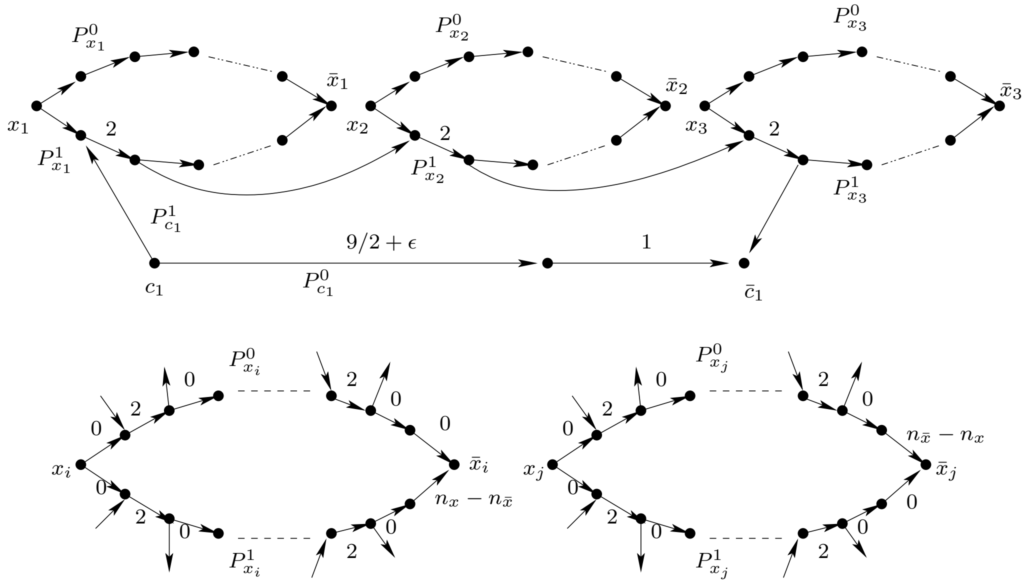

Figure 2.5: Network of players $p_x$ and $p_c$. Two paths of player $p_{c_1}$, for $c_1 = x_1 \vee x_2 \vee x_3$, are illustrated.

$c = c_j = (x_{j_1} \vee x_{j_2} \vee x_{j_3})$ with $j_1 < j_2 < j_3$, path $P^1_c$ starts with the edge connecting source $c$ to the first inner node $v_1$ on path $P^1_{x_{j_1}}$ that has only two incident edges so far. The second edge is the unique edge $(v_1, v_2)$ of path $P^1_{x_{j_1}}$ that has $v_1$ as its start vertex. The third edge connects $v_2$ to the first inner node $v_3$ on path $P^1_{x_{j_2}}$ that has only two incident edges so far. The fourth edge is the only edge $(v_3, v_4)$ on $P^1_{x_{j_2}}$ with start vertex $v_3$. Similarly, the fifth edge is the edge connecting $v_4$ to the first inner node $v_5$ of $P^1_{x_{j_3}}$ which has only two incident edges so far, followed by $(v_5, v_6)$. The last edge of $P^1_c$ connects $v_6$ to sink $\bar{c}$. For a negative clause, the construction is similar. The difference is that a positive clause concerns only paths (of literal players in the clause) of superscript 1 while a negative clause concerns only with those of superscript 0.

The second part of the network consists of the four players $p_1, p_2, p_3$ and $p_4$. First three players have a network (with edge costs) defined in Lemma 2.4. In addition, this network has an additional edge $e_{10}$ of cost 0. Player $p_4$ has two paths $P^0_4, P^1_4$ connecting her source $s_4$ and her sink $t_4$. Path $P^1_4$ consists of edge $e_8$ and an additional edge $(t, t_4)$ of cost $m - c_8/(w+1) - \epsilon$. Path $P^0_4$ shares edges of cost 1 with all paths $P^0_c, \forall c \in C$ and contains some additional edges (of cost 0) connecting those. The network with its edge costs is shown in Figure 2.6. Note that each player in our network possesses two strategies. We call a *0-path* (*1-path*, resp) of a player the path with superscript 0 (1, resp).

Given a satisfying assignment, consider the following strategy profile. Players $p_x$ ($x \in X$) use their $b$-path ($b \in \{0, 1\}$) corresponding to the value $b$ of $x$ in the solution, players $p_c$ ($c \in C$) use their 1-path, player $p_4$ uses her 1-path and players $p_1, p_2, p_3$ use paths $(e_7), (e_3 \to e_6 \to e_9), (e_{10} \to e_8)$ respectively. We argue that this strategy profile is a Nash equilibrium. Player $p_x$ has no incentive to switch her strategy because her cost stays the same (it equals $\max\{n_x, n_{\bar{x}}\}$) even if she changes the strategy (by the trick that we add a new edge of cost $|n_x - n_{\bar{x}}|$ to some paths). Player $p_c$'s cost is at most 5 since in a satisfying assignment, she shares at least an edge of cost 2 with a player $p_x, x \in X$. Observing that if using 0-path,

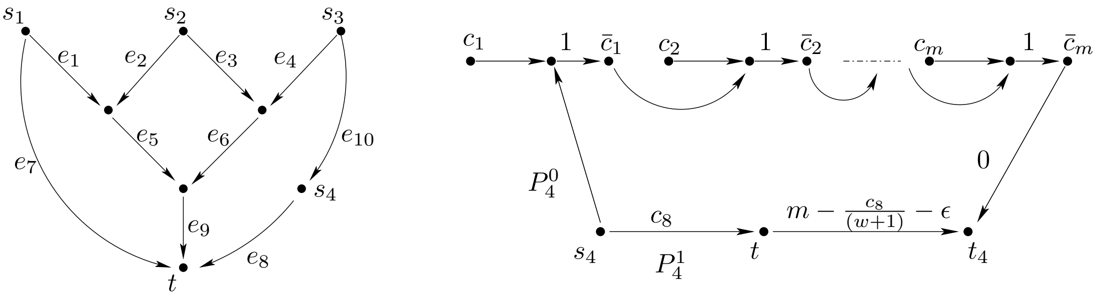

Figure 2.6: Network of players $p_1, p_2, p_3$ and $p_4$ (an edge has cost 0 if not explicitly given).

player $p_c$ may share only one edge of cost 1 with player $p_4$ so if she switches to 0-path, in the best case, her cost would be $9/2 + \epsilon + 1/2 > 5$. Hence, player $p_c$ is happy on the 1-path. The cost of player $p_4$ is $m - \epsilon$ and she is happy on her current strategy. It is easy to verify that all three players $p_1, p_2$ and $p_3$ are also happy.

Suppose there is a Nash equilibrium for this game. Hence, in this equilibrium, player $p_4$ must use her 1-path since otherwise the strategy profile is not an equilibrium (by Lemma 2.4). The cost of $p_4$ is at least $m - \epsilon$ and this happens in case $p_4$ shares edge $e_8$ with $p_3$. Therefore all players $p_c$ use her 1-path because if there is a player $p_c$ using her 0-path, player $p_4$ has an incentive to change her strategy and get a cost at most $m - 1/2$. The fact that players $p_c$ ($\forall c \in C$) play their 1-path means that, for each $c \in C$, there is at least one player $p_x$ sharing an edge with $p_c$ (otherwise, $p_c$ will change her strategy). Hence, the assignment, in which $x_i = 1$ if $p_x$ uses 1-path and $x_i = 0$ otherwise, is a satisfying assignment for the MONOTONE3SAT instance. □

### 2.3.2 Cost-Sharing mechanism for Good Equilibrium

The inefficiency of equilibria is measured by the price of anarchy (PoA) and the price of stability (PoS). In the previous subsection, we consider the Shapley cost-sharing mechanism as the allocation of the cost of the constructed network to players. Under this cost-sharing mechanism, the PoA may be as $\Omega(n)$ and the PoS may be as $\Omega(\log n)$ [2] where $n$ is the number of players in the game. A natural question is whether there exists a cost-sharing mechanism which guarantees a small inefficiency of the game. Chen, Roughgarden and Valiant [21] have studied the inefficiency of the Connection Games while considering the set of *admissible* cost-sharing mechanisms. A cost-sharing mechanism is *admissible* if it satisfies:

(i) *Budget-balance*: the cost of each edge in the constructed network is fully passed on to its users;

(ii) *Separability*: the cost shares of an edge are completely determined by the set of players that use it;

(iii) *Stability*: for every network using the cost-sharing mechanism, there is at least one Nash equilibrium.

The purpose is to design an admissible cost-sharing mechanism such that for all networks, the PoS induced from this mechanism is small, say constant.

In Connection Games where all players have the same sink, there is a cost-sharing mechanism which induces PoS = 1 for every network [3]. Nevertheless, the best Nash equilibrium is not social optimum in general case. Precisely, [21] obtained that for any admissible cost-sharing mechanism, the PoS is at least 3/2. Here, we present our proof for this lower bound.

**Lemma 2.6** ([21])**.** *There is a directed network whose PoS is at least 3/2 for any admissible cost-sharing mechanism.*

*Proof.* Consider the network illustrated in Figure 2.7 where player $p_i$ has source/sink $(s_i, t_i)$ for $1 \leq i \leq n$. The *backbone* path consists of edges $e_1, e_2, \ldots, e_n$ of cost alternating between $1 + 1/n$ and 0. Players can use the backbone path or other paths to connect her terminals. Player $p_i$ $(2 \leq i \leq n)$ has a path containing three edges of cost 1/2. we call *one-hop-source edge* and *one-hop-sink edge* the first and the last among these three edges, respectively. The optimum, where all players use the backbone path, is of cost $n + 1$. We will prove that for any admissible cost-sharing mechanism, an equilibrium of $n$ players on this network has cost at least $3(n - 1)/2 + 1$.

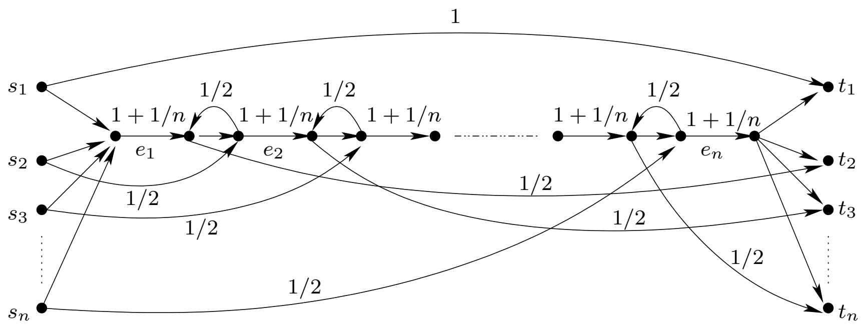

Figure 2.7: Network of $n$ players whose PoS = 3/2 (an edge has cost 0 if it is not explicitly given).

Fix an admissible cost-sharing mechanism. We claim that in an equilibrium induced by the mechanism, the first player does not use the backbone path. Assume that in an equilibrium $S$, $p_1$ uses the backbone path. Player $p_i$'s cost $(2 \leq i \leq n)$ restricted to the backbone path is at most 1/2 before edge $e_i$ and at most 1/2 after (including) edge $e_2$ since otherwise $p_i$ has incentive to use her one-hop-source edge or one-hop-sink edge. Hence, the total cost of all players $p_i$ $(2 \leq i \leq n)$ restricted to the backbone path is at most $n - 1$, so $p_1$ pays at least 2 on the backbone path. In this case, $p_1$ would switch to the one-hop path of cost 1, that contradicts to the assumption of equilibrium.

Moreover, we claim that in an equilibrium, no player entirely uses the backbone path. Suppose in an equilibrium there are $k - 1$ players (different from the first player) entirely using the backbone path, i.e., $(n - k)$ other players use at least one of their one-hop edges in order to connect their terminals. The cost restricted on the backbone path of each one in these $(n - k)$ players is at most 1 (otherwise a player, whose restricted cost is larger than 1, would change her path restricted on the backbone by an one-hop edge with or without a backward edge of cost 1/2). Therefore, the total cost shared by the $(k - 1)$ players is at least $k$. In other words, there is a player $p_i$ who strictly pays more than 1 on the backbone path. It means that she strictly pays more than 1/2 on her path before $e_i$ or after (including) $e_i$. Hence, this player can decrease her cost by switching her strategy.

Consider an equilibrium $S$ and let $R = \{j : e_j \in S\}$. If $R = \emptyset$ then in $S$ each player $p_i$ $(2 \leq i \leq n)$ pays $3/2$ and player $p_1$ pays 1. The total cost is $3/2 \cdot (n-1)+1$. If $R \neq \emptyset$, let $i$ be the smallest element in $R$. If $i > 1$ then $p_i$ uses her one-hop-source edge and no player $p_j$ uses an edge $e_j$ with $j \leq i$ on the backbone by minimality of $i$. Thus, player $p_i$ fully pays edge $e_i$. In this case, the total cost of $p_i$ is $3/2 + 1/n$ and she has an incentive to switch her strategy. Hence $i = 1$. Let $T$ be the set of players using $e_1$ in $S$ and $k := \min\{j : p_j \in T\}$. Remark that no one uses the full backbone path. Since $S$ is an equilibrium, edge $e_k$ is shared by $p_k$ and some other player of index $\ell < k$ (otherwise, $p_k$ is the only one who pays edge $e_k$ and she is not happy). Since $\ell < k$ and $p_\ell$ uses an edge $e_k$, player $p_\ell$ must use a path containing $\{e_k, e_{k+1}, \ldots, e_n\}$ (she can not use her one-hop-sink edge). That means the backbone path is entirely bought. Moreover, no one uses the backbone path, i.e., each player uses at least an edge of cost $1/2$. Hence, the total cost in $S$ is $n + 1/2 \cdot (n-1)$. □

Consider the Shapley cost-sharing mechanism. This mechanism has a good property, namely fairness — it divides evenly the cost of an edge to players using this edge. We are interested in designing admissible cost-sharing mechanisms which also possess a fairness property. We define the property $\epsilon$-fairness which is desired in a cost-sharing mechanism. Intuitively, in these mechanisms, if an edge is shared by several players, no one pays a too large fraction of the cost.

**Definition 2.7.** Given $0 < \epsilon < 1/2$, an *$\epsilon$-fair* cost-sharing mechanism is a cost-sharing mechanism in which if there are at least two players sharing an edge then no one pays more than $1 - \epsilon$ fraction of this edge cost.

Unfortunately, the following theorem shows that it is intractable to find an $\epsilon$-fair admissible cost-sharing mechanism which ensures small inefficiency of equilibria of the game. It also highlights an intuition in designing a cost-sharing mechanism for the Connection Games with good PoS: this kind of cost-sharing mechanism may not be fair.

**Theorem 2.8.** *Given a network $G(V,E)$ with edge costs $c : E \to \mathbb{Q}$ and a set of players, deciding whether there exists an $\epsilon$-fair cost-sharing mechanism such that the PoS $\leq 3/2 - \delta$ is $\mathcal{NP}$-hard, where $\delta > 0$ can be chosen arbitrarily small.*

*Proof.* Again, we reduce from Monotone3Sat. Consider an instance of Monotone3Sat with variable set $X = \{x_1, \ldots, x_n\}$ and clause set $C = \{c_1, \ldots, c_m\}$. Let $\alpha, \beta, M, k$ be parameters satisfying the following inequalities:

$$2\alpha + (1-\epsilon)\alpha < \beta < 3\alpha \tag{2.3}$$

$$(m-1)\beta + (1-\epsilon)\beta < M < M+2 < m\beta \tag{2.4}$$

$$\frac{2}{\delta}(M + m\beta + nm\alpha) < k \tag{2.5}$$

We can choose, for instance, $\alpha = \frac{2}{\epsilon}, \beta = \frac{6}{\epsilon} - 1, M = m\beta - 4, k = 2(M + m\beta + nm\alpha)/\delta$.

We will create a game which has one player $p_x$ with origin $x$ and destination $\bar{x}$ for every literal $x \in X$ and one player $p_c$ with origin $c$ and destination $\bar{c}$ for every clause $c \in C$. Besides, we have $k$ additional players who play the role of the gadget from Lemma 2.6. One more additional player $p$ has terminals $s, t$.

The network for all players $p_x$ and $p_c$ for $x \in X, c \in C$ is similar to that in Theorem 2.5. For player $p_x$, there are two paths $P_x^0, P_x^1$ from $x$ to $\bar{x}$. Let $n_x := |\{c \in C | x \in c\}|$ and $n_{\bar{x}} :=$

$|\{c \in C | \bar{x} \in c\}|$. Path $P_x^1$ consists of $(2n_x + 2)$ edges and path $P_x^1$ consists of $(2n_{\bar{x}} + 2)$ edges. On each path, the cost of all odd$^{th}$ edges is 0 and that of all even$^{th}$ edges is $\alpha$ except the last edge. If $n_x > n_{\bar{x}}$ then the cost of the last edge on path $P_x^0$ is $(n_x - n_{\bar{x}})\epsilon\alpha$ and that on path $P_x^1$ is 0. Otherwise, the cost of the last edge on path $P_x^0$ is 0 and that on path $P_x^1$ is $(n_{\bar{x}} - n_x)\epsilon\alpha$. For each player $p_c$, she also has two path $P_c^0, P_c^1$ from $c$ to $\bar{c}$. Path $P_c^0$ consists of one edge with cost $\beta$. Path $P_c^1$ consists of seven edges and are constructed for $c = c_j$ in the order $j = 1, \dots, m$ in the same manner as that in Theorem 2.5.

The second part of the network consists of $k$ players forming a gadget as described in Lemma 2.6. Additional player $p$ has two path-strategies $P_p^0, P_p^1$ connecting her terminals $s, t$. Path $P_p^0$ consists of $2m$ edges, each even$^{th}$ edge is shared with a player $p_c, c \in C$ (these edges have cost $\beta$) and each other edge has cost 0. Path $P_p^1$ consists of $2k + 1$ edges, where $2k - 1$ edges are on the backbone path of the gadget, an edge of cost $M$ connects source $s$ to the beginning point of the backbone path and an edge of cost 0 connecting the endpoint of the backbone path to sink $t$. Again, we call a *0-path* (*1-path*, resp.) of a player is her path with super index 0 (1, resp.).

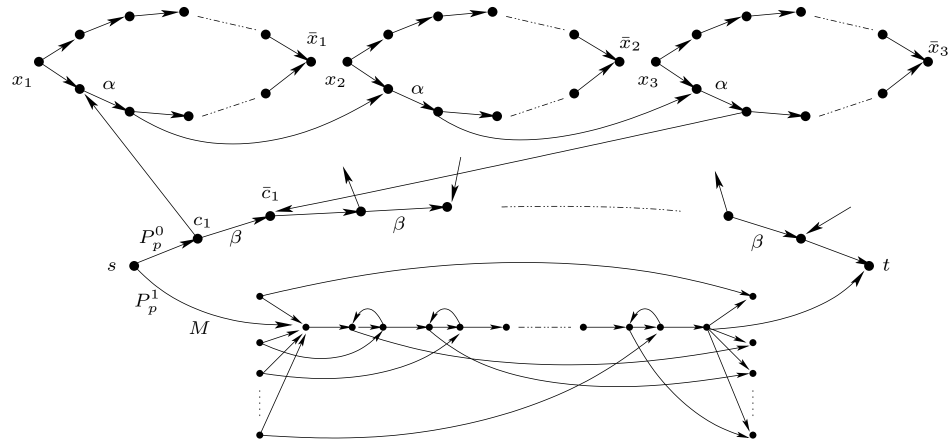

Figure 2.8: A part of network with player $p$, players on the gadget and players $p_{c_1}, p_{x_1}, p_{x_2}, p_{x_3}$ where $c_1 = (x_1 \vee x_2 \vee x_3)$.

Given a satisfying truthful assignment, we argue that the following strategy profile together with a cost-sharing mechanism is a Nash equilibrium and the cost of this equilibrium is within a factor 3/2 of $OPT$. Consider the strategy profile in which players $p_x$ $(x \in X)$ use their $b$-path $(b \in \{0, 1\})$ corresponding to the value $b$ of $x$ in the truthful assignment, players $p_c$ $(c \in C)$ use their 1-path, player $p$ uses her 1-path and players in the gadget use the backbone path. Let $\xi$ be a cost-sharing mechanism such that

(i) if a player $p_x$, $x \in X$ and a player $p_c$, $c \in C$ share an edge then $p_c$ pays an $(1 - \epsilon)$ fraction of the edge, the rest is paid by $p_x$;

(ii) in the gadget, player $p_i$ $(2 \le i \le n)$ pays 1/2 on edge $e_{i-1}$ and 1/2 on edge $e_i$, player $p_1$ pays 0 and player $p$ pays $1/2 + 1/n$ on edges $e_1, e_n$ and she pays $1/n$ on edges $e_i$ for $2 \le i \le n - 1$.

Clearly, $\xi$ is an $\epsilon$-fair admissible cost-sharing mechanism.

We show that everyone in this strategy profile is happy and bound the total cost. Player $p_x$ has no incentive to switch her strategy because her cost stays the same even if she changes her path (due to the trick that we add a new edge of cost $\epsilon(n_x - n_{\bar{x}})\alpha$ or $\epsilon(n_{\bar{x}} - n_x)\alpha$ to some paths). Player $p_c$ is happy because her current cost is at most $2\alpha + (1-\epsilon)\alpha$ which is smaller than $\beta$ – the cost she would pay if she unilaterally switches the strategy. The same holds for player $p$, since she must pay $m\beta$ if she changes strategy instead of the cost $M+2$. All players on the gadget are also happy. Observing that the optimum of the gadget is the backbone path of cost $k$ and since no one in the gadget can use a path outside, the cost of $OPT$ of the network is at least $k$. By the choice of parameters, value $k$ dominates the cost of all edges outside the gadget. Therefore the cost of the Nash equilibrium described above is smaller than $3/2k$, or in the other words, it is within a factor $3/2 - \delta$ of the $OPT$ by inequality (2.5).

Suppose there is a Nash equilibrium derived from an $\epsilon$-fair cost-sharing mechanism $\xi$ such that the cost of this equilibrium is within a factor $3/2$ of the $OPT$ in this game. Hence, in this equilibrium, player $p$ uses her 1-path and shares the backbone path of the gadget (otherwise, the cost of this equilibrium is at least $3/2k$ by Lemma 2.6 and is greater than $(3/2-\delta)OPT$ by (2.5)). It means that no player $p_c, c \in C$ uses her 0-path (one-hop path of cost $\beta$) since otherwise player $p$ will switch her strategy (by (2.4)). Therefore, on her 1-path, player $p_c$ shares some edge with some player $p_x$ (by (2.3)). For all $x \in X$, assign $x$ to the value 0 or 1 depending on 0-path or 1-path that player $p_x$ uses in the equilibrium. This assignment is satisfied because each clause $c$ ($c \in C$) has value 1. □

# Chapter 3

# Voronoi Games on Graphs

## 3.1 Introduction

Summer is a holiday season for everyone and it is also the season of competition between ice-cream sellers on beaches. On a colorful crowded beach, an ice-cream seller need to choose a "good" location that maximize his profit known that tourists will go to the closest seller to buy ice-creams. A similar competition holds between service providers, enterprises to decide where to open a new facility, a new market in order to attract as much of clients as possible.

The Voronoi Game is a simple geometric model for the competitive facility location. Competitive facility location studies the placement of sites by competing market players. Voronoi game is widely studied on a continuous space, for example on a 2-dimensional rectangle. Players alternatively place points in the space. Then the Voronoi diagram is considered. Every player gains the total surface of the Voronoi cells of his points. The geometric concepts are combined with game theory arguments to study if there exists any winning strategy.

We consider the discrete version of the Voronoi game which plays on a given graph instead on a continuous space. Formally, the discrete Voronoi game plays on a given undirected graph $G(V, E)$ with $n = |V|$ and $k$ players. Every player has to choose a vertex (*facility*) from $V$, and every vertex (*customer*) is assigned to the closest facilities. If there are more than one closest facility then the vertex is assigned in equal fraction to these closest facilities. One may think that each vertex consists of a group of clients in which one half go to a facility and the other half go to another facility if there are two closest facilities to this group. A player's *payoff (utility)* is the number of vertices assigned to his facility. The *social cost* is the total distance that customers go to their closest facilities, i.e. it is defined as the sum of the distances to the closest facility over all vertices. The optimization problem with the objective function as this social cost is the well-known *k-median* problem which is $\mathcal{NP}$-complete [65, chapter 25].

In this chapter, we show that the existence of Nash equilibria is a graph property for a fixed number of players. There exist Nash equilibria for some graphs but there are none for others. We show that the best-response dynamic does not converge to a Nash equilibria on cycle and line graphs, but does converge for a variant of this game. Moreover, we show that deciding this graph property is an $\mathcal{NP}$-hard problem using the negated gadget technique described in Chapter 2.

We also study the social cost discrepancy in the game and show that it is $\Omega(\sqrt{n/k})$ and $O(\sqrt{kn})$. Hence for a constant number of players we have tight bounds. Interestingly, in analyzing this measure, we introduce the notion of Delaunay triangulation on discrete setting and that matches the usual Delaunay triangulation in continuous surface.

**Related work** Most prior work studies Voronoi game on continuous surface. Ahn et al. [1] were the firsts to demonstrate that there exists a winning strategy for the one dimensional arena case, i.e. a line segment $[0, 1]$. In two dimensional case, the scenario differs significantly. Intuitively, the difficulty is arisen since the Voronoi cells in one dimension are simply lines or curves but in two dimension, they becomes much more complicated. Cheong et al. [22], Fekete and Meijer [36] characterized the winning strategy for the game played on 2D rectangle as a function of width-height ratio of the rectangle.

The closest game related to our work is the facility location game introduced in [66]. The facility location game plays on bipartite weighted graph in which there are $k$ suppliers, each chooses to open a single facility within the set of facilities, and $m$ markets that the suppliers will serve. Given a strategy profile, a supplier serves the markets closest to it and receives a payment from these markets. A market receives a value if it is served. The payment in this game makes it fundamentally different to the Voronoi game. Vetta [66] proved that there always exist a Nash equilibrium in this game and the price of anarchy is at most 2 with respect to the total profit of the game (the formal definition is in [66]).

Inspired by the Voronoi game on graphs, introduced in [29], Mavronicolas et al. [55] study the game on cycle graphs, particularly on the characterization of equilibrium and on the price of anarchy on such graph. Zhao et al. [68] study the isolation game, which may be considered as the dual of the Voronoi game.

**Organization** In Section 3.2, we give the formal description of the Voronoi game. We warm up by studying the game on cycle graph in Section 3.3. In Section 3.4, we prove that it is $\mathcal{NP}$-complete to decide whether a given game admits an equilibrium. We bound the social cost discrepancy of the game in Section 3.5.

## 3.2 The game

For this game we need to generalize the notion of vertex partition of a graph: A *generalized partition* of a graph $G(V, E)$ is a set of $n$-dimensional non-negative vectors, which sum up to the vector with 1 in every component, for $n = |V|$.

The Voronoi game on graphs consists of:

- A graph $G(V, E)$ and $k$ players. We assume $k < n$ for $n = |V|$, otherwise the game has a trivial structure. The graph induces a distance between vertices $d : V \times V \to \mathbb{N} \cup \{\infty\}$, which is defined as the minimal number of edges of any connecting path, or infinite if the vertices are disconnected.

- The strategy set of each player is $V$. A strategy profile of $k$ players is a vector $f = (f_1, \ldots, f_k)$ associating each player to a vertex.

- For every vertex $v \in V$ — called *customer* — the distance to the closest facility is denoted as $d(v, f) := \min_{f_i} d(v, f_i)$. Customers are assigned in equal fractions to the closest facilities as follows. The strategy profile $f$ defines the generalized partition

$\{F_1, \ldots, F_k\}$, where for every player $1 \le i \le k$ and every vertex $v \in V$,

$$F_{i,v} = \begin{cases} 1/|\arg\min_j d(v, f_j)| & \text{if } d(v, f_i) = d(v, f) \\ 0 & \text{otherwise.} \end{cases}$$

We call $F_i$ the *Voronoi cell* of player $i$. The *radius* of the Voronoi cell of player $i$ is defined as $\max_v d(v, f_i)$ where the maximum is taken over all vertices $v$ such that $F_{i,v} > 0$.

- The *payoff (utility)* of player $i$ is the (fractional) amount of customers assigned to it, that is $p_i := \sum_{v \in V} F_{i,v}$. (see figure 3.1 for an example)
- The *social cost* of strategy profile $f$ is $\text{cost}(f) := \sum_{v \in V} d(v, f)$.

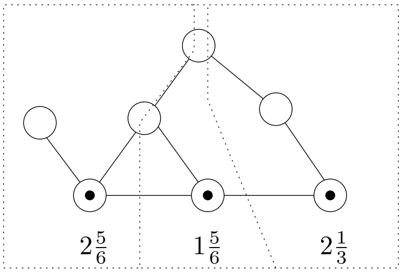

Figure 3.1: A strategy profile of a graph (players are dots) and the corresponding payoffs.

We defined players' payoffs in such a way, that there is a subtle difference between the Voronoi game played on graphs and the Voronoi game played on a continuous surface. Consider a situation where a player $i$ moves to a location already occupied by a single player $j$ who is not neighbor of $i$. Then, in the continuous case player $i$ gains exactly a half of the previous payoff of player $j$ (since it is now shared with $i$). However, in our setting (the discrete case), player $i$ can sometimes gain more than a half of the previous payoff of player $j$ (see figure 3.2).

A simple observation leads to the following bound on the players payoff.

**Lemma 3.1.** *In a Nash equilibrium the payoff $p_i$ of every player $i$ is bounded by $n/2k < p_i < 2n/k$.*

*Proof.* If a player gains $p$ and some other player moves to the same location then both payoffs are at least $p/2$. Therefore the ratio between the largest and the smallest payoffs among all players can be at most 2. If all players have the same payoff, it must be exactly $n/k$, since the payoffs sum up to $n$. Otherwise there is at least one player who gains strictly less than $n/k$, and another player who gains strictly more than $n/k$. This concludes the proof. □

## 3.3 The cycle graph

Let $G(V, E)$ be the cycle on $n$ vertices with $V = \{v_i : i \in \mathbb{Z}_n\}$ and $E = \{(v_i, v_{i+1}) : i \in \mathbb{Z}_n\}$, where addition is modulo $n$. The game plays on the undirected cycle, but it will be convenient to fix an arbitrary orientation. Let $u_0, \ldots, u_{\ell-1}$ be the distinct facilities chosen by $k$ players in a strategy profile $f$ with $\ell \leq k$, numbered according to the orientation of the cycle. For every $j \in \mathbb{Z}_\ell$, let $c_j \geq 1$ be the number of players who choose the facility $u_j$ and let $d_j \geq 1$ be the length of the directed path from $u_j$ to $u_{j+1}$ following the orientation of $G$. Now the strategy profile is defined by these $2\ell$ numbers, up to permutation of the players. We decompose the distance into $d_j = 1 + 2a_j + b_j$, for $0 \leq b_j \leq 1$, where $2a_j + b_j$ is the number of vertices between facilities $u_j$ and $u_{j+1}$. So if $b_j = 1$, then there is a vertex in midway at equal distance from $u_j$ and $u_{j+1}$.

With these notations the payoff of player $i$ located on facility $u_j$ is

$$p_i := \frac{b_{j-1}}{c_{j-1} + c_j} + \frac{a_{j-1} + 1 + a_j}{c_j} + \frac{b_j}{c_j + c_{j+1}}.$$

All Nash equilibria of the game on the cycle graph are explicitly characterized. The intuition is that the cycle is divided by the players into segments of different length, which roughly differ at most by a factor 2. The exact statement is more subtle because several players can be located at a same facility and the payoff is computed differently depending on the parity of the distances between facilities.

**Proposition 3.2** ([55]). *For a given strategy profile, let $\gamma$ be the minimal payoff among all players, i.e., $\gamma := \min\{p_i | 1 \leq i \leq k\}$. Then this strategy profile is a Nash equilibrium if and only if, for all $j \in \mathbb{Z}_\ell$:*

*P1.* $c_j \leq 2$.

*P2.* $d_j \leq 2\gamma$.

*P3.* *If* $c_i \neq c_{i+1}$ *then* $\lfloor 2\gamma \rfloor$ *is odd.*

*P3.* *If* $c_j = 1$ *and* $d_{j-1} = d_j = 2\gamma$ *then* $2\gamma$ *is odd.*

*P4.* *If* $c_j = c_{j+1} = 1$ *and* $d_{i-1} + d_i = d_{i+1} = 2\gamma$ *then* $2\gamma$ *is odd.*
*If* $c_j = c_{j-1} = 1$ *and* $d_{i-1} = d_i + d_{i+1} = 2\gamma$ *then* $2\gamma$ *is odd.*

A method to find Nash equilibrium in some games is to apply the best-response dynamic from an initial strategy profile. However, in our game, even in the simple cycle graph in which all Nash equilibria can be exactly characterized, in general there is no hope to use the best-response dynamic to find Nash equilibria in the game.

**Proposition 3.3.** *On the cycle graph, the best response dynamic does not converge.*

*Proof.* Figure 3.2 shows an example of a graph, where the best response dynamic can iterate forever.

□

Nevertheless there is a slightly different Voronoi game in which the best response dynamic converges : The *Voronoi game with disjoint facilities* is identical with the previous game, except that players who are located on the same facility now gain zero.

**Proposition 3.4.** *On the cycle graph, for the* Voronoi game with disjoint facilities*, the best response dynamic does converge to pure Nash equilibria.*

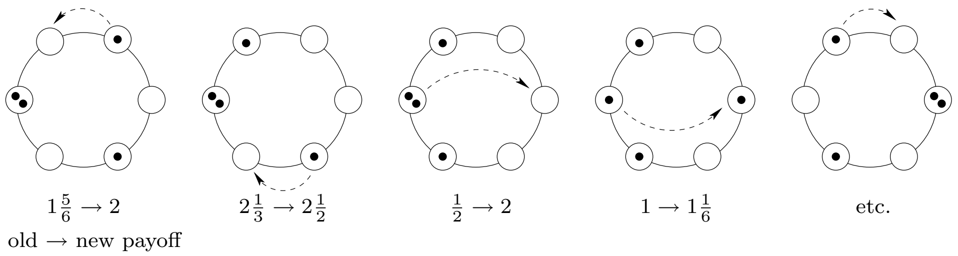

Figure 3.2: The best response dynamic does not converge on this graph.

*Proof.* To show convergence we use a potential function. For this purpose we define the *dominance order*: Let $A, B$ be two multisets. If $|A| < |B|$ then $A \succ B$. If $|A| = |B| \geq 1$, and $\max A > \max B$ then $A \succ B$. If $|A| = |B| \geq 1, \max A = \max B$ and $A \backslash \{\max A\} \succ B \backslash \{\max B\}$ then $A \succ B$. This is a total order.

The potential function is the multiset $\{d_0, d_1, \ldots, d_{k-1}\}$, that is all distances between successive occupied facilities. Player $i$'s payoff — renumbered conveniently — is simply $(d_i + d_{i+1})/2$ since there is at most one player located on a vertex. Now consider a best response for player $i$ moving to a vertex not yet chosen by another player, say between player $j$ and $j+1$. Therefore in the multiset $\{d_0, d_1, \ldots, d_{k-1}\}$, the values $d_i, d_{i+1}, d_j$ are replaced by $d_i + d_{i+1}, d', d''$ for some values $d', d'' \geq 1$ such that $d_j = d' + d''$. The new potential value is dominated by the previous one. This proves that after a finite number of iterations, the best response dynamic converges to a Nash equilibrium. □

## 3.4 Existence of a Nash equilibrium is $\mathcal{NP}$-hard

In this section we show that it is $\mathcal{NP}$-hard to decide whether for a given graph $G(V, E)$ there is a Nash equilibrium for $k$ players. For this purpose we define a more general but equivalent game, which simplifies the reduction.

In the *generalized Voronoi game* $\langle G(V, E), U, w, k\rangle$ we are given a graph $G$, a set of facilities $U \subseteq V$, a positive weight function $w$ on vertices and a number of players $k$. Here the set of strategies of each player is only $U$ instead of $V$. Also the payoff of a player is the weighted sum of fractions of customers assigned to it, i.e., the payoff of player $i$ is $p_i := \sum_{v \in V} w(v) F_{i,v}$.

**Lemma 3.5.** *For every generalized Voronoi game $\langle G(V, E), U, w, k\rangle$ there is a standard Voronoi game $\langle G'(V', E'), k\rangle$ with $V \subseteq V'$, which has the same set of Nash equilibria and which is such that $|V'|$ is polynomial in $\sum_{v \in V} w(v)$.*

*Proof.* To construct $G'$ we will augment $G$ in two steps. Start with $V' = V$.

First, for every vertex $u \in V$ such that $w(u) > 1$, let $H_u$ be a set of $w(u) - 1$ new vertices. Set $V' = V' \cup H_u$ and connect $u$ with every vertex from $H_u$.

Second, let $H$ be a set of $k(a+1)$ new vertices where $a = |V'| = \sum_{v \in V} w(v)$. Set $V' = V' \cup H$ and connect every vertex of $U$ with every vertex of $H$.

Now in $G'(V', E')$ every player's payoff can be decomposed in the part gained from $V' \backslash H$ and the part gained from $H$. We claim that in a Nash equilibrium every player chooses a vertex from $U$. If there is at least one player located in $U$, then the gain from $H$ for any other player is 0 if located in $V' \backslash (U \cup H)$, is 1 if located in $H$ and is at least $a+1$ if located in $U$.

Since the total payoff from $V'\backslash H$ over all players is $a$, this forces all players to be located in $U$.

Clearly by construction, for any strategy profile $f \in U^k$, the payoffs are the same for the generalized Voronoi game in $G$ as for the standard Voronoi game in $G'$. Therefore we have equivalence of the set of Nash equilibria in both games. □

Our $\mathcal{NP}$-hardness proof will need the following gadget.

**Lemma 3.6.** *For the graph $G$ shown in figure 3.3 and $k = 2$ players, there is no Nash equilibrium.*

*Proof.* We will simply show that given an arbitrary location of one player, the other player can move to a location where he gains at least 5. Since the total payoff over both players is 9, this will prove that there is no Nash equilibrium.

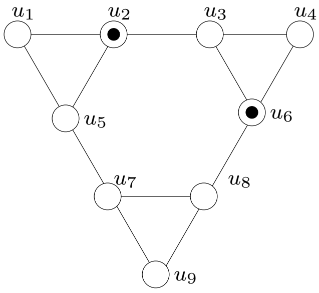

Figure 3.3: Example of a graph with no Nash equilibrium for 2 players.

By symmetry without loss of generality the first player is located at the vertices $u_1$ or $u_2$. Now if the second player is located at $u_6$, his payoff is at least 5. □

**Theorem 3.7.** *Given a graph $G(V, E)$ and a set of $k$ players, deciding the existence of Nash equilibrium for $k$ players on $G$ is $\mathcal{NP}$-complete for arbitrary $k$, and polynomial for constant $k$.*

*Proof.* The problem is clearly in $\mathcal{NP}$, since best responses can be computed in polynomial time, therefore it can be verified efficiently if a strategy profile is a Nash equilibrium. Since there are $n^k$ different strategy profiles, for $n = |V|$, the problem is polynomial when $k$ is constant.

For the proof of $\mathcal{NP}$-hardness, we will reduce 3-Partition — which is unary $\mathcal{NP}$-complete [42] — to the generalized Voronoi game, which by Lemma 3.5 is itself reduced to the original Voronoi game. In the 3-Partition problem we are given integers $a_1, \ldots, a_{3m}$ and $B$ such that $B/4 < a_i < B/2$ for every $1 \leq i \leq 3m$, $\sum_{i=1}^{3m} = mB$ and have to partition them into disjoint sets $P_1, \ldots, P_m \subseteq \{1, \ldots, 3m\}$ such that for every $1 \leq j \leq m$ we have $\sum_{i \in P_j} a_i = B$.

We construct a weighted graph $G(V, E)$ with the weight function $w : V \rightarrow \mathbb{N}$ and a set $U \subseteq V$ such that for $k = m+1$ players ($m \geq 2$) there is a Nash equilibrium to the generalized Voronoi game $\langle G, U, w, k\rangle$ if and only if there is a solution to the 3-Partition instance. We define the constants $c = \binom{3m}{3} + 1$ and $d = \left\lfloor \frac{Bc-c+c/m}{5} \right\rfloor + 1$. The graph $G$ consists of 3 parts. In the first part $V_1$, there is for every $1 \leq i \leq 3m$ a vertex $v_i$ of weight $a_i c$. There is also an additional vertex $v_0$ of weight 1. In the second part $V_2$, there is for every triplet $(i, j, k)$

with $1 \leq i < j < k \leq 3m$ a vertex $u_{ijk}$ of unit weight[1]. Every vertex $u_{ijk}$ is connected to $v_0, v_i, v_j$ and $v_k$. The third part $V_3$, consists of the 9 vertex graph of Figure 3.3 where each of the vertices $u_1, \ldots, u_9$ has weight $d$. To complete our construction, we define the facility set $U := V_2 \cup V_3$. Note that although the graph for the generalized Voronoi game is disconnected, the reduction of Lemma 3.5 to the original Voronoi game will connect $V_2$ with $V_3$.

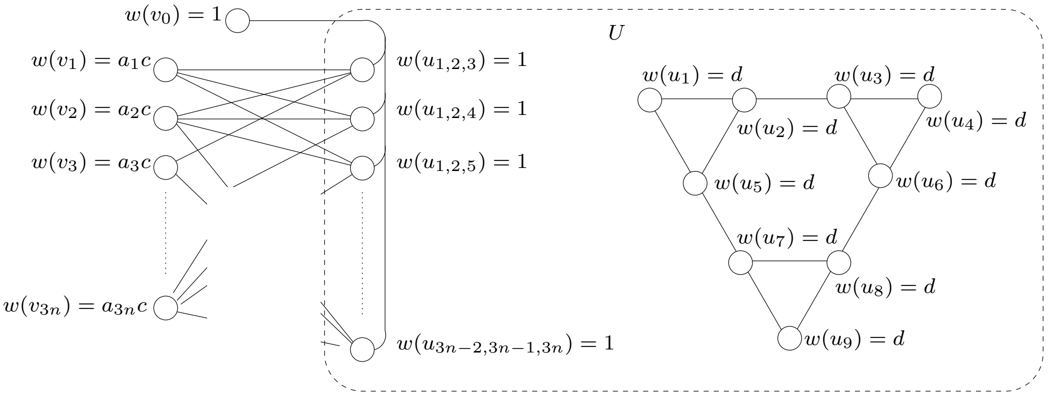

Figure 3.4: Reduction from 3-Partition.

First we show that if there is a solution $P_1, \ldots, P_m$ to the 3-Partition instance then there is a Nash equilibrium for this graph. Simply for every $1 \leq q \leq m$ if $P_q = \{i, j, k\}$ then player $q$ is assigned to the vertex $u_{ijk}$. Player $m + 1$ is assigned to $u_2$. Now player $(m + 1)$'s payoff is $9d$, and the payoff of each other player $q$ is $Bc + c/m$. To show that this is a Nash equilibrium we need to show that no player can increase his payoff. There are different cases. If player $m + 1$ moves to a vertex $u_{ijk}$, his payoff will be at most $\frac{3}{4}Bc + c/(m + 1) < 9d$, no matter if that vertex was already chosen by another player or not. If player $1 \leq q \leq m$ moves from vertex $u_{ijk}$ to a vertex $u_\ell$ then his gain can be at most $5d < Bc + c/m$. But what can be his gain, if he moves to another vertex $u_{i'j'k'}$? In case where $i = i', j = j', k \neq k'$, $a_ic + a_jc$ is smaller than $\frac{3}{4}Bc$ because $a_i + a_j + a_k = B$ and $a_k > B/4$. Since $a_{k'} < B/2$, and player $q$ gains only half of it, his payoff is at most $a_ic + a_jc + a_{k'}c/2 + c/m < Bc + c/m$ so he again cannot improve his payoff. The other cases are similar.

Now we show that if there is a Nash equilibrium, then it corresponds to a solution of the 3-Partition instance. So let there be a Nash equilibrium. First we claim that there is exactly one player in $V_3$. Clearly if there are 2 players, this contradicts equilibrium by Lemma 3.6. If there are 3 players or more, then by a counting argument there are vertices $v_i, v_j, v_k$ which are at distance more than one from any player. One of the players located at $V_3$ gains at most $3d$ and if he moves to $u_{ijk}$, his payoff would be at least $\frac{3}{4}Bc + c/m > 3d$. Now if there is no player in $V_3$, then any player moving to $u_2$ will gain $9d > \frac{3}{2}Bc + c/m$ which is an upper bound for the payoff of players located in $V_2$. So we know that there is a single player in $V_3$ and the $m$ players in $V_2$ must form a partition, since otherwise there is a vertex $v_i \in V_1$ at distance at least 2 to any player. So, by the previous argument, there would be a player in $V_2$ who can increase his payoff by moving to the other vertex in $V_2$ as well. (He moves in such a way that his new facility is at distance 1 to $v_i$.) Moreover, in this partition, each player gains exactly $Bc + c/m$ because if one gains less, given all weights in

[1] Ideally we would like to give it weight zero, but there seems to be no simple generalization of the game which allows zero weights, while preserving the set of Nash equilibria.

$V_1$ are multiple of $c$, he gains at most $Bc - c + c/m$ and he can always augment his payoff by moving to $V_3$ ($5d > Bc - c + c/m$). □

## 3.5 Social cost discrepancy

In this section, we study how much the social cost of Nash equilibria can differ for a given graph, assuming Nash equilibria exist. Recall that the *social cost* of a strategy profile $f$ is $\text{cost}(f) := \sum_{v \in V} d(v, f)$. Since we assumed $k < n$ the cost is always non-zero. The *social cost discrepancy* of the game is the maximal fraction $\text{cost}(f)/\text{cost}(f')$ over all Nash equilibria $f, f'$. For unconnected graphs, the social cost can be infinite, and so can be the social cost discrepancy. Therefore in this section we consider only connected graphs.

**Lemma 3.8.** *There are connected graphs for which the social cost discrepancy is* $\Omega(\sqrt{n/k})$, *where* $n$ *is the number of vertices and* $k$ *the number of players.*

*Proof.* We construct a graph $G$ as shown in figure 3.5. The total number of vertices in the graph is $n = k(2a+b+2)$. We distinguish two strategy profiles $f$ and $f'$: the vertices occupied by $f$ are marked with round dots, and the vertices of $f'$ are marked with triangular dots.

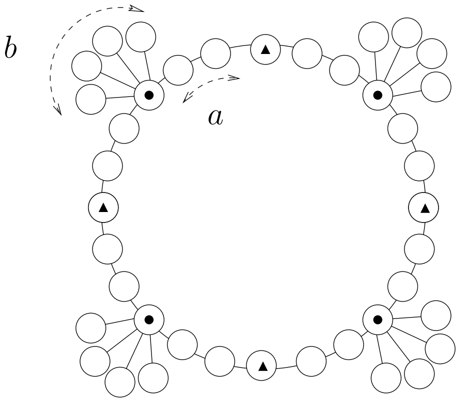

Figure 3.5: Example of a graph with high social cost discrepancy.

By straightforward verification, it can be checked that both $f$ and $f'$ are Nash equilibria. However the social cost of $f$ is $\Theta(kb + ka^2)$ while the social cost of $f'$ is $\Theta(kab + ka^2)$. The ratio between both costs is $\Theta(a) = \Theta(\sqrt{n/k})$ when $b = a^2$ and thus the cost discrepancy is lower bounded by this quantity. □

In continuous setting, Delaunay triangulation for a set $P$ of points in the plane is a triangulation such that no point in $P$ is inside the circumcircle of any triangle in the triangulation. Moreover, Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid skinny triangles. The Delaunay triangulation, considered as the dual of Voronoi diagram together with these nice properties, is an important object as well as a tool in mathematics and computational geometry. Here, we give the definition of *Delaunay triangulation* on graph which is a generalization of Delaunay triangulation in continuous setting (see Figure 3.6 for an illustration). Interestingly, this notion is useful in analyzing the social cost discrepancy of Voronoi game.

**Definition 3.9.** Given a strategy profile $f$, the *Delaunay triangulation* corresponding to $f$ is a graph $H_f$ on the $k$ players in profile $f$. There is an edge $(i, j)$ in $H_f$ either if there is a vertex $v$ in $G$ with $F_{i,v} > 0$ and $F_{j,v} > 0$ or if there is an edge $(v, v')$ in $G$ with $F_{i,v} > 0$ and $F_{j,v'} > 0$.

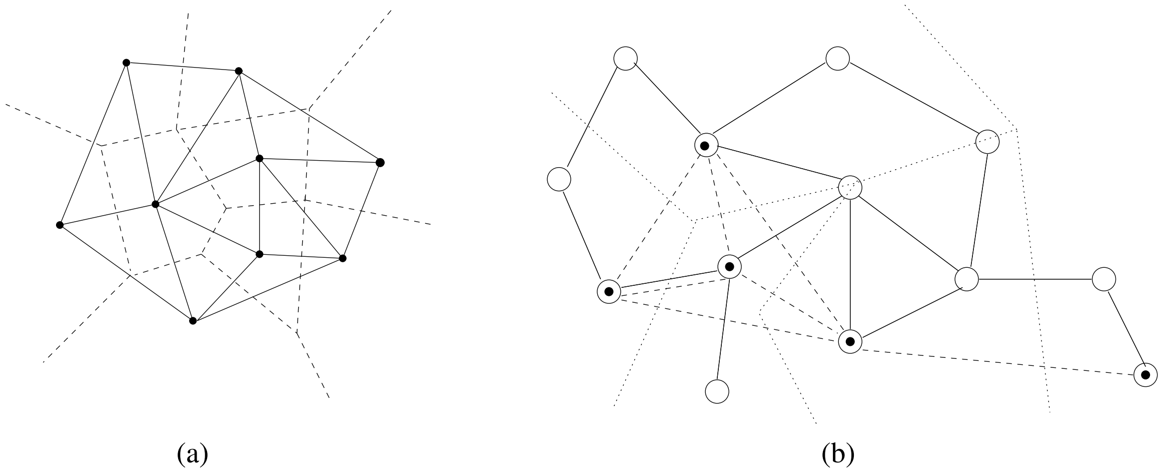

Figure 3.6: The Voronoi diagram and Delaunay triangulation in continuous and discrete settings. In continuous setting (a), the Voronoi diagram and the Delaunay triangulation are characterized by dashed and continuous lines, respectively. In discrete setting (b), the edges of graph are drawn in continuous and players are dots. The Voronoi diagram and the Delaunay triangulation in this case are characterized by dotted and dashed lines, respectively.

We will need to partition the Delaunay triangulation into small sets, called stars. For a given graph $G(V, E)$ a *star* is vertex set $A \subseteq V$ such that $|A| \geq 2$, and there is a *center* vertex $v_0 \in A$ such that for every $v \in A, v \neq v_0$ we have $(v_0, v) \in E$. Note that our definition allows the existence of additional edges between vertices from $A$.

**Lemma 3.10.** *For any connected graph $G(V, E)$, $V$ can be partitioned into stars.*

*Proof.* We define an algorithm to partition $V$ into stars.

As long as the graph contains edges, we do the following. We start choosing an edge: If there is a vertex $u$ with a unique neighbor $v$, then we choose the edge $(u, v)$; otherwise we choose an arbitrary edge $(u, v)$. Consider the vertex set consisting of $u, v$ as well as of any vertex $w$ that would be isolated when removing edge $(u, v)$. Add this set to the partition, remove it as well as adjacent edges from $G$ and continue.

Clearly the set produced in every iteration is a star. Also when removing this set from $G$, the resulting graph does not contain an isolated vertex. This property is an invariant of this algorithm, and proves that it ends with a partition of $G$ into stars. □

Note that, when a graph is partitioned into stars, the centers of these stars form a dominating set of this graph. Nevertheless, vertices in a dominating set are not necessarily centers of any star-partition of a given graph.

The following lemma states that given two different Nash equilibria $f$ and $f'$, every player in $f$ is not too far from some player in $f'$. For this purpose we partition the Delaunay

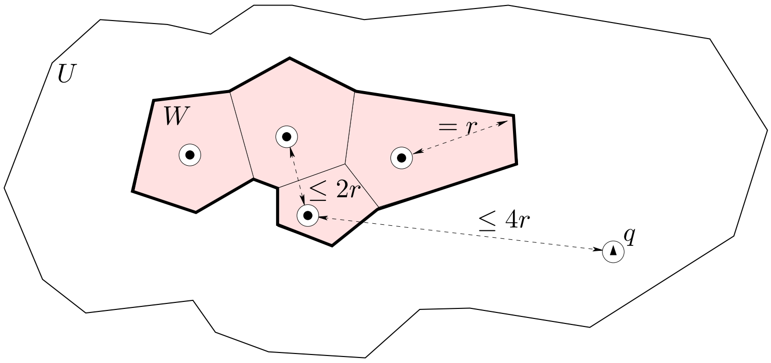

Figure 3.7: Illustration of Lemma 3.11.

triangulation $H_f$ into stars, and bound the distance from any player of a star to $f'$ by some value depending on the star.

**Lemma 3.11.** *Let $f$ be an equilibrium and $A$ be a star of a star partition of the Delaunay triangulation $H_f$. Let $r$ be the maximal radius of the Voronoi cells over all players $i \in A$. Then, for any equilibrium $f'$, there exists a player $f'_j$ such that $d(f_i, f'_j) \leq 6r$ for every $i \in A$.*

*Proof.* Let $U = \{v \in V : \min_{i \in A} d(v, f_i) \leq 4r\}$. If we can show that there is a facility $f'_j \in U$ we would be done, since by definition of $U$ there would be a player $i \in A$ such that $d(f_i, f'_j) \leq 4r$ and the distance between any pair of facilities of $A$ is at most $2r$. This would conclude the lemma.

So for a proof by contradiction, assume that in the strategy profile $f'$ there is no player located in $U$. Now consider the player with smallest payoff in $f'$. His payoff is not more than $n/k$. However if this player would choose as a facility the center of the star $A$, then he would gain strictly more: By the choice of $r$, any vertex in $W$ is at distance at most $3r$ to the center of the star. However, by assumption and definition of $U$, any other facility of $f'$ would be at distance strictly greater than $3r$ to any vertex in $W$. So the player would gain at least all vertices around it at distance at most $3r$, which includes $W$. Since any player's payoff is strictly more than $n/2k$ by Lemma 3.1, and since a star contains at least two facilities by definition, this player would gain strictly more than $n/k$, contradicting that $f'$ is an equilibrium. This concludes the proof. □

**Theorem 3.12.** *For any connected graph $G(V, E)$ and any number of players $k$ the social cost discrepancy is $O(\sqrt{kn})$, where $n = |V|$.*

*Proof.* Let $f, f'$ be arbitrary equilibria on $G(V, E)$. We will consider a generalized partition of $V$ and for each part bound the cost of $f'$ by $c\sqrt{kn}$ times the cost of $f$ for some constant $c$.

For a non-negative $n$-dimensional vector $W$ we define the cost restricted to $W$ as $\text{cost}_W(f) = \sum_{v \in V} W_v \cdot d(v, f)$. Now the cost of $f$ would write as the sum of $\text{cost}_W(f)$ over the vectors $W$ from some fixed generalized partition.

Fix a star partition of the Delaunay triangulation $H_f$. Let $A$ be an arbitrary star from this partition, $a = |A|$, and let $W$ be the *sum* of the corresponding Voronoi cells, i.e., $W = \sum_{i \in A} F_i$. We will show that $\text{cost}_W(f') = O(\sqrt{kn} \cdot \text{cost}_W(f))$, which would conclude the proof. There will be two cases $k \leq n/4$ and $k > n/4$.

By the previous lemma there is a vertex $f'_j$ such that $d(f_i, f'_j) \leq 6r$ for all $i \in A$, where $r$ is the largest radius of all Voronoi cells corresponding to the star $A$. So the cost of $f'$ restricted to the vector $W$ is

$$\begin{aligned}
\mathrm{cost}_W(f') &= \sum_{v \in V} W_v \cdot d(v, f') \leq \sum_{v \in V} W_v \cdot d(v, f'_j) \\
&= \sum_{v \in V} \sum_{i \in A} F_{i,v} \cdot d(v, f'_j) \\
&\leq \sum_{v \in V} \sum_{i \in A} F_{i,v} \cdot \big(d(v, f_i) + d(f_i, f'_j)\big) \\
&\leq \mathrm{cost}_W(f) + 6r \cdot |W|, \qquad (3.1)
\end{aligned}$$

where $|W| := \sum_{v \in V} W_v$.

Moreover by definition of the radius, there is a vertex $v$ with $W_v > 0$ such that the shortest path to the closest facility in $A$ has length $r$. So the cost of $f$ restricted to $W$ is bigger than the cost restricted to this shortest path:

$$\mathrm{cost}_W(f) \geq (\frac{1}{k} \cdot 1 + \frac{1}{k} \cdot 2 + \ldots + \frac{1}{k} \cdot r) \geq \frac{1}{k} \cdot r(r-1)/2.$$

(The fraction $\frac{1}{k}$ appears because a vertex can be assigned to at most $k$ players.)

First we consider the case $k \leq n/4$. We have

$$\mathrm{cost}_W(f) \geq |W| - |A| \geq a(n/2k - 1) \geq an/4k.$$

The first inequality is because the distance of all customers which are not facilites to a facility is at least one. The second inequality is due to Lemma 3.1 and $|W|$ is the sum of payoffs of all players in $A$.

Note that $|W| \leq n$ and $2 \leq a \leq k$ . Now if $r \leq \sqrt{an}$, then

$$\frac{\mathrm{cost}_W(f')}{\mathrm{cost}_W(f)} \leq 1 + \frac{6r|W|}{\mathrm{cost}_W(f)} \leq 1 + \frac{6r \cdot a \cdot 2n/k}{an/4k} = O(r) = O(\sqrt{kn}).$$

And if $r \geq \sqrt{an}$, then

$$\frac{\mathrm{cost}_W(f')}{\mathrm{cost}_W(f)} \leq 1 + \frac{6r|W|}{\mathrm{cost}_W(f)}) \leq 1 + \frac{6r \cdot a \cdot 2n/k}{r(r-1)/2k} = O(an/r) = O(\sqrt{kn}).$$

Now we consider case $n > k > n/4$. In any equilibrium, the maximum payoff is at most $2n/k$. Moreover the radius $r$ of any Voronoi cell is upper bounded by $n/k + 1$, otherwise the player with minimum gain (which is at most $n/k$) could increase his gain by moving to a vertex which is at distance at least $r$ from every other facility. Therefore $r = O(1)$. Summing (3.1) over all stars with associated partition $W$, we obtain $\mathrm{cost}(f') \leq \mathrm{cost}(f) + cn$, for some constant $c$. Remark that the social cost of any equilibrium is at least $n - k$. Hence, $\frac{\mathrm{cost}(f')}{\mathrm{cost}(f)} = O(n) = O(\sqrt{kn})$. □

# Chapter 4

# Scheduling Games in the Dark

## 4.1 Introduction

With the development of the Internet, large-scale autonomous systems became more and more important. The systems consist of many independent and selfish agents who compete for the usage of shared resources. Every configuration has some social cost, as well as individual costs for every agent. Due to the lack of coordination, the equilibrium configurations may have high cost compared to the global social optimum and this inefficiency can be captured by the price of anarchy. Since the behavior of the agents is influenced by the individual costs, it is natural and necessary to come up with mechanisms that both force the existence of Nash equilibria and reduce the price of anarchy. The idea is to try to reflect the social cost in the individual costs, so that selfish agents' behaviors result in a socially desired solution. In particular, we are interested in scheduling games, where every player has to choose one machine on which to execute its job. The individual cost of a player is the completion time of its job, and the social cost is the largest completion time over all jobs, the *makespan*. A *coordination mechanism* is based on a *local policy* for every machine, that specifies a schedule for the jobs assigned to the machine, and the schedule depends only on these jobs. Most prior studied policies depend on the processing times and need the jobs to announce their processing times. The jobs could try to influence the schedule to their advantage by announcing not their correct processing times. There are two ways to deal with this issue. One is to design *truthful coordination mechanisms* where jobs have an incentive to announce their real processing times. Another approach is to design mechanisms that do not depend on the processing times at all and this is the subject of this paper: we study coordination mechanisms based on so called *non-clairvoyant policies*.

### 4.1.1 Preliminaries

**Scheduling** The *machine scheduling problem* is defined as follows: we are given $n$ jobs, $m$ machines and each job needs to be scheduled on exactly one machine. In the most general case the processing times of jobs are unrelated: for every job $1 \leq i \leq n$ and every machine $1 \leq j \leq m$ we are given an arbitrary processing time $p_{ij}$, which is the time spend by job $i$ on machine $j$. A schedule $\sigma$ is a function mapping each job to some machine. The *load* of a machine $j$ in schedule $\sigma$ is the total processing time of all jobs assigned to this machine, i.e., $\ell_j = \sum_{i:\sigma(i)=j} p_{ij}$. The *makespan* of a schedule is the maximal load over all machines.

**Machine environments** We consider four different machine environments, which all have their own justification. The most general environment concerns unrelated machines as defined above and is denoted $R||C_{\max}$. In the *identical* machine scheduling model, denoted $P||C_{\max}$, every job $i$ comes with a length $p_i$ such that $p_{ij} = p_i$ for every machine $j$. In the *uniform* machine scheduling model, denoted $Q||C_{\max}$, again every job has *length* $p_i$ and every machine $j$ a speed $s_j$ such that $p_{ij} = p_i/s_j$. For the *restricted identical* machine model, every job $i$ comes with a length $p_i$ and a set of machines $S_i$ on which it can be scheduled, such that $p_{ij} = p_i$ for $j \in S_i$ and $p_{ij} = \infty$ otherwise. In [14] this model is denoted $PMPM||C_{\max}$, and in [47] it is denoted $B||C_{\max}$.

**Scheduling Game** Consider the situation where each of the $n$ jobs is owned by an independent agent. The agents' goal is to reduce their individual cost as much as possible. Each agent selects a single machine to process its job. Such a mapping $\sigma$ is a strategy profile. The *individual cost* of an agent is defined as the completion time of its job. An agent is aware of the decisions made by other agents and behaves selfishly. For simplicity, from now on we identify an agent with his job and use the term "job" instead of "agent". In games, there are two types of social costs which are widely considered: *utilitarian* and *egalitarian.* Informally, the utilitarian cost aims to measure the total individual costs while the egalitarian cost studies the maximal cost over all individual ones. Each cost is useful in different contexts and reflects the goal of the game designer. In scheduling game, we are interested in the latter. The *social cost* of a game is defined as its makespan. Intuitively, a small social cost means that jobs are well-balanced distributed and no machine is overload. It is $\mathcal{NP}$-hard to compute the *global optimum* even for identical machines, that is when $p_{ij}$ does not depend on $j$. We denote by OPT the makespan of the optimal schedule.

**Coordination Mechanism** A *coordination mechanism* is a set of *scheduling policies*, one for each machine, that determines how to schedule jobs assigned to a machine. The idea is to connect the individual cost to the social cost, in such a way that the selfishness of the agents will lead to equilibria that have low social cost. How good is a given coordination mechanism? This is measured by the well-known price of anarchy (PoA). We also consider the strong price of anarchy (SPoA) which is the extension of the price of anarchy applied to strong Nash equilibria.

**Policies** A *policy* is a rule that specifies how the jobs that are assigned to a machine are to be scheduled.

We distinguish between *local, strongly local* and *non-clairvoyant* policies. Let $S_j$ be the set of jobs assigned to machine $j$. A policy is *local* if the schedule of jobs on machine $j$ depends only on the parameters of jobs in $S_j$, i.e., it may look at the processing time $p_{ik}$ of a job $i \in S_j$ on any machine $k$. A policy is *strongly local* if it depends only on the processing time of jobs in $S_j$. We call a policy *non-clairvoyant* if the scheduling of jobs on machine $j$ does not depend on the processing time of any job on any machine.

Consider some clairvoyant policies. SPT and LPT are policies that schedule the jobs without preemption respectively in order of increasing or decreasing processing times with a deterministic tie-breaking rule for each machine. SPT (LPT) stands for "shortest (longest) remaining processing time first". An interesting property of SPT is that it minimizes the sum of the completion times, while LPT has a better price of anarchy, because it incites small jobs to go on the least loaded machine which smoothes the loads. A policy that relates individual costs even stronger to the social cost is MAKESPAN, where jobs are scheduled in parallel on one machine using time-multiplexing and assigned each job a fraction of the CPU that is proportional to its processing time. As a result all jobs complete at the same time, and the individual cost is the load of the machine. Note that MAKESPAN is not a non-clairvoyant policy since in order to complete all jobs at the same time as the load, the machine needs to know the processing times of jobs.

What could a scheduler do in the non-clairvoyant case? He could either schedule the jobs in a random order or in parallel. The RANDOM policy schedules the jobs in a random order without preemption. Consider a job $i$ assigned to machine $j$ in the schedule $\sigma$, then the cost of $i$ under the RANDOM policy is its expected completion time [47], i.e.,

$$c_i = p_{ij} + \frac{1}{2} \sum_{i' : \sigma(i') = j,\ i' \neq i} p_{i'j}.$$

In other words the expected completion time of $i$ is half of the total load of the machine, where job $i$ counts twice. Again, as for MAKESPAN, the individual and social cost in RANDOM are strongly related, and it is likely that these policies should have the same price of anarchy. That is is indeed the case except for unrelated machines.

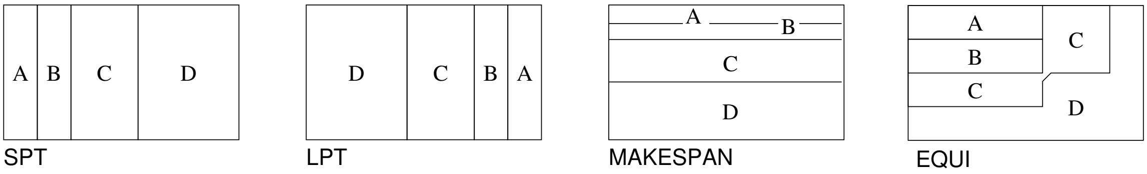

Figure 4.1: Illustration of different scheduling policies for $p_A = 1, p_B = 1, p_C = 2, p_D = 3$. Tie is broken arbitrarily between jobs $A$ and $B$. The rectangles represent the schedules on a single machine with time going from left to right and the hight of a block being the proportion of CPU assigned to the job.

Another natural non-clairvoyant policy is EQUI. As MAKESPAN it schedules the jobs in parallel preemptively using time-multiplexing and assigns to every job the same fraction of the CPU. Suppose that there are $k$ jobs assigned to machine $j$ and we renumber jobs from 1 to $k$ in such a way that $p_{1j} \leq p_{2j} \leq \ldots \leq p_{kj}$. As each job receives the same amount of resource, job 1 is completed at time $c_1 = kp_{1j}$. At that time, all jobs have remaining processing time $(p_{2j} - p_{1j}) \leq (p_{3j} - p_{1j}) \leq \ldots \leq (p_{kj} - p_{1j})$. Now the machine splits its resource into $k-1$ parts until the moment job 2 is completed, which is at $kp_{1j} + (k-1)(p_{2j} - p_{1j}) = p_{1j} + (k-1)p_{2j}$. In general, the completion time of job $i$, which is also its cost, under EQUI policy is:

$$\begin{aligned} c_i &= c_{i-1} + (k - i + 1)(p_{ij} - p_{i-1,j}) \\ &= p_{1j} + \ldots + p_{i-1,j} + (k - i + 1)p_{ij} \qquad (4.1) \end{aligned}$$

We already distinguished policies depending on what information is needed from the jobs. In addition we distinguish between *preemptive* and *non-preemptive* policies, depending on the schedule that is produced. Among the policies we considered so far, only MAKESPAN and EQUI are preemptive, in the sense that they rely on time-multiplexing, which consists in executing arbitrary small slices of the jobs.

### 4.1.2 Previous and related work

Coordination mechanism are related to local search algorithms. The local improvements in the local search algorithm correspond to the best-response moves of players in the game. Some results on local search algorithms for scheduling problem are surveyed in [67].

Most previous work concerned strongly local policies, in particular the MAKESPAN policy. The scheduling game under MAKESPAN policy also models congestion game in parallel links. In this game, there are players who have same source and sink. Each one can choose a link to go from the source to the sink. The cost of each player is the congestion of the link that he used which is defined as the total sum of loads induced from all players using this link. Czumaj and Vöcking [26] gave tight results $\Theta(\log m/\log\log m)$ of its price of anarchy for pure Nash equilibria on uniform machines. Fiat et al. [38] extended this result for the strong price of anarchy, and obtained the tight bound $\Theta(\log m/(\log\log m)^2)$. In addition, Gairing et al. [41] and Awerbuch et al. [7] gave tight bounds for the price of anarchy for restricted identical machines (see Table 4.1).

Coordination mechanism design was introduced by Christodoulou et al. [24]. They studied the LPT policy on identical machines. Immorlica et al. [47] studied coordination mechanism for all four machine environments and gave a survey on the results for non-preemptive strongly local policies. They also analyzed the existence of pure Nash equilibria under SPT, LPT and RANDOM for certain machine environments and the convergence speed of best-response dynamic to an equilibrium. Precisely, they proved that the game is a potential game under the policies SPT on unrelated machines, LPT on uniform or restricted identical machines, and RANDOM on restricted identical machines. The policy EQUI has been studied in [32] for its competitive ratio. The results are summarized in Table 4.1.

For local policies, Azar et al. [9] introduced the inefficiency-based policy which has price of anarchy $O(\log m)$ on unrelated machines and modified it to get a policy which always admits an equilibrium and has price of anarchy $O(\log^2 m)$. Moreover, they also proved that every non-preemptive strongly local policy with an additional assumption on policies has price of anarchy at least $m/2$, which shows a sharp difference between strongly local and local policies. Caragiannis [16] gave three local policies with price of anarchy $O(\log m)$, $O(\log m/\log\log m)$ and $O(\log^2 m)$, respectively, in which the games under first and the third policies always admit an equilibrium.

| model \ policy | MAKESPAN | SPT | LPT | RANDOM | EQUI |
|---|---|---|---|---|---|
| identical | $2 - \frac{2}{m+1}$<br>[39, 61] | $2 - \frac{1}{m}$<br>[43, 47] | $\frac{4}{3} - \frac{1}{3m}$<br>[44, 24] | $2 - \frac{2}{m+1}$<br>[39, 61] | $2 - \frac{1}{m}$ |
| uniform | $\Theta(\frac{\log m}{\mathrm{loglog} m})$<br>[26] | $\Theta(\log m)$<br>[5, 47] | $1.52 \leq PoA \leq 1.59$<br>[27, 40, 47] | $\Theta(\frac{\log m}{\mathrm{loglog} m})$<br>[26] | $\Theta(\log m)$ |
| restricted id. | $\Theta(\frac{\log m}{\mathrm{loglog} m})$<br>[41, 7] | $\Theta(\log m)$<br>[5, 47] | $\Theta(\log m)$<br>[8, 47] | $\Theta(\frac{\log m}{\mathrm{loglog} m})$<br>[41, 7] | $\Theta(\log m)$ |
| unrelated | unbounded<br>[61] | $\Theta(m)$<br>[23, 46, 9] | unbounded | $\Theta(m)$<br>[47] | $\Theta(m)$ |

Table 4.1: Price of anarchy under different strongly local and non-clairvoyant policies. The right most column is our contribution.

### 4.1.3 Our contribution

We are interested in *admissible* non-clairvoyant policies – policies that always induce a Nash equilibrium for any instance of the game. Maybe more important than the question of existence of Nash equilibrium is the question of convergence to an equilibrium. Since no processing times are known to the coordination mechanism, it is impossible to compute some equilibria. As all processing times are known to all jobs, it makes sense to let the jobs evolve according to the best-response dynamics, until they eventually reach an equilibria. Therefore it is important to find out under which conditions the dynamics converges.

For the unrelated machine model, we call a job $i$ *balanced* if the ratio of its processing times is bounded by 2, meaning $\max_j p_{ij} / \min_j p_{ij} \leq 2$. In addition for the uniform machine model, we say that machines have *balanced speeds* if the maximum and minimum speeds differ at most by factor 2. Note that, in the model of uniform machines with balanced speeds, jobs are all balanced.

In Section 4.2, we study the existence of Nash equilibrium under the non-clairvoyant policies RANDOM and EQUI. We show that in the RANDOM policy, the game always possesses an equilibrium on uniform machines with balanced speeds. We also show that on two unrelated machines, it is a potential game for balanced jobs, but on three unrelated machines or more the best-response dynamic does not converge. These results partly answer open questions in [47]. The question of the existence of equilibria under RANDOM policy remains open in general model (without balanced jobs). Moreover, we prove that for the EQUI policy, the game is a potential game, see Table 4.2. Note that, in our proofs of equilibrium existence, it is sufficient to show that there exist potential functions which strictly decrease at each step of the dynamic. As the game is finite, after a finite number of steps, the potential function will converge to an equilibrium.

| model \ policy | MAKESPAN | SPT | LPT | RANDOM | EQUI |
|---|---|---|---|---|---|
| identical | Yes [34] | Yes [47] | Yes [47] | Yes [47] | Yes |
| uniform | | | | (*) | |
| restricted id. | | | | Yes [47] | |
| unrelated | | | No | (**) | |

Table 4.2: Convergence of the best response dynamic and existence of equlibria. (*) A refinement of the best response dynamic converges when machines have balanced speeds. (**) Does not converge for $m \geq 3$ machines, but converges for $m = 2$ machines and balanced jobs.

In Section 4.3, we analyze the price of anarchy and the strong price of anarchy of EQUI. We observe that RANDOM is slightly better than EQUI except for the unrelated model. In the unrelated model, interestingly, the price of anarchy of EQUI reaches the lower bound in [9] on the PoA of any strongly local policy with some additional condition. The latter showed that although there is a clear difference between strongly local and local policies with respect to the price of anarchy, our results indicate that in contrast, restricting strongly local policies to be non-clairvoyant does not really affect the price of anarchy. Moreover, EQUI policy does not need any knowledge about jobs' characteristics, even their identities (IDs) which are useful in designing policies with low price of anarchy in [9, 16].

## 4.2 Existence of Nash equilibrium

The results in this section are summarized as follows.

**Summary of results on the existence of Nash equilibrium:** *We consider the scheduling game under different policies in different machine environments.*

1. *For the RANDOM policy on unrelated machines, it is not a potential game for 3 or more machines, but it is a potential game for 2 machines and balanced jobs. On uniform machines with balanced speeds, the RANDOM policy induces a Nash equilibrium.*

2. *For the EQUI policy it is an exact potential game.*

### 4.2.1 The RANDOM policy for unrelated machines

In the RANDOM policy, the cost of a job is its expected completion time. If the load of machine $j$ is $\ell_j$ then the cost of job $i$ assigned to machine $j$ is $\frac{1}{2}(\ell_j + p_{ij})$. We see that a job $i$ on machine $j$ has an incentive to move to machine $j'$ if and only if $p_{ij} + \ell_j > 2p_{ij'} + \ell_{j'}$. In the following, we will characterize the game under the RANDOM policy in the unrelated model as a function of the number of machines.

**Theorem 4.1.** *The game is a potential game under the RANDOM policy on 2 machines with balanced jobs.*

*Proof.* The proof uses a potential function. Let $\sigma : \{1, \dots, n\} \to \{1, 2\}$ be the current strategy profile, meaning that job $i$ is assigned to machine $\sigma(i)$. By $\overline{\sigma}(i)$ we denote the opposite machine to machine $\sigma(i)$. Let $\ell_j$ be the load of machine $j$ in strategy profile $\sigma$, which is $\sum_{i:\sigma(i)=j} p_{ij}$. Define

$$\Phi := |\ell_1 - \ell_2| + 3\sum_i \max\{p_{i\sigma(i)} - p_{i,\overline{\sigma}(i)}, 0\}$$

We claim that the potential function $\Phi$ strictly decreases at every best response move. Let $i$ be a job moving from say machine 1 to machine 2, while strictly decreasing its cost, i.e.

$$\ell_2 + 2p_{i2} - \ell_1 - p_{i1} < 0, \tag{4.2}$$

where $\ell_1, \ell_2$ are the loads before the move. We show now that the potential function decreases strictly by considering different cases.

By (4.2) and since all jobs are balanced, we have $\ell_1 > \ell_2$, meaning machine 1 is the most loaded machine before the move.

1. Case $\ell_1 - p_{i1} \geq \ell_2 + p_{i2}$, i.e., machine 1 is still the most loaded after the move. Let $\Phi'$ be the value of $\Phi$ after the move. Since $i$ is balanced and all other jobs do not change the strategy, we have

$$\begin{aligned}\Phi' - \Phi &= [(\ell_1 - p_{i1}) - (\ell_2 + p_{i2}) + 3\max\{p_{i2} - p_{i1}, 0\}] - [\ell_1 - \ell_2 + 3\max\{p_{i1} - p_{i2}, 0\}] \\ &= 2p_{i2} - 4p_{i1} < 0.\end{aligned}$$

where the equality uses the fact that for any real numbers $a$ and $b$, $\max\{a - b, 0\} - \max\{b - a, 0\} = a - b$.

2. Case $\ell_1 - p_{i1} < \ell_2 + p_{i2}$, i.e., machine 2 is the most loaded after the move. We have

$$\begin{aligned}\Phi' - \Phi &= [(\ell_2 + p_{i2}) - (\ell_1 - p_{i1}) + 3\max\{p_{i2} - p_{i1}, 0\}] - [\ell_1 - \ell_2 + 3\max\{p_{i1} - p_{i2}, 0\}] \\ &= 2\ell_2 - 2\ell_1 + p_{i2} - p_{i1} + 3(p_{i2} - p_{i1}) \\ &= 2(\ell_2 + 2p_{i2} - \ell_1 - 2p_{i1}) < 0,\end{aligned}$$

where the last inequality is due to (4.2).

Therefore, the potential function $\Phi$ decreases strictly lexicographically at every best move. □

However, for 3 or more machines, the best-response dynamic does not converge.

**Lemma 4.2.** *The best-response dynamic does not converge under the* RANDOM *policy on 3 or more machines.*

*Proof.* We give a simple four-job instance, with the following processing times. For convenience we name the jobs $A, B, C, D$.

| $p_{ij}$ | 1 | 2 | 3 |
|---|---|---|---|
| $A$ | 90 | 84 | $\infty$ |
| $B$ | 96 | 2 | $\infty$ |
| $C$ | 138 | 100 | $\infty$ |
| $D$ | $\infty$ | 254 | 300 |

Now we describe a cyclic sequence of best moves, where each job strictly decreases its cost, showing that the game does not converge. In the following table, we describe in each line, the current strategy profile, a best move of an unhappy job and its cost improvement.

| 1 | 2 | 3 | move | cost improvement |
|---|---|---|---|---|
| $AB$ | $C$ | $D$ | $1 \xrightarrow{A} 2$ | $138 > 134$ |
| $B$ | $AC$ | $D$ | $1 \xrightarrow{B} 2$ | $96 > 94$ |
|  | $ABC$ | $D$ | $2 \xrightarrow{C} 1$ | $143 > 138$ |
| $C$ | $AB$ | $D$ | $3 \xrightarrow{D} 2$ | $300 > 297$ |
| $C$ | $ABD$ |  | $2 \xrightarrow{B} 1$ | $171 > 165$ |
| $BC$ | $AD$ |  | $2 \xrightarrow{A} 1$ | $211 > 207$ |
| $ABC$ | $D$ |  | $1 \xrightarrow{C} 2$ | $231 > 227$ |
| $AB$ | $CD$ |  | $2 \xrightarrow{D} 3$ | $304 > 300$ |
| $AB$ | $C$ | $D$ |  |  |

□

Although there exists a cycle in best-reponse dynamic of the game under RANDOM policy, this does not mean that the game possesses no equilibrium. In fact, this question remains open.

### 4.2.2 The RANDOM policy on uniform machines with balanced speeds

Let $p_1 \leq p_2 \leq \ldots \leq p_n$ be the job lengths and $s_1 \geq s_2 \geq \ldots \geq s_m$ be the machine speeds. Now the processing time of job $i$ on machine $j$ is $p_i/s_j$. A *new unhappy* job with respect to a move is a job that was happy before the move and has become unhappy by this move.

**Lemma 4.3.** *Consider a job $i$ making a best move from machine $a$ to $b$ on uniform machines with balanced speeds. We have that if there is a new unhappy job with index greater than $i$ then $s_a > s_b$.*

*Proof.* Let $i'$ be a new unhappy job with respect to the best move of $i$. Let $\ell_j$ be the load of machine $j$ before the move of $i$. We distinguish three cases.

1. $i'$ was happy on some machine $c$ and $i'$ has incentive to move to machine $a$ when $i$ leaves machine $a$. We have the following inequalities.

$$\ell_a + \frac{p_i}{s_a} > \ell_b + \frac{2p_i}{s_b} \iff \ell_a - \ell_b > \left(\frac{2}{s_b} - \frac{1}{s_a}\right) p_i \tag{4.3}$$

$$\ell_c + \frac{p_{i'}}{s_c} > \left(\ell_a - \frac{p_i}{s_a}\right) + \frac{2p_{i'}}{s_a} \iff \ell_c - \ell_a > \left(\frac{2}{s_a} - \frac{1}{s_c}\right) p_{i'} - \frac{p_i}{s_a} \tag{4.4}$$

$$\ell_c + \frac{p_{i'}}{s_c} \leq \ell_b + \frac{2p_{i'}}{s_b} \iff \ell_c - \ell_b \leq \left(\frac{2}{s_b} - \frac{1}{s_c}\right) p_{i'} \tag{4.5}$$

These inequalities translate the facts that (4.3) $i$ has an incentive to migrate from $a$ to $b$; (4.4) $i'$ has incentive to move from $c$ to $a$ after $i$ moves; (4.5) $i'$ was happy before the move and had no incentive to move to $b$.

Summing up inequalities (4.3), (4.4) and using (4.5), we get:

$$\left(\frac{2}{s_b} - \frac{1}{s_c}\right) p_{i'} > \left(\frac{2}{s_b} - \frac{1}{s_a}\right) p_i + \left(\frac{2}{s_a} - \frac{1}{s_c}\right) p_{i'} - \frac{p_i}{s_a}$$

$$\iff \left(\frac{1}{s_b} - \frac{1}{s_a}\right) (p_{i'} - p_i) > 0.$$

Hence, if there is a new unhappy job with index greater than $i$, then $s_a > s_b$.

2. $i'$ was happy on machine $b$ and has an incentive to move to some machine $c \neq a$ when $i$ moves to $b$. We have:

$$\ell_b + \frac{p_i + p_{i'}}{s_b} > \ell_c + \frac{2p_{i'}}{s_c} \iff \ell_b - \ell_c > \frac{2p_{i'}}{s_c} - \frac{p_i + p_{i'}}{s_b}$$

$$\ell_c + \frac{2p_i}{s_c} \geq \ell_b + \frac{2p_i}{s_b} \iff \ell_b - \ell_c \leq \frac{2p_i}{s_c} - \frac{2p_i}{s_b}$$

where the first inequality translates the fact that $i'$ has an incentive to move from $b$ to $c$ after the move of $i$ and the second one describes that $i$ prefers machine $b$ to $c$.

From these inequalities, we get:

$$\frac{2p_{i'}}{s_c} - \frac{p_i + p_{i'}}{s_b} < \frac{2p_i}{s_c} - \frac{2p_i}{s_b}$$

$$\iff (p_i - p_{i'}) \left(\frac{2}{s_c} - \frac{1}{s_b}\right) > 0$$

By the property of balanced speeds, $2s_b \geq s_c$ so we have $p_i > p_{i'}$. In other words, there is no new unhappy job with greater index than $i$ in this case.

3. $i'$ was happy on machine $b$ and has an incentive to move to machine $a$ when $i$ moves to $b$. We have:

$$\ell_a + \frac{p_i}{s_a} > \ell_b + \frac{2p_i}{s_b}$$
$$\ell_b + \frac{p_i + p_{i'}}{s_b} > \ell_a + \frac{2p_{i'} - p_i}{s_a}$$

where the first inequality translates the fact that $i$ has an incentive to move from $a$ to $b$ and the second one describes the unhappiness of $i'$ after the move of $i$.

From these inequalities, we get $(p_i - p_{i'})\left(\frac{2}{s_c} - \frac{1}{s_b}\right) > 0$ as in previous case.

The claim immediately follows from these cases. □

**Theorem 4.4.** *On uniform machines with balanced speeds, there always exist Nash equilibria under the* RANDOM *policy.*

*Proof.* We use a potential argument on a refinement of the best-response dynamic. Consider a best-response dynamic in which among all unhappy jobs, the one with the greatest index makes the best move. By numbering convention, this job has the greatest length among all unhappy jobs. Given a strategy profile $\sigma$, let $t$ be the unhappy job of greatest index. We encode $t$ by a characteristic function $f_\sigma : \{1, 2, \ldots, n\} \to \{0, 1\}$ defined as:

$$f_\sigma(i) = \begin{cases} 1 & \text{if } 1 \leq i \leq t, \\ 0 & \text{otherwise.} \end{cases}$$

Define the potential function

$$\Phi(\sigma) := (f_\sigma(1), s_{\sigma(1)}, f_\sigma(2), s_{\sigma(2)}, \ldots, f_\sigma(n), s_{\sigma(n)}).$$

We claim that in each step of the best-response dynamic described above, the potential function decreases strictly lexicographically.

Let $t$ be the unhappy job of greatest index in the strategy profile $\sigma$, let $t'$ be the unhappy job of greatest index in $\sigma'$ – the strategy profile after the move of $t$. Note that the unique difference between $\sigma$ and $\sigma'$ is the machine to which job $t$ moved.

- If $t' < t$, we have that $f_\sigma(i) = f_{\sigma'}(i) = 1$, $s_{\sigma(i)} = s_{\sigma'(i)}$ for all $i \leq t'$ and $1 = f_\sigma(t'+1) > f_{\sigma'}(t'+1) = 0$. Thus, $\Phi(\sigma) > \Phi(\sigma')$.
- If $t' > t$, we also have that $f_\sigma(i) = f_{\sigma'}(i) = 1$, $s_{\sigma(i)} = s_{\sigma'(i)}$ for all $i < t$ and $f_\sigma(t) = f_{\sigma'}(t) = 1$. However, $t' > t$ means that there are some new unhappy jobs with lengths greater than $p_t$, hence by Lemma 4.3, $s_{\sigma(t)} > s_{\sigma'(t)}$. In the other words, $\Phi(\sigma) > \Phi(\sigma')$.

Therefore, the dynamic converges, showing that there always exists a Nash equilibrium. □

Intuitively, the potential function describes the following idea. We choose a dynamic satisfying either after each move, the number of unhappy player decreases (the case $t' < t$ in the proof) or otherwise there is something decreases (in the proof, this is some machine speed). We manage to combine coherently these two things into one function. Even the potential looks unusual, the idea behind is quite clear. The question about existence of equilibria remains open without assumption on machine speeds. Note that Lemma 4.3, which uses an assumption of balanced speeds, is crucial in our proof.

### 4.2.3 The EQUI policy

In the EQUI policy, the cost of job $i$ assigned to machine $j$ is given by expression (4.1). Here is an alternative formulation for the cost

$$c_i = \sum_{\substack{i':\sigma(i')=j \\ p_{i'j} \le p_{ij}}} p_{i'j} + \sum_{\substack{i':\sigma(i')=j \\ p_{i'j} > p_{ij}}} p_{ij}$$

**Theorem 4.5.** *The game with the EQUI policy is an exact potential game. In addition, it is a strong potential game, in the sense that the best-response dynamic converges even with deviations of coalitions.*

*Proof.* First, we prove that the game is an exact potential game. Let $\sigma$ be the current strategy profile, meaning $\sigma(i)$ is the current machine on which job $i$ is scheduled. Consider the following potential function.

$$\Phi(\sigma) = \frac{1}{2}\sum_{i=1}^{n} \left(c_i + p_{i\sigma(i)}\right)$$

We prove that if a job makes a best move then the potential function strictly decreases. Let $t$ be a job that moves from machine $a$ to $b$, while stricly decreasing its cost from $c_t$ to $c'_t$. We have:

$$c_t = \left( \sum_{\substack{i:\sigma(i)=a,i\neq t \\ p_{i,a}\le p_{t,a}}} p_{i,a} + \sum_{\substack{i:\sigma(i)=a,i\neq t \\ p_{i,a}>p_{t,a}}} p_{t,a} \right) + p_{t,a} > \left( \sum_{\substack{i:\sigma(i)=b,i\neq t \\ p_{i,b}\le p_{t,b}}} p_{i,b} + \sum_{\substack{i:\sigma(i)=b,i\neq t \\ p_{i,b}>p_{t,b}}} p_{t,b} \right) + p_{t,b} = c'_t$$

Let $\sigma'$ be the strategy profile after the move of job $t$. Note that in $\sigma'$ the processing time of all jobs except $i$ and the cost of all jobs scheduled on machine different to $a$ and $b$ stay the same. Thus, the change in the potential depends only on the jobs scheduled on machines $a$ and $b$.

$$\begin{aligned}
2\cdot\Delta\Phi = & \left( \sum_{i:\sigma'(i)=a} (c'_i + p_{ia}) \; + \sum_{i:\sigma'(i)=b,i\neq t} (c'_i + p_{ib}) + p_{t,b} \right) \\
& - \left( \sum_{i:\sigma(i)=a,i\neq t} (c_i + p_{ia}) \; + \sum_{i:\sigma(i)=b} (c_i + p_{ib}) \; + p_{t,a} \right) + (c'_t - c_t) \\
= & \sum_{i:\sigma(i)=a,i\neq t} (c'_i - c_i) \; + \sum_{i:\sigma(i)=b,i\neq t} (c'_i - c_i) \; + (c'_t - c_t) + p_{t,b} - p_{t,a}
\end{aligned}$$

since $\sigma(i) = \sigma'(i)\ \forall i \neq t$.

Consider a job $i \neq t$ on machine $a$. If the processing time of $i$ is at most that of $t$ then the difference between its new and old cost is exactly $-p_{i,a}$. Otherwise if the processing time of $i$ is strictly greater than that of $t$ then this difference is exactly $-p_{t,a}$. Analogously for jobs on machine $b$. Hence,

$$\begin{aligned}2\cdot\Delta\Phi = &\left(\sum_{\substack{i:\sigma(i)=b,i\neq t\\ p_{i,b}\le p_{t,b}}} p_{i,b} + \sum_{\substack{i:\sigma(i)=b,i\neq t\\ p_{i,b}> p_{t,b}}} p_{t,b} + p_{t,b}\right) + \\ &+\left(\sum_{\substack{i:\sigma(i)=a,i\neq t\\ p_{i,a}\le p_{t,a}}} -p_{i,a} + \sum_{\substack{i:\sigma(i)=a,i\neq t\\ p_{i,a}> p_{t,a}}} -p_{t,a} - p_{t,a}\right) + (c'_t - c_t)\\ &= 2\cdot(c'_t - c_t) < 0\end{aligned}$$

Therefore, the game with the EQUI policy is an exact potential game.

Moreover, the game is also an exact strong potential game. Let $S$ be a coalition and define its total cost $c(S) := \sum_{i\in S} c_i$. We study the best move of $S$ by dividing the process into two phases: in the first phase, all jobs in $S$ move out (disappear) from the game and in the second phase, jobs from $S$ move back (appear) into the game at their new strategies. We argue that after the first phase, the change in the potential is $\Delta\Phi = -c(S)$ and after the second phase $\Delta\Phi = c'(S)$. Since the argument is the same, we only prove it for the first phase; the second phase can be done similarly. Fix a machine $a$ and suppose without loss of generality that all the jobs assigned to $a$ are $1,\ldots,k$ for some $k$. Also to simplify notation we denote $q_i = p_{i,a}$ and assume $q_1 \le \ldots \le q_k$. Let $R = S\cap\sigma^{-1}(j) = \{i_1 \le \ldots \le i_r\}$ be the set of jobs in the coalition that are scheduled on this machine. Then,

$$c(R) = \sum_{j=1}^{r} c_{i_j} = \sum_{j=1}^{r}\big(q_1 + q_2 + \ldots + q_{i_j-1} + (k - i_j + 1)q_{i_j}\big)$$

The jobs in $R$ partition the jobs $\{1,\ldots,k\}$ into $r+1$ parts: part $j\in\{0,\ldots,r\}$ is $[i_j + 1, j_{j+1}]$, where for convenience we denote $i_0 = 0$ and $i_{r+1} = k$. After the move out of $R$, the change in cost of a job $t\notin R$ scheduled on the machine with index in $[i_j+1, i_{j+1}]$ is $q_{i_1} + q_{i_2} + \ldots + q_{i_{j-1}} + (r-j)q_t$. Hence, the difference in the potential restricted to machine $a$ after the first phase $\Delta\Phi|_a$ satisfies:

$$\begin{aligned}-2\Delta\Phi|_a &= \left[\sum_{j=0}^{r}\sum_{\substack{t\notin R\\ t\in[i_j+1,i_{j+1}]}} q_{i_1} + q_{i_2} + \ldots + q_{i_{j-1}} + (r-j)q_t\right] + [c(R) + (q_{i_1} + q_{i_2} + \ldots + q_{i_r})]\\ &= \left[\sum_{j=1}^{r}(k-i_j)q_{i_j} + \sum_{j=0}^{r}\sum_{\substack{t\notin R\\ t\in[i_j+1,i_{j+1}]}} (r-j)q_t\right] + [c(R) + (q_{i_1} + q_{i_2} + \ldots + q_{i_r})]\\ &= \sum_{j=1}^{r}\big(q_1 + q_2 + \ldots + q_{i_j-1} + (k-i_j+1)q_{i_j}\big) + c(R)\\ &= 2\cdot c(R)\end{aligned}$$

where in the first term of these equalities, we distinguish between the cost change of all jobs not in the coalition and the cost change of the jobs in the coalition, disapearing from the game. The potential change after the first phase is simply the sum of all the changes over all machines, so $\Delta\Phi = -c(S)$.

By the same argument, after the second phase we have $\Delta\Phi = c'(S)$. Therefore, the net change over both phases is $c'(S) - c(S)$. In conclusion, the game is a strong potential game. □

## 4.3 Inefficiency of Equilibria under the EQUI policy

In this section, we study the inefficiency of the game under the EQUI policy which is captured by the price of anarchy (PoA) and the strong price of anarchy (SPoA). Note that the set of strong Nash equilibria is a subset of that of Nash equilibria so the SPoA is at most as large as the PoA. We state the main theorem of this section. Whenever we bound *(S)PoA* we mean that the bound applies to both the price of anarchy and the strong price of anarchy.

**Summary of results on the price of anarchy:** *The game under the EQUI policy has the following inefficiency.*

1. *For identical machines, the (S)PoA is* $2 - \frac{1}{m}$.
2. *For uniform machines, the (S)PoA is* $\Theta(\min\{\log m, r\})$ *where* $r$ *is the number of different machine's speeds in the model.*
3. *For restricted identical machines, the (S)PoA is* $\Theta(\log m)$.
4. *For unrelated machines, the (S)PoA is* $\Theta(m)$.

We first give a characterization for strong Nash equilibrium in the game, which connects the equilibria to the strong ones. This characterization is useful in settling tight bounds of the strong price of anarchy in the game.

**Lemma 4.6.** *Suppose in a Nash equilibrium there is a coalition* $T$ *that makes a collective move such that each job in* $T$ *improves strictly its cost. Then this move preserves the number of jobs on every machine.*

*Proof.* For a proof by contradiction, let $\eta$ be an equilibrium that is not strong, and let $T$ be a coalition as stated in the claim. Suppose that the number of jobs on the machines is not preserved by the move of $T$. Let $j$ be a machine that has strictly more jobs after the move, and among all jobs migrating to $j$, let $o \in T$ be the job with smallest length $p_o$. Let $k$ and $k'$ be the numbers of jobs on $j$ before and after the move of $T$, respectively ($k' > k$). We claim that job $o$ could already improve its cost by unilaterally moving to $j$, contradicting that $\eta$ is a Nash equilibrium. Consider equilibrium $\eta$, if $o$ moves to machine $j$, its cost would be:

$$\begin{aligned} c_o &= (k + 1 - w)p_{oj} + \sum_{i: p_{ij} < p_{oj}} p_{ij} \\ &= (k - w + 1)p_{oj} + \sum_{i: p_{ij} < p_{oj}, i \notin T} p_{ij} + \sum_{i: p_{ij} < p_{oj}, i \in T} p_{ij} \end{aligned}$$

where $w$ is the number of jobs on machine $j$ in $\eta$ with length strictly less than $p_{oj}$.

Let $w'$ be the number of jobs on machine $j$ after the move of $T$ with length strictly less than $p_{oj}$. Since $o$ has the smallest length among all jobs migrating to $j$, $w' \leq w$. The cost of $o$ after the move of $T$ is:

$$c'_o = (k' - w')p_{oj} + \sum_{i:p_{ij}<p_{oj}, i \notin T} p_{ij}$$

We have:

$$\begin{aligned} c'_o - c_o &= \left[(k' - w') - (k - w + 1)\right] p_{ij} - \sum_{i:p_{ij}<p_{oj}, i \in T} p_{ij} \\ &\geq (w - w')p_{ij} - \sum_{i:p_{ij}<p_{oj}, i \in T} p_{ij} \\ &\geq (w - w')p_{ij} - (w - w')p_{ij} = 0 \end{aligned}$$

where the first inequality follows from $k' \geq k + 1$ and the second inequality uses $|\{i : p_{ij} < p_{oj}, i \in T\}| = w - w'$. Since job $o$ has incentive to cooperate and move to machine $j$, $o$ also get better off by unilaterally changing its strategy, so $\eta$ is not an equilibrium. □

### 4.3.1 Identical machines

In case of identical machines, the analysis of the PoA is quite similar to the well-known analysis of Graham's greedy load balancing algorithm that assigns the jobs to the least load machine, processing jobs in arbitrary order. Here we show that the (S)PoA matches exactly the approximation factor of the greedy algorithm.

**Proposition 4.7.** *For identical machines, the (S)PoA is* $2 - \frac{1}{m}$*. Moreover, there is an instance in which all equilibria have cost at least* $(2 - \frac{2}{m})OPT$*.*

*Proof. (Upper bound)* First we prove that PoA is upper-bounded by $2 - 1/m$. Let $\sigma$ be an equilibrium and $\ell_{\max}$ be the makespan of this equilibrium. Let $i$ be a job (with processing time $p_i$) that has cost $\ell_{\max}$. Hence, $p_i \leq OPT$. Since $\sigma$ is an equilibrium, the fact that job $i$ has no incentive to move to any other machine $j$ implies $\ell_{\max} \leq \ell_j + p_i$ for all machines $j$ different to $\sigma(i)$, where $\ell_j$ is the load of machine $j$. Summing up these inequalities over all machines $j$ we get $m\ell_{\max} \leq \sum_{j=1}^{m} \ell_j + (m-1)p_i$. Moreover, for any assignment of jobs to identical machines, $\sum_{j=1}^{m} \ell_j \leq mOPT$. Therefore, $m\ell_{\max} \leq (2m-1)OPT$, i.e., PoA $\leq 2 - 1/m$.

*(Lower bound)* Now we give an instance in which $OPT$ equals $m$ and all equilibria have cost at least $2m - 2$. In the instance, there are $m$ machines and $m(m-1)+1$ jobs in which all jobs have processing time 1 except one with processing time $m$. In an optimum assignment, the big job is scheduled on one machines and all $m(m-1)$ unit jobs are evenly assigned to the other machines, producing makespan $m$. We claim that in any equilibrium, every machine has at least $(m-1)$ unit jobs. Suppose there is a machine with at most $m-2$ jobs. Since there are $m(m-1)$ jobs of unit processing time, there must be a machine $j$ with at least $m$ unit jobs in the equilibrium. A unit job on machine $j$ has cost at least $m$ and it has incentive to move to the machine with less than $m-2$ jobs and get a smaller cost (at most $m-1$). This gives a contradiction and shows that any equilibrium, every machine has at least $m-1$ unit jobs. Now consider the machine with the big job. In addition this machine has at least $m-2$ unit jobs, so its load is at least $m + (m-2)$. Therefore, the makespan of the equilibrium is at least $(2 - 2/m)OPT$.

Consider the schedule in which there are $(m-1)$ unit jobs on every machine and the job with processing time $m$ on some arbitrary machine. It is straightforward that this is an equilibrium. By Lemma 4.6, this equilibrium is also a strong one. Hence, (S)PoA $\geq 2 - 1/m$. □

### 4.3.2 Uniform machines

We first give an upper bound on the PoA of any deterministic policy in this machine environment.

**Lemma 4.8.** *For uniform machines, the PoA of EQUI is* $\min\{O(\log m), 2r+1\}$*.*

*Proof.* Using ideas from [26], Immorlica et al. [47] proved that for any deterministic policy on uniform machines, the PoA is $O(\log m)$. We will prove the bound with respect to the number of machines' speeds. We classify all machines into $r$ groups $G_1, \ldots, G_r$ such that machines in group $G_j$ have the same speed, that we denote $s_j$ and $s_1 > s_2 > \ldots > s_r$. Recall that $OPT$ is the makespan of an optimal schedule. Let $\sigma$ be an equilibrium and $k$ be an integer such that the makespan of $\sigma$ is in the interval $(kOPT, (k+1)OPT]$. We claim and prove by induction on $j$, for $1 \leq j \leq r$, that the loads of machines in group $G_j$ are strictly greater than $(k+1-2j)OPT$.

Consider a machine in group $G_1$. If this machine has load at most $(k-1)OPT$ then a job with greatest cost (which equals the makespan of the equilibrium) would move to this machine. Since the machine has the greatest speed, the processing time of the job on this machine is at most $OPT$, which induces the cost at most $(k-1)OPT + OPT = kOPT$. Hence, the job has an incentive to change its strategy – this contradicts the assumption that $\sigma$ is an equilibrium.

Next, assume that all machines from groups $G_j$ for $j \leq t-1$ have load strictly greater than $(k+1-2j)OPT$. Let $W$ be the set of jobs with completion time at least $(k-2j)OPT$ and scheduled on some machine from $G_j$ for $1 \leq j \leq t-1$. Since the optimal schedule has cost $OPT$, in this optimal schedule there must be a job $i \in W$ which is assigned to a machine of group $G_{j'}$ with $j' \geq t$. Let $j$ be the machine on which $i$ is currently scheduled. The processing time of $i$ on a machine in group $G_t$ is $p_i/s_t \leq p_i/s_{j'} \leq OPT$. Hence, no machine in group $G_t$ has load at most $(k+1-2(j+1))OPT$ in $\sigma$ because otherwise, job $i$ can decrease its cost by moving to such a machine. This completes our induction step.

The optimal schedule implies that the total job length is at most $OPT/s_1 + \ldots + OPT/s_r$. Observe that in the Nash equilibrium, there exists a machine with load at most $OPT$ since otherwise the total job length would exceed $OPT/s_1 + \ldots + OPT/s_r$. Let $G_{j_0}$ be a group containing a machine of load at most $OPT$. By the claim, the load of this machine is at least $(k+1-2j_0)OPT$, thus $k+1-2j_0 \leq 1$, i.e., $k \leq 2j_0 \leq 2r$. This shows that the price of anarchy is at most $2r+1$. □

In the following, we present a family of game instances in which the PoA, together with the SPoA, are $\Omega(\log m)$ or $\Omega(r)$. The instances are inspired by the ones proving the lower bound of the competitive ratio of the greedy algorithm for related machine in [5].

**Family of Game Instances** There are $k+1$ groups of machines $G_0, G_1, \ldots, G_k$, each machine in group $G_j$ has speed $2^{-j}$ for $0 \leq j \leq k$. Group $G_0$ has $m_0 = 1$ machine, group $G_j$ has $m_j$ machines which is recursively defined as $m_j = \sum_{t=0}^{j-1} m_t \cdot 2^{j-t}$. Moreover, there are

$k+1$ groups of jobs $J_0, J_1, \dots, J_k$, for $0 \le j \le k-1$ each group $J_j$ consists of $2m_j$ jobs of length $2^{-j}$ and group $J_k$ consists of $3m_k$ jobs of length $2^{-k}$. The total number of machines is $m = \sum_{j=0}^{k} m_j = 1 + \frac{2}{3}(4^k - 1) + 2 \cdot 4^{k-1}$, thus $k = \Omega(\log m)$. The number of different speeds in the instance is $k+1$.

Consider a schedule that is a two-to-one mapping from the job group $J_j$ to the machine group $G_j$, for every $j < k$, and that is a three-to-one mapping from the job group $J_k$ to the machine group $G_k$. The load on a machine in group $G_j$ for $j < k$ is 2 and each machine in $G_k$ has load 3. Hence, $OPT \le 3$.

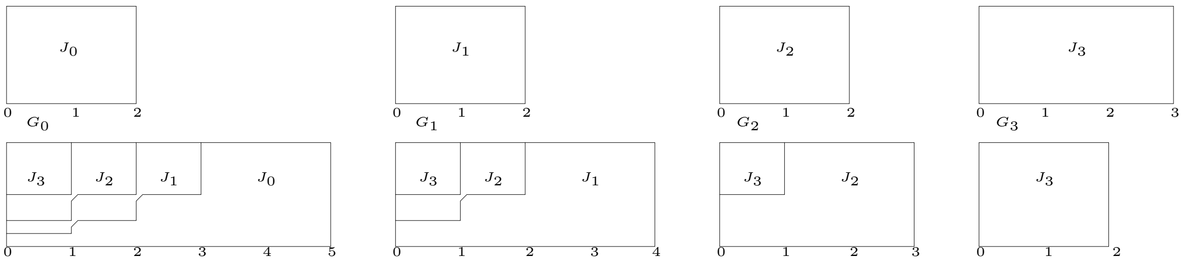

Figure 4.2: Illustration of the schedule with makespan 3 (upper part) and the strategy profile $\sigma$ (lower part) in the game instance for $k = 3$. Each machine group is represented by one of its machines.

Consider a schedule (strategy profile) $\sigma$ such that for every $0 \le j \le k$, in each machine of group $G_j$ (with speed $2^{-j}$), there are 2 jobs of length $2^{-j}$, and for every $j < i \le k$, there are $2^{i-j}$ jobs of length $2^{-i}$. Each machine in group $G_j$ has $2^{k-j+1}$ jobs and has load $2 \cdot 2^{-j}/2^{-j} + \sum_{t=1}^{k-j} 2^t \cdot 2^{-(j+t)}/2^{-j} = k - j + 2$, so the makespan of this schedule is $k+2$. We claim that this strategy profile is a Nash equilibrium, moreover it is a strong one.

**Lemma 4.9.** *Strategy profile $\sigma$ is a Nash equilibrium.*

*Proof.* First, we show that, in strategy profile $\sigma$, the cost of a job in $J_j$ is equal to $k - j + 2$. Fix a machine in $G_t$. We are only interested in case $t \le j$ since in $\sigma$, no job in $J_j$ is assigned to a machine of group $G_t$ with $t > j$. On this machine, there are exactly $2^{j-t+1}$ jobs with processing time at least $2^{-j}$. So, the cost of a job in $J_j$ scheduled on this machine is:

$$\frac{1}{2^{-t}}\left[\left(2^{k-t} \cdot 2^{-k} + 2^{(k-1)-t} \cdot 2^{-(k-1)} + \ldots + 2^{(j+1)-t} \cdot 2^{-(j+1)}\right) + 2^{j-t+1} \cdot 2^{-j}\right] = k + 2 - j.$$

Now we argue that $\sigma$ is an equilibrium. Suppose that a job $i$ in $J_j$ moves from its current machine to a machine of group $G_t$. If $j \le t$, $i$ has the greatest length among all jobs assigned to this new machine, so the new cost of $i$ is the new load of the machine which is $(k - t + 2) + 2^{-j}/2^{-t} > k - j + 2$. If $j > t$ then there are $(2^{j-t+1} + 1)$ jobs with length at least $2^{-j}$ on $i$'s new machine. Hence, the new cost of $i$ is:

$$\frac{1}{2^{-t}}\left[\left(2^{k-t} \cdot 2^{-k} + 2^{(k-1)-t} \cdot 2^{-(k-1)} + \ldots + 2^{(j+1)-t} \cdot 2^{-(j+1)}\right) + (2^{j-t+1} + 1) \cdot 2^{-j}\right] > k+2-j.$$

Therefore, no job can improve its cost by changing its strategy. □

Using Lemma 4.6, we show that $\sigma$ is indeed a strong equilibrium.

**Lemma 4.10.** *Strategy profile $\sigma$ is a strong Nash equilibrium.*

*Proof.* Suppose $\sigma$ is not a strong Nash equilibrium, then there exists a coalition $T$ such that all jobs in $T$ strictly decrease their costs and after the move of $T$, all machines have the same number of jobs as in $\sigma$ (by Lemma 4.6). Observe that the cost of a job in $J_k$ (with the least length among all jobs in the instance) depends only on the number of jobs scheduled on its machine. With such a move of $T$, if there are some jobs in $J_k$ involved in the coalition, none of them can strictly decrease its cost. Hence, $T \cap J_k = \emptyset$. Consider jobs in $J_{k-1}$. Since jobs in $J_k$ stay in their machines and they incur the same load 1 on each machine, the cost of a job in $J_{k-1}$, if it involves in $T$, depends only on the number of jobs which are not in $J_k$ and are scheduled on its new machine. However, this number is preserved after the move of $T$ (by Lemma 4.6 and $T \cap J_k = \emptyset$), so the cost of a job in $J_{k-1}$ stays the same, i.e., the job has no incentive to involve in $T$. The argument holds for groups of jobs $J_{k-2}, \ldots, J_0$. Therefore, $T = \emptyset$ meaning that $\sigma$ is a strong equilibrium. □

The previous lemmas imply that for uniform machines, the PoA of EQUI is at least $2r+1$ and $\Omega(\log m)$. Together with Proposition 4.8, we have the following theorem.

**Theorem 4.11.** *For uniform machines, the (S)PoA of EQUI is* $\min\{\Theta(\log m), 2r+1\}$.

### 4.3.3 Restricted Identical Machines

The upper bound of the price of anarchy of EQUI on restricted identical machines follows immediately by Immorlica et al. [47]. In [47], an instance was given which shows that any deterministic non-preemptive coordination mechanism has PoA $\Omega(\log m)$. However, EQUI is a preemptive policy, and the instance cannot be adapted. In this section, we show that the price of anarchy of the EQUI policy on restricted identical machines is also $\Omega(\log m)$ using another instance.

**Theorem 4.12.** *For restricted identical machines, the (S)PoA is* $\Theta(\log m)$.

*Proof.* The upper bound follows from [47]. We show now the lower bound. We adapt a game instance from a previous proof for uniform machines. Let $(m_j)_{j=0}^{k}$ be a sequence defined as $m_0 = 1, m_1 = 2$ and $m_j = m_0 + \ldots + m_{j-1}$, i.e., $m_j = 3 \cdot 2^{j-2}$ for every $j \geq 2$. Let $m = \sum_{j=0}^{k} m_j = 3 \cdot 2^{k-1}$. Hence $k = \Omega(\log m)$.

In the instance, there are $m$ machines which are divided into $k+1$ groups $G_0, \ldots, G_k$ where group $G_j$ consists of $m_j$ machines. There are also $k+1$ job groups $J_0, J_1, \ldots, J_k$ where group $J_j$ contains $3 \cdot 2^j m_j$ jobs of processing time $2^{-j}$.

We first describe a schedule $\mu$ which will be proved to be a Nash equilibrium. On each machine in group $G_j$ for $0 \leq j \leq k$, there are $2^{j+1}$ jobs of length $2^{-j}$ and for every $j < i \leq k$, there are $2^i$ jobs of length $2^{-i}$. The strategy set of each job is the following. Jobs in group $J_j$ can be scheduled on all $m_j$ machines of group $G_j$. Moreover, a job in $J_j$ can be additionally scheduled on its current machine in $\mu$.

We claim that $\mu$ is a equilibrium. Observe that on each machine of group $G_j$, there are exactly $2^{i+1}$ jobs of processing time at least $2^{-i}$ for all $j \leq i \leq k$ and the total load of jobs with processing time strictly smaller than $2^{-i}$ (on the machine) is $k-i$. Thus, the cost of each job in group $J_i$ is $k-i+2$ in $\mu$ and if a job switches the strategy, its cost would be strictly greater than $k-j+2$. In addition, using Lemma 4.6 and by the same argument as in Lemma 4.10, we have that this equilibrium is indeed a strong one.

If we schedule evenly all jobs of group $J_j$ on $m_j$ machines of $G_j$ for $0 \leq j \leq k$ then the makespan is bounded by 3, so $OPT \leq 3$. The makespan of the strong equilibrium above is $k+1$, which gives the (S)PoA is at least $(k+1)/3 = \Omega(\log m)$. □

### 4.3.4 Unrelated Machines

In this section, we prove that the PoA of the game under the EQUI policy is upper bounded by $2m$. Interestingly, without any knowledge of jobs' characteristics, the inefficiency of EQUI – a non-clairvoyant policy – is the same up to a constant compared to that of SPT – the best strongly local policy with price of anarchy $\Theta(m)$.

**Theorem 4.13.** *For unrelated machines, the price of anarchy of policy EQUI is at most $2m$.*

*Proof.* For job $i$, let $q_i$ be the smallest processing time of $i$ among all machines, i.e., $q_i := \min_j p_{ij}$ and let $Q(i) := \text{argmin}_j p_{ij}$ be the corresponding machine. Without loss of generality we assume that jobs are indexed such that $q_1 \leq q_2 \leq \ldots \leq q_n$. Note that $\sum_{i=1}^{n} q_i \leq m \cdot OPT$, where $OPT$ is the optimal makespan, as usual. First, we claim the following lemma.

**Lemma 4.14.** *In any Nash equilibrium, the cost $c_i$ of job $i$ is at most*

$$2q_1 + \ldots + 2q_{i-1} + (n-i+1)q_i. \tag{4.6}$$

*Proof.* The proof is by induction on $i$. The cost of job 1 on machine $Q(1)$ would be at most $nq_1$, simply because there are at most $n$ jobs on this machine. Therefore the cost of job 1 in the Nash equilibrium is also at most $nq_1$. Assume the induction hypothesis holds until index $i-1$. Consider job $i$. Since the strategy profile is a Nash equilibrium, $i$'s current cost is at most its cost if moving to machine $Q(i)$. We distinguish different cases. In these cases, denote $c'_i$ as the new cost of $i$ if it moves to machine $Q(i)$

**Case all jobs $t$ scheduled on machine $Q(i)$ satisfy $t > i$.** This case is very similar to the basis case. There are at most $n-i$ jobs on machine $Q(i)$, beside $i$. The completion time of job $i$ is then at most $(n-i+1)q_i$ which is upper bounded by (4.6). For the remaining cases, we assume that there is a job $i' < i$ scheduled on $Q(i)$.

**Case there is a job $t < i$ on machine $Q(i)$ such that $p_{tQ(i)} \geq p_{iQ(i)} (= q_i)$.** Since $p_{tQ(i)} \geq q_i$, the new cost of job $i$ is not more than the new cost of job $t$. Moreover, the new cost of job $t$ is increased by exactly $q_i$, so the new cost of $i$ is bounded by

$$\begin{aligned} c'_i &\leq c_t + q_i \\ &\leq 2q_1 + \ldots + 2q_{t-1} + (n-t+1)q_t + q_i \\ &= 2q_1 + \ldots + 2q_{t-1} + 2(i-t)q_t + (n-2i+t+1)q_t + q_i \\ &\leq 2q_1 + \ldots + 2q_{t-1} + 2q_t + \ldots + 2q_{i-1} + (n-i+1)q_i, \end{aligned}$$

where the first inequality uses the induction hypothesis and the last inequality is due to $t < i$ and $q_t \leq q_{t+1} \leq \ldots \leq q_i$.

**Case every job $t$ scheduled on machine $Q(i)$ with $p_{tQ(i)} \geq q_i$ satisfies $t \geq i$.** Since we are not in the first two cases, there is a job $t < i$ on machine $Q(i)$ with $p_{tQ(i)} < q_i$. Let $i'$ be the job of greatest index among all jobs scheduled on $Q(i)$ with smaller processing time than $q_i$. All jobs $t$ scheduled on $Q(i)$ and having smaller processing time than that of $i$, also have smaller index because $q_t \le p_{tQ(i)} \le q_i$. Therefore $i'$ is precisely the last job to complete before $i$. At the completion time of $i'$ there are still $q_i - p_{i'Q(i)} \le q_i - q_{i'}$ units of $i$ to be processed. By the case assumption, there are at most $(n-i)$ jobs with processing time greater than that of $i$. Therefore the new cost of $i$ is at most

$$\begin{aligned} c'_i &= c_{i'} + (n-i+1)(q_i - q_{i'}) \\ &\le 2q_1 + \ldots + 2q_{i'-1} + (n-i'+1)q_{i'} + (n-i+1)(q_i - q_{i'}) \\ &= 2q_1 + \ldots + 2q_{i'-1} + (i-i')q_{i'} + (n-i+1)q_i \\ &\le 2q_1 + \ldots + 2q_{i'-1} + (q_{i'} + \ldots + q_{i-1}) + (n-i+1)q_i \\ &\le 2q_1 + \ldots + 2q_{i-1} + (n-i+1)q_i \end{aligned}$$

where the first inequality uses the induction hypothesis and the third inequality is due to the monotonicity of the sequence $(q_j)_{j=1}^n$.

□

Since the term $2q_1 + \ldots + 2q_{i-1} + (n-i+1)q_i$ is increasing in $i$ and at $i = n$ this term is $2\sum_{i=1}^n q_i \le 2m \cdot OPT$, the cost of each job in an equilibrium is bounded by $2m \cdot OPT$, so the price of anarchy is at most $2m$. □

We provide a game instance showing that the upper bound analyzed above is tight. The instance is inspired by the work of Azar et al. [9]. In the following lemma, we prove the lower bound of the PoA of the game under the EQUI policy.

**Lemma 4.15.** *The (strong) price of anarchy of EQUI is at least $(m+1)/4$.*

*Proof.* Let $n_j := \frac{2(m-1)!}{(j-1)!}$ and $n := \sum_{j=1}^m n_j$. Consider the set of $m$ machines and $m$ groups of jobs $J_1, J_2, \ldots, J_m$. In group $J_j$ $(1 \le j \le m-1)$, there are $n_j$ jobs that can be scheduled on machine $j$ or $j+1$ except the last group $(J_m)$ which has a single job that can be only scheduled on machine $m$. Each job in group $J_j$ $(1 \le j \le m-1)$ has processing time $p_{jj} = \frac{(j-1)!}{(m-1)!} = \frac{2}{n_j}$ on machine $j$ and has processing time $p_{j,j+1} = \frac{j!}{2(m-1)!} = \frac{1}{n_{j+1}}$ on machine $j+1$. The job in $J_m$ has processing time $p_{mm} = 1$ on machine $m$.

Consider the strategy profile in which half of the jobs in $J_j$ $(1 \le j \le m-1)$ are scheduled on machine $j$ and the other half are scheduled on machine $j+1$ (jobs in $J_m$ are scheduled on machine $m$). We claim that this strategy profile is a Nash equilibrium. Note that the cost of jobs in the same group and scheduled on the same machine are the same. The cost of each job in group $J_j$ on machine $j$ is the load of the machine, because its processing time is greater than that of jobs in group $J_{j-1}$ on machine $m$, and this load equals $\frac{n_{j-1}}{2}p_{j-1,j} + \frac{n_j}{2}p_{jj} = \frac{j-1}{2} + 1 = \frac{j+1}{2}$. Each job in group $J_j$ has smaller processing time than that of each job in group $J_{j+1}$ on machine $j+1$, thus the cost of the former is $\frac{n_j + n_{j+1}}{2}p_{j,j+1} = \frac{j+1}{2}$. Hence, no job in group $J_j$ $(1 \le j \le m-1)$ has an incentive to move and the job in group $J_m$ cannot switch its strategy. Therefore, the strategy profile is an equilibrium.

Moreover, we prove that this equilibrium is indeed a strong one. Suppose that it is not a strong equilibrium, i.e., there is a coalition $S$ such that all jobs in $S$ can strictly decrease their cost. Again, by Lemma 4.6, the number of jobs on each machine remains the same after the move of $S$. We call a job in group $J_j$ *moving up* if it moves from machine $j$ to $j+1$ and *moving down* if it moves from machine $j+1$ to $j$. First, we claim that no job has an incentive to move up. If a job in group $J_j$ moves up, as only jobs in $J_j$ and $J_{j+1}$ can use machine $j+1$ and $p_{j,j+1} < p_{j+1,j+1}$, its new cost would be $p_{j,j+1} \cdot (n_j + n_{j+1})$ which equals its old cost. Hence, no one can strictly decrease its cost by moving up. Among all jobs in $S$, consider the one who moves down to the machine $j^*$ of smallest index. By the choice of $j^*$, there is no job moving down from machine $j^*$ and as claimed above, no job moving up from $j^*$. Hence, the job moving to machine $j^*$ cannot strictly decrease its cost – that contradicts to the assumption that all jobs in $S$ strictly get better off. Therefore, the equilibrium is a strong one.

Consider a schedule in which jobs in group $J_j$ $(1 \leq j \leq m)$ are assigned to machine $j$ and this schedule has makespan 2, hence $OPT \leq 2$. The makespan of the above (strong) Nash equilibrium is the load on machine $m$, that is equal to $(m+1)/2$. Then, the (strong) price of anarchy is at least $(m+1)/4$. □

# Chapter 5

# Online Auction of Perishable Items

## 5.1 Introduction

**Online Auction** Auction design is an important part of Mechanism Design. Roughly speaking, in an auction, the auctioneer first collects bids from bidders and then decides who receive the service and how much they have to pay. The auction is designed in such a way that it yields the desired goal for the auctioneer, for example: maximizing the profit, maximizing the welfare or making the auction to be truthful.

Inspired by Adwords of Google and Yahoo!, a phenomenally fast-growing market, auction is a motivation and a new source of important problems that attract a lot of attention and have been extensively studing by economists, computer scientist, .... In this chapter, we investigate the online auction truthfulness as the desired property.

**Auction problem** Consider the auction in which at each regular period of time, the auctioneer has a non-storable *perishable* identical good that he has to sell immediately to some bidder (for example, electricity, ice-cream ... which are hardly or costly to be stored). Bidders arrive online. The presence of bidder $i$ is only known to the auctioneer at $i$'s arrival time $r_i$. Each bidder $i$ has a departure time $d_i$ and has an own *private* value $w_i$ on the allocation of $p_i$ goods and 0 if he gets less than $p_i$ goods before his departure. At the arrival time, bidder $i$ announces his departure time $d_i$, his demand $p_i$ and his bid $b_i$ on the demand. We are interested in the case all bidders' demands are bounded, meaning $p_i \leq k \ \forall i$ for some $k$. This assumption is natural since an auction is usually held for a period of time, say a day, so a bidder cannot ask for a huge quantity since the providing capacity of the auctioneer in a day is limited. The auction of course can begin again next day. The goal of the auctioneer is to design a truthful auction which maximizes the total *welfare* – the sum $\sum_i w_i$ over all satisfied bidders.

This problem belongs to the single-parameter domain, introduced in Section 1.3. Hence, using the framework in that section, it is sufficient to design a monotone allocation algorithm for the optimization problem and verify that the critical-payment scheme can be computed efficiently.

**Optimization problem** This can be seen as an online-scheduling problem on a single machine. More formally, we consider the online scheduling problem with preemption, where each job $j$ is revealed at release time $r_j$, has processing time $p_j$, deadline $d_j$ and weight $w_j$

where $p_j \leq k$ for some constant $k$. Preemption of the jobs is allowed, meaning that a job $i$ could be scheduled in different separated time intervals, as long as the total processing time is $p_i$. The goal is to maximize the total weight of all jobs completed on time. This problem could be denoted as $1|\text{online-}r_i; \text{pmtn}|\sum w_i(1-U_i)$, according to the notation of [19].

### 5.1.1 Related work

**Optimization** This model is also called the *preemptive model with resume*, which contrasts with the *preemptive model with restarts* [25], where a job can be interrupted, but when it is scheduled again, it must be scheduled from the beginning.

It is known that the problem without the condition of bound on job processing times has an unbounded deterministic competitive ratio [12], so different directions of research were considered. One is to see if randomization helps, and indeed in [48] a constant competitive randomized algorithm was given, although with a big constant. Another direction of research is to consider resource augmentation, and in [50] a deterministic online algorithm was presented which has constant competitive ratio provided that the algorithm is allowed a constant speedup of its machine compared to the adversary.

**Bounded processing time, unit weights** (Case $\forall j : p_j \leq k, w_j = 1$) The offline problem can be solved in time $O(n^4)$ [10] already when the processing time is unbounded. Baruah et al. [12] showed that any deterministic online algorithm is $\Omega(\log k/\log\log k)$-competitive where the processing times, release times and deadlines of jobs in their model can be arbitrary fractions. Moreover, algorithm Shortest Remaining Processing Time First is $O(\log k)$-competitive, which is claimed by [48]. The same paper provides a constant competitive randomized algorithm, however with a large constant.

**Bounded processing time, arbitrary weights** (Case $\forall j : p_j \leq k$) For fixed $k$ the offline problem has not been studied to our knowledge, and when the processing times are unbounded the offline problem is $\mathcal{NP}$-hard by a trivial reduction from Knapsack Problem. It is known that any deterministic online algorithm for this case has competitive ratio $k/(2\ln k) - 1$ [63]. For the variant with only tight jobs, [15] provide an $O(\log k)$-competitive randomized online algorithm and show a $\Omega(\sqrt{\log k/\log\log k})$ lower bound for any randomized competitive algorithm against an oblivious adversary.

**Equal processing time, unit weights** (Case $\forall j : p_j = k, w_j = 1$) The offline problem can be solved in time $O(n \log n)$ [51], and it is well known that the same algorithm can be turned into a 1-competitive online algorithm, see for example [64].

**Equal processing time, arbitrary weights** (Case $\forall j : p_j = k$) The offline problem can be solved in time $O(n^4)$ [11]. For $k = 1$ the problem is well studied, and the deterministic competitive ratio is between 1.618 and 1.83 [53, 33]. A deterministic lower bound 2.59 is given in [17].

**Mechanism Design** Hajiaghayi et al. [45] considered an online auction problem similar to ours but bidders are interested in a single item. They showed that the greedy algorithm — the algorithm allocates every item to the available customer who has not been satisfied yet and the highest bid — was monotone. Since Kesselman et al. [49] showed that greedy algorithm is 2-competitive, this gave a truthful mechanism that is 2-competitive. Hajiaghayi et al. [45]

also proved that for no deterministic truthful mechanism has competitive ratio smaller than 2.

### 5.1.2 Contribution

We design monotone algorithms for this online optimization problem and analyze the competitive ratio as a function of $k$ in different cases.

**Bounded processing time, unit weights** (Case $\forall j : p_j \leq k, w_j = 1$) As mentioned earlier, Baruah et al. [12] showed that any deterministic online algorithm is $\Omega(\log k/\log\log k)$-competitive where the processing times, release times and deadlines can be arbitrary fractions. To get a ratio $R$, Baruah et al. construct a sequence with largest to smallest processing time ratio $\frac{1}{2}(2R+1)^R$, and rational values of job parameters (release times, deadlines, processing times). In our construction all these parameters are integers and yet the largest processing time of job in our lower bound construction is only $R \cdot R!$. Thus using more restricted sequences we get a better constant in the $\Omega(\log k/\log\log k)$ lower bound. Additionally, we revisit and give a concise proof of the known fact that algorithm Shortest Remaining Processing Time First is $O(\log k)$-competitive.

**Bounded processing time, arbitrary weights** (Case $\forall j : p_j \leq k$) We show that any deterministic online algorithm for this case has competitive ratio at least $k/\ln k$. In addition we show that the standard Smith Ratio Algorithm has ratio $\Theta(k)$, and provide an optimal online algorithm that reaches the ratio $O(k/\log k)$.

**Equal processing time, arbitrary weights** (Case $\forall j : p_j = k$) We show that for $k \geq 2$ the ratio is upper-bounded by 5. Moreover, we revisit the lower bound and give the exact value $3\sqrt{3}/2 \approx 2.598$ as a lower bound for any deterministic algorithm.

As our algorithms are monotone, using the framework described in Section 1.3, we get a truthful optimal online auction in case all demands of bidders are bounded and a truthful constant-competitive auction in case all demands are the same.

### 5.1.3 Organization

In Section 5.2, we represent our general technique to design competitive algorithms for the optimization problem and to analyze these algorithms. In Section 5.3, we study the case in which jobs have the same weight with bounded processing time. In Section 5.4, we investigate the case of weighted jobs with bounded processing times. In Section 5.5, we study an interesting restricted case in which all weighted jobs have the same length.

## 5.2 Preliminaries

For a job $i$ we denote its release time by $r_i$, its deadline by $d_i$, its processing time by $p_i$ and its weight by $w_i$. All these quantities, except possibly $w_i$, are integers. Let $q_i(t)$ be the remaining processing time of job $i$ for the algorithm at time $t$. When there is no confusion, we simply write $q_i$. We say that an algorithm schedules a job at time $t$ meaning that the algorithm executes an unit of this job in interval $[t, t+1)$. A job $i$ is *pending* for the algorithm at time $t$ if it has not been completed before and $r_i \leq t$ and $t + q_i(t) < d_i$. Let $j$ be a job which is not completed by the algorithm. The *critical time* of $j$ is the latest time when $j$ was still pending for the algorithm. In the other words, the critical time $s$ of job $j$ for the algorithm is the time that if the algorithm does not schedule $j$, it cannot schedule $j$ anymore, i.e. $s = \max\{\tau : \tau + q_j(\tau) = d_j\}$. A direct observation from the definition is that (w.l.o.g.) at the critical time of a job $j$, which is not completed by the algorithm, the algorithm schedules some unit of another job.

Throughout the paper we will analyze our algorithms with similar charging schemes, that have the rough following outline: For every job $j$ completed by the adversary we consider its $p_j$ units. Each unit of job $j$ will charge $w_j/p_j$ to some job $i_0$ completed by the algorithm. The charging scheme will satisfy that every job $i_0$ completed by the algorithm receives no more than $Rw_{i_0}$ in total, which implies competitive ratio $R$ for the algorithm.

More precisely we distinguish individual units scheduled by the algorithm, where unit $(i, a)$ stands for an execution of job $i$ when its remaining processing time was $a$. In particular a job $i$ completed by the algorithm consists of the units $(i, p_i), (i, p_i - 1), \dots, (i, 1)$. With every unit $(i, a)$ we associate a *capacity* $\pi(i, a)$ that depends on $w_i$ and $a$, the exact value will be different from proof to proof. The algorithms, with their capacities, will be designed in such a way that they satisfy these following properties.

**Increasing property:** If the algorithm schedules $(i, a)$ at $t$ with $a > 1$ and $(i', a')$ at $t + 1$, then $\pi(i', a') \geq \pi(i, a - 1) \geq \pi(i, a)$,

**Validity:** Let $(i, a)$ be an unit scheduled by the algorithm while $j$ was pending. Then, $\pi(i, a) \geq w_j/p_j$.

Informally, the increasing property means that between two consecutive moments that the algorithm completes some jobs, the capacities of scheduled units are increasing. The validity property ensures that the capacity of unit $(i, a)$ is large enough to receive the charge from an unit of job $j$.

In general there will be 3 types of charges in the charging scheme. Let $(j, b)$ be a unit of job $j$ scheduled by the adversary at time $t$.

**Type 1:** If the algorithm already completed $j$ by time $t$, then charge $w_j/p_j$ to $j$.

**Type 2:** Otherwise if the algorithm schedules a job unit $(i, a)$ at time $t$ that has capacity at least $w_j/p_j$ then we charge $w_j/p_j$ to $i_0$, where $i_0$ is the next job completed by the algorithm from time $t$ on (including $t$).

**Type 3:** In the remaining case, the job $j$ is not pending anymore for the algorithm by algorithm's validity and the increasing property. Let $s$ be the critical time of $j$. We charge $w_j/p_j$ to $i_0$, where $i_0$ is the first job completed by the algorithm from time $s$ on.

Clearly every job $i_0$ completed by the algorithm can get at most $w_{i_0}$ type 1 charges in total. We can bound the other types as well.

**Lemma 5.1.** *Let $\mathcal{J}$ be the set of job units that are type 3 charged to a job $i_0$ completed by the algorithm, which satisfies the validity and the increasing property. Then there are strictly less than $p$ units $(j, b) \in \mathcal{J}$ with $p_j \leq p$. In particular $|\mathcal{J}| \leq k - 1$, if all jobs have processing time at most $k$. Moreover for each $(j, b) \in J$ it holds that $w_j/p_j \leq \pi(i_0, 1)$.*

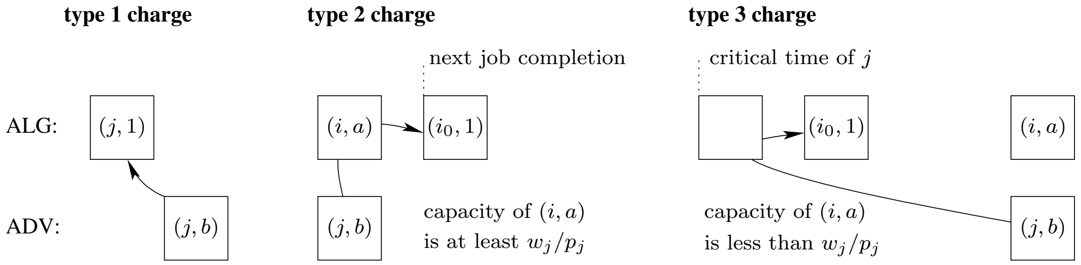

Figure 5.1: The general charging scheme

*Proof.* To be more precise we denote the elements of $\mathcal{J}$ by triplets $(s,t,j)$ such that a job unit $(j,b)$ scheduled at time $t$ by the adversary is type 3 charged to $i_0$ and the critical time of $j$ is $s$. Let $t_0 \geq s$ be the completion time of $i_0$ by the algorithm. As the charge is of type 3, between $s$ and $t_0$ there is no idle time, nor a job completion, so by increasing property and validity of the algorithm the capacities of all units in $[s,t_0]$ are at least $w_j/p_j$. However by definition of type 3 charges, the algorithm schedules at $t$ some unit with capacity strictly smaller than $w_j/p_j$, so $t_0 < t$.

Now let $a = q_j(s)$. Since $s$ is the critical time we have $s + a = d_j$. However since the adversary schedules $j$ at time $t$ we have $t < d_j$. Thus $t - s < a \leq p_j$. Note that all triplets $(s,t,j) \in \mathcal{J}$ have distinct times $t$. The first part of the lemma follows from the observation that there can be at most $c-1$ pairs $(s,t)$ with distinct $t$ that satisfy $s \leq t_0 < t$ and $t - s < c$.

Since $j$ was pending at time $s$, whatever the algorithm scheduled then had capacity at least $w_j/p_j$. By increasing property of the algorithm this holds also at time $t_0$, so $\pi(i_0,1) \geq w_j/p_j$. □

**Lemma 5.2.** *Suppose that there is a constant $0 < \rho < 1$ such that $\pi(i,a) \geq \pi(i,a+1)/\rho$ where $\pi$ is the capacity of the algorithm which satisfies the validity and the increasing property. Then the total charge of type 2 that a job $i_0$ receives is at most $w_{i_0}/(1-\rho)$.*

*Proof.* The algorithm has the property that as long as it does not complete a job, it schedules units that have capacities increasing at least by factor $1/\rho$ each. More formally, let $t_0$ be the completion time of $i_0$, and let $s$ be the smallest time such that $[s,t_0)$ contains no idle time and no job completion. Then every unit scheduled at time $t_0 - i$ for $0 \leq i \leq t_0 - s$ has capacity at most $w_{i_0}\rho^i$, since $\pi(i_0,1) = w_{i_0}$. Thus the total type 2 charge is bounded by a geometric series,

$$w_{i_0}(1 + \rho + \rho^2 + \rho^3 \ldots) = w_{i_0}/(1-\rho).$$

□

Actually all our algorithms, except the one from the last section, will be *priority based algorithms*, in the sense that the capacity $\pi$ plays also the role of a priority, and every time the highest priority job is scheduled. More formally, the priority of $j$ is $\pi(j,q_j)$. For such algorithms, increasing property and validity are properties of the function $\pi$ rather than of the algorithm itself. We will adapt this general scheme to individual algorithms in the next sections. Note that our algorithms need to select jobs only at release times or completion times of some jobs.

## 5.3 Bounded Processing Time, Unit Weight Model

In this section we consider instances over jobs $j$ such that $p_j \leq k$ for some $k$ and $w_j = 1$. First, we study the lower bound of the competitive ratio for deterministic algorithms.

**Theorem 5.3.** *Any deterministic online algorithm has ratio $\Omega(\log k/\log\log k)$.*

*Proof.* Fix some deterministic algorithm. We will define an instance denoted $I(\ell, 0, 0)$ from which the algorithm can complete at most a single job, and the adversary can complete $\ell$ jobs. Moreover all jobs have processing time at most $(\ell+1)!$. So if we choose $\ell = \lfloor \ln k/\ln\ln k\rfloor - 1$, then the processing time is not more than

$$\begin{aligned}(\ell+1)! &\leq \left\lfloor \frac{\ln k}{\ln\ln k}\right\rfloor ! \leq \left(\frac{\ln k}{2\ln\ln k}\right)^{\frac{\ln k}{\ln\ln k}} \\ &= \exp\left((\ln\ln k - \ln 2 - \ln\ln\ln k)\cdot\frac{\ln k}{\ln\ln k}\right) \leq \exp(\ln k) = k.\end{aligned}$$

Let $\ell \geq 1, s, e \geq 0$ be integers. Let $f$ be a function defined as $f(1, e) = e+1$ and for $\ell > 1$,

$$f(\ell, e) = \max\{e, f(\ell-1, 0)\} + f(\ell-1, 0) + f(\ell-1, \max\{e, f(\ell-1, 0)\}) \ . \tag{5.1}$$

We construct an instance $I(\ell, s, e)$ with the following properties.

- The adversary can schedule $\ell$ jobs from this instance.
- The algorithm can schedule at most one job from this instance, and if he does then he spends strictly more than $e$ units on jobs from this instance, including jobs that he does not complete.
- All jobs $i$ from the instance satisfy $s \leq r_i$ and $d_i \leq s + f(\ell, e)$, and therefore also $p_i \leq f(\ell, e)$.

The basis case is easy, for $I(1, s, e)$ at time $s$ we release a tight job of length $e+1$. It satisfies the required properties.

Now we show how to construct $I(\ell+1, s, e)$. Let $b = f(\ell, 0)$, $a = \max\{e, b\}$ and $c = f(\ell, a)$. At time $s$ we release a job $A$ of length $a+c$ and deadline $s+a+b+c$, as well as a job $B$ of length $a+b$ and tight deadline. At time $s+a$, if the algorithm scheduled only $B$ in $[s, s+a)$, then we release instance $I(\ell, s+a, 0)$. Otherwise at time $s+a+b$ we release $I(\ell, s+a+b, a)$, see figure 5.2.

Let's verify that the construction satisfies the required properties, by induction on $\ell$. We already settled the basis case $\ell = 1$, so assume the claim holds for instances $I(\ell, s', e')$ for all $s', e' \geq 0$, and we will show it holds for $I(\ell+1, s, e)$ as well. By construction and induction every job $i$ from instance $I(\ell+1, s, e)$ is released not before $s$ and has deadline not after $s+a+b+c$, which is $f(\ell+1, e)$. So the third property is satisfied.

**In case the algorithm scheduled only $B$ in $[s, s+a)$:** At this point, if the algorithm completes $A$ or $B$, then in the interval $[s+a, s+a+b)$ there is not a single idle time left for another job. Therefore by induction hypothesis the algorithm can only schedule a single job. The algorithm already spent $a$ units on $B$, so if he does complete a job, then he spends strictly more than $a \geq e$ units on jobs from this instance. By induction hypothesis, the adversary can schedule $\ell$ jobs from the subinstance in the interval $[s+a, s+a+b)$, and schedule $A$ in the remaining time units $[s, s+a)\cup[s+a+b, s+a+c)$.

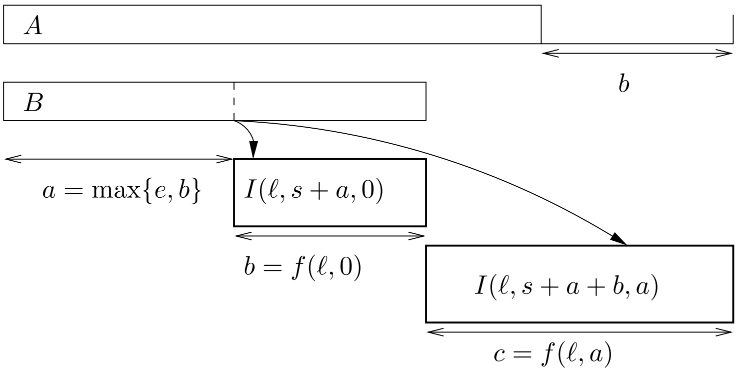

Figure 5.2: The construction of $I(\ell + 1, s, e)$

**Otherwise:** The algorithm cannot complete $B$, since the job is tight. Also if the algorithm completes some job from $I(\ell, s + a + b, a)$, then by induction hypothesis he spends strictly more than $a \geq e$ units on jobs from the sub-instance. This does not leave enough space to complete job $A$ in addition. And if the algorithm completes job $A$, then he spends $a + c > e$ units on it. Now the adversary can complete $B$ plus $\ell$ jobs from the sub-instance.

To complete the proof of the theorem, it remains to show that all jobs from $I(\ell, 0, 0)$ have processing time at most $(\ell + 1)!$. To this end, we prove by induction that

$$f(\ell, e) = \ell \cdot \max\{\ell!, (\ell - 1)! + e\}, \tag{5.2}$$

which implies that all jobs from $I(\ell, 0, 0)$ have processing time at most $\ell \cdot \ell! < (\ell + 1)!$. This trivially holds for $\ell = 1$. Now assume the claim holds for $\ell - 1$, and in particular $f(\ell - 1, 0) = (\ell - 1)(\ell - 1)!$. Therefore

$$\begin{aligned} f(\ell, e) &= \max\{e, f(\ell - 1, 0)\} + f(\ell - 1, 0) + f(\ell - 1, \max\{e, f(\ell - 1, 0)\}) \\ &= \max\{e, (\ell - 1) \cdot (\ell - 1)!\} + (\ell - 1) \cdot (\ell - 1)! + \\ &\qquad + (\ell - 1) \max\{(\ell - 1)!, (\ell - 2)! + \max\{e, (\ell - 1) \cdot (\ell - 1)!\}\} \\ &= \max\{e, (\ell - 1) \cdot (\ell - 1)!\} + (\ell - 1) \cdot (\ell - 1)! + \\ &\qquad + (\ell - 1) \cdot ((\ell - 2)! + \max\{e, (\ell - 1) \cdot (\ell - 1)!\}) \qquad (5.3) \\ &= \ell \cdot \max\{e, (\ell - 1) \cdot (\ell - 1)!\} + \ell! \\ &= \ell \cdot \max\{e + (\ell - 1)!, \ell!\} \end{aligned}$$

The equality (5.3) follows from $(\ell - 1)! < (\ell - 2)! + \max\{e, (\ell - 1) \cdot (\ell - 1)!\}$. □

The Shortest Remaining Processing Time Algorithm: is the greedy online algorithm that schedules at every step the pending job with the smallest remaining processing time. In other words, it is a priority-based algorithm with the priority function $\pi(i, a) = w_i/a = 1/a$. It was analyzed in [48] and we provide a concise proof for completeness.

**Proposition 5.4** ([48]). Shortest Remaining Processing Time First *is* $2H_k$*-competitive, where* $H_k$ *denotes the* $k$*-th harmonic number,* $1 + 1/2 + 1/3 + \ldots + 1/k$.

*Proof.* We use our general charging scheme. The algorithm with the definition of capacity satisfies properties of increasing property and validity. A simple observation of the algorithm is that whenever the algorithm schedules some job $i$ at time $t$, then some job will complete in $[t, t+k)$, either $i$ itself or some job with smaller processing time. In particular if $t_0$ is the completion time of some job $i_0$ by the algorithm, and $s$ is the smallest time such that $[s, t_0)$ contains no idle time nor completion, then $t_0 - s < k$ and the unit scheduled at time $t_0 - i$ for $0 \leq i \leq t_0 - s$ has capacity at most $1/(i+1)$. As a result the total type 2 charge to $i_0$ is at most $H_k$.

Lemma 5.1 states that there are at most $p-1$ type 3 charges to $i_0$ from jobs units $j$ with $p_j \leq p$. The worst case is when there is exactly one job unit $j$ with $p_j = p$ charging $1/p$ to $i_0$ for every $p = 2, 3, \ldots, k$. Therefore the total type 3 charge to $i_0$ is at most $H_k - 1$.

Total type 1 charge is at most $w_{i_0} = 1$, so this concludes the proof. □

The following lemma show that our analysis of Shortest Remaining Processing Time First is tight up to a constant factor.

**Lemma 5.5.** *There exists an instance in which the competitive ratio of* Shortest Remaining Processing Time First *is* $\frac{1}{2}\log k$.

*Proof.* The construction of the instance is similar to and simpler than the construction in Theorem 5.3. For simplicity, suppose that $k = 4^{\ell}$. Define a sequence $(t_u)$ where $t_0 = 0$ and $t_u = t_{u-1} + 4^{\ell-u}$ for all $1 \leq u \leq \ell - 1$. Consider the instance in which at time $t_u$, there are two released jobs: a job $A_u$ with processing time $3 \cdot 4^{\ell-u-1}$, deadline $t_u + 4^u$ and a tight job $B_u$ with processing time $2 \cdot 4^{\ell-u-1} + 1$ (deadline $t_u + 2 \cdot 4^{u-1} + 1$) for all $0 \leq u \leq \ell - 1$ except the last job $B_{\ell-1}$ has processing time 2 (instead of 3).

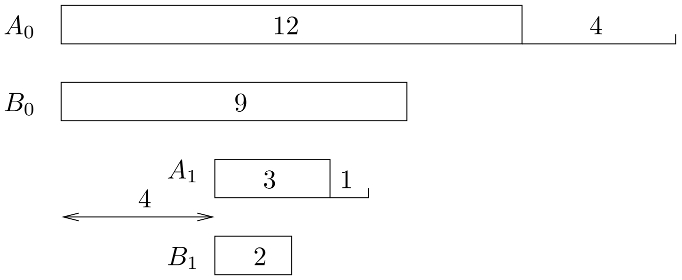

Figure 5.3: The instance in case $k = 16$ ($\ell = 2$)

In this instance, each time between $[t_u, t_{u+1})$ for $0 \leq u \leq \ell$, Shortest Remaining Processing Time First schedules an unit of job $B_u$ — the job with smallest remaining processing time. At time $t_{\ell-1}$, the algorithm completes only one job which is $B_\ell$ and it cannot complete any other jobs. However the adversary may schedule all jobs $A_u$ which gives the competitive ratio as $\ell = \log_4 k = \frac{1}{2}\log k$. □

## 5.4 Bounded Processing Time, Arbitrary Weight Model

In this section, we consider instances with arbitrary weights. First, we look at a straight forward generalization of Shortest Remaining Processing Time First.

The Smith Ratio Algorithm: schedules at every step the pending job $j$ that maximizes the Smith ratio $w_j/q_j$. In other words, it is a priority-based algorithm with the following priority function: $\pi(j, q_j) = w_j/q_j$.

**Theorem 5.6.** *The* Smith Ratio Algorithm *is* $2k$*-competitive.*

*Proof.* We use the general charging scheme. Monotonicity and validity of the algorithm easily follow from its definition and the priority function it uses. A job $i_0$ completed by the algorithm receives at most $w_{i_0}$ type 1 charge in total. Lemma 5.2 implies that each $i_0$ receives at most $kw_{i_0}$ type 2 charges in total, as for $\pi(i, a) = w_i/a$ the value of $\rho$ is $k/(k-1)$. By Lemma 5.1 $i_0$ receives at most $k-1$ type 3 charges, and each such charge is at most $\pi(i_0, 1) = w_{i_0}$. This concludes the proof. □

However, $2k$ does not match to the current lower bound. Ting [63] showed that the deterministic competitive ratio is at least $(k/2\ln k) - 1$. Here we slightly improve the lower bound by a multiplicative constant.

**Lemma 5.7.** *For any deterministic algorithm and* $k \geq 16$*, its competitive ratio is at least* $k/\ln k - o(1) > k/\ln k - 0.06$*.*

*Proof.* For convenience denote $R = k/\ln k$, and assume $k \geq 16$. Fix any deterministic algorithm. Consider the following instance, see figure 5.4. At time 0, the adversary releases a big job $B$ with weight $w_B = R$, processing time $k$ and deadline $k$, as well as a small job $A_1$ with weight, processing time and deadline all 1. Moreover, at each moment $0 \leq t \leq k-1$, if the algorithm scheduled only job $B$ in $[0, t)$, then the adversary releases a tight job $A_{t+1}$ of unit processing time at time $t$, and does not release any new job otherwise. The jobs $A_t$ have weights:

$$w(A_t) := \begin{cases} 1 & \text{if } t < R \ , \\ e^{t/R-1} & \text{if } t \geq R \ . \end{cases}$$

Note, job $A_t$ is released at time $t-1$.

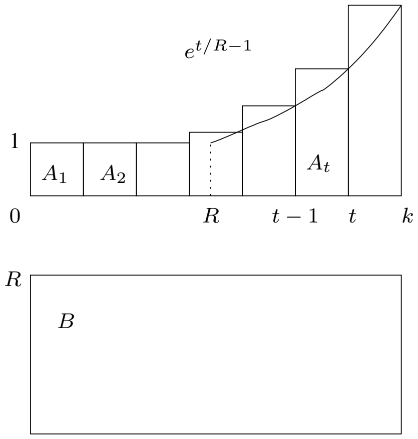

Figure 5.4: The construction of the lower bound

For any algorithm, there are three cases.

1. If the algorithm schedules a job $A_{t_0}$ with $t_0 < R$, then the adversary schedules job $B$ and the ratio is $R$.

2. If the algorithm schedules a job $A_{t_0}$ with $t_0 \geq R$, then the adversary schedules all jobs $A_t$ for $t = 1, \ldots, t_0$. The adversary's gain is

$$\begin{aligned} \lceil R\rceil - 1 + \sum_{t=\lceil R\rceil}^{t_0} e^{t/R-1} &\geq \lceil R\rceil - 1 + \int_{\lceil R\rceil -1}^{t_0} e^{t/R-1}\mathrm{d}t \\ &= \lceil R\rceil - 1 + \left[Re^{t/R-1}\right]_{\lceil R\rceil-1}^{t_0} \\ &= \lceil R\rceil - 1 - Re^{(\lceil R\rceil -1)/R-1} + Re^{t_0/R-1} \\ &\geq f(R) + Re^{t_0/R-1} = f(R) + Rw(A_{t_0}) \ , \qquad (5.4) \end{aligned}$$

where the inequalities follow from monotonicity of the function $e^{t/R-1}$, and

$$f(R) = \lceil R\rceil - 1 - Re^{(\lceil R\rceil -1)/R-1} \ .$$

So the adversary gain is at least $k/\ln k$ times the algorithm's gain plus $f(R)$. Eventually we will prove that $f(R) < 0$, but it tends to 0 as $R$ grows, and in particular $f(R) > -0.06$ for $k \geq 16$.

3. If the algorithm schedules job $B$, gaining $k/\ln k$, the adversary schedules all $k$ jobs $A_t$ from $t = 0$ to $k-1$. In that case, by (5.4) its gain is at least

$$f(R) + Re^{k/R-1} = f(R) + Re^{\ln k - 1} = f(R) + R \cdot k/e,$$

and we need it to be more than $f(R) + Rw(B) = f(R) + R^2$. This is true if $e \leq \ln k$ which holds for $k \geq e^e$ and in particular when $k \geq 16$.

Now we analyze the function $f(R)$. To this end note that

$$f(R) = f(m, \xi) = m - (m + \xi)\, e^{-\xi/(m+\xi)} \ ,$$

where $m$ is the largest integer strictly smaller than $R$ and $\xi = R - m \in (0, 1]$. As one can easily check, $f(m, \xi)$ is an increasing function of $m$ and a decreasing function of $\xi$. Since $f(m, 0) = 0$ for all $m$, $f(R) < 0$ for all $R$, but it tends to 0 as $R$ grows. As $R(k) = k/\ln k$ is an increasing function of $k$ ($k \geq 16$), the lowerbound on $f(R(k))$ follows, since $f(R(k)) > -0.06$ for $k = 16, 17, \ldots, 21$, and for $k > 21$ we have $R(k) > 7$, as well as $f(7, 1) > -0.06$.

As the algorithm's gain (w.l.o.g.) is at least 1, $f(R)$ divided by that gain is at least $f(R)$. □

Now we present a deterministic online algorithm that is optimal up to a multiplicative constant.

The Exponential Priority Algorithm: It is the priority-based algorithm with priority function of the form $\pi(j, a) = w_j \cdot \alpha^{a-1}$ for $\alpha = \alpha(k) = 1 - c^2 \cdot \ln k/k$, where $c = 1 - \epsilon$ for arbitrarily small $\epsilon > 0$.

In fact, the constant $\alpha$ depends on $k$, seemingly making Exponential Priority Algorithm a semi-online algorithm. However, the $\alpha(k)$ we use is an increasing function of $k$, so we can make the algorithm fully online by using the value $\alpha(k^*)$ in each step, where $k^*$ is the maximum processing time among all jobs released before that step. One can check that this fully online algorithm is still increasing — in the sense of section 5.2 — and one can derive bounds on competitive ratio of the algorithm by analyzing only the $\pi$ function defined by $\alpha(k^*)$ for the final value of $k^*$.

**Theorem 5.8.** *The* Exponential Priority Algorithm *is* $(3 + o(1))\, k/\ln k$*-competitive.*

*Proof.* As before, we use the general charging scheme. First of all we claim that the algorithm is increasing — it follows from $\alpha \leq 1$ — and that the algorithm is valid, which follows from the inequalities (5.5) and (5.6) below.

Let $i_0$ be a job completed by the algorithm. We bound the type 3 charges it can receive. Let $(j, b)$ be a unit scheduled by the adversary at time $t$ and that is type 3 charged to $i_0$, and let $s$ be the critical time of $j$. We will relate the weight of $j$ to the weight of $i$. At time $s$ the priority of $j$ was $\pi(j, a)$, where $a$ is the remaining processing time of $j$ at time $s$. By increasing property of the algorithm $\pi(j, a) \leq \pi(i_0, 1)$, in other words $w_j \alpha^{a-1} \leq w_{i_0}$, implying $w_j/p_j \leq w_{i_0}/(p_j \alpha^{a-1})$. This upper bound on the charge to $i_0$ is maximized when $a = p_j$.

Hence we turn our attention to the function $f(x) = x\alpha^{x-1}$, i.e. the denominator from the last bound. We are going to prove that for any set $\mathcal{J}$ of units type 3 charged to any job $i_0$ completed by the algorithm, it holds that $\sum_{(j,b)\in\mathcal{J}} 1/f(p_j) \in O(k/\log k)$. To this end, recall that Lemma 5.1 states that for every $p \leq k$ the number of $(j, b) \in \mathcal{J}$ such that $p_j \leq p$ is at most $p - 1$. We will apply it for $p = k/(c^2 \ln k)$.

We claim that for any natural $x$ and large enough $k$ the following holds.

$$f(x) \geq 1 \qquad \text{for } 1 \leq x \leq \frac{k}{c^2 \ln k} \ , \tag{5.5}$$

$$f(x) \geq \ln k \qquad \text{for } \frac{k}{c^2 \ln k} < x \leq k \ , \tag{5.6}$$

In particular,

$$\sum_{(j,b)\in\mathcal{J}} 1/f(p_j) \leq \frac{k}{c^2 \ln k} + \frac{k}{\ln k} = \frac{k}{\ln k}\left(1 + \frac{1}{c^2}\right) \ .$$

Putting things together, each job $i_0$ completed by the algorithm receives a type 1 charge of at most $w_{i_0}$. By Lemma 5.2 for $\rho = \alpha$ it can receive at most $w_{i_0} k/c^2 \ln k$ type 2 charges in total. And we just showed that type 3 charges are at most $w_{i_0}\left(1 + 1/c^2 + o(1)\right) k/\ln k$ in total. Together, this is $w_{i_0}(1 + 2/c^2 + o(1)) \cdot k/\ln k = w_{i_0}(3 + o(1)) \cdot k/\ln k$.

It remains to prove the claims (5.5) and (5.6). First let us observe that for every constant $c < 1$ and large enough $x$,

$$\left(1 - \frac{c}{x}\right)^x \geq \frac{1}{e} \ , \tag{5.7}$$

as for $x$ tending to infinity the left hand side tends to $e^{-c} > e^{-1}$.

Clearly $f(1) = 1$ and, by (5.7),

$$\begin{aligned} f(k) = k\left(1 - \frac{c^2 \ln k}{k}\right)^{k-1} &= k\left(1 - \frac{c^2 \ln k}{k}\right)^{\frac{ck}{\ln k}(k-1)\frac{\ln k}{ck}} \\ &\geq k\left(\frac{1}{e}\right)^{c(k-1)\frac{\ln k}{k}} \\ &= k \cdot k^{c(1-k)/k} = k^{(1-\epsilon+k\epsilon)/k} \geq \ln k \ , \end{aligned}$$

if $k$ is sufficiently large.

Now we observe that the sequence $(f(x))_{x=1}^{k}$ is non-decreasing for $x \leq k/(c^2 \ln k)$ and decreasing for $x > k/(c^2 \ln k)$. For this we analyse the ratio $f(x)/f(x-1) = \alpha x/(x-1)$, and see that it is at least 1 if and only if $x \geq k/(c^2 \ln k)$. Inequalities (5.5) and (5.6) follow. This completes the proof. □

**Theorem 5.9.** *There exists a truthful mechanism which is $(3 + o(1))\, k/\ln k$-competitive for online auction of perishable items where $k$ is the maximal demand over all bidders, i.e., $k = \max_i p_i$.*

*Proof.* Consider the mechanism that receives all bids $b_i$ and then invokes the Exponential Priority Algorithm to allocate items to bidders. Now, the priority function is of the form $\pi(i, a) = b_i \cdot \alpha^{a-1}$ where $\alpha$ is defined earlier and $(p_i - a)$ is the number of items that bidder $i$ have received so far. It is straightforward that if a satisfied bidder raises his bid then he is still satisfied since the priority is increased. Hence, it is sufficient to efficiently compute the critical payment for bidders. The following algorithm computes the payment for a satisfied bidder $i$.

**Payment algorithm** Let $B$ be the set of bidders arriving before time $d_i$.

1. Run the Exponential Priority Algorithm on $B \setminus \{i\}$.
2. Let $\pi(j, a)$ be the $p_i$-th smallest value of priority function in time interval $[r_i, d_i]$ of the run in step 1. (Note that, $\pi(j, a)$ may be equal to 0 and we can always choose the $p_i$-th smallest value as $d_i - r_i \geq p_i$ since $i$ is satisfied by the run of the algorithm on $B$.) Set the price to bidder $i$ to be $v_i = \pi(j, a) \cdot \alpha^{1-p_i}$. (Remark that $v_i \leq b_i$.)

The algorithm shows the high-level idea of computing the price for each player. In fact, we can compute the price for all bidders more efficiently. Let $B_t$ and be the set of bidders with their received items that are considered by the run of the allocation algorithm Exponential Priority Algorithm on $B$ at time $t$. Let $\tau(i)$ be the moment that $i$ receives his first item in the run of Exponential Priority Algorithm on $B$. Keep the sets $B_t$ and $\tau(i)$ in memory. In order to compute $v_i$, it is sufficient to run Exponential Priority Algorithm initially on $B_{\tau(i)}$ from time $\tau(i)$ to time $d_i$ without considering bidder $i$. Then take $\pi(j, a)$ as the $p_i$-th smallest priority value of this run.

If bidder $i$ bids less than price $v_i$ then by the allocation algorithm, he can receive at most $(p_i - 1)$ items. If he bids more than that price, he will receive at least $p_i$ items. Therefore, the payment algorithm above computes the critical payment. □

## 5.5 Identical Processing Time Model

In this section we consider restricted instances where every job has the same processing time $k \geq 2$ and an arbitrary weight. Let $\beta > 1$ be a constant that we will define later.

The conservative algorithm: If the previous time step was idle or a job completion, schedule the highest weight pending job. Otherwise if in the previous step some job $i$ was scheduled that is still pending now, then continue executing $i$, until a new job $j$ is released that is at least $\beta$ times heavier than $i$, in which case $j$ is executed (choosing the heaviest such job $j$ if there is a choice).

Using the general charging scheme, we can prove that this algorithm is 6.83-competitive.

**Theorem 5.10.** *The* Conservative Algorithm *has ratio* $2 + 2\beta + 1/(\beta - 1)$ *which is at most* 6.83 *when* $\beta = 1 + 1/\sqrt{2}$.

*Proof.* We adapt the general charging scheme from the previous section. A job unit $(i, a)$ scheduled by the algorithm has capacity $\beta w_i/k$. Note that the algorithm is increasing, since any job may only be interrupted by a heavier job. Now we turn to the algorithm's validity. Suppose the algorithm schedules $(i, a)$ in some step $t$. Then any job $j$ pending for the algorithm in that step has weight $w_j < \beta w_i$ by the algorithm's definition. As $\pi(i) = \beta w_i/k$, the algorithm is valid. Now we will bound the total charge received by a job $i_0$ completed by the algorithm, say at time $t_0$. Total type 1 charge is at most $w_{i_0}$. For the type 2 charge, let $s$ be the smallest time such that there is no idle time and no completion time in $[s, t_0)$. Let the jobs scheduled by the algorithm in this interval be $i_b, i_{b-1}, \ldots, i_0$, respectively. Then for every $c$: $b > c \geq 0$ the job $i_{c+1}$ was interrupted by $i_c$, hence $\beta w_{i_{c+1}} \leq w_{i_c}$ . So the total type 2 charge to $i_0$ is at most

$$\beta w_{i_0}(1 + 1/\beta + 1/\beta^2 + \ldots + 1/\beta^b) \leq \beta/(1 - 1/\beta) = \beta^2/(\beta - 1) = \beta + 1 + 1/(\beta - 1).$$

For the type 3 charge, we simply apply Lemma 5.1, and conclude that every job unit $j$ type 3 charged to $i_0$ is bounded by $w_j/k \leq \beta w_{i_0}/k$, and since there are at most $k-1$ type 3 charges, that makes at most $(k - 1)\beta w_{i_0}/k < \beta w_{i_0}$ in total. Adding the charges of all 3 types gives the required bound on the ratio. □

In Conservative Algorithm, a job $i$ can be interrupted by a new job with weight larger than $\beta$ time that of job $i$ where $\beta$ is fixed. This seems strong since the algorithm does not consider the remaining processing time of the current scheduling job. Intuitively, it is reasonable that to interrupt a job with large remaining processing time, a new job may have a slightly larger weight. However, to interrupt a job with small remaining processing time, a new job must have a weight significantly larger than that of the currently scheduling job. The following algorithm takes into account that idea and its analysis is more involved to the fact that all jobs have the same processing time.

The Priority Algorithm: At every step execute the pending job which maximizes the priority $\pi(j, q_j) = 2^{-q_j/k} \cdot w_j$.

**Theorem 5.11.** *The* Priority Algorithm *is 5-competitive.*

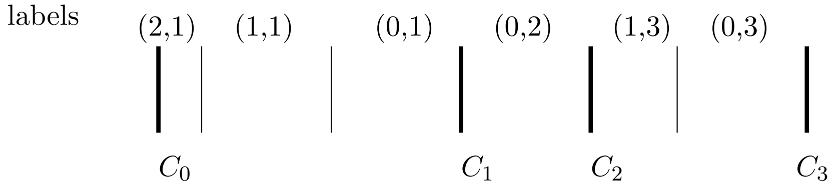

Figure 5.5: The intervals as used by the charging procedure.

*Proof.* The proof is based on a charging scheme, different from the general charging scheme.

Fix some instance. Consider the jobs scheduled by the algorithm and jobs scheduled by the adversary. Without loss of generality we assume that the adversary completes every job that he starts, and that he follows the Earliest Deadline First policy.

Every job $j$ scheduled by the adversary that is also completed by the algorithm, is charged to itself. From now on we ignore those jobs, and focus on remaining ones.

All jobs scheduled by the adversary will be charged to some jobs completed by the algorithm, in such a way that job $i$ completed by the algorithm receives a charge of at most $4w_i$ in total.

For convenience we renumber the jobs completed by the algorithm from 1 to $n$, such that the completion times are ordered $C_1 < \ldots < C_n$. Also we denote $C_0 = 0$. For every $i = 1, \ldots, n$ we divide $[C_{i-1}, C_i)$ further into intervals: Let $a = \lceil (C_i - C_{i-1})/k \rceil$. The first interval is $[C_{i-1}, C_i - (a-1)k)$. The remaining intervals are $[C_i - (b+1)k, C_i - bk)$ for every $b = a-2, \ldots, 0$. We label every interval $I$ with a pair $(b, i)$ such that $I = [s, C_i - bk)$ for $s = \max\{C_{i-1}, C_i - (b+1)k\}$.

The charging will be done by the following procedure, which maintains for every interval $[s, t)$ a set of jobs $P$ that are started before $t$ by the adversary and that are not yet charged to some job of the algorithm.

Initially $P = \emptyset$.
**For all** intervals $[s, t)$ as defined above in left to right order, do

- Let $(b, i)$ be the label of the interval.
- Add to $P$ all jobs $j$ started by the adversary in $[s, t)$.
- If $P$ is not empty, then remove from $P$ the job $j$ with the smallest deadline and charge it to $i$. Mark $[s, t)$ with $j$.
- If $P$ is empty, then $[s, t)$ is not marked.
- Denote by $P_t$ the current content of $P$.

**Lemma 5.12.** *For every interval $[s, t)$, all jobs $j \in P_t$ are still pending for the algorithm at time $t$.*

*Proof.* Assume that $P_t$ is not empty, and let $j$ be the job in $P_t$ with the smallest deadline.

First we claim that there is a time $s_0$, such that every interval contained in $[s_0, t)$ is marked with some job $j'$ satisfying $s_0 \le r_{j'}$ and $d_{j'} \le d_j$.

The existence of $s_0$ is shown by a kind of pointer chasing: Let $[s', t')$ be the interval where the adversary started $j$. So $j$ entered $P$ by the charging procedure at this interval. Job $j$ was in $P$ during all the iterations until $[s, t)$, so every interval between $t'$ and $t$ is marked with some job of deadline at most $d_j$. Let $\mathcal{M}$ be the set of these jobs. If for every $j' \in \mathcal{M}$ we have $s' \le r_{j'}$, we choose $s_0 = s'$ and we are done. Otherwise let $j' \in \mathcal{M}$ be the job with smallest release time. So $r_{j'} < s'$. Let $[s'', t'')$ be the interval where the adversary started $j'$. By the same argment as above, during the iteration over the intervals between $s''$ and $s'$, job $j'$ was in $P$. Therefore every such interval was marked with some job with deadline at most $d_{j'} \le d_j$. Now we repeat for $s''$ the argument we had for $s'$. Eventually we obtain a valid $s_0$, since $P$ was initially empty.

Now let $\mathcal{M}$ be the set of jobs charged during all intervals in $[s_0, t)$. In an Earliest Deadline First schedule of the adversary, job $j$ would complete not before $s_0 + (|\mathcal{M}| + 1)k$. But any interval has size at most $k$, so $t - s_0 \le |\mathcal{M}|k$. We conclude that $d_j \ge t + k$, which shows that $j$ is still pending for the algorithm at time $t$. □

**Lemma 5.13.** *Let $[s, t)$ be an interval with label $(b, i)$ and $j$ a job pending for the algorithm at some time $t_0 \in [s, t)$. Then $w_j \le 2^{1-b} w_i$.*

*Proof.* Let $u = C_i$ and let $x_{t_0}, x_{t_0+1}, \ldots, x_{u-1}$ be the respective priorities of the job units scheduled in $[t_0, u)$. Clearly the algorithm is $2^{-1/k}$-monotone, i.e. $x_{t'} \le 2^{-1/k} x_{t'+1}$ for every $t' \in [t_0, u)$.

We have $x_{u-1} = 2^{-1/k} w_i$, since $i$ completes at $u$ and the remaining processing time of $i$ at time $(u-1)$ is 1. Now the priority of $j$ at time $t_0$ is at most $2^{-1} w_j$, therefore

$$2^{-1} w_j \le x_{t_0} \le 2^{-(u-1-t_0)/k} x_{u-1} \le 2^{-(u-t_0)/k} w_i = 2^{-b} w_i.$$

□

This lemma permits to bound the total charge of a job $i$ completed by the algorithm. Let $a = \lceil (C_i - C_{i-1})/k \rceil$. Then $i$ gets at most one charge of weight at most $2^{1-b} w_i$ for every $b = a-1, \ldots, 0$. Summing the bounds shows that job $i$ receives at most 4 times its own weight, plus one possible self-charge.

At time $t = C_n$ the algorithm is idle, so by Lemma 5.12, $P_t = \emptyset$. Therefore all jobs scheduled by the adversary have been charged to some job of the algorithm, and this completes the proof. □

**Theorem 5.14.** *There exists a truthful mechanism which is 5-competitive for online auction of perishable items where the bidders' demands are the same, i.e., $k = p_i$ $\forall i$.*

*Proof.* Consider the mechanism that receives all bids $b_i$ then invokes Priority Algorithm to allocate items to bidders. Once again, it is straightforward that if a satisfied bidder raises his bid then he is still satisfied. Moreover, the critical payment for satisfied bidders can be efficiently computed as in Theorem 5.9. Therefore, there exists a 5-competitive truthful mechanism for online auction of perishable items where the bidders' demands are the same. □

We will show a $(3\sqrt{3}/2)$-lower bound on competitive ratio of any deterministic algorithm. This counterpart clearly indicates that the problem becomes harder when jobs do not have unit length and preemption matters; compare to the unit length variant [53, 33]. The same construction was used before in [17], and it was claimed to yield 2.59 lower bound on the competitive ratio. However, in their proof, no exact value of the lower bound is given. Here we give the complete proof with the exact value of the instance.

**Lemma 5.15.** *Any deterministic online algorithm for the equal processing time with $k \geq 2$ has competitive ratio at least $3\sqrt{3}/2 \approx 2.598$.*

*Proof.* We describe the adversary's strategy for $k = 2$ only, as it can be easily adapted to larger values of $k$. Every job $j$ will have processing time 2 and will be tight, i.e. $d_j = r_j + p_j = r_j + 2$. W.l.o.g. the adversary completes the heaviest feasible subset of jobs, which can be specified once the sequence is finished. For the time being we need only describe what jobs are released in each step. We also assume that when there are pending jobs with positive weights, Alg will process one of them, and that it will never process a job with non-positive weight.

Initially ($t = 0$) the adversary releases a job with weight $x_0 = 1$. In every step $t > 0$ the adversary releases a job with weight $x_t$ that we specify later, unless the algorithm has already completed one job (this has to be the one with weight $x_{t-2}$). In that case the adversary releases no job at time $t$ and the sequence is finished. The adversary, in that case, completes every other job starting from the last one, for a total gain of

$$X_{t-1} = x_{t-1} + x_{t-3} + \ldots + x_{b+2} + x_b \ ,$$

where $b = t - 1 \bmod 2$, while Alg's gain is only $x_{t-2}$.

Now we describe the sequence $x_i$ that forces ratio at least $R = 1.5\sqrt{3} - \epsilon$ for arbitrarily small epsilon. As we later prove, there is a non-positive element $x_{i_0}$ in the sequence, so by previous assumptions the algorithm completes some job released before the step $i_0$.

If Alg completes a job released in step $t$, the ratio is

$$R_t = \frac{X_{t+1}}{x_t} = \frac{X_{t+1}}{X_t - X_{t-2}} \ ,$$

assuming $X_{-2} = X_{-1} = 0$. As we want to force ratio $R$, we let $R_t = R$, i.e.

$$X_{t+1} = R\,(X_t - X_{t-2})$$

for each $t > 0$. Note that this defines the sequence $x_i$, as $x_i = X_i - X_{i-2}$.

To prove existence of $i_0$, we introduce two sequences: $q_i = R \cdot X_{i-1}/X_{i+1}$ and $s_i = R - q_i = R(1 - X_{i-1}/X_{i+1})$. We shall derive a recursive formula defining $q_i$ and $s_i$, and then prove that $s_i$ is a strictly decreasing sequence. Next we prove by contradiction that $s_i \leq 0$ for some $i$. That will conclude the proof, as (assuming both $X_{i-1}$ and $X_{i+1}$ are positive). Assume that $s_i > 0$ for all $i$.

First observe that

$$X_i = R\,(X_{i-1} - X_{i-3}) = X_{i-1}\left(R - R\frac{X_{i-3}}{X_{i-1}}\right) = X_{i-1}\,(R - q_{i-2}) \ ,$$

which implies

$$q_i = R \cdot \frac{X_{i-1}}{X_{i+1}} = \frac{R}{(R - q_{i-1})(R - q_{i-2})} \ . \tag{5.8}$$

Rewriting (5.8) in terms of $s_i$ we get

$$s_i = R\left(1 - \frac{1}{s_{i-1}s_{i-2}}\right) \ , \tag{5.9}$$

and one can calculate that $s_0 = R$, $s_1 = R - 1/R$ and $s_2 = R(R^2 - 2)/(R^2 - 1)$, in particular $s_0 > s_1 > s_2 > 0$.

We prove by induction that $s_i$ is a decreasing sequence. Observe that

$$s_{i+1} - s_i = R\left(\frac{1}{s_{i-1}s_{i-2}} - \frac{1}{s_i s_{i-1}}\right) = R \cdot \frac{s_i - s_{i-2}}{s_i s_{i-1} s_{i-2}} < 0 \ ,$$

since by induction hypothesis $s_{i-2} > s_{i-1} > s_i$. Since the sequence $s_i$ is decreasing and by assumption lower bounded by 0, it converges to $g = \inf s_i$, such that $g \geq 0$. Moreover it is impossible that $g < 1$ since otherwise we would have $0 < s_{i_0-2}, s_{i_0-1} < 1$ for some $i_0$, which implies $s_{i_0} < 0$ by (5.9), a contradiction. So we have $g \geq 1$ and by (5.9)

$$g = R\left(1 - \frac{1}{g^2}\right) \ ,$$

or, equivalently,

$$P(g) = g^3 - Rg^2 + R = 0 \tag{5.10}$$

Since $R = 1.5\sqrt{3} - \epsilon$, the discriminant of $P$, which is $4R^2(R^2 - 27/4)$, is negative, i.e. $P$ has a single real root. As $P(-1) = -1$ and $P(0) = R > 0$, the sole real root of $P$ lies in $(-1, 0)$. In particular, it is negative, which proves $s_i$ is not lower-bounded by any non-negative constant. □

# Chapter 6

# Recent and Future Work

In this thesis, we have studied the existence and inefficiency of pure Nash equilibria in different games. We have also designed some coordination mechanism and truthful mechanism in different contexts by eliciting the self-interested behaviors of players. While we have succeeded in answering a few basic questions, we see our work as a very small step in order to well understand the complex interactions between selfish players and the structure of games. Inspired by this small step, we suggest some open questions and interesting unexplored directions as future work. We also summarize recent work related to these topics.

## 6.1 Cost-sharing mechanism in Connection Games

In Chapter 2, we have studied the Connection Games (on directed networks) and showed that it is $\mathcal{NP}$-hard to decide whether there exists an $\epsilon$-fair cost-sharing mechanism that induces the price of stability smaller than 3/2. The main question raised by Chen et al. [21] is still open.

**Open question 1.** Does there exist a cost-sharing mechanism that induces a constant price of stability?

Although there are many ways to share the cost of a network, the most natural is the Shapley cost-sharing mechanism. One hopes that this mechanism in fact is the best in sense of inefficiency for Connection Games on undirected networks. However, there is still a large gap between the upper and lower bound of the price of stability. The best lower bound is 12/7 [37] and the best upper bound is $\log(n)$ [2] where $n$ is the number of players in the game. In a special case where all players have the same source, the price of stability is recently settled to $O(\log n/ \log\log n)$ in [54]. A more special case, in which all vertices in the graph network are players and they all want to connect to a specified source $s$, is studied by Fiat et al. [37]. They proved that the price of stability is upper bounded by $O(\log\log n)$ in this case.

**Open question 2.** Close the gap of the price of stability of Connection Games on undirected networks

## 6.2 Voronoi Games

In Chapter 3, we have studied the social cost discrepancy of Voronoi Games on graphs and proved that it is $\Omega(\sqrt{n/k})$ and $O(\sqrt{n \cdot k})$. A straight-forward question is to close this gap.

**Open question 3.** Close the gap of the social cost discrepancy of Voronoi Games on graphs.

Inspired by this work, we believe that the social cost discrepancy is worth to be studied in different games and in a larger context. Moreover, the game is interesting for restricted graphs such as cycle graphs (studied by [55]) or lattice graphs.

## 6.3 Coordination Mechanism for Scheduling Games

In Chapter 4, we have first studied the existence of equilibria of the games under RANDOM and EQUI policies. While there always exists an equilibrium for EQUI, we only proves that the best-response dynamic may cycle in general and we settled the existence of Nash equilibria in restricted cases where jobs are balanced. Even RANDOM is a specific policy, studying whether the game under this policy possesses equilibria may contribute new ideas or new techniques as we realized in the unusual potential argument in case of uniform machines with balanced speeds.

**Open question 4.** Does the game under RANDOM policy possess pure Nash equilibrium?

Coordination mechanism aim to design policies such that the game under those policies always admits equilibria and the price of anarchy is small. As we have seen, there are three type of policies: non-clairvoyant, strongly local and local which correspond to different level of available information to machines. Between non-clairvoyant and strongly local policies, there is asymptotically no difference in term of inefficiency measured by the price of anarchy. However, there are a large gap of the price of anarchy between the best-known strongly local policy ($PoA = O(m)$ where $m$ is the number of machines) and the best-known local policy ($PoA = O(\log^2 m)$ [9, 16]). Note that, when all machines are controlled by a central server, meaning information about jobs are globally available, there is a 2-approximation algorithm [52]. Does the barrier of information is really an insurmountable obstacle?

**Open question 5.** Design a policy that always induces Nash equilibria with price of anarchy $o(\log^2 m)$.

In Scheduling Games, the egalitarian social cost (the makespan) is considered. However, it is interesting to consider the utilitarian function — defined as the sum of all players' costs — as the social cost and study the loss of performance of the game with respect to this cost.

## 6.4 Online Auctions

In chapter 5, we have designed truthful auctions which are $\Theta(k/\log k)$-competitive where the demands are bounded by $k$ and 5-competitive where all the demands are exactly $k$. The lower bound (2.59) for the latter is constructed using only online adversary without forcing the auction to be truthful. As designing truthful competitive auction is at least as hard as designing competitive algorithm, it is interesting to combine the truthful property in order to get a larger lower bound for truthful auctions.

**Open question 6.** Reduce the gap between the upper and lower bound for truthful auctions (and also for online optimization problem) in case all demands are the same.

# Appendix A

# Potential and equilibria

In the appendix, we present some results that are related but do not fall into the scope of the main text. In this chapter, we illustrate that the existence of equilibria in a game is not equivalent to the fact that the game is potential, i.e. the game may possess equilibria even thought the best-response dynamic has cycles. This fact is widely known and admitted in game theory and in mathematics, particularly in dynamical system theory. Informally, it is not hard to construct a normal form game with well chosen entries such that the best-response dynamic, with a specific initial strategy profile, cycles but the game always admits a Nash equilibrium. However, such games seem artificial in the sense that these games, with artificial matrix entries, may be inspired by no real situation. The issue here is that in concrete succinct games which are natural, for example Connection Games, Voronoi Games, Scheduling Games, ..., maybe the existence of equilibria is equivalent to the existence of potential functions. Until now, no concrete example illustrated that fact. In the following, we describe a concrete game in which the best-response dynamic cycles, meaning the game is not a potential game, but the game always admit an equilibrium.

**The game:** Given an undirected graph $G(V, E)$ with cost $c : E \rightarrow \mathbb{R}^+$. There are $n$ players that have the same sink $t$ and player $i$ has source $s_i$. Each player want to connect her source to the sink and they can do it by buying edges. An edge can be only bought entirely, meaning that an edge is sold only to one player and that player pays entirely the cost of the edge. When an edge is bought, every player can use the edge for free. The *cost* $c_i$ of a player $i$ is the total cost of edges bought by $i$. The cost $c_i = \infty$ if $s_i$ is not connected to $t$. The goal of players is to minimize their cost. The *social cost* is defined as the total cost of edges bought by all players.

Anshelevich et al. [3] considered a game with a different model. In their game, players can share the cost of an edges and the cost of each player is the total cost she has to pay. The social cost is defined as the same. They asked a question whether there exists a cost-sharing mechanism such that there exists an equilibrium which is also the optimum. From an optimization point of view, the minimization problem is the well-known Steiner tree problem. In fact, they gave an algorithm that given an optimum solution for social cost, the algorithm computes an cost-sharing mechanism such that all players are happy in using the path in the optimum solution. Moreover, one property in their cost-sharing mechanism is that each edge is entirely bought by one player. Therefore, we can deduce the existence of equilibrium from [3].

**Proposition A.1** ([3])**.** *The game always admits an equilibrium.*

In the following, we represent an instance of game in which the best-response dynamic does not necessarily converge.

**Proposition A.2.** *The game is not a potential game.*

*Proof.* Consider the following network. There are two players with source $s_1$ and $s_2$.

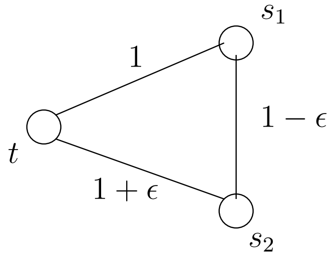

Figure A.1: An instance in which the best-response does not necessarily converge.

A Nash equilibrium (which is also the optimum) is that player 1 buys edge $(s_1, t)$; player 2 buys edges $(s_2, s_1)$. However, if in the initial strategy profile, player 2 buys edge $(s_2, t)$ then the best response for player 1 is to buy edge $(s_1, u)$. Then, player 2 will change her strategy by buying edge $(s_1, t)$ to get a cost 1 instead of cost $1 + \epsilon$. Now, player 1 leaves edge $(s_1, u)$ and use edge $(s_1, t)$ for free. Player 2 is disconnected from the sink and the best response for her is to buy the edge $(s_2, t)$. The strategy profile is the same as the initial one.

Thus, the best-response dynamic does not necessarily converge or in the other words, the game is not potential. □

# Appendix B

# LPT policy in Scheduling Games

In this chapter, we answer the question about the existence of equilibrium in Scheduling Games (Chapter 4) under the LPT policy. It is known that for the uniform machine and the restricted identical machine models, the game under LPT is potential game. However, it is not known whether there exist equilibria in unrelated machine model. This question is raised in [16, 47]. Unfortunately, the answer is negative. We hope that this result is useful in studying an open question (in Chapter 6) about the existence of equilibrium in Scheduling Games under RANDOM policy (as the cost of job under RANDOM policy is related to a combination of the costs of job under the SPT and LPT policies).

**Proposition B.1.** *Consider the Scheduling Games under LPT policy.*

1. *There exists an instance with two machines in which the best response dynamic has a cycle.*

2. *There exists an instance with three machines in which there is no Nash equilibrium.*

| machines \ jobs | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|
| 1 | 35 | 13 | 1 | 14 |
| 2 | 8 | 26 | 25 | 7 |

Figure B.1: Instance 1 in which the best response dynamic cycles.

| machines \ jobs | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|
| 1 | $\infty$ | $\infty$ | $\infty$ | 9 |
| 2 | 13 | 7 | 1 | 8 |
| 3 | 12 | 4 | 15 | $\infty$ |

Figure B.2: Instance 2 in which there is no equilibrium.

*Proof.* The instances are shown in Figure B.3 and Figure B.4. As machines use LPT policy, the order of jobs scheduled on machines are as follows.

**Instance 1** : $A, D, B, C$ on machine 1 and $B, C, A, D$ on machine 2.

**Instance 2** : $D$ on machine 1, $A, D, B, C$ on machine 2 and $C, A, B$ on machine 3. Note that, if a job has processing time $\infty$ on a machine, it means that the job cannot be scheduled on the machine in an equilibrium.

We consider the graph of best-response dynamic. In the graph, nodes represent strategy profiles. There is an arc from node $u$ to node $v$ if and only if in strategy profile corresponding to $u$, there exists a job such that after the best response of this job, the new strategy profile is the one that $v$ represents.

In Figure B.3, we show a cycle of the graph of the best-response dynamic of Instance 1. In the label of each node, the upper part stands for jobs scheduled on machine 1 and the lower part stands for jobs scheduled on machine 2. An arc between two node is characterized by the job who makes the best response and its improvement in terms of cost. Note that, the instance has an equilibrium: jobs $D, C$ on machine 1 and jobs $B, A$ on machine 2.

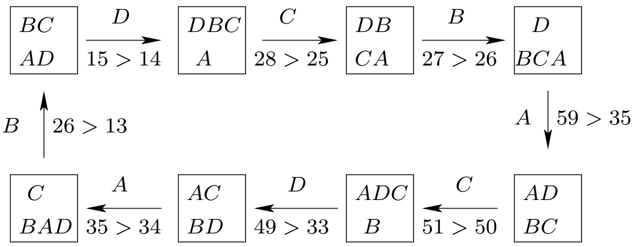

Figure B.3: The best-response dynamic of instance 1 cycles.

In Figure B.4, we show the best-response dynamic graph of Instance 2 omitting some arcs. In the label of each node, the upper, middle and lower parts stand for jobs scheduled on machine 1, 2 and 3, respectively. Similarly as before, an arc between two node is characterized by the job who makes the best response and its improvement in terms of cost. Note that, a node in Figure B.4 may have more than one out-going arc in the best-response dynamic graph but it is sufficient to represent one.

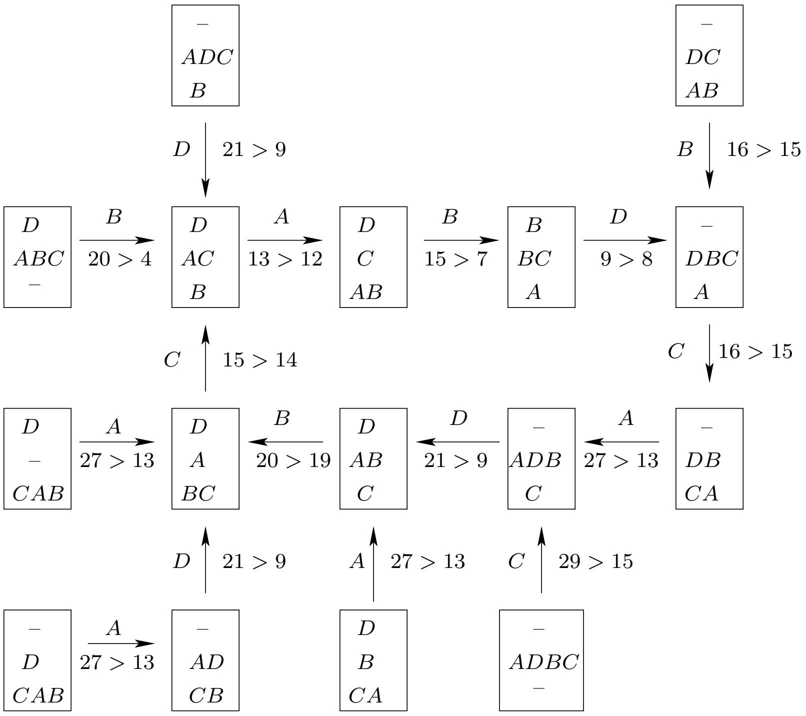

Figure B.4: No equilibrium for instance 2.

□

# Bibliography

[1] Hee-Kap Ahn, Siu-Wing Cheng, Otfried Cheong, Mordecai Golin, and Renée van Oostrum. Competitive facility location: the Voronoi game. *Theoretical Computer Science*, 310:457–467, 2004.

[2] Elliot Anshelevich, Anirban Dasgupta, Jon M. Kleinberg, Éva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[3] Elliot Anshelevich, Anirban Dasgupta, Éva Tardos, and Tom Wexler. Near-optimal network design with selfish agents. *Theory of Computing*, 4(1):77–109, 2008.

[4] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42th Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2001.

[5] James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.

[6] Robert J. Aumann. Acceptable points in general cooperative $n$-person games. *Contributions to the Theory of Games IV, Annals of Math. Study*, 1959.

[7] Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. Tradeoffs in worst-case equilibria. *Theoretical Computer Science*, 361(2-3):200–209, 2006.

[8] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18(2):221–237, 1995.

[9] Yossi Azar, Kamal Jain, and Vahab S. Mirrokni. (Almost) optimal coordination mechanisms for unrelated machine scheduling. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 323–332, 2008.

[10] Philippe Baptiste. An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Operations Research Letters*, 24(4):175–180, 1999.

[11] Philippe Baptiste, Marek Chrobak, Christoph Dürr, Wojciech Jawor, and Nodari Vakhania. Preemptive scheduling of equal-length jobs to maximize weighted throughput. *Operations Research Letters*, 32(3):258–264, 2004.

[12] S.K. Baruah, J. Haritsa, and N. Sharma. On-line scheduling to maximize task completions. *Real-Time Systems Symposium*, pages 228–236, Dec 1994.

[13] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, New York, NY, USA, 1998.

[14] Peter Brucker. *Scheduling Algorithms.* Springer, 3rd edition, 2001.

[15] Ran Canetti and Sandy Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998.

[16] Ioannis Caragiannis. Efficient coordination mechanisms for unrelated machine scheduling. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 815–824, 2009.

[17] Wun-Tat Chan, Tak Wah Lam, Hing-Fung Ting, and Prudence W. H. Wong. New results on on-demand broadcasting with deadline via job scheduling with cancellation. In *Proceedings of the 10th International on Computing and Combinatorics Conference*, pages 210–218, 2004.

[18] Chandra Chekuri, Julia Chuzhoy, Liane Lewin-Eytan, Joseph Naor, and Ariel Orda. Non-cooperative multicast and facility location games. In *Proceedings of the 7th ACM Conference on Electronic Commerce (EC)*, pages 72–81, 2006.

[19] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. *Handbook of Combinatorial Optimization*, volume 3, chapter A review of machine scheduling: Complexity, algorithms and approximability, pages 21–169. Kluwer Academic Publishers, 1998.

[20] Ho-Lin Chen and Tim Roughgarden. Network design with weighted players. In *Proceedings of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 29–38, 2006.

[21] Ho-Lin Chen, Tim Roughgarden, and Gregory Valiant. Designing networks with good equilibria. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 854–863, 2008.

[22] O. Cheong, S. Har-Peled, N. Linial, and J. Matousek. The one-round Voronoi game. *Discrete and Computational Geometry*, 31(1):125–138, 2004.

[23] Y. Cho and S. Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9:91–103, 1980.

[24] George Christodoulou, Elias Koutsoupias, and Akash Nanavati. Coordination mechanisms. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 345–357, 2004.

[25] Marek Chrobak, Wojciech Jawor, Jiri Sgall, and Tomás Tichý. Online scheduling of equal-length jobs: Randomization and restarts help. *SIAM Journal on Computing*, 36 (6):1709–1728, 2007.

[26] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 413–420, 2002.

[27] G. Dobson. Scheduling independent tasks on uniform processors. *SIAM Journal on Computing*, 13:721–716, 1984.

[28] Juliane Dunkel and Andreas S. Schulz. On the Complexity of Pure-Strategy Nash Equilibria in Congestion and Local-Effect Games. In *Internet and Network Economics, Second International Workshop (WINE)*, pages 62–73, 2006.

[29] Christoph Dürr and Nguyen Kim Thang. Nash Equilibria in Voronoi Games on Graphs. In *Proceedings of the 15th European Symposium on Algorithms (ESA)*, pages 17–28, 2007.

[30] Christoph Dürr and Nguyen Kim Thang. Non-clairvoyant scheduling games. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT)*, 2009.

[31] Christoph Dürr, Łukasz Jeż, and Nguyen Kim Thang. Online scheduling of bounded length jobs to maximize throughput. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA)*, 2009.

[32] Jeff Edmonds. Scheduling in the dark. In *Proceedings of the 31st ACM Symposium on Theory of Computing (STOC)*, pages 179–188, 1999.

[33] Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in QoS switches. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 209–218, 2007.

[34] Eyal Even-Dar, Alexander Kesselman, and Yishay Mansour. Convergence time to Nash equilibrium in load balancing. *ACM Transactions on Algorithms*, 3(3), 2007.

[35] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

[36] Sándor P. Fekete and Henk Meijer. The one-round Voronoi game replayed. *Computational Geometry: Theory and Applications*, 30:81–94, 2005.

[37] Amos Fiat, Haim Kaplan, Meital Levy, Svetlana Olonetsky, and Ronen Shabo. On the price of stability for designing undirected networks with fair cost allocations. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 608–618, 2006.

[38] Amos Fiat, Haim Kaplan, Meital Levy, and Svetlana Olonetsky. Strong price of anarchy for machine load balancing. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming*, pages 583–594, 2007.

[39] G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19:312–320, 1979.

[40] D. K. Friesen. Tighter bounds for LPT scheduling on uniform processors. *SIAM Journal on Computing*, 16:554–560, 1987.

[41] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. Computing Nash equilibria for scheduling on restricted parallel links. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 613–622, 2004.

[42] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[43] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[44] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 45:416–429, 1969.

[45] Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, Mohammad Mahdian, and David C. Parkes. Online auctions with re-usable goods. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*, pages 165–174, 2005.

[46] O. H. Ibarra and C. E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM*, 24:280–289, 1977.

[47] Nicole Immorlica, Li Li, Vahab S. Mirrokni, and Andreas Schulz. Coordination mechanisms for selfish scheduling. In *Proceedings of the 1st International Workshop on Internet and Network Economics (WINE)*, pages 55–69, 2005.

[48] Bala Kalyanasundaram and Kirk R. Pruhs. Maximizing job completions online. *J. Algorithms*, 49(1):63–85, 2003.

[49] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[50] Chiu-Yuen Koo, Tak-Wah Lam, Tsuen-Wan Ngan, Kunihiko Sadakane, and Kar-Keung To. On-line scheduling with tight deadlines. *Theoretical Computer Science*, 295(1-3):251 – 261, 2003.

[51] E. L. Lawler. Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the “tower of sets” property. *Mathematical and Computer Modelling*, 20(2):91–106, 1994.

[52] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[53] Fei Li, Jay Sethuraman, and Clifford Stein. Better online buffer management. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 199–208, 2007.

[54] Jian Li. An $O(\log n/ \log\log n)$ upper bound on the price of stability for undirected shapley network design games, 2008. URL `http://arXiv.org:0812.2567`.

[55] Marios Mavronicolas, Burkhard Monien, Vicky G. Papadopoulou, and Florian Schoppmann. Voronoi games on cycle graphs. In *Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science 2008 (MFCS)*, pages 503–514, 2008.

[56] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[57] Dilip Mookherjee and Stefan Reichelstein. Dominant strategy implementation of Bayesian incentive compatible allocation rules. *Journal of Economic Theory*, 56(2): 378–399, 1992.

[58] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[59] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory.* Cambridge University Press, 2007.

[60] Robert W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[61] Petra Schuurman and Tjark Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *Informs Journal on Computing*, 361(1):52–63, 2007.

[62] Nguyen Kim Thang. $\mathcal{NP}$-hardness of pure Nash equilibrium in Scheduling and Connection Games. In *Proceedings of the 35th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 413–424, 2009.

[63] Hing-Fung Ting. A near optimal scheduler for on-demand data broadcasts. *Theoretical Computer Science*, 401(1-3):77 – 84, 2008.

[64] Nodari Vakhania. A fast on-line algorithm for the preemptive scheduling of equal-length jobs on a single processor. In *Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications*, pages 158–161, 2008.

[65] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2004.

[66] A. Vetta. Nash equilibria in competitive societies with applications to facility location. In *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 416–425, 2002.

[67] Tjark Vredeveld. *Combinatorial Approximation Algorithms: Guaranteed Versus Experimental Performance.* PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 2002.

[68] Yingchao Zhao, Wei Chen, and Shang-Hua Teng. The isolation game: A game of distances. In *Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC)*, pages 148–158, 2008.