Truthful Mechanism Design for Cooperative Cost Sharing and Congestion Games

vorgelegt von

Dipl.-Math. Janina Alexandra Brenner

Hamburg

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation.

Tag der wissenschaftlichen Aussprache: 17. Mai 2010

Berlin 2010

D 83

CONTENTS

1 INTRODUCTION

The joint usage of infrastructures in today's world imposes a new line of mathematical challenges. On the one hand, one seeks efficient ways to install and use common infrastructures such that desirable global properties are met. On the other hand, one has to cope with the diverse interests of their users. Often, numerous participants contribute to the installation by providing money or relevant personal data. To reach their individual goals, they attempt to influence the outcome as much as they can.

One of the simplest examples for such a setting is a *single item auction*: A number of bidders are interested in a single indivisible good. At the end of the day, this good is allocated to at most one of the bidders. Say our global goal is to give the item to the bidder who values it the most. How do we find out who this is? If we endow the item to the bidder who *declares* the highest valuation for the item, all bidders clearly have incentives to exaggerate their valuations. If we additionally charge the winner an amount equal to her declared value, rational bidders will stop submitting arbitrarily high valuations. But can we be sure that the highest bidder submits the highest valuation? As a matter of fact, in this case the winner is better off understating her valuation as long as she still wins.

A solution to this problem is the *second price auction*: Allocate the item to the bidder who declares the highest value, and charge her the value of the second highest bid. It can be verified by a simple case analysis that in this auction, no bidder can improve by exaggerating or understating her true valuation. Thus, assuming that bidders behave rationally, we gather the necessary information to achieve our global goal of maximizing the value obtained by a player.

The single item auction has an extremely simple structure: The possible outcomes of the global decision process are allocations of the good to at most one user. In general, there is a wide range of problems with much richer structures. For instance, in network design problems, users wish to be connected to a network or a specific point of interest. In routing problems, agents want their goods or information to be transferred from one point to another. In the scheduling context, we can imagine both machines or jobs to be owned by agents who follow their own selfish interests. Similarly, almost every combinatorial optimization problem yields an interesting application when equipped with the additional complexity of dealing with selfish interests. Many of these problems are hard to solve even without a game-theoretic component.

Further, in the single item auction, each bidder distinguishes only two types of outcomes – those in which she loses and those in which she wins. In general, players may submit complex information about their requests and have complicated valuation functions. The various ways to model the private contributions of participants determine the extent to which they can manipulate a global process. This thesis studies models in which users contribute *money*, and the only private data they control is their *valuation* for the different outcomes. Intrinsically, a valuation indicates how happy a user is with a given outcome. This may play a significant role in measuring the aspired global objective. At the same time, in our model, a valuation designates the maximum monetary contribution a player is willing to make.

The challenges described above are addressed in the realm of *algorithmic mechanism design*. The groundlaying paradigm in this area is the *revelation principle*, a famous result in economics from the seventies. It states that every realistic outcome of a global decision process – more formally one that is in Nash equilibrium – can be implemented by a truthful direct revelation mechanism. A *direct revelation mechanism* works in a very simple way: It collects all relevant data and chooses a global outcome and a payment for every player. A mechanism is called *truthful* if no player can benefit from misreporting her data.

This thesis is devoted to the design of direct revelation mechanisms and the investigation of the limitations that truthfulness and computational efficiency impose on them. We are interested in mechanisms that meet the following objectives:

1. *Computational efficiency*: The mechanism runs in polynomial time.
2. *Truthfulness*: The selection and payment scheme implemented by the mechanism guarantee that it is a dominant strategy for every player to reveal her private valuation. This objective is studied with different assumptions concerning the cooperative behavior of players.
3. *Global Objectives*: Assuming that all players report their data truthfully, the selected outcome optimizes a globally desirable objective.

There are various reasonable global objectives that arise in reality and are therefore worth investigating. In this thesis, we study mechanisms for the following two settings:

I. The first and larger part of the thesis, containing Chapters 3, 4 and 5, is on *cooperative cost sharing mechanisms*. In this framework, the global goal is to find socially attractive allocations where the sum of the payments reflects the cost of the shared infrastructure. We study this question in *cooperative* settings, where the mechanism is required to be truthful even against manipulation of groups of players.

II. The second part, presented in Chapter 6, introduces a new model to incorporate the disutility of *sharing* resources. This is an issue which is not respected in the standard mechanism design approach used in Part I. We assume that a player's valuation depends on the number of other players with whom she shares resources. Our work thus belongs to the area of *mechanism design with externalities*.

In the following, we introduce both topics and give an overview of our contributions in the two areas.

Cooperative Cost Sharing.

In a cost sharing problem, we are given a set of players who are interested in receiving a common service, e.g. connectivity to a network. The provision of the service incurs some cost that is specified by a given cost function. The task is to design truthful mechanisms which run in polynomial time. The global goal that one seeks to optimize is twofold:

3a. *Budget balance*: The sum of all payments charged to the players equals or approximates the cost to establish the service.

3b. *Social cost*: Assuming that players bid truthfully, the servicing cost of the selected players plus the sum of the valuations of the excluded players is minimal.

In the cost sharing setting, it is assumed that players act strategically to maximize their own *quasi-linear* utility functions, defined as their valuation for the service minus the payment they have to make. We consider *cooperative* cost sharing games, i.e. players can form coalitions in order to coordinate their bids and collectively attempt to manipulate the outcome of the mechanism.

In the late nineties, Moulin [73] proposed a framework to derive truthful mechanisms for cooperative cost sharing games. The resulting *Moulin mechanisms* realize the strong notion of *group-strategyproofness*, which ensures that no coordinated bidding of a coalition of players can ever strictly increase the utility of some player without strictly decreasing the utility of another player in the coalition. In the years following his work, a lot of research has gone into designing mechanisms using Moulin's framework for the cost sharing variants of many classical optimization problems (see e.g. [10, 16, 25, 45, 46, 54, 60, 64, 80, 88, 89]).

In **Chapter 3**, we investigate cost sharing mechanisms for combinatorial optimization problems within this framework, with a particular focus on scheduling problems. Our contribution is threefold: First, for a large class of optimization problems that satisfy a certain cost-stability property, we prove that no budget balanced Moulin mechanism can approximate social cost better than $\Omega(\log n)$, where n denotes the number of players. Second, we present a group-strategyproof cost sharing mechanism for the minimum makespan scheduling problem with tight approximation guarantees with respect to budget balance and social cost. Finally, we derive a general lower bound on the budget balance factor of Moulin mechanisms, which can be used to prove a lower bound of $\Omega(n)$ on the budget balance factor for all completion and flow time scheduling objectives. Even in the single-machine case and if all jobs have unit processing times, no Moulin mechanism can achieve a budget balance approximation factor of less than $(n + 1)/2$, although one can easily compute an optimal schedule for this problem in polynomial time.

In addition to our results mentioned above, recent negative results showed that for many fundamental cost sharing games, Moulin mechanisms inevitably suffer from poor approximation factors with respect to budget balance [10, 16, 52, 60, 88] or social cost [16, 25, 88, 89]. Driven by the limitations of Moulin mechanisms, Mehta, Roughgarden, and Sundararajan [69] recently introduced a general framework to derive cost sharing mechanisms, termed *acyclic mechanisms*. These mechanisms implement a weaker notion of truthfulness, called *weak group-strategyproofness* [28, 69], but therefore open new ground for improving the approximation guarantees with respect to budget balance and social cost. Weak group-strategyproofness ensures that no coordinated bidding of a coalition of players can ever strictly increase the utility of *every* player in the coalition.

In **Chapter 4**, we present an approach to derive weakly group-strategyproof cost sharing mechanisms from approximation algorithms. Given a ρ-approximation algorithm for the underlying optimization problem of a cost sharing game, we define a *generalized incremental mechanism* that is ρ-budget balanced and weakly group-strategyproof. We also provide a method for proving social cost approximation guarantees of our generalized incremental mechanisms. Our techniques are particularly effective in the scheduling context. We demonstrate the applicability of our framework by developing weakly group-strategyproof mechanisms for completion time (and flow time) scheduling problems with and without release dates and preemption. Specifically, using the three-field notation scheme by Graham et al. (see Section 2.5.1), we achieve 1-budget balance and 2-approximate social cost for $P||\sum C_i$, 1.21-budget balance and 2.42-approximate social cost for $P||\sum w_iC_i$, and 1.25-budget balance and 5-approximate social cost for $P|r_i, pmtn|\sum C_i$. Very notably, our new mechanisms outperform the strong lower bounds of $(n+1)/2$ on the budget balance factor of Moulin mechanisms for all completion time related objectives that we proved in Chapter 3. We emphasize that these are the first cost sharing mechanisms that are weakly group-strategyproof and achieve constant approximation guarantees with respect to both budget balance and social cost.

To the best of our knowledge, cooperative cost sharing games have so far only been studied in *offline* settings, where the entire instance is known in advance. Hence, the mechanism can take into account all input data associated with every player for determining the allocation and payment scheme. However, many natural cost sharing games inherently bear an *online* characteristic in the sense that players arrive over time and reveal their input data only at their arrival. In such settings, a mechanism must instantaneously make irreversible decisions without any knowledge about players that arrive in the future. The standard measure for assessing the quality of an online algorithm is its *competitive ratio*, i.e. the worst case ratio of the cost of the solution produced by the online algorithm compared to the cost of an optimal offline algorithm that knows the entire input data in advance.

In **Chapter 5**, we consider cooperative cost sharing games in an *online* setting. First, we devise an online model for *general demand* cost sharing games, in which each player can request not only one but several levels of service. We then give a complete characterization of both weakly group-strategyproof and group-strategyproof online cost sharing mechanisms in this model. Moreover, we present a simple method to derive online cost sharing mechanisms from competitive online algorithms which preserves the competitive ratio of the online algorithm. Based on our general results, we develop truthful online mechanisms for several binary demand and general demand cost sharing games derived from network design and scheduling problems.

Mechanism Design with Congestion.

In the standard mechanism design model, it is assumed that each player assigns a private valuation to every possible outcome. A particularly well-tractable special case is when each player distinguishes only between outcomes in which she loses and outcomes in which she wins. A very natural generalization of this special case with various applications is to include negative externalities that arise between winners of an outcome. One approach to model such externalities is via *congestion games*. In a congestion game, we are given a set of resources and a set of players.

Each player has a set of strategies, each of which corresponds to a subset of the resources, e.g. a route between a source and a destination in a given network. A congestion game is called symmetric if all players have the same strategy set. In the classical congestion games literature, it is assumed that each player chooses a strategy to minimize her individual cost, e.g. the total delay of her route. The cost is usually related to the *congestion* of a resource, i.e. the number of players to using that resource.

In **Chapter 6**, we take a mechanism design approach to resource allocation problems inspired by congestion games. We assume that players assign a private valuation if they are allocated one of their desired strategies. The resources can be shared among players; however, there is a loss for resource congestion, which we model by a decrement in the player's valuation for that resource. We assume that players experience the loss of their most congested resource (also called *bottleneck*). Our goal is to design truthful mechanisms that maximize social welfare. Hence, the global goal in this chapter is

3. *Social welfare*: Assuming that players bid truthfully, the sum of the valuations over all players is maximum.

We design a generic truthful mechanism for symmetric bottleneck congestion games. Our approach is to reduce the strategy set to a set of pairwise disjoint strategies for which we find an optimal allocation using dynamic programming. Our mechanism applies to various settings and approximates social welfare by different factors which can be proven using duality in hypergraphs. As an example, our mechanism selects an outcome with optimal social welfare for single-commodity network congestion games. We also provide sufficient conditions for truthfulness of mechanisms for congestion games which we use to prove that our generic mechanism is truthful. We complement our results by identifying several special cases of the problem that are hard to approximate.

How to read this thesis. We assume that the reader is familiar with basic concepts in combinatorial optimization and game theory. In Chapter 2, we introduce the more specific models from both areas on which this thesis is based. Chapters 3 to 6 contain the contributions of this thesis as described above. Each of these four chapters is self-contained but relies on the concepts introduced in the preliminaries chapter.

2

PRELIMINARIES

In this chapter, we review the basic concepts and related work from algorithmic game theory and combinatorial optimization upon which this thesis is built. Section 2.1 introduces some notations that we use throughout the thesis. In Section 2.2, we give a brief overview over the relevant parts of mechanism design. Section 2.3 introduces the cost sharing settings that we study. In Section 2.4, we sketch existing techniques to design cost sharing mechanisms and review the related work in the area. In Section 2.5, we define several combinatorial optimization problems on which our game-theoretic problems are based. We provide a short introduction to congestion games in Section 2.6.

2.1 Notation

For convenience, the following notations are frequently used throughout this thesis.

We use $[n]$ to denote the set $\{1, \dots, n\}$. Many variables will be defined for all players $i \in [n]$. For sets, as e.g. the players' type sets T_i, we denote by the bold variable name $\boldsymbol{T}$ the cartesian product of all players' sets, i.e. $\boldsymbol{T} := T_1 \times \ldots \times T_n$. For functions or numbers like v_i, we use the bold variable name $\boldsymbol{v}$ to denote the vector of all v_i, i.e. $\boldsymbol{v} := (v_1, \dots, v_n)$. For any vector $\boldsymbol{x}$ or cartesian product X, x_{-i} or X_i denotes the same object without its ith entry. Along these lines, we define $(y_i, x_{-i}) := (x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n)$ for vectors, and $(Y_i, X_{-i}) := (X_1 \times \ldots \times X_{i-1} \times Y_i \times X_{i+1} \times \ldots \times X_n)$ for cartesian products.

We denote by H_n the nth *harmonic number*, i.e. $H_n := \sum_{i=1}^{n} 1/i$. As n goes to infinity, H_n approaches $\log n + \gamma$, where $\gamma \approx 0.577$ denotes the Euler-Mascheroni constant. Hence, $H_n = \Theta(\log n)$ and we use both values interchangeably.

2.2 Mechanism Design

Mechanism design is a research field at the boundary of computer science and economics. It aims at implementing a *social choice* in an environment with strategic players. In this thesis, we study mechanisms in the presence of *money*. That is, we assume that the players' preferences for different social choices can be expressed in monetary values. By allowing to request payments from the players, this assumption opens the ground for many positive results, the most remarkable being the *revelation principle*. Moreover, the set of implementable social choice functions can be characterized quite precisely in some cases. We summarize the most important notions and results in the following. For a more comprehensive survey on mechanism design see for instance Chapter 23 in [65] or Chapter 9 in [78].

2.2.1 General Setting

In this thesis, we study several special cases of the following general mechanism design setting: There is a set Θ of *outcomes* which affect a universe U of n *players*. Each player has a private *type* t_i which is only known to the player herself. The set of possible types of each player i is restricted to a publicly known set T_i. The preference of a player $i \in U$ is given by her *valuation function* $w_i : t_i \times \Theta \to \mathbb{R}$, where $w_i(t_i, \theta)$ is the (monetary) valuation that i assigns to outcome $\theta \in \Theta$ if she has type $t_i \in T_i$. The valuation functions, as well as all other information besides the private types t_i of the players, are public information.

Definition 2.1 (Mechanism Design Problem). *A* mechanism design problem *is defined by a set Θ of* outcomes *and a set U of n players, each of whom has a set of possible types T_i, and a valuation function $w_i : T_i \times \Theta \to \mathbb{R}$.*

An important subclass of mechanism design problems are *binary demand* problems, in which each player only distinguishes "winning" and "losing" outcomes. Both Chapters 3 and 4 study this strongly restricted but still very rich subclass. We briefly describe the binary demand setting to illustrate what types and valuation functions can be.

Example 2.2 (Binary Demand Problem). *In a binary demand (or single parame-*

ter) problem, each player distinguishes only two types of outcomes: those in which she wins and those in which she loses. In this setting, it is usually assumed that a player has zero valuation for the outcomes in which she loses, and her type t_i is a real number specifying her valuation for any of the winning outcomes. A player's valuation function is thus of the form

$$w_i(t_i, \theta) := \begin{cases} t_i & \text{if } \theta \in \Theta_i, \\ 0 & \text{otherwise,} \end{cases}$$

where $\Theta_i \subseteq \Theta$ denotes the (publicly known) subset of outcomes in which player i wins.

A *social choice function* is a function $f : T_1 \times \ldots \times T_n \to \Theta$. Given the types $t_1, \ldots, t_n$ of all players, it selects a deterministic outcome $f(\boldsymbol{t}) \in \Theta$. The most important concept of this thesis is that of a *direct revelation mechanism.* In addition to making a social choice, it charges each player a payment.

Definition 2.3 (Direct Revelation Mechanism)**.** *A* (direct revelation) mechanism *$M = (f, p_1, \ldots, p_n)$ is a function $M : T_1 \times \ldots \times T_n \to \Theta \times \mathbb{R}^n$. Given the types t_i of all players $i \in U$, it selects an outcome $f(\boldsymbol{t}) \in \Theta$ and a payment $p_i(\boldsymbol{t})$ to be made by every player $i \in U$.*

We assume that each player's selfish goal is to maximize her own *utility.* In this thesis, we only consider the important case of *quasilinear* utilities. The quasilinear utility of a player is defined as her valuation for the outcome minus the price she has to pay.

Definition 2.4 (Quasilinear Utility)**.** *If player $i \in U$ of type t_i is charged an amount of money p_i, her* quasilinear utility *for outcome θ is defined as*

$$u_i(t_i, \theta) := w_i(t_i, \theta) - p_i.$$

A direct revelation mechanism is said to be *strategyproof*, *incentive compatible* or *truthful* (in the non-cooperative sense) if no player can gain utility by misreporting her type.

Definition 2.5 (Strategyproof)**.** *A direct revelation mechanism $M = (f, p_1, \ldots, p_n)$ is* strategyproof *if for every player $i \in U$, every type profile $t_1, \ldots, t_n$ and every alternative type t'_i, we have*

$$w_i(t_i, f(\boldsymbol{t})) - p_i(\boldsymbol{t}) \geq w_i(t_i, f(t'_i, t_{-i})) - p_i(t'_i, t_{-i}).$$

As we will motivate in the following section, the concept of truthful direct revelation mechanisms is well suited to study the central questions of mechanism design.

2.2.2 Equilibria and the Revelation Principle

Perhaps the central question in mechanism design is the following: How can a social choice function f be "implemented" without knowing the players' types? For instance, assume that we want to maximize the utilitarian *social welfare*, i.e. choose an outcome that maximizes the sum of valuations over all players. The mechanism needs to somehow gather information from the players.

In the following, we describe a more general model to address this question and discuss outcomes that one may expect to be generated in this model. This discussion will enable us to describe the *revelation principle* and narrow down our further studies to direct revelation mechanisms. The revelation principle has been proven for various settings including so-called *Bayesian* settings which assume a distribution over player's types. Our presentation is largely along the lines of [78].

The most general type of game that we discuss in this thesis is the following setting:

Definition 2.6 (Private Information Game). *A* private information game $(\boldsymbol{X}, \boldsymbol{T}, \boldsymbol{u})$ *is defined by a set of n players each of whom has (i) a set of* actions X_i*, (ii) a set of* types T_i*, and (iii) a utility function $u_i : T_i \times X_1 \times \ldots \times X_n \to \mathbb{R}$.*

Again, the *type* of a player models the private information that is only known to the player herself. Besides the actions that all players make, her own type is the crucial piece of information that captures a player's utility for an outcome of the game. The intuition behind this definition is that the utility functions of all players are common knowledge, but the key to determining a player's strategic behavior is her type, which is not known to anybody but the player herself.

Since a player i has no information about other players' types or actions, the action she chooses solely depends on her own type. We define a *strategy* of player i as a function $s_i : T_i \to X_i$. That is, a strategy s_i determines which action a player will choose depending on which type she receives.

We can now define two important equilibrium concepts. The first one, the *ex-post Nash equilibrium*, captures strategy profiles in which no player has an incentive to switch to a different action, given that all other players stick to their strategy functions, no matter what the actual type profile is.

We call a private information game in which each player's type set is restricted to exactly one type a *full information game.*

Definition 2.7 (Ex-post Nash)**.** *A strategy profile* $\boldsymbol{s} = (s_1, \ldots, s_n)$ *is an* ex-post Nash equilibrium *if* $\boldsymbol{s}(\boldsymbol{t}) = (s_1(t_1), \ldots, s_n(t_n))$ *is a Nash equilibrium for every full information game defined by a fixed vector* $t_1, \ldots, t_n$*. That is, if for every player* i*, every type profile* $\boldsymbol{t}$ *and every alternative action* $x_i \in X_i$*, we have*

$$u_i(t_i, \boldsymbol{s}(\boldsymbol{t})) \geq u_i(t_i, (x_i, s_{-i}(\boldsymbol{t}))).$$

The second concept we are interested in is that of *dominant strategy equilibria.* It characterizes a more restricted set of strategy profiles: Here, given his own type, no player has an incentive to switch to an action different from the one determined by her strategy, no matter what the other players do.

Definition 2.8 (Dominant Strategy Equilibrium)**.** *A strategy profile* $s_1, \ldots, s_n$ *is a* dominant strategy equilibrium *if for every type profile* $t_1, \ldots, t_n$ *and every player* i*,* $s_i(t_i)$ *is a dominant strategy in the full information game defined by* $t_1, \ldots, t_n$*. That is, if for every player* i*, every type* $t_i \in T_i$ *and every action profile* $\boldsymbol{x} \in \boldsymbol{X}$*, we have*

$$u_i(t_i, (s_i(t_i), x_{-i})) \geq u_i(t_i, \boldsymbol{x}).$$

Hence, the crucial difference is that in the dominant strategy equilibrium, we also pay attention to actions that are not played in any strategy profile. The following fact is an easy observation:

Fact 2.9 ([78])**.** *Let* $\boldsymbol{s}$ *be an ex-post Nash equilibrium in the private information game* $(\boldsymbol{X}, \boldsymbol{T}, \boldsymbol{u})$*. Then,* $\boldsymbol{s}$ *is a dominant strategy equilibrium in the modified private information game* $(\boldsymbol{X'}, \boldsymbol{T}, \boldsymbol{u})$*, where each player's action set is restricted to the image of her strategy function, i.e.* $X'_i := \{s_i(t_i) \mid t_i \in T_i\}$*.*

In order to reconnect with our original setting, we identify an outcome $a(\boldsymbol{x})$ and a payment vector $\boldsymbol{p}(\boldsymbol{x})$ with every possible action profile $\boldsymbol{x} \in \boldsymbol{X}$. Intuitively, we "observe" the outcomes and payments realized in equilibria. The utility $u_i(t_i, \boldsymbol{x})$ that player i of type t_i has when $\boldsymbol{x}$ is the set of actions played is naturally identified with the utility $u_i(t_i, a(\boldsymbol{x}))$ of this player when $a(\boldsymbol{x})$ is the outcome and p_i is the payment requested from her.

Due to the inherent properties of ex-post Nash equilibria and, even stronger, dominant strategy equilibria, outcomes in both these states are likely to occur and remain stable. With the above correspondence, we therefore define *implementable* social choice functions as social choice functions that are realized in equilibrium by a private information game. We intentionally leave open the equilibrium concept meant in the following definition to fit both concepts defined above.

Definition 2.10 (Implementable social choice)**.** *A social choice function* f *is* implementable *by a private information game if there exists an equilibrium* $\boldsymbol{s}$ *of the game such that for every type profile* $\boldsymbol{t}$*, we have* $a(\boldsymbol{s}(\boldsymbol{t})) = f(\boldsymbol{t})$.

The following influential theorem known as the "revelation principle" states that if a social choice function is implementable in dominant strategies, it is also directly implementable by a truthful direct revelation mechanism. The intuitive idea is to introduce the direct revelation mechanism as a mediator who asks each agent for her type and then implicitly plays her dominant strategy for her.

Theorem 2.11 (Revelation Principle [39, 65])**.** *Given a private information game and a social choice function f, if there exists an arbitrary mechanism that implements f in dominant strategies, then there exists a strategyproof direct revelation mechanism that implements f. Further, the equilibrium payments for all players are equal in both mechanisms.*

Proof. Let $\boldsymbol{s}$ be a dominant strategy equilibrium of the private information game such that $a(\boldsymbol{s}(\boldsymbol{t})) = f(\boldsymbol{t})$ for all $\boldsymbol{t} \in \boldsymbol{T}$. We define the direct revelation mechanism $M = (f, p_1, \dots, p_n)$ by f and the equilibrium prices $p_i(\boldsymbol{t}) := p_i(\boldsymbol{s}(\boldsymbol{t}))$.

We need to prove that M is strategyproof, i.e. for every player i and type profile $\boldsymbol{t}$, we have $w_i(t_i, f(\boldsymbol{t})) - p_i(\boldsymbol{t}) \geq w_i(t_i, f(t'_i, t_{-i})) - p_i(t'_i, t_{-i})$ for every alternative type t'_i. Assume that player i misreports her type as t'_i. Then by definition, the mechanism selects output $a(\boldsymbol{s}(t'_i, t_{-i}))$ and prices $\boldsymbol{p}(\boldsymbol{s}(t'_i, t_{-i}))$. Since s_i is a dominant strategy for player i in the private information game, we have $u_i(t_i, (s_i(t_i), x_{-i})) \geq u_i(t_i, \boldsymbol{x})$ for every action profile $\boldsymbol{x} \in \boldsymbol{X}$. This is in particular true for $\boldsymbol{x} := \boldsymbol{s}(t'_i, t_{-i})$, which yields $u_i(t_i, \boldsymbol{s}(\boldsymbol{t})) \geq u_i(t_i, \boldsymbol{s}(t'_i, t_{-i}))$, hence in the mechanism design notation, $w_i(t_i, a(\boldsymbol{s}(\boldsymbol{t}))) - p_i(\boldsymbol{s}(\boldsymbol{t})) \geq w_i(t_i, a(\boldsymbol{s}(t'_i, t_{-i}))) - p_i(\boldsymbol{s}(t'_i, t_{-i}))$. □

Thanks to the revelation principle, one can avoid simulating the strategic behavior of individuals and instead restrict to designing truthful direct revelation mechanisms where it can be assumed that all data is reported truthfully. We will therefore restrict our attention to direct revelation mechanisms throughout the remainder of this thesis, as is common in mechanism design theory.

2.2.3 Implementable Social Choice Functions

We have motivated in the previous section that to characterize which social choice functions are realizable, or more technically speaking implementable, reduces to studying which social choice functions can be turned into strategyproof direct revelation mechanisms. It turns out that the domains of players' types play a major role in characterization truthful mechanisms.

The following property of *weak monotonicity* is a necessary condition for implementability of a social choice function. It states that if the unilateral deviation of one player causes a different outcome to be chosen, then it must be because this player benefits more from switching to this outcome under her new type than she did under the one she started with. The formal definition is as follows:

Definition 2.12 (Weak Monotonicity [9]). *A social choice function f is* weakly monotone *if for every player $i \in U$, every type profile $t_1, \ldots, t_n$ and every alternative type t'_i such that $f(\boldsymbol{t}) = \theta \neq \theta' = f(t'_i, t_{-i})$ we have*

$$w_i(t_i, \theta) - w_i(t_i, \theta') \geq w_i(t'_i, \theta) - w_i(t'_i, \theta').$$

In fact, if the type domains are convex, weak monotonicity of a social choice function is also sufficient for its implementability. Theorem 2.13 states that in this case, given a weakly monotone social choice function, we can always find payments such that the induced mechanism is strategyproof. Proving this theorem is quite involved and we refer the reader to the original proofs in [9, 91].

Theorem 2.13 ([9, 91]). *If a mechanism $M = (f, p_1, \ldots, p_n)$ is strategyproof, then f satisfies weak monotonicity. If all type sets T_i are convex, then for every social choice function f satisfying weak monotonicity there exist payment functions $p_1, \ldots, p_n$ such that the corresponding mechanism is strategyproof.*

The following theorem tells us even more: Once we found an implementable social choice function (i.e. one satisfying weak monotonicity), the payment of every player is unique for every fixed vector of types t_{-i} of other players.

Theorem 2.14 (Uniqueness of Prices [75]). *Let $M = (f, p_1, \ldots, p_n)$ be a strategyproof mechanism. Then, the mechanism $M' = (f, p'_1, \ldots, p'_n)$ with modified payments is strategyproof if and only if $p'_i(\boldsymbol{t}) = p_i(\boldsymbol{t}) + h_i(t_{-i})$ for some functions h_i depending only on the types of players other than i.*

As a consequence, the payment functions of a strategyproof mechanism are fully determined by its social choice function if we find a way of normalizing the functions h_i defined in Theorem 2.14. As we will see in Section 2.2.5, this is straightforward for single-parameter domains.

2.2.4 Maximizing Social Welfare – VCG Mechanisms

One of the most well-studied social choice functions is that of maximizing the *social welfare*. The social welfare of an outcome $\theta \in \Theta$ is defined as the sum of valuations for this outcome over all players, i.e. $W(\theta) := \sum_{i=1}^{n} w_i(t_i, \theta)$. This social choice function is implemented by the famous *Vickrey-Clarke-Groves (VCG)* mechanisms due to Vickrey [99], Clarke [26], and Groves [43]. The central idea of VCG mechanisms is to let every player pay the difference in the social welfare obtained by other players when she is present versus when she is not present. It turns out that VCG mechanisms are basically the only truthful mechanisms that maximize social welfare.

As an easy example, we revisit the single item auction described in the introduction:

Example 2.15 (Single Item Auction)**.** *A number of bidders are interested in a single item that is auctioned off by an auctioneer. Every bidder has a private valuation $v_i \geq 0$ for getting the item, and has zero valuation for not getting it. The social choice function maximizing social welfare is to allocate the item to the bidder with highest valuation, say v_1. In this allocation, the social welfare gained by the rest of the population is zero. However, if the winning player was not present in the auction, the second highest valuation, say v_2, would win, yielding a social welfare of v_2. Hence, according to the VCG payment rule, the winner should pay an amount of $v_2 - 0$ for getting the item.*

The VCG mechanism for this auction is also called the *Second Price Auction*: The item is allocated to the bidder with highest valuation at the price of the second highest valuation. If we require an auction that is normalized such that a bidder pays nothing when not getting an item, this is in fact the only truthful mechanism that optimizes social welfare.

More generally, a VCG mechanism is defined as follows:

Definition 2.16 (Vickrey-Clarke-Groves Mechanism [99, 26, 43])**.** *A mechanism is a* Vickrey-Clarke-Groves (VCG) *mechanism if its social choice function maximizes the social welfare $W(\theta) = \sum_{i=1}^{n} w_i(t_i, \theta)$ and there are functions $h_i : T_{-i} \to \mathbb{R}$ such that for every player i,*

$$p_i(\boldsymbol{t}) = h_i(t_{-i}) - \sum_{j \neq i} w_i(t_j, f(\boldsymbol{t})).$$

The second term in the definition of the players' payments aligns the players' incentives with that of maximizing social welfare – each player is implicitly paid an amount equal to the sum of valuations of the other players. In the non-cooperative setting, the first term has no strategic meaning for a player since it only depends on other players' types, on which she has no influence. Together, this is the intuitive reason why every VCG mechanism is strategyproof. We leave the formal proof to the reader.

A sensible way of defining the prices is by the *Clarke pivot rule* which fixes each h_i to the maximum social welfare obtainable by players other than player i. This ensures that no player has a negative payment or negative utility.

The class of VCG mechanisms is very powerful and has been widely applied for various settings. However, VCG mechanisms have several drawbacks that we are concerned about in this thesis. On the one hand, it is not always possible to calculate its payments or even the socially optimal outcome in polynomial time. On the other hand, VCG *payments* in general do not reflect any desirable properties of the game – they neither maximize the revenue of the seller nor approximate any cost function that may be associated with the game. Both of these issues will be approached in the framework of *cost sharing games* that we introduce in Section 2.3.

2.2.5 Single Parameter Domains

Many of the results contained in this thesis concern *single parameter games.* In single parameter games, the valuation functions of the players are fully determined by a single private real-valued variable. This is a strong confinement of the original setting, but still allows for a lot of interesting questions to be studied, some of which simply cannot be solved for the general model. In fact, Roberts' theorem [86] states that if the type domains are unrestricted and the social choice is between more than two outcomes, only *affine maximizers* of the social welfare are implementable. However, in the single parameter setting, our first question of characterizing incentive compatible mechanisms has a nice and easy to state answer which allows for a lot more social choice functions to be implemented.

We first provide a formal definition of the most natural way to define single parameter games. The basic idea is that the player's preferences are binary in that they are only interested in winning or losing. In other words, every player has a (publicly known) set of outcomes which they consider as winning outcomes and value highly with some constant. On the other hand, players have zero valuation for all outcomes in which they lose. Hence, the valuation function of each player is modeled by a publicly known "winning" set $\Theta_i \subseteq \Theta$ and a private value $v_i \in \mathbb{R}^+$ that they assign if one of the outcomes in Θ_i are chosen. This value corresponds to the previously defined *type* t_i of a player.

Definition 2.17 (Single-Parameter Game)**.** *In a single parameter game, there exists publicly known sets $\Theta_i \subseteq \Theta$ for all $i \in U$ such that the valuation function of every player $i \in U$ is of the form*

$$w_i(v_i, \theta) = \begin{cases} v_i & \textit{if } \theta \in \Theta_i, \\ 0 & \textit{otherwise.} \end{cases}$$

One example of a single parameter game is the single item auction discussed in Section 2.2.4. Clearly, the set of winning outcomes of every player is publicly known – it is the unique outcome in which the player obtains the item.

The property of weak monotonicity that we identified in Section 2.2.3 reduces to a much more comprehensive condition in the single parameter case. It simply states that given the input of all other players, the higher her value v_i the more likely a player is to win. It is easy to verify that a mechanism cannot be truthful if this condition is not satisfied.

Definition 2.18 (Monotone Social Choice)**.** *A social choice function f for a single parameter game is called* monotone *if for every player $i \in U$ and every valuation vector $\boldsymbol{v}$, if $f(\boldsymbol{v}) \in \Theta_i$ then for every $v'_i \geq v_i$, $f(v'_i, v_{-i}) \in \Theta_i$.*

Figure 2.2.5 depicts the output function as seen by a single-parameter player depending on her value v_i: It is zero if the chosen outcome is not among her winning outcomes Θ_i, and v_i if the outcome is in Θ_i.

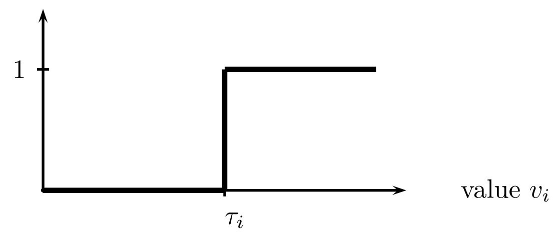

Figure 2.1: Output function as seen by player i given a fixed vector v_{-i}

Since our mechanisms are deterministic, the monotonicity requirement limits the number of "jumps" of this function to one. Hence, there exists a threshold value $\tau_i(v_{-i})$ for every player i and valuation profile of players other than i, such that player i wins if her value lies above this threshold and loses if her value lies below it. As we will see in Section 2.3, it can play an important role for cooperative games whether or not a player wins if her valuation equals this threshold value.

As we have seen in Section 2.2.3, every monotone social choice function permits a payment function combined with which it yields a strategyproof mechanism. Further, the payment function for every player is unique up to an additive term we called $h_i(v_{-i})$. This term only depends on the valuations of the other players. In single-parameter games, there is a very straightforward way to define these terms and thus "normalize" the payment functions of a mechanism. We call a mechanism *normalized* if every losing player pays zero. With this quite intuitive standardization, it is not very hard to prove the following theorem which nicely characterizes the set of strategyproof mechanisms for single-parameter domains.

Theorem 2.19. *A normalized mechanism $M = (f, p_1, \dots, p_n)$ for a single-parameter game is strategyproof if and only if f is monotone and every winning player pays her threshold value τ_i.*

2.2.6 Cooperative Games

As we have seen in the previous sections, we can characterize strategyproof mechanisms depending on the type spaces of players. Especially in the single-parameter case, this characterization is very comprehensive. However, the property of strategyproofness implicitly makes one major assumption: It assumes that players act selfishly in a very restricted way, namely by forming their strategies to maximize their own utility without making any arrangements with other players. This assumption is dropped in *cooperative* games.

The area of *cooperative game theory* studies scenarios in which players can make contracts among each other in some form and thus "cooperate". Two classical models are cooperative games *with nontransferable utilities* (*NTU* games) and cooperative games *with transferable utilities* (*TU* games). For an overview over both models see e.g. [79]. We remark that both models do not correspond to mechanism design problems as described above.

In this thesis, we consider the model in which subsets of players can cooperate by agreeing on choosing a concerted set of strategies. We assume that players cannot exchange money as in other models sometimes referred to by the term "with transferable utility". However, some properties of the cost sharing mechanisms we develop are related to the *core* concept of classical cooperative games. Therefore, the model with transferable utilities (TU) is of interest for this thesis.

In a cooperative game with transferable utilities [78, 79], we are given a set of players U and a player-set dependent function $C : 2^U \to \mathbb{R}$ which models the cost (or utility) generated by each subset of players if this set would form a coalition. The intuition is that if a coalition S forms, the players in $U \setminus S$ do not participate in the game. An outcome of a TU game is defined by a vector $(\xi_i)_{i \in U}$ which specifies for every player $i \in U$ the share of the cost (or utility) that she obtains. The central fairness concept for these games is the *core*. Roughly speaking, the core of a cooperative game seeks to characterize outcomes in which no coalition of players can benefit from breaking away from the grand coalition U. Here, it is implicitly assumed that players in a coalition can redistribute the cost generated by forming this coalition. The formal definition is as follows.

Definition 2.20 (Core)**.** *An outcome $(\xi_i)_{i \in U}$ is in the* core *of a cooperative game if $\sum_{i \in U} \xi_i = C(U)$, and for every subset of players $S \subseteq U$, $\sum_{i \in S} \xi_i \leq C(S)$.*

In many settings, the cost $C(S)$ of a coalition is defined as the optimal objective value of a combinatorial optimization problem and can thus not even be efficiently calculated. In these settings, it makes sense to relax the core condition to the following approximate version.

Definition 2.21 (β-Core). *An outcome $(\xi_i)_{i\in U}$ is in the* β-core *of a cooperative game if*

$$\frac{1}{\beta}\cdot C(U) \leq \sum_{i\in U} \xi_i \leq C(U),$$

and for every subset of players $S \subseteq U$, $\sum_{i\in S} \xi_i \leq C(S)$.

The core and the β-core of TU games have been studied extensively and characterized for various cost functions; we refer the reader to [79] and the references therein.

2.3 Cost Sharing Games

The major part of this thesis is on mechanisms for cooperative cost sharing games. In cost sharing, the foremost concern lies on the payments specified by a mechanism. The goal is to share the *cost* of an outcome among the players that have positive valuation for it. Intuitively, one often takes a reverse approach to mechanism design here: First, we determine payments that reflect the given costs for the possible outcomes. Then, we check whether these payments can be accompanied by a social choice function to obtain a truthful mechanism. In many cases, it is hard to additionally fulfill a socially oriented global objective. However, our results show that this is sometimes remarkably well possible. In the following, we define the different cost sharing models that are studied in Chapters 3, 4 and 5 of this thesis.

2.3.1 Binary Demand Model

A *binary demand cost sharing game* is defined as follows. We are given a universe U of n players (also called agents or users) that are interested in a certain service. The *servicing cost* is given by a cost function $C : 2^U \to \mathbb{R}^+$. For every subset of players $S \subseteq U$, $C(S)$ denotes the cost to establish the service for player set S. We require that $C(\emptyset) = 0$. We will often assume that C is given implicitly by the cost of an optimal solution to an underlying cost-minimization problem $\mathcal{P}$. Examples for such optimization problems are given in Section 2.5 of this thesis.

Every player $i \in U$ has a private *value* $v_i \geq 0$ for receiving the service, i.e. v_i is known to i only. Additionally, each player i announces a non-negative *bid* b_i which represents the maximum price player i is willing to pay for the service. In this setting v_i thus represents the true type of a player while b_i denotes her reported type.

A (direct revelation) *cost sharing mechanism* M solicits the bids of all players and based on these bids determines a player set which receives the service a payment for each served player. The formal definition is as follows.

Definition 2.22 (Binary Demand Cost Sharing Mechanism)**.** *Given a bid vector* $\boldsymbol{b} \in \mathbb{R}^n_+$*, a* cost sharing mechanism *for a binary demand cost sharing game determines a binary allocation vector* $\boldsymbol{x} \in \{0,1\}^n$ *and a payment vector* $\boldsymbol{p} \in \mathbb{R}^n$*.*

Let S^M be the subset of players associated with the allocation vector $\boldsymbol{x}$, i.e. $i \in S^M$ iff $x_i = 1$. We say that S^M is the player set that receives service.

We require that a cost sharing mechanism complies with the following three standard assumptions:

1. *Individual rationality*: A player is charged only if she receives service and her payment is at most her bid, i.e. $p_i = 0$ if $i \notin S^M$ and $p_i \leq b_i$ if $i \in S^M$.

2. *No positive transfer*: A player is not paid for receiving service, i.e. $p_i \geq 0$ for all $i \in S^M$.

3. *Consumer sovereignty*: A player is guaranteed to receive service if she is willing to bid high enough, i.e. there exists a threshold value b_i^* for every player $i \in U$ such that $i \in S^M$ for all $b_i \geq b_i^*$.

In addition, the mechanism has to compute a (possibly suboptimal) feasible solution to the underlying optimization problem $\mathcal{P}$ on the player set S^M. We denote the cost of the computed solution by $\bar{C}(S^M)$. A cost sharing mechanism is called β-budget balanced if it the sum of payments collected by the mechanism does not deviate from the servicing cost by more than a factor β.

Definition 2.23 (β-Budget Balance)**.** *A mechanism M is* β-budget balanced *for some $\beta \geq 1$ if*

$$\bar{C}(S^M) \leq \sum_{i \in S^M} p_i \leq \beta \cdot C(S^M).$$

We say that the cost shares satisfy *cost recovery* if the first inequality holds; they are *β-competitive* if the latter inequality is fulfilled. If $\beta = 1$, we simply call the cost sharing mechanism budget balanced.

In Chapter 3, we use an alternative definition for budget balance that is also often used in the literature:

$$\frac{1}{\beta} \cdot \bar{C}(S^M) \leq \sum_{i \in S^M} p_i \leq C(S^M). \tag{2.1}$$

It is easy to see that given a fixed value of β, both definitions are equivalent since one can simply multiply or divide all cost shares by β.

2.3.2 Strategic Behavior

We assume that players act strategically and every player's goal is to maximize her own utility. The *utility* of player i is defined as

$$u_i(\boldsymbol{x}, \boldsymbol{p}) := v_i x_i - p_i.$$

That is, player i's utility is $v_i - p_i$ if she receives service and zero otherwise. This definition is in line with the single parameter games described in Section 2.2.5. Remark that the *valuation function* we defined for mechanism design problems in Section 2.2 is thus $w_i(\boldsymbol{x}) := v_i x_i$. However, due to its particularly simple structure, it is common to refer to the private value v_i as a player's valuation. We adopt this slight abuse of language in this thesis.

Since the outcome computed by a cost sharing mechanism solely depends on the bids $\boldsymbol{b}$ of the players (and not on their true valuations), a player may have an incentive to declare a bid b_i that differs from her valuation v_i. As defined in Section 2.2.1, a mechanism is called *strategyproof* if bidding truthfully is a dominant strategy for every player. That is, for every player $i \in U$ and every two bid vectors $\boldsymbol{b}, \boldsymbol{b}'$ with $b_i = v_i$ and $b_j = b'_j$ for all $j \neq i$, we have

$$u_i(\boldsymbol{x}, \boldsymbol{p}) \geq u_i(\boldsymbol{x}', \boldsymbol{p}'),$$

where $(\boldsymbol{x}, \boldsymbol{p})$ and $(\boldsymbol{x}', \boldsymbol{p}')$ are the solutions output by the mechanism for bid vectors $\boldsymbol{b}$ and $\boldsymbol{b}'$, respectively.

In this thesis, we consider *cooperative cost sharing games*, i.e. we assume that players can form coalitions in order to coordinate their bids. A mechanism is called *group-strategyproof* if no coordinated bidding of a coalition $T \subseteq U$ can ever strictly increase the utility of some player in S without strictly decreasing the utility of another player in S. The formal definition is as follows.

Definition 2.24 (Group-Strategyproof). *A cost sharing mechanism is* group-strategyproof *if for every subset $S \subseteq U$ of players and every two bid vectors $\boldsymbol{b}, \boldsymbol{b}'$ with $b_i = v_i$ for every $i \in S$ and $b_i = b'_i$ for every $i \notin S$,*

$$u_i(\boldsymbol{x}', \boldsymbol{p}') \geq u_i(\boldsymbol{x}, \boldsymbol{p}) \quad \forall\, i \in S \quad \Longrightarrow \quad u_i(\boldsymbol{x}', \boldsymbol{p}') = u_i(\boldsymbol{x}, \boldsymbol{p}) \quad \forall\, i \in S.$$

A mechanism is *weakly group-strategyproof* [28, 68] if no coordinated bidding can ever strictly increase the utility of *every* player in the coalition.

Definition 2.25 (Weakly Group-Strategyproof). *A cost sharing mechanism is* weakly group-strategyproof *if for every subset $S \subseteq U$ of players and every two bid vectors $\boldsymbol{b}, \boldsymbol{b'}$ with $b_i = v_i$ for every $i \in S$ and $b_i = b'_i$ for every $i \notin S$,*

$$\exists\, i \in S: \;\; u_i(\boldsymbol{x}', \boldsymbol{p}') \leq u_i(\boldsymbol{x}, \boldsymbol{p}).$$

Intuitively, weak group-strategyproofness suffices if we assume that players adopt a slightly more conservative attitude with respect to their willingness of joining a coalition: Group-strategyproofness is needed if a player will participate in a coalition even if her utility is not affected, while weak group-strategyproofness suffices if she only joins when she is strictly better off doing so.

2.3.3 Social Welfare vs. Social Cost

An important measure for the performance of a cost sharing mechanism that has recently gained more attention is its *efficiency* with respect to a social objective. Traditionally, a mechanism is said to be efficient if it selects a set of players that maximizes the *social welfare* (assuming truthful bidding). For a set $S \subseteq U$, define $v(S) := \sum_{i \in S} v_i$ as the sum of valuations of all players in S.

Definition 2.26 (Social Welfare). *The* social welfare *of a player set $S \subseteq U$ is defined as the sum of valuations of all served players minus the servicing cost, i.e.*

$$v(S) - C(S).$$

Note that at first sight, this definition differs slightly from the one given in Section 2.2. However, both notions coincide if one considers the service provider as an additional player whose (negative) valuation is the cost of providing the service.

Classical results in economics [42, 86] state that no truthful mechanism can achieve budget balance and maximum social welfare at the same time; even for simple cost functions and if only strategyproofness is required. Moreover, Feigenbaum et al. [33] showed that for the multicast cost sharing game these two objectives cannot even be approximated simultaneously, even if only strategyproofness is required.

Motivated by these shortcomings, Roughgarden and Sundararajan [88] recently introduced an alternative measure of efficiency that circumvents the intractability results in [34, 42, 86] at least partially. They define the *social cost* of a player set as the cost for serving this player set plus the sum of valuations over all players that are not served.

Definition 2.27 (Social Cost [88])**.** *The* social cost *of a set $S \subseteq U$ is defined as*

$$\Pi(S) := \bar{C}(S) + \sum_{i \notin S} v_i.$$

Observe that for every set $S \subseteq U$, $\Pi(S) = v(U) - (v(S) - C(S))$, i.e. the social cost of a set S equals the total valuation $v(U)$ of all players minus the social welfare of the set S. Since $v(U)$ is a constant parameter of the game, a player set minimizes social cost if and only if it maximizes social welfare. As a consequence, the optimal allocations coincide for both efficiency objectives; however, the two objectives differ with respect to their approximability.

A mechanism is said to be *α-approximate* if it computes a final set of social cost at most α times the optimal social cost Π^*, where

$$\Pi^* := \min_{S \subseteq U} \left(C(S) + \sum_{i \notin S} v_i \right).$$

Definition 2.28 (α-Approximate)**.** *Let S^M denote the served set of players computed by a mechanism M. Then, M is* α-approximate *for some $\alpha \geq 1$ if (assuming that all players $i \in U$ bid their true values $b_i = v_i$)*

$$\Pi(S^M) \leq \alpha \cdot \Pi^*.$$

Dobzinski et al. [30] very recently showed that for cost functions comprising the *public excludable good problem*, a logarithmic gap between the budget balance and social cost approximation guarantees is inevitable even if only strategyproofness is required.

2.3.4 General Demand Model

We now review *general demand* cost sharing games, a generalization of the well-studied binary demand case. Here, in contrast to binary demand cost sharing games, players request several levels of service. This model has been introduced by Moulin in [73] and was recently studied by [14, 29, 69].

In a general demand cost sharing game, every player $i \in U$ has valuation for a finite number of service levels. The maximum service level is bounded by a constant $L \in \mathbb{N}$. The private type of a player $i \in U$ is her *valuation vector $v_i \in \mathbb{R}_+^L$*. Here, $v_{i,l}$ denotes the additional valuation that player assigns for receiving l levels of service over receiving $l-1$ levels of service. Accordingly, each player i announces a *bid vector $b_i \in \mathbb{R}_+^L$*. $b_{i,l}$ represents the maximum price player i is willing to pay for receiving service level l (in addition to service levels 1 to $l-1$).

An *allocation* of goods or service to the set of players U is denoted by a vector $\boldsymbol{x} \in \mathbb{N}_0^U$, where $x_i \in \mathbb{N}_0$ indicates the level of service that player i obtains; $x_i = 0$ means that player i does not receive service. Note that as a characteristic of this model, only subsequent service levels can be allocated to a player, i.e. if a player obtains service level l, then she also obtains service levels $1, \dots, l-1$.

The *cost* of an allocation $\boldsymbol{x} \in \mathbb{N}_0^U$ is given by a cost function $C : \mathbb{N}_0^U \to \mathbb{R}_+$. We assume that C is non-decreasing in every component, and $C(\mathbf{0}) = 0$ for the all-zero allocation $\mathbf{0}$. In the examples we study in Chapter 5, the common service is represented by a combinatorial optimization problem like e.g. Steiner tree, machine scheduling, etc. (see Section 2.5 for definitions). In these cases, we define $C(\boldsymbol{x})$ as the cost of an offline optimal solution to the underlying optimization problem.

Definition 2.29 (General Demand Cost Sharing Mechanism)**.** *A* general demand cost sharing mechanism *solicits the bid vectors b_i from all players $i \in U$, and computes a service allocation $\boldsymbol{x} \in \mathbb{N}_0^U$ and a payment $\phi_{i,l} \in \mathbb{R}$ for every player $i \in U$ and service level $l \leq L$.*

In the general demand setting, the standard assumptions generalize as follows:

1. *Individual rationality*: A player is charged only for service levels that she receives, and for any service level, her payment is at most her bid, i.e. for all i, l: $\phi_{i,l} = 0$ if $x_i < l$ and $\phi_{i,l} \leq b_{i,l}$ if $x_i \geq l$.

2. *No positive transfer*: A player is not paid for receiving service, i.e. $\phi_{i,l} \geq 0$ for all i, l.

3. *Consumer sovereignty*: A player is guaranteed to receive an additional service level if she bids high enough, i.e. there exists a threshold value $b^*_{i,l}$ for each player i and service level l such that $x_i \geq l$ if $b_{i,l} \geq b^*_{i,l}$ and $x_i \geq l - 1$.

For notational convenience, we define $v_{i,0} = \phi_{i,0} = 0$ for all players $i \in U$.

Let $\bar{C}(\boldsymbol{x})$ denote the cost of the actually computed solution for allocation $\boldsymbol{x}$. Along the lines of Definition 2.23, a general demand cost sharing mechanism is *β-budget balanced* if

$$\bar{C}(\boldsymbol{x}) \leq \sum_{i \in U} \sum_{l=1}^{L} \phi_{i,l} \leq \beta \cdot C(\boldsymbol{x}).$$

2.3.5 Classes of Cost Functions

Often, restricting the class of cost functions for a cost sharing game can be exploited to derive cost sharing mechanisms with more desirable properties. We define four classes of cost functions that are frequently referred to in this thesis.

Subadditive costs. The cost of the union of two sets is less or equal to the sum of the individual costs of both sets, i.e. $C(S \cup T) \leq C(S) + C(T)$ for all $S, T \subseteq U$.

Superadditive costs. The cost of the union of two disjoint sets is greater or equal to the sum of the individual costs of both sets, i.e. $C(S \cup T) \geq C(S) + C(T)$ for all $S, T \subseteq U$ with $T \cap S = \emptyset$.

Submodular costs. The incremental cost of adding a player to a set S is non-increasing in the size of S, i.e. $C(T \cup \{i\}) - C(T) \leq C(S \cup \{i\}) - C(S)$ for all $S \subseteq T \subseteq U$ and $i \in S$. This is equivalent to requiring that $C(S \cup T) + C(S \cap T) \leq C(S) + C(T)$ for all $S, T \subseteq U$.

Supermodular costs. The incremental cost of adding a player to a set S is non-decreasing in the size of S, i.e. $C(T \cup \{i\}) - C(T) \geq C(S \cup \{i\}) - C(S)$ for all $S \subseteq T \subseteq U$ and $i \in S$. This is equivalent to requiring that $C(S \cup T) + C(S \cap T) \geq C(S) + C(T)$ for all $S, T \subseteq U$.

2.4 Design Techniques and Classes of Cost Sharing Mechanisms

The development of truthful mechanisms for binary demand cooperative cost sharing games has attracted a lot of attention in the theoretical computer science literature in recent years. Most notably, Moulin [73] proposed a class of cost sharing mechanisms widely known as *Moulin mechanisms* that realize the strong notion of group-strategyproofness. For a long time, most cost sharing mechanisms that were developed for binary demand cost sharing games were Moulin mechanisms. More recently, Mehta, Roughgarden, and Sundararajan [69] introduced a new class of cost sharing mechanisms called *acyclic mechanisms*. These mechanisms generalize Moulin mechanisms and thereby leave room for better approximation guarantees with respect to budget balance and social cost. However, they achieve the slightly weaker notion of truthfulness called weak group-strategyproofness. Another class of cost sharing mechanisms that we adopt in this thesis are *sequential mechanisms* that were introduced by Moulin in [73] and later rediscovered by Juarez [56]. In the following, we review these three classes of cost sharing mechanisms and discuss the related work on cost sharing mechanisms.

2.4.1 Cost Sharing Methods

An important ingredient to both classes of Moulin and acyclic mechanisms are *cost sharing methods*. For every player set $S \subseteq U$, a cost sharing method defines the cost share each player would have to pay if S was the served player set.

Definition 2.30 (Cost Sharing Method)**.** *A* cost sharing method *is a function* $\xi : U \times 2^U \to \mathbb{R}^+$ *that assigns to each user* $i \in U$ *and subset* $S \subseteq U$ *a non-negative cost share* $\xi(i, S)$*. We define* $\xi(i, S) := 0$ *for all* $i \in U \setminus S$ *and* $S \subseteq U$*.*

Similar to Definition 2.23 for cost sharing mechanisms, a cost sharing method ξ is called *β-budget balanced* if for every possible subset of players, the sum of the cost shares with respect to this set does not deviate by more than a factor of β from the cost of serving this set, i.e.

$$\forall\, S \subseteq U : \quad \frac{1}{\beta} \cdot \bar{C}(S) \leq \sum_{i \in S} \xi(i, S) \leq C(S).$$

We say that ξ satisfies *β-cost recovery* if the first inequality holds; it is *competitive* if the latter inequality is fulfilled.

2.4.2 Moulin Mechanisms

The most well-known class of cost sharing mechanisms are the so-called *Moulin mechanisms* based on a framework by Moulin and Shenker [74]. Moulin mechanisms are the only general class of mechanisms that satisfy the strong notion of group-strategyproofness. A Moulin mechanism can roughly be viewed as an iterative ascending auction: In each iteration, the mechanism proposes a cost share to every player. If all players accept their cost shares, the mechanism halts and returns the respective player set and their cost shares. Otherwise, the mechanism removes all players who reject their cost shares from the game and continues with the next iteration.

The crucial input to a Moulin mechanism are cost shares that are *cross-monotonic*. Intuitively, a cost sharing method is cross-monotonic if the cost share of a player does not decrease when some of the other players are removed from the game.

Definition 2.31 (Cross-Monotonic)**.** *A cost sharing method* ξ *is* cross-monotonic *if for all* $S' \subseteq S \subseteq U$ *and for every* $i \in S'$*, it holds that*

$$\xi(i, S') \geq \xi(i, S).$$

The Moulin mechanism $M(\xi)$ induced by a cross-monotonic cost sharing method ξ is formally defined as specified in Algorithm 1.

Algorithm 1: Moulin mechanism $M(\xi)$ induced by ξ.

```
Input: Set of players $U$ and bid vector $\boldsymbol{b} = (b_i)_{i \in U}$
Output: Allocation vector $\boldsymbol{x} = (x_i)_{i \in U}$ and payment vector $\boldsymbol{p} = (p_i)_{i \in U}$
1 Initialize $S := U$.
2 if $\xi(i, S) \leq b_i$ for every player $i \in S$ then halt and output the
  characteristic vector $\boldsymbol{x}$ of $S$ and payments $\boldsymbol{p} := (\xi(i, S))_{i \in U}$.
3 Remove an arbitrary player with $\xi(i, S) > b_i$ from $S$ and return to Step 2.
```

Moulin and Shenker [74] showed that, given a budget balanced and cross-monotonic cost sharing method ξ, the Moulin mechanism $M(\xi)$ satisfies budget balance and group-strategyproofness. Jain and Vazirani [54] observed that this result also carries over to approximately budget balanced and cross-monotonic cost sharing methods:

Theorem 2.32 ([74, 54])**.** *Let ξ be a β-budget balanced and cross-monotonic cost sharing method. Then, the induced Moulin mechanism $M(\xi)$ is group-strategyproof and β-budget balanced.*

Proof. The inheritance of β-budget balance follows directly from the definition of a Moulin mechanism. To prove group-strategyproofness, consider a cost sharing game and let ξ be a cross-monotonic cost sharing method. Assume for contradiction that there is a coalition $S \subseteq U$ whose members can all increase or maintain their utilities by bidding according to a bid vector $\boldsymbol{b'}$ instead of a bid vector $\boldsymbol{b}$ with $b_i = v_i$ for all $i \in S$ (where $b_i = b'_i$ for all $i \notin S$). Let Q and Q' denote the final served player sets output by $M(\xi)$ for $\boldsymbol{b}$ and $\boldsymbol{b}'$, respectively.

We first prove that $Q' \subseteq Q$. Assume for contradiction that there is a player who is served in the run on $\boldsymbol{b'}$ but not in the run on $\boldsymbol{b}$. Among all such players, let i be the one who is dropped first in the run on $\boldsymbol{b}$. Let $T \subseteq U$ be the remaining player set at the beginning of this iteration. By choice of i, all players that have previously been dropped in the run on $\boldsymbol{b}$ are also dropped in the run of $\boldsymbol{b}'$, i.e. $T \supseteq Q'$. Thus, by cross-monotonicity and by the decisions of the mechanism in both runs, we can conclude that $b_i < \xi(i, T) \leq \xi(i, Q') \leq b'_i$. Since only members of the coalition can submit two different bids $b'_i \neq b_i$, we thus have $i \in S$. However, this means that $v_i = b_i < \xi(i, T) \leq \xi(i, Q')$ and hence player i has negative utility in the run on $\boldsymbol{b}'$. On the other hand, player i has zero utility in the run on $\boldsymbol{b}$ and therefore $u_i(\boldsymbol{b}') < u_i(\boldsymbol{b})$, contradicting the first assumption. We conclude that $Q' \subseteq Q$.

Now, by cross-monotonicity, $\xi(i, Q') \geq \xi(i, Q)$ for all $i \in Q'$, and hence no member of the coalition can strictly improve her utility by misreporting. $\square$

As we mentioned in Section 2.2.6, a standard fairness concept in classical cooperative game theory is the β*-core*. A cost sharing method is said to be in the β-core if for each player set $S \subseteq U$, the vector $(\xi(i, S))_{i \in S}$ is in the β-core of the cooperative game with transferable utilities on the set of players S. Formally, this results in the following definition.

Definition 2.33 (β-Core)**.** *A cost sharing method ξ is in the* β-core *iff it is β-budget balanced and*

$$\sum_{i \in S'} \xi(i, S) \leq C(S')$$

for all $S' \subseteq S \subseteq U$.

It is not hard to see that every cross-monotonic and β-budget balanced cost sharing method is in the β-core: We simply use cross-monotonicity to deduce that $\sum_{i \in S'} \xi(i, S) \leq \sum_{i \in S'} \xi(i, S') = C(S')$. As a consequence, the cost shares computed by any β-budget balanced Moulin mechanism fulfill the fairness property of being in the β-core.

Following the publication of Theorem 2.32, a lot of research in recent years has gone into designing cross-monotonic and approximately budget balanced cost sharing methods cross-monotonic for the cost sharing variants of many classical optimization problems such as fixed tree multicast [3, 33, 34], submodular cost sharing [74], minimum spanning tree [54, 59], Steiner tree [54, 59], price-collecting Steiner tree [45], Steiner forest [60], facility location [80], connected facility location [46, 64, 80], single-source rent-or-buy problems [80, 64, 46] parallel machine scheduling [10, 12], and vertex and set cover [52].

Very recently, Roughgarden and Sundararajan [88] introduced the *social cost* as a new benchmark for evaluating the efficiency of cost sharing mechanisms. In the same work, the authors revealed a relation between the social cost approximation factor of a Moulin mechanism $M(\xi)$ and a property of the input cost sharing method ξ. Assume we are given an arbitrary order σ on a subset $S \subseteq U$ of players, i.e. $S = \{i_1, \dots, i_{|S|}\}$, where $i_j \prec_\sigma i_k$ if $1 \leq j < k \leq |S|$. We define $S_j \subseteq S$ as the set of the first j players of S according to the order σ.

Definition 2.34 (α-Summable)**.** *A cost sharing method ξ is* α-summable *if for every subset $S \subseteq U$ and every order σ on S,*

$$\sum_{j=1}^{|S|} \xi(i_j, S_j) \leq \alpha \cdot C(S).$$

Roughgarden and Sundararajan [88] proved that the Moulin mechanism $M(\xi)$ is $(\alpha + \beta)$-approximate and β-budget balanced if the underlying cost sharing method ξ is α-summable and β-budget balanced. Moreover, they showed that $\max\{\alpha, \beta\}$ is a lower bound on the approximability of $M(\xi)$.

Following the work of Roughgarden and Sundararajan, researchers started to investigate cost sharing mechanisms in light of their social cost approximation guarantees. Results were found for Steiner tree and forest, facility location, single-source rent-or-buy network design and machine scheduling [16, 25, 45, 88, 89]. However, negative results showed that for several fundamental cost sharing games, Moulin mechanisms inevitably suffer from poor approximation factors with respect to budget balance [10, 16, 52, 60, 88] or social cost [16, 25, 88, 89]. Among these results are our lower bounds presented in Chapter 3.

2.4.3 Characterizing Group-Strategyproof Mechanisms

It has been a long-standing open question to characterize the full range of group-strategyproof mechanisms. Until today, there are only very few group-strategy-proof mechanisms that do not belong the class of Moulin mechanisms. These mechanisms are typically defined for very constrained cases, like e.g. the 2-price mechanisms by Bleischwitz et al. [13]. Moulin [73] proved that for submodular cost functions, Moulin mechanisms are the only cost sharing mechanisms that are budget balanced and group-strategyproof. Moreover, Immorlica et al. [52] showed that group-strategyproof cost sharing mechanisms that satisfy some additional conditions correspond to Moulin mechanisms.

Only very recently, the question of characterizing group-strategyproof mechanisms has been completely settled [56, 84]. An important point in this context is whether a player that is *indifferent*, i.e. whose valuation is equal to the requested payment, is accepted or rejected from receiving service (see [56] or [84] for a detailed discussion). Juarez [56] very recently characterized group-strategyproof cost sharing mechanisms that either always accept or always reject indifferent players. Two months ago, Pountourakis and Vidali gave a complete characterization of group-strategyproof mechanisms [84]. Throughout this thesis, we assume that cost sharing mechanisms always accept indifferent players.

2.4.4 Acyclic Mechanisms

Motivated by the shortcomings inherent to Moulin mechanisms, Mehta, Roughgarden, and Sundararajan [69] recently introduced *acyclic mechanisms* which realize the slightly weaker notion of weak group-strategyproofness. Acyclic mechanisms are also driven by cost sharing methods that have to obey certain properties. However, these properties are less restrictive than the cross-monotonicity requirement for Moulin mechanisms and therefore leave more flexibility for improving the approximation guarantees with respect to budget balance and social cost.

An acyclic mechanism is defined in terms of a cost sharing method ξ and an offer function τ.

Definition 2.35 (Offer Function). *An* offer function $\tau : U \times 2^U \to \mathbb{R}^+$ *defines for every subset $S \subseteq U$ and every player $i \in S$ a non-negative* offer time $\tau(i, S)$.

The *acyclic mechanism* $A(\xi, \tau)$ induced by ξ and τ receives the bid vector $\boldsymbol{b}$ as input and proceeds as described in Algorithm 2.

Algorithm 2: Acyclic mechanism $A(\xi, \tau)$ induced by ξ and τ.

```
Input: Set of players $U$ and bid vector $\boldsymbol{b} = (b_i)_{i \in U}$
Output: Allocation vector $\boldsymbol{x} = (x_i)_{i \in U}$ and payment vector $\boldsymbol{p} = (p_i)_{i \in U}$
1 Initialize $S := U$.
2 if $\xi_i(S) \leq b_i$ for every player $i \in S$ then halt and output the
    characteristic vector $\boldsymbol{x}$ of $S$ and payments $\boldsymbol{p} := (\xi(i, S))_{i \in U}$.
3 Among all players in $S$ with $\xi(i, S) > b_i$, let $i^*$ be one with minimum
    $\tau(i, S)$ (breaking ties arbitrarily).
4 Set $S := S \setminus \{i^*\}$ and return to Step 2.
```

Intuitively, acyclic mechanisms generalize Moulin mechanisms by allowing to constrain the order in which players can be rejected through the offer function τ. For a given subset $S \subseteq U$ and a player $i \in S$, we partition the player set S into three sets with respect to the offer time of i: let $L(i, S)$, $E(i, S)$ and $G(i, S)$ be the sets of players with offer times $\tau(\cdot, S)$ strictly less than, equal to, or strictly greater than $\tau(i, S)$, respectively. The following definition is crucial to achieve weak group-strategyproofness.

Definition 2.36 (Valid Offer Function). *Let ξ and τ be a cost sharing method and an offer function on U. The offer function τ is* valid *for ξ if the following two properties hold for every subset $S \subseteq U$ and player $i \in S$:*

(P1) $\xi(i, S \setminus T) = \xi(i, S)$ *for every subset* $T \subseteq G(i, S)$;

(P2) $\xi(i, S \setminus T) \geq \xi(i, S)$ *for every subset* $T \subseteq G(i, S) \cup (E(i, S) \setminus \{i\})$.

Thus intuitively, Definition 2.36 requires cross-monotonicity only with respect to players with greater or equal offer times. We summarize the main result of Mehta, Roughgarden, and Sundararajan [69] in the following theorem:

Theorem 2.37 ([69]). *Let ξ be a β-budget balanced cost sharing method and let τ be an offer function that is valid for ξ. Then, the induced acyclic mechanism $A(\xi, \tau)$ is β-budget balanced and weakly group-strategyproof.*

Mehta, Roughgarden, and Sundararajan [69] showed that primal-dual approximation algorithms for several combinatorial optimization problems naturally give rise to acyclic mechanisms with attractive approximation guarantees both with respect to budget balance and social cost. Bleischwitz et al. [12] recently defined *egalitarian mechanisms*, which belong to the class of acyclic mechanisms. Egalitarian mechanisms iteratively add a most cost efficient player set and charge each player in the set an equal amount. The authors show how to construct egalitarian mechanisms from approximation algorithms that fulfill a (rather strong) monotonicity property, requiring that the approximate solution cost does not increase when any player's *size* (e.g. its processing time) is reduced. They apply their results primarily to makespan scheduling and bin packing problems. Bleischwitz et al. [12] also proved that all acyclic mechanisms are *weakly group-strategyproof against collectors*, a notion that strengthens weak group-strategyproofness to the setting where indifferent players are assumed to strictly prefer receiving service at their valuation price over not receiving service.

2.4.5 SEQUENTIAL MECHANISMS

Another class of cost sharing mechanisms that has been introduced by Moulin [73] are *incremental mechanisms.* An incremental mechanism considers players sequentially according to an arbitrary but fixed order. The cost share offered to a player is equal to her *incremental cost*, i.e. the increase in cost caused by adding this player to the set of previously selected players. The player is granted service if she accepts her cost share. Moulin claimed that for supermodular cost functions, incremental mechanisms are basically the only cost sharing mechanisms that are group-strategyproof and budget balanced. However, the original characterization given in [73] is flawed (as indicated in [56]) and the positive part of the statement holds only under the assumption that players are never indifferent. Moulin also extended the class of incremental mechanisms by allowing that the order on the set of players that have not been considered so far may change during the course of the mechanism. He calls these mechanisms *generalized incremental mechanisms.*

In his original work [73], Moulin gives several equivalent definitions for incremental and generalized incremental mechanisms. The most intuitive definition is by means of a *binary decision tree.* A binary decision tree is a routed directed tree in which each vertex is labeled with a player and has two successors. Starting from the root, the mechanism proposes in each iteration an incremental cost share to the player associated with the current vertex. Depending on whether the player accepts her cost share or not, it moves on to the right or left successor of the current vertex. The mechanism stops when it has reached a sink vertex. In the binary decision tree corresponding to an incremental mechanism, all vertices that have the same distance to the root are labeled with the same player. In the *generalized* setting, it is only required that every path contains the name of each player exactly once (for binary demand cost sharing games) or up to L times (for general demand cost sharing games).

Juarez [56] very recently showed that group-strategyproof cost sharing mechanisms correspond to *sequential mechanisms* if indifferent players are *rejected* from obtaining service. Sequential mechanisms correspond to generalized incremental mechanisms in which the vertices of the binary decision tree can be labeled with arbitrary cost shares (see [56] for precise definitions).

Incremental and generalized incremental mechanisms are much less prevailing in literature than Moulin mechanisms (at least in the context of deriving cost sharing mechanisms for optimization problems). However, our results in Chapter 4 reveal that generalized incremental mechanisms naturally arise as weakly group-strategyproof mechanisms with attractive budget balance and social cost approximation guarantees for several fundamental cost sharing games for which Moulin mechanisms inevitably fail.

2.5 Combinatorial Optimization Problems

Many of the problems considered in this thesis are based on *combinatorial optimization problems* in which one seeks to identify a combinatorial structure with minimum cost. The problems we are interested in can broadly be divided into two domains: The first domain is scheduling, where the major challenge is to find good sequences or temporary plans for a given set of tasks. In this thesis, we consider the fundamental class of *parallel machine scheduling* problems, which we introduce in Section 2.5.1. The second domain can be subsumed by the term *network design*. It includes various problems in which one seeks a minimum cost structure that establishes certain connectivity requirements. These problems are often defined on graphs. In Section 2.5.2 we give an introduction to the central problems of this domain.

2.5.1 Parallel Machine Scheduling

In general, a *parallel machine scheduling problem* can be described as follows: We are given a set U of n jobs that are to be executed on m parallel machines. Every machine can execute at most one job at a time. The goal is to assign all jobs to the machines such that a certain objective function, such as the completion time of the last job or the sum of all completion times, is minimized.

Depending on the scheduling applications, there are various meaningful objective functions and input characteristics for machine scheduling problems. In order to differentiate between the various settings, we use the standard notation scheme by Graham et al. [40]. In this notation scheme, a scheduling problem is classified by a three-field expression of the form $\alpha|\beta|\gamma$. In the following, we review the instantiations used in this thesis.

First field (α). The first field describes the machine environment. In this thesis, we only consider *identical parallel machines* which are denoted by **P**. If there is only a single machine, i.e. $m = 1$, the parameter α is to set to **1**.

Second field (β). The second field describes the job types and execution options. In the identical machine problems that we study, each job $i \in U$ has a positive *processing time* p_i and a non-negative *weight* w_i. The processing time describes the time needed to execute i on one of the machines (thus, intuitively, all machines have the same speed). In case all jobs have the same length, we speak of *unit processing times* and add the term $\mathbf{p_i = 1}$ to the second field.

Additionally, a job may have a non-negative *release date* r_i. We describe this by adding the term $\mathbf{r_i}$ to the second field. The release date specifies the time when job i becomes available for execution. When jobs have release dates, it is often allowed to interrupt the execution of a job at any point of time and resume is later, possibly on a different machine. This setting is called the *preemptive setting* and is indicated by the term **pmtn**. In contrast, in the (standard) *non-preemptive* setting, a job that has started must be completed on the same machine without interruption.

Third field (γ). The third field indicates the objective function (or cost function) that is sought to be minimized. We consider the makespan, completion time and flow time objectives and their respective weighted counterparts. Let C_i denote the *completion time* of job $i \in U$ in a given schedule, i.e. the point of time when job i has been fully processed.

The *makespan* of a given schedule is denoted by $\mathbf{C}_{\max}$ and refers to the point of time when all jobs have been completed. Job weights are generally not considered in this setting.

The *(weighted) completion time* objective is referred to by setting the parameter γ to $\sum_\mathbf{i} \mathbf{w_i C_i}$. Here, the cost of a schedule is the sum of the (weighted) completion times over all jobs. The *average (weighted) completion time* objective is denoted by $\sum_\mathbf{i} \mathbf{w_i C_i / n}$. In the standard setting, i.e. when all jobs have to be executed, this objective is equivalent to the (weighted) completion time objective. However, in the mechanism design settings studied in this thesis, we usually have to choose a subset of jobs to execute. In this case, the parameter n varies with the executed set of jobs and we need to differentiate between the two objective functions.

The *flow time* F_i of a job $i \in U$ is defined as the difference between its completion time and its release date, i.e. $F_i := C_i - r_i$. We set the third parameter to $\sum_\mathbf{i} \mathbf{w_i F_i}$ if the objective is to minimize the *weighted flow time.* In all the three min-sum scheduling cases, we omit the weights w_i in the case of *unweighted* or *unit weight* jobs.

The problem of scheduling independent jobs on parallel machines is well-studied for various settings described by the above parameters. We refer the reader to Brucker [21] for an excellent overview. The most relevant results for our work are the following:

The minimum makespan problem $P|\,|C_{\max}$ is NP-complete, as shown by Garey and Johnson [36]. Hochbaum and Shmoys [50] gave a polynomial-time approximation scheme (PTAS) for this problem. Graham's *largest processing time* (LPT) algorithm [41] is a 4/3-approximation. The versions with unit processing times or equal weights are are optimally solved by the LPT algorithm [94, 67]. Lenstra proves that the minimum weighted completion time scheduling problem $P|\,|\sum_i w_i C_i$ is NP-complete (see [21]). This result also implies NP-completeness of the average weighted completion time problem $P|\,|\sum_i w_i C_i/n$. A PTAS for both problems has been given in [2]. Smith's rule [96] schedules jobs by non-increasing weight per processing time ratios and approximates both problems by a factor of $\frac{1}{2} \cdot (1 + \sqrt{2}) \approx 1.21$. For unit processing times or equal weights, Smith's rule delivers optimal solutions. With release dates and preemption, minimizing the sum of (unweighted) completion times $P|r_i, pmtn|\sum_i C_i$ becomes NP-hard [82]. Only the single machine case is solved optimally by the *shortest remaining processing time* (SRPT) algorithm [93]. Sitters [95] very recently showed that SRPT achieves an approximation guarantee of 1.25 for the parallel machines case.

In the mechanism design variant of scheduling problems, each job is identified with a player who wants her job to be processed on one of the m machines.

Scheduling with Rejection.

In scheduling problems *with rejection*, the algorithm may choose to schedule only a subset of the jobs and pay a certain penalty for each job that is omitted. Consider an arbitrary scheduling problem with job set U and objective function C. In the respective problem with rejection, every job $i \in U$ has a non-negative *penalty* z_i. For every job $i \in U$, we can either schedule i, which incurs a respective contribution to the cost of the schedule, or reject i and pay its penalty z_i. Summing up, the problem is to compute a solution for a subset $S \subseteq U$ of jobs such that the overall cost $C(S) + \sum_{i \notin S} z_i$ is minimized.

This setting has been introduced by Bartal et al. [7] for an online minimum makespan scheduling problem. Since then, there has been some follow-up work on makespan scheduling with rejection. Engels et al. [32] study the offline version for completion time related problems. They give randomized algorithms for minimizing the weighted sum of completion times on related machines which achieve expected approximation guarantees of 2 with and 3/2 without release dates, respectively. For the single machine case, they were able to design optimal algorithms; however, their running-time is only pseudopolynomial unless either weights or processing times are all equal. Bunde [22] gives an optimal algorithm for the single machine case with release dates and unit processing times. He also proves that the completion time scheduling problem with rejection is NP-complete even on a single machine if there are release dates. Bansal et al. [6] study the online preemptive single machine case where flow time or job idle time is concerned.

In scheduling problems with due dates, problems in which a subset of jobs can be rejected are sometimes called *scheduling with penalties*. Similar variants for network design problems are usually referred to as *price-collecting* variants. It is easily verified that a cost sharing mechanism that approximates social cost by a factor of α defines an α-approximate algorithm for the underlying optimization problem with rejection. Our mechanisms presented in Chapter 4 therefore yield several constant approximation algorithms for scheduling problems with rejection.

2.5.2 Network Design

We briefly review network design and related problems. A comprehensive survey on these problems can be found in [61].

A *network design* problem, we are given an undirected graph $G = (V, E)$ with edge weights $w_e \geq 0$ for all edges $e \in E$. Further, we are given a connectivity requirement $d_{uv} \in \mathbb{N}_0$ for every unordered pair of vertices $u, v \in V$. The task is to find a minimum weight subgraph H of G such that for all $u, v \in V$, there are d_{uv} edge-disjoint paths in H.

One of the easiest examples is the *minimum spanning tree* (MST) problem. Here, the connectivity requirement is $d_{uv} = 1$ for all $u, v \in V$, i.e. we are asking for a spanning tree of G with minimum weight. There are several well-know polynomial algorithms that solve the MST problem optimally [61]. In the corresponding cost sharing game, we assume that we are given a designated root vertex $r \in V$. Every player $i \in U$ is associated with a unique vertex in V and the goal of player i is to connect her vertex to the root r.

Another example is the *Steiner tree* problem. Here, we ask for a minimum weight tree that spans a subset of prespecified terminal vertices $T \subseteq V$ and may contain non-terminal vertices. The connectivity requirement is thus $d_{uv} = 1$ if both $u, v \in T$ and $d_{uv} = 0$ otherwise. The Steiner tree problem is NP-hard even for unit weights [37], but there are several approximation algorithms with small constant approximation guarantees, the currently best ones being around 1.55 (compare e.g. [61]).

In the *Steiner forest* problem, we are given a set of terminal pairs $s_i, t_i \in V$. The goal is to select a minimum cost set of edges such that each terminal pair is connected by a path. Hence, the connectivity requirement is $d_{s_i t_i} = 1$ for every terminal pair, and $d_{uv} = 0$ otherwise. The Steiner forest problem contains both the MST and the Steiner tree problem as special cases.

A problem that is very related to network design problems is the *traveling salesman* (TSP) problem. Here, we are given a graph with edge weights, and the goal is to determine a minimum weight tour which contains each vertex exactly once. Like the Steiner tree problem, TSP is NP-hard [37] but can be approximated well. For instance, a 2-approximate solution can be constructed from a minimum spanning tree (see [61]).

2.6 Congestion Games

Congestion games were originally introduced by Rosenthal [87] as a class of games which possess pure Nash equilibria. Generally speaking, a congestion game is a *strategic game* [79] in which the utility of a player depends on the number of players who use the same or overlapping strategies. Since Rosenthal's original work, there has been an immense amount of research on congestion games, most of which study different types of equilibria and their respective inefficiency compared to optimal allocations. We briefly review congestion games in this section.

Definition 2.38 (Congestion Game). *A congestion game $\Gamma = (U, R, (\mathcal{S}_i)_{i\in U})$ contains a set of players U and a set R of* resources. *Every player $i \in U$ has a* strategy set $\mathcal{S}_i \subseteq \mathcal{P}(R)$, *where $\mathcal{P}(R) := \{S | S \subseteq R\}$ denotes the power set of R. A congestion game is called* symmetric *if the strategy sets of all players coincide, i.e. $\mathcal{S}_i = \mathcal{S}$ for all $i \in U$.*

Each player chooses to use one of the strategies in her strategy set. Let $S_i \in \mathcal{S}_i$ denote the strategy chosen by player $i \in U$. The *congestion* x_r of a resource $r \in R$ is defined as the number of players using it, i.e.

$$x_r := |\{i \in U : r \in S_i\}|.$$

Depending on its congestion, each resource has a *payoff* that is specified by a function $g : \mathbb{N} \to \mathbb{R}_+$. There are several ways to define the payoff π_i of a player (compare e.g. [27]). We review the following two standard definitions:

- **Total Payoff.** In the standard congestion games model, the payoff of a player is defined as her *total* payoff over all resources that she uses, i.e.

$$\pi_i = \sum_{r \in S_i} g(x_r).$$

In this model, $g(x_r)$ is usually interpreted as the *delay* of a resource, and each player aims at minimizing π_i, which corresponds to her total delay.

- **Bottleneck Payoff.** A different way of defining players' payoff is by means of their *minimum* payoff

  $$\pi_i = \min_{r \in S_i} g(x_r).$$

  This definition especially makes sense when $g(x_r)$ models the *throughput* of a resource. π_i then corresponds to the throughput of the most congested or *bottleneck* resource used by player i. For this reason, we call congestion games with these payoff functions *bottleneck congestion games.*

There is a significant amount of work in the field of congestion games, most of which study several variations of the price of anarchy or stability. Besides, several generalizations of the above setting have been defined. For instance, Monderer and Shapley [72] introduced congestion games with resource-dependent payoff functions g_r. Milchtaich [71] studied the case with player-dependent payoff functions g_i. Cole et al. [27] study the price of anarchy for different types of players' payoff functions in a model in which an optimal allocation is not required to include all players.

In Chapter 6, we adopt the bottleneck payoff definition in which g represents the troughput of a resource and is therefore assumed to be non-decreasing. In the following, we describe three important classes of congestion games that we consider.

Singleton Congestion Games.

In a *singleton* congestion game, every strategy contains exactly one resource. We denote a symmetric singleton congestion game on a resources by $\Gamma_a = (U, [a], [a])$. Here, each player's strategy set is identified with the set of resources, i.e. $\mathcal{S} = [a]$.

Among many others, Koutsoupias and Papadimitriou [62] and Gairing and Schoppmann [35] study the price of anarchy of singleton congestion games in different settings.

Network Congestion Games.

In a *network* congestion game, we are given a graph $G = (V, E)$. The resources of the game are the edges of G. Each player $i \in U$ has a source vertex $s_i \in V$ and a sink vertex $t_i \in V$. Player i's strategy set is defined as the set of paths from s_i to t_i. In a *symmetric* or *single-commodity* network congestion game, the source and sink vertices of all players coincide, respectively. In network congestion games, the congestion of an edge is often called the *flow* on this edge.

There are many works on network congestion games (see e.g. [62, 90]). Banner and Orda [5] and Busch and Magdon-Ismail [23] study the efficiency of Nash equilibria for bottleneck network congestion games with the global objective of optimizing the (overall) worst payoff of a resource. Mazalov et al. [66] study the performance of bottleneck network congestion games compared to the minimum weighted sum of the bottleneck latencies of all used paths.

Matroid Congestion Games.

In a *matroid* congestion game [1], the strategy set of each player corresponds to the bases of a matroid.

Definition 2.39 (Matroid [61])**.** *A* matroid *is a tuple $(R, \mathcal{I})$, where R is a finite set of resources, and $\mathcal{I}$ is a non-empty family of subsets of R such that*

(i) If $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$, and

(ii) If $I, J \in \mathcal{I}$ and $|J| \leq |I|$, then there exists $r \in I \setminus J$ such that $J \cup \{r\} \in \mathcal{I}$.

A set $I \in \mathcal{I}$ is called an *independent* set of R. An maximal independent set B is called a *basis* of the matroid. It can be proven that all maximal independent sets have the same size, which is also referred to as the *rank* of the matroid (see e.g. [61]). In a *cycle* matroid, R corresponds to the set of edges of a given graph G, and an independent set of the matroid defined as a forest in G. A basis of a cycle matroid is thus a spanning tree in G.

Ackermann et al. [1] study matroid congestion games with respect to the total payoff function defined above. Voice et al. [100] study Nash equilibria of matroid congestion games with general "congestion-averse" payoff functions.

3

GROUP-STRATEGYPROOF COST SHARING

Classical results in economics show that no truthful mechanism can achieve budget balance and efficiency simultaneously. Roughgarden and Sundararajan recently proposed an alternative efficiency measure, which was subsequently used to exhibit that many previously known cost sharing mechanisms approximate both budget balance and efficiency. In this chapter, we investigate cost sharing mechanisms for combinatorial optimization problems using this novel efficiency measure, with a particular focus on scheduling problems. Our contribution is threefold: First, for a large class of optimization problems that satisfy a certain cost-stability property, we prove that no budget balanced Moulin mechanism can approximate efficiency better than $\Omega(\log n)$, where n denotes the number of players. Second, we present a group-strategyproof cost sharing mechanism for the minimum makespan scheduling problem that is tight with respect to budget balance and efficiency. Finally, we show a general lower bound on the budget balance factor of cost sharing methods, which can be used to prove a lower bound of $\Omega(n)$ on the budget balance factor for completion and flow time scheduling objectives.

Publication Note. The results contained in this chapter have been presented at the *24th International Symposium on Theoretical Aspects of Computer Science (STACS 07)* [16] and were published in *Theoretical Computer Science* [17].

3.1 INTRODUCTION

In this chapter, we study *group-strategyproof* cost sharing mechanisms for binary demand cost sharing games. A particular focus is laid on cost sharing games that are derived from parallel machine scheduling problems. The general setting is as described in Section 2.3.1: We are given a set of n players that are interested in a certain service. Every player has a private valuation v_i for receiving this service and announces a bid b_i which designates the maximum price she is willing to pay. Associated with the underlying optimization problem, we are given a cost function describing the minimum cost of serving each subset of players. In this chapter, we use the alternative definition for β-budget balance given in Equation 2.1.

A large class of group-strategyproof cost sharing mechanisms are so-called *Moulin mechanisms*, based on a framework due to Moulin and Shenker [74]. This framework provides a means to obtain group-strategyproof cost sharing mechanisms from cross-monotonic cost sharing methods. Roughgarden and Sundararajan [88] revealed a relation between the efficiency of a Moulin mechanism with respect to social cost and a property of the underlying cost sharing method, which they termed *α-summability.* There is a fair amount of related work that is based on these findings. Details are given in Section 2.4.2.

One focus of this chapter is on cost sharing mechanisms for *parallel machine scheduling* problems, which we introduced in Section 2.5.1. In the classical setting, we are given a set of jobs that have to be executed on m parallel machines. The goal is to assign all jobs to the machines such that a certain objective function, such as the makespan or the sum of all completion times, is minimized. In the cost sharing context, we assume that every job is owned by a player who acts strategically in order to get her job processed at a low cost. The cost that is to be distributed among the players depends on the objective function of the scheduling problem. It is very natural to suppose that the cost incurred by the service provider is the amount of time that she needs until all jobs are completed, leading to the *minimum makespan* cost function. However, one can also imagine that the service provider aims at minimizing the total time that jobs spend in the system or other *completion time* related objective functions.

Very notably, although network design problems have been studied extensively in a cost sharing context, very little attention has been given to scheduling problems; in particular if jobs are assumed to act strategically, and group-strategyproofness is a desirable objective. In most of the previous works, authors have either concentrated on scheduling problems where machines act selfishly [77, 4, 63], or strategyproofness (but not group-strategyproofness) is an issue [83, 48].

Contributions. In this chapter, we study cost sharing mechanisms for optimization problems in light of the social cost efficiency measure introduced by Roughgarden and Sundararajan [88]. Our contribution is threefold:

1. *Lower Bound on Approximability of Cost Sharing Mechanisms.*

   We present a general inapproximability result for cost sharing methods for combinatorial optimization problems. In particular, we prove that there is no cost sharing method that is α-summable and satisfies β-cost recovery for any $\alpha < H_n/\beta$, where n denotes the number of players. Our proof holds if the underlying cost function satisfies a certain cost-stability property. As a consequence, our result implies a lower bound of $\Omega(\log n)$ on the approximability of Moulin mechanisms for various optimization problems, such as, for instance, facility location, minimum spanning tree (and thus also minimum Steiner tree and forest), single-source rent-or-buy, minimum makespan scheduling, etc. Despite its generality, our lower bound is tight for some specific problems such as facility location and minimum makespan scheduling.

2. *Optimal Cost Sharing Method for Makespan Scheduling.*

   We study the *minimum makespan scheduling problem*, one of the most fundamental problems in scheduling theory, in a cost sharing context. In this problem, we are given a set of jobs N, each of which is owned by a selfish player. The objective is to assign the chosen set of jobs to m parallel machines such that the maximum completion time is minimized. We develop a cross-monotonic cost sharing method for this problem which is $(2 - 2/(m + 1))$-budget balanced and $(H_n + 1)$-approximate; this is tight with respect to both budget balance and approximability.

   Related to this result is the recent work of Bleischwitz and Monien [11]. The authors present a cross-monotonic $(2-2/(m+1))$-budget balanced cost sharing method for the minimum makespan scheduling problem. However, we show that their cost sharing mechanism does not approximate social cost.

3. *Lower Bound on Budget Balance of Cost Sharing Mechanisms.*

   We present a generic lower bound showing that no cross-monotonic and β-budget balanced cost sharing method exists for any $\beta < f(n)$, where f is a function that measures the maximum rate of increase of the underlying cost function C. For example, for every fixed player set, f is at least the ratio between the cost of the whole set and the sum of the costs of all its singleton subsets. We prove that this lower bound even holds for cost sharing methods in the *β-core.* We use this general approach to prove negative results for several fundamental scheduling problems in a cost sharing context. Namely, we show a lower bound of $\Omega(n)$ on the budget balance factor of cross-monotonic cost sharing methods for all scheduling problems in which we aim at minimizing the total (weighted) completion (or flow) time. We prove that the same bound also applies to average (weighted) completion (or flow) time objectives.

Organization of Chapter. The general lower bound on the approximability of cost sharing mechanisms is presented in Section 3.2. Our tight cost sharing mechanism for the minimum makespan scheduling is given in Section 3.3. The negative results with respect to approximating the budget balance factor for certain cost functions together with its applications to completion time related scheduling problems is stated in Section 3.4. Finally, we offer some conclusions in Section 3.5.

3.2 A Lower Bound for Social Cost Approximation

In this section, we prove a lower bound of $\Omega(\log n)$ on the summability of cost sharing methods. Our lower bound holds for every optimization problem which contains a so-called *cost stable* instance. Intuitively, we call an instance cost stable if it contains a significantly large player set whose cost does not deviate too much from the cost of any of its subsets. This property is fulfilled by a variety of combinatorial optimization problems such as facility location, Steiner tree, minimum makespan scheduling, etc. In particular, it includes the *public excludable good problem*, in which the cost $C(S)$ is equal for every player set $S \subseteq U$. Together with the recent result of Roughgarden and Sundararajan [88], our result shows that for all these problems, the approximability of Moulin mechanisms cannot be better than $\Omega(\log n)$.

Theorem 3.1. *Consider an instance of a combinatorial optimization problem on a player set U inducing a cost function C. Suppose that there is a set $S \subseteq U$ of size $|S| \geq |U|/\gamma$ for some constant $\gamma \geq 1$ such that*

$$C(S') \geq \frac{1}{\delta} \cdot C(S)$$

for all $S' \subseteq S$ and some constant $\delta \geq 1$. Let ξ be a cost sharing method for this problem that satisfies the β-cost recovery condition. Then, ξ is not α-summable for any $\alpha < H_{\lceil n/\gamma \rceil}/(\beta \cdot \delta)$.

Proof. It is sufficient to prove that there exists an order σ on U such that

$$\sum_{j=1}^{|S|} \xi(i_j, S_j) \geq \frac{H_{\lceil n/\gamma \rceil}}{\beta \cdot \delta} \cdot C(S),$$

where S_j is the set of the first j players in S and i_j is the jth player of S (ordered according to σ).

We construct σ by determining the sets S_j and players i_j inductively as follows. Initially, set $j = |S|$ and assign $S_j = S$. Now, suppose we have determined sets $S_{|S|}, \dots, S_j$. By an average argument, there must exist a player $i \in S_j$ such that

$$\xi(i, S_j) \geq \frac{C(S_j)}{\beta \cdot |S_j|} = \frac{C(S_j)}{\beta \cdot j} \geq \frac{C(S)}{\beta\delta \cdot j},$$

since ξ satisfies the β-cost recovery condition. The last inequality holds because $S_j \subseteq S$. Assign $i_j := i$ and $S_{j-1} := S_j \setminus \{i_j\}$.

Let $S = \{i_1, \dots, i_{|S|}\}$ be the set of players in S ordered according to the order σ constructed above. We have

$$\sum_{j=1}^{|S|} \xi(i_j, S_j) \geq \left(1 + \frac{1}{2} + \dots + \frac{1}{|S|}\right) \cdot \frac{C(S)}{\beta\delta} \geq \frac{H_{\lceil n/\gamma \rceil}}{\beta\delta} \cdot C(S),$$

where we exploit that $|S| \geq n/\gamma$ and $|S| \in \mathbb{N}$. □

This lower bound applies to many problems, as e.g. to the following ones:

Example 3.1 (Fixed-tree Multicast Problem)**.** *Players are located at vertices of an undirected graph and wish to receive a broadcasting service which is produced in a root vertex. The cost of serving a set of players S is the cost of a minimum spanning tree containing S and the root. An instance fulfilling the conditions of the above theorem is the one in which all players are located on the same vertex which is connected to the root by an edge of length 1. The lower bound for this problem has been shown in [88].*

Example 3.2 (Facility Location Problem)**.** *Players are located at vertices and wish to be connected to an open facility. Facilities can be opened at a given subset of vertices. Here, a sample instance is the one in which there is only one vertex v at which a facility may be opened, and all players are located directly on v. Then, the cost of a solution is independent of the number of players and equal to the opening cost of the facility. This lower bound is tight, as has been shown in [89].*

Another example for which Theorem 3.1 applies is the makespan scheduling problem that we consider in Section 3.3. We show there that the bound on summability is tight for this problem. We remark that there exist stronger lower bounds for e.g. the Steiner tree and Steiner forest problems [89].

3.3 Optimal Cost Sharing Method for Makespan Scheduling

We consider the classical *minimum makespan scheduling problem* $P| |C_{\max}$. We are given a set of n jobs N that have to be scheduled on m identical machines. Each job $i \in N$ has a non-negative *processing time* p_i, which is the time needed to execute i on one of the machines. We denote the completion time of job i by C_i. Every machine can execute at most one job at a time; preemption of jobs is not allowed. The objective is to schedule all jobs in N on the m machines such that the *makespan* $\max_{i \in N} C_i$ is minimized.

In the cost sharing variant of this machine scheduling problem, each job is associated with a player who wants her job to be processed on one of the m machines. We therefore identify the universe of players U with the set of jobs N. The cost $C(S)$ incurred to schedule all jobs in S is the minimum makespan. We are interested in designing a cost sharing mechanism for the minimum makespan scheduling problem that is β-budget balanced and α-approximate for every possible instance.

For a given set of jobs $S \subseteq U$, let $p_{\max}(S)$ denote the maximum processing time over all jobs in S. Define $\mu(S)$ as the average machine load, i.e. $\mu(S) := \sum_{i \in S} p_i/m$.

3.3.1 Cross-Monotonic Cost Shares

Bleischwitz and Monien [10] describe a cross-monotonic cost sharing method ξ^{BM} for the above machine scheduling problem. We briefly review their cost sharing method.[1]

We call a job i *large* with respect to S if $p_i = p_{\max}(S)$ and *small* otherwise. Let $\ell(S)$ be the number of large jobs in S. Given a subset $S \subseteq U$ of the jobs, we define the cost share of $i \in S$ as:

$$\xi^{\mathrm{BM}}(i,S) := \begin{cases} \dfrac{p_i}{m} + \dfrac{p_i - \mu(S)}{\ell(S)} & \text{if } p_i = p_{\max}(S) \text{ and } p_i > \mu(S), \\ \dfrac{p_i}{m} & \text{otherwise.} \end{cases} \tag{3.1}$$

The intuition is as follows: Every job gets a cost share of p_i/m. If the average machine load $\mu(S)$ is less than the maximum processing time $p_{\max}(S)$, every large job additionally obtains an equal share of the cost $p_{\max}(S) - \mu(S)$. We summarize one of the main results of Bleischwitz and Monien [10] in the following theorem.

[1] At first sight, the cost shares that we state here differ from the ones defined by Bleischwitz and Monien in [10]. However, it can easily be verified that both definitions are in fact equivalent; we feel that the definition we present here is more intuitive.

Theorem 3.2 ([10]). *ξ^{BM} is a $(2-2/(m+1))$-budget balanced cross-monotonic cost sharing method for the minimum makespan scheduling problem. Moreover, there is no β-budget balanced cross-monotonic cost sharing method ξ for this problem, for any $\beta < 2 - 2/(m+1)$.*

Albeit Theorem 3.2 proves that the Moulin mechanism $M(\xi^{\text{BM}})$, driven by the cost sharing method ξ^{BM} by Bleischwitz and Monien, is optimal with respect to budget balance, we show below that it is far from being optimal with respect to social cost. In fact, the social cost of the final set S^M output by $M(\xi^{\text{BM}})$ can be as large as $n/2$ times the optimal social cost.

Lemma 3.3. *For every $n \in \mathbb{N}$, there exists an instance of the minimum makespan scheduling problem such that the cost sharing method ξ^{BM} is not α-summable for any $\alpha < n/2$.*

Proof. It is sufficient to define an instance of the minimum makespan scheduling problem on n jobs and a permutation σ for which the cost share sum in Definition 2.34 with respect to ξ^{BM} is at least $n/2$ times the minimum makespan.

Let $U := \{i_1, \dots, i_m\}$ be an (ordered) set of m jobs, where $m = n$ is the number of machines. Define the processing time of job i_j to be $p_{i_j} := 1 + (j-1)\epsilon$ for all $j \in [m]$ and some small $\epsilon > 0$. Since the number of jobs equals the number of machines, the makespan of an optimal assignment for U is $C(U) = 1 + (m-1)\epsilon$.

Observe that the processing time of job i_j, $j \in [m]$, is maximum among all jobs in the set $S_j = \{i_1, \dots, i_j\}$, i.e. i_j is large. Furthermore, i_j is the only large job in S_j and thus $\ell(S_j) = 1$. The average machine workload of S_j is

$$\mu(S_j) = \frac{1}{m}\sum_{l=1}^{j} p_{i_l} = \frac{1}{m}\left(j + \frac{j(j-1)\epsilon}{2}\right) \leq 1 + (j-1)\epsilon = p_{\max}(S_j).$$

Hence, the cost share that job i_j obtains with respect to S_j is

$$\xi^{\text{BM}}(i_j, S_j) = \frac{p_{i_j}}{m} + p_{i_j} - \mu(S_j) = p_{i_j} - \mu(S_{j-1}),$$

where we define $S_0 := \emptyset$. We obtain

$$\xi^{\text{BM}}(i_j, S_j) = (1 + (j-1)\epsilon) - \frac{1}{m}\left((j-1) + \frac{(j-1)(j-2)\epsilon}{2}\right) \geq 1 - \frac{j-1}{m}.$$

Therefore,

$$\sum_{j=1}^{m} \xi^{\text{BM}}(i_j, S_j) \geq m - \frac{m(m-1)}{2m} = \frac{m}{2} + \frac{1}{2} \geq \frac{m}{2}(1 + (m-1)\epsilon) = \frac{m}{2} \cdot C(U),$$

where the last inequality holds if we choose ϵ sufficiently small. $\square$

Intuitively, this high summability gives voice to the fact that processing times exceeding the average workload $\mu(S)$ are punished in an unfair manner: Instead of sharing the additional cost of $p_{\max}(S) - \mu(S)$ among all jobs for which $p_i > \mu(S)$, only those jobs attaining the maximum processing time come up for it. We tackle this problem in the next section.

3.3.2 Approximate Cost Shares

We continue by proposing new cost shares ξ^{BS} for the minimum makespan scheduling problem that are still $(2 - 2/(m+1))$-budget balanced and cross-monotonic, but concurrently (H_n+1)-summable. This is tight in terms of both budget balance and summability.

We use a different definition of *small* and *large* jobs here: A job i is *large* with respect to S iff $p_i > \mu(S)$ and *small* otherwise. The cost share of a job $i \in S$ with respect to S is defined as

$$\xi^{\text{BS}}(i,S) := \begin{cases} \dfrac{p_i}{m} + \displaystyle\int\limits_{\mu(S)}^{p_i} \frac{1}{|\{j \in S : p_j \geq t\}|}\,\mathrm{d}t & \text{if } p_i > \mu(S), \\ \dfrac{p_i}{m} & \text{otherwise.} \end{cases} \tag{3.2}$$

Intuitively, every job receives a cost share of p_i/m. A large job i obtains some additional cost share: for every time instant $t \in [\mu(S), p_i]$, i shares the cost of $1\mathrm{d}t$ evenly with all other jobs in S whose processing time is at least t.

We show that ξ^{BS} is a cost sharing method that satisfies cross-monotonicity, approximate budget balance and summability.

Theorem 3.3. *ξ^{BS} is a cross-monotonic, $(2 - 2/(m+1))$-budget balanced and (H_n+1)-summable cost sharing method for the minimum makespan scheduling problem.*

Our cost sharing method is essentially tight: Bleischwitz and Monien [10] proved that no cross-monotonic cost sharing method for this problem achieves a budget balance factor better than $2 - 2/(m+1)$. Moreover, using Theorem 3.1 we show (see Corollary 3.7 below) that no cost sharing method that satisfies the β-cost recovery condition can be α-summable for any $\alpha < H_n/\beta$.

The proof of Theorem 3.3 follows from Lemmas 3.4, 3.5 and 3.6 that are given below.

Lemma 3.4. *ξ^{BS} is cross-monotonic.*

Proof. Consider some set $S \subseteq U$ and a job $i \in S$. We prove that if a new job $j \notin S$ is added to S, the cost share of i does not increase.

If i was small in S, then it remains small, and hence i's cost share stays p_i/m. If i was large in S and becomes small in $S \cup \{j\}$, then i's cost share decreases to p_i/m. It remains to show that the cost share of i does not increase if i stays large. Note that by adding job j, the number of jobs whose processing time is at least t for some $t \geq 0$ does not decrease. Moreover, we have

$$\int_{\mu(S)}^{p_i} \frac{1}{|\{j \in S : p_j \geq t\}|}\,\mathrm{d}t \geq \int_{\mu(S \cup \{j\})}^{p_i} \frac{1}{|\{j \in S \cup \{j\} : p_j \geq t\}|}\,\mathrm{d}t,$$

since $\mu(S) \leq \mu(S \cup \{j\})$. This concludes the proof. □

We show next that the budget balance condition is satisfied.

Lemma 3.5. ξ^{BS} *is* $(2 - 2/(m+1))$*-budget balanced.*

Proof. With the cost share definition in (3.2) we have

$$\begin{aligned}\sum_{i \in S} \xi(i,S) &= \sum_{i \in S} \frac{p_i}{m} + \sum_{i \in S:\, p_i > \mu(S)} \int_{\mu(S)}^{p_i} \frac{1}{|\{j \in S : p_j \geq t\}|}\,\mathrm{d}t \\ &= \mu(S) + \int_{\mu(S)}^{p_{\max}(S)} 1\,\mathrm{d}t = \max\{\mu(S), p_{\max}(S)\}.\end{aligned}$$

It is a well-known fact (see e.g. [49]) that $C(S) \geq \max\{\mu(S), p_{\max}(S)\}$, which proves competitiveness. To show that the cost shares satisfy $(2 - 2/(m+1))$-cost recovery, we refer to the proof of Theorem 3.1 in [11]. Since $\sum_{i \in S} \xi^{\text{BM}}(i,S) = \max\{\mu(S), p_{\max}(S)\}$ holds for Bleischwitz and Monien's cost shares as well, their budget balance analysis directly applies here. □

Finally, we prove that the cost shares fulfill $(H_n + 1)$-summability.

Lemma 3.6. ξ^{BS} *is* $(H_n + 1)$*-summable.*

Proof. Let σ be an arbitrary order on the jobs in U, and let $S := \{i_1, \ldots, i_{|S|}\} \subseteq U$

be a subset of U ordered according to σ. First, observe that

$$\begin{aligned}\sum_{j=1}^{|S|} \xi^{\mathrm{BS}}(i_j, S_j) &= \sum_{j=1}^{|S|} \left(\frac{p_{i_j}}{m} + \int_{\mu(S)}^{p_{i_j}} \frac{1}{|\{k \in S_j : p_k \geq t\}|} \,\mathrm{d}t \right) \\ &\leq \sum_{j=1}^{|S|} \left(\frac{p_{i_j}}{m} + \int_{0}^{p_{i_j}} \frac{1}{|\{k \in S_j : p_k \geq t\}|} \,\mathrm{d}t \right) \\ &\leq \mu(S) + \sum_{j=1}^{|S|} \int_{0}^{p_{i_j}} \frac{1}{|\{k \in S_j : p_k \geq t\}|} \,\mathrm{d}t.\end{aligned}$$

Fix a point in time $t \in [0, p_{\max}(S)]$. Define $r(t)$ as the number of jobs in S whose processing time is at least t. Using this definition, we obtain

$$\sum_{j=1}^{|S|} \int_{0}^{p_{i_j}} \frac{1}{|\{k \in S_j : p_k \geq t\}|} \,\mathrm{d}t = \int_{0}^{p_{\max}(S)} \sum_{r=1}^{r(t)} \frac{1}{r} \,\mathrm{d}t = \int_{0}^{p_{\max}(S)} H_{r(t)} \,\mathrm{d}t \leq p_{\max}(S) \cdot H_{|S|}.$$

Thus,

$$\sum_{j=1}^{|S|} \xi^{\mathrm{BS}}(i_j, S_j) \leq \mu(S) + p_{\max}(S) \cdot H_{|S|} \leq (H_n + 1) \cdot C(S).$$

□

Using Theorem 3.1, we can prove that Lemma 3.6 is essentially tight.

Corollary 3.7. *Let ξ be a cost sharing method for the minimum makespan scheduling problem $P|p_i = 1|C_{\max}$ that satisfies the β-cost recovery condition. Then the summability of ξ is no better than H_n/β.*

Proof. Consider an instance that consists of n jobs with unit processing times and $m := n$ machines. Clearly, $C(S) = 1 = C(U)$ for all $S \subseteq U$. Theorem 3.1 now gives a lower bound of H_n/β. □

3.4 A General Lower Bound on Budget Balance

While we have given cross-monotonic $(2-2/(m+1))$-budget balanced and (H_n+1)-summable cost shares for the minimum makespan scheduling problem, we identify a class of problems to which no constantly budget balanced and cross-monotonic cost sharing method exists in this section. We show that both weighted completion time scheduling and average completion time scheduling belong to this class, as well as all of their generalizations.

Consider a cost sharing game on a universe U of n players whose cost function $C : 2^U \to \mathbb{R}$ is non-decreasing, i.e. $C(S') \le C(S)$ for all $S' \subseteq S \subseteq U$. If there is an instance to the cost sharing game for which $C(U)$ exceeds $\sum_{i \in U} C(\{i\})$ by a factor of $f(n)$, then the β-core of this game is empty for all $\beta < f(n)$. This is due to the fact that players can never be charged more than the cost they incur when being served alone, and therefore the players in a set S cannot pay more than $\sum_{i \in S} C(\{i\})$.

In the case of general (not necessarily non-decreasing) cost functions, using sets T_i containing i instead of the singletons $\{i\}$ itself can yield even better lower bounds. Intuitively, we choose the subset $T_i \subseteq S$ for which the amount that player $i \in S$ is allowed to pay is smallest.

Theorem 3.4. *Consider a cost sharing game on a universe U of n players and its cost function C. Let $f : \mathbb{N} \to \mathbb{R}$ be a non-decreasing function. Suppose there is a set S of size $|S| \ge |U|/\gamma$ for some constant $\gamma \ge 1$, and arbitrary sets $T_i \subseteq S$ with $i \in T_i$ such that*

$$C(S) \ge f(|S|) \cdot \sum_{i \in S} C(T_i).$$

Then, there is no cost sharing method ξ in the β-core for any $\beta < f(\frac{n}{\gamma})$ for this game.

Proof. Assume that ξ is a cost sharing method in the β-core for this problem. First, the core property implies that the cost share of player i in the set $S \supseteq T_i$ is at most the cost induced by the set T_i, i.e. $\xi(i, S) \le \sum_{j \in T_i} \xi(j, S) \le C(T_i)$ for all $i \in S$. Second, we assume $C(S) \ge f(|S|) \cdot \sum_{i \in S} C(T_i)$. The condition of β-cost recovery now implies that for every $S \subseteq U$:

$$\beta \ge \frac{C(S)}{\sum_{i \in S} \xi(i, S)} \ge \frac{f(|S|) \cdot \sum_{i \in S} C(T_i)}{\sum_{i \in S} C(T_i)} \ge f(\frac{n}{\gamma}).$$

□

Since every *cross-monotonic* β-budget balanced cost sharing method is in the β-core, this theorem implies the same lower bound on the budget balance factor of a cross-monotonic cost sharing method for the respective problem. In the following two sections, we apply Theorem 3.4 to the parallel machine scheduling problems with completion time and average completion time objectives.

3.4.1 Weighted Completion Time Scheduling

In the *minimum weighted completion time scheduling problem*, we are given a set of n jobs N and m identical machines. Each job $i \in N$ has a processing time p_i and a weight w_i. The objective is to assign all n jobs to the m machines such that the total weighted completion time $\sum_{i \in N} w_i C_i$ is minimized.

In the cost sharing context, we define $U := N$ as before, and let C be the total weighted completion time of an optimal schedule. We show that the β-core of this scheduling problem is empty for $\beta < (n+1)/2$, even for the unweighted single machine case with unit processing times.

Corollary 3.8. *Consider the single machine minimum completion time scheduling problem with unit processing times $1|p_i = 1|\sum_i C_i$. There is no cost sharing method ξ that is in the β-core for any $\beta < (n+1)/2$ for this game.*

Proof. Clearly, the cost of every singleton set $\{i\}$, $i \in U$, is $C(\{i\}) = 1$. Set $T_i := \{i\}$. On the other hand, $C(U) = n(n+1)/2$. Thus,

$$C(U) \geq \frac{n+1}{2} \cdot \sum_{i \in U} C(T_i),$$

and using Theorem 3.4 with $S = U$ and $f(n) = (n+1)/2$ yields the claim. $\square$

This lower bound carries over to all generalizations of the single machine minimum completion time scheduling problem, e.g. to the *minimum weighted flow time scheduling problem* and problems with additional constraints such as release or due dates. Note that the trivial cost sharing method $\xi^{\text{WCT}}(i, S) := w_i p_i$ for all $i \in S$ and $S \subseteq U$ is cross-monotonic and n-budget balanced for $P|\,|\sum_i w_i C_i$, as shown by the following lemma.

Lemma 3.9. *Consider the minimum weighted completion time scheduling problem $P|\,|\sum_i w_i C_i$. ξ^{WCT} is a cross-monotonic n-budget balanced cost sharing method for this problem.*

Proof. ξ^{WCT} is obviously cross-monotonic. It is competitive since $p_i \leq C_i$ for every job $i \in S$ and all $S \subseteq U$.

To show n-cost recovery, we first consider the single machine case. Take an optimal schedule and number the jobs accordingly. Smith [96] proved that if job i is scheduled before job j in an optimal schedule, i.e. $i < j$, then $p_i/w_i \leq p_j/w_j$. Thus, either $p_i \leq p_j$ or $w_i > w_j$ (or both) are true, and the following inequality holds for all $i < j$:

$$w_j p_i \leq \max\{w_i p_i, w_j p_j\} \leq w_i p_i + w_j p_j.$$

Using this, we can bound the cost of an optimal schedule for a set $S \subseteq U$ by

$$\begin{aligned}\sum_{j=1}^{|S|} w_j C_j &= \sum_{j=1}^{|S|} w_j \cdot \Big(\sum_{i=1}^{j} p_i\Big) = \sum_{j=1}^{|S|}\sum_{i=1}^{j} w_j p_i \leq \sum_{j=1}^{|S|} \Big(\sum_{i=1}^{j-1}(w_i p_i + w_j p_j) + w_j p_j\Big) \\ &= \sum_{j=1}^{|S|}\big(j \cdot w_j p_j + (|S| - j) w_j p_j\big) = |S| \cdot \sum_{j=1}^{|S|} w_j p_j \leq n \cdot \sum_{j=1}^{|S|} \xi^{\text{WCT}}(j, S),\end{aligned}$$

which proves n-cost recovery for the single machine case.

For the general case, consider the set $S_k \subseteq S$ of jobs that are scheduled on machine $k \in [m]$ in an optimal schedule. Clearly, the schedule for machine k is optimal for the corresponding single machine problem on the set of jobs S_k, for which the above inequality holds. Summing up over all machines, we obtain

$$\sum_{j \in S} w_j C_j = \sum_{k \in [m]} \sum_{j \in S_k} w_j C_j \leq \sum_{k \in [m]} n \cdot \sum_{j \in S_k} \xi^{\text{WCT}}(j, S_k) = n \cdot \sum_{j \in S} \xi^{\text{WCT}}(j, S).$$

□

3.4.2 Average Completion Time Scheduling

In the *minimum average completion time scheduling problem*, the setting is as above, but with the objective of minimizing the total *average* weighted completion time, i.e. $C(S) = \sum_{i \in S} w_i C_i / |S|$ for all $S \subseteq U$. In classical machine scheduling, where an optimal (or approximate) solution for the whole set U of players is sought, the problems with average weighted completion time and weighted completion time objectives coincide, since the objectives only differ by a constant factor of $|U|$. However, during the run of a Moulin mechanism, the size of the current player set varies, and thus $|S|$ can no more be seen as a constant. As a matter of fact, due to the division by $|S|$, the cost function is not monotone for this game, as the following example shows:

Example 3.10. *Consider an instance on a single machine and three jobs* $\{1, 2, 3\}$ *with processing times* $p_i = i$ *and unit weights. The average completion time is* 3 *if only job* 3 *is scheduled,* $(1 + 4)/2 = 2.5$ *if jobs* 1 *and* 3 *are scheduled, and* $(1 + 3 + 6)/3 > 3$ *if all three jobs are scheduled. Hence, the cost of an optimal schedule can increase as well as decrease when a job is added to the scheduled set.*

For this reason, we need a slightly more elaborated instance with non-uniform processing times to show that the β-core of this game is empty for $\beta < (n + 4)/8$. Nevertheless, the lower bound holds even for the unweighted single machine case.

Corollary 3.11. *Consider the single machine minimum average completion time scheduling problem* $1| \,|\sum_i C_i/n$*. There is no cost sharing method* ξ *that is in the* β*-core for any* $\beta < (n+4)/8$ *for this game.*

Proof. Let $U = S \mathbin{\dot\cup} L$ be a set of n jobs, where $|S| = n/2 - 1$ and $|L| = n/2 + 1$; we call the jobs in S *small* and those in L *large*. Define $p_i := \epsilon$ for all $i \in S$, and $p_i := 1$ for all $i \in L$. The optimal cost for every singleton set $\{i\}$, $i \in S$, is $C(\{i\}) = \epsilon$. Set $T_i := \{i\}$ for all small jobs $i \in S$. For the large jobs $i \in L$, set $T_i := S \cup \{i\}$. In an optimal schedule for T_i, first all small jobs in S are processed and finally the large job i. The cost of an optimal schedule is thus

$$C(T_i) = \frac{1}{|T_i|}\left(\sum_{j=1}^{|S|} j \cdot \epsilon + (|S| \cdot \epsilon + 1)\right) \leq \frac{2}{n}\Big(1 + \epsilon n\Big(\frac{n}{8} + \frac{1}{2}\Big)\Big).$$

We obtain

$$\begin{aligned}\sum_{i \in U} C(T_i) &\leq \Big(\frac{n}{2} - 1\Big) \cdot \epsilon + \Big(\frac{n}{2} + 1\Big)\left(\frac{2}{n}\Big(1 + \epsilon n\Big(\frac{n}{8} + \frac{1}{2}\Big)\Big)\right) \\ &\leq \frac{n+2}{n}\Big(1 + \epsilon n\Big(\frac{n}{8} + 1\Big)\Big). \qquad (3.3)\end{aligned}$$

Define $\epsilon' := \epsilon n(n/8 + 1)$. On the other hand,

$$C(U) \geq \frac{(\frac{n}{2} + 1)(\frac{n}{2} + 2)}{2n} = \frac{(n+2)(n+4)}{8n}. \qquad (3.4)$$

Combining inequalities (3.3) and (3.4), we obtain

$$C(U) \geq \frac{n+4}{8(1+\epsilon')} \cdot \sum_{i \in U} C(T_i).$$

By Theorem 3.4, we obtain a lower bound of $\beta \geq (n+4)/(8(1+\epsilon'))$ for any cost sharing method in the β-core. The claim now follows by choosing ϵ sufficiently small. □

Note that the trivial cost sharing method $\xi^{\text{ACT}}(i, S) := w_i p_i/n$ for all $i \in S$ and $S \subseteq U$ is cross-monotonic and n-budget balanced for $P| \,|\sum_i w_i C_i/n$.

Lemma 3.12. *Consider the minimum average weighted completion time scheduling problem* $P| \,|\sum_i w_i C_i/n$*.* ξ^{ACT} *is a cross-monotonic* n*-budget balanced cost sharing method for this problem.*

Proof. ξ^{ACT} is obviously cross-monotonic. It is competitive since $w_i p_i/n \leq w_i C_i/|S|$ for every job $i \in S$ and all $S \subseteq U$. The proof of n-cost recovery is analogous to the non-average case.

On a single machine, Smith's rule still holds for every optimal schedule for S since the average cost is only a constant factor times the non-average cost for fixed S. Thus, the cost of an optimal schedule for a set $S \subseteq U$ is bounded by

$$\sum_{j=1}^{|S|} w_j C_j/|S| \leq \sum_{j=1}^{|S|} w_j p_j = n \cdot \sum_{j=1}^{|S|} \xi^{\text{ACT}}(j, S).$$

For the general case, again, considering the sets $S_k \subseteq S$ for all machines $k \in [m]$ and summing up yields

$$\sum_{j \in S} w_j C_j/|S| \leq \sum_{k \in [m]} \sum_{j \in S_k} w_j C_j/|S_k| \leq \sum_{k \in [m]} n \cdot \sum_{j \in S_k} \xi^{\text{ACT}}(j, S_k) = n \cdot \sum_{j \in S} \xi^{\text{ACT}}(j, S),$$

proving n-cost recovery. □

3.5 Conclusion

We proved that the efficiency of Moulin mechanisms is not approximable within less than logarithmic factors even with the new social cost efficiency measure. Our lower bound holds if the underlying optimization problem satisfies a certain cost-stability property and in particular holds for all problems containing the public excludable good problem. This reduces the hope to find truly efficient cost sharing mechanisms for these problems. On the other hand, the new efficiency measure allows us to characterize cost sharing mechanisms in terms of their best polylogarithmic approximation factor.

Although most of the previously known cross-monotonic and approximately budget balanced cost sharing methods for combinatorial optimization problems turned out to simultaneously achieve the best possible social cost efficiency [88, 45, 89, 25], our work reveals that different cost sharing methods achieving the same budget balance factor may indeed behave very differently with respect to social cost.

We studied cost sharing methods for makespan and completion time related scheduling problems. Our results demonstrate that the tractability of these problems in a cost sharing context heavily depends on the respective objective function. Our negative result on the budget balance factor for cross-monotonic cost sharing methods motivates the investigation of alternative cost sharing models; perhaps with a weaker notion of truthfulness for cooperative games.

4

WEAKLY GROUP-STRATEGYPROOF COST SHARING

We study the problem of devising truthful mechanisms for cooperative cost sharing games that realize (approximate) budget balance and social cost. Recent negative results show that group-strategyproof mechanisms can only achieve very poor approximation guarantees for several fundamental cost sharing games. Driven by these limitations, we consider cost sharing mechanisms that realize the weaker notion of *weak group-strategyproofness.* Mehta et al. [69] recently introduced the broad class of weakly group-strategyproof *acyclic mechanisms* and show that several primal-dual approximation algorithms naturally give rise to such mechanisms with attractive approximation guarantees. In this chapter, we provide a very simple yet powerful approach that enables us to turn any ρ-approximation algorithm into a ρ-budget balanced acyclic mechanism. We demonstrate the applicability of our approach by deriving weakly group-strategyproof mechanisms for several fundamental scheduling problems that outperform the best possible approximation guarantees of Moulin mechanisms. The mechanisms that we develop for completion time scheduling problems are the first mechanisms that achieve constant budget balance and social cost approximation factors. Interestingly, our mechanisms belong to the class of *generalized incremental mechanisms* proposed by Moulin [73].

Publication Note. The results contained in this chapter have been presented at the *1st International Symposium on Algorithmic Game Theory (SAGT 08)* [18].

4.1 Introduction

In this chapter, we reconsider the problem of designing efficient mechanisms for cooperative cost sharing games arising from combinatorial optimization problems, with a particular focus on scheduling problems. We consider the binary demand cost sharing setting introduced in Section 2.3.1. That is, we are given a set of n players that are interested in receiving a common service and a player-set dependent cost function. Each player has a private valuation for receiving the service and announces a bid which represents the maximum price she is willing to pay. Based on these bids, a cost sharing mechanism decides which players receive the service and at what price. We assume that every player acts strategically in that she aims for maximizing her own quasi-linear utility.

In recent years, considerable progress has been made in devising truthful mechanisms for cooperative cost sharing games. Section 2.4 gives an overview over previous results in the area. Most optimization problems that have been considered in the literature bear the common characteristic that larger sets of served players have a larger potential for individual cost savings because the underlying cost functions are subadditive. In general, scheduling problems do not have this characteristic since the objective functions are usually superadditive and thus adding additional players corresponds to a larger potential of individual cost increases. To the best of our knowledge, there are only a few articles that study scheduling problems in the cost sharing context [10, 12, 16].

Recent negative results showed that for several fundamental cost sharing games, Moulin mechanisms inevitably suffer from poor approximation factors with respect to budget balance [10, 16, 52, 60, 88] or social cost [16, 25, 88, 89]. As an example, consider the *minimum completion time scheduling game*. Here the task is to schedule n jobs (each owned by a selfish player) non-preemptively on m parallel machines such that the sum of the completion times of all jobs is minimized. We showed in Chapter 3 that no cross-monotonic cost sharing method can achieve a budget balance approximation factor of less than $(n+1)/2$, even in the single-machine case and if all jobs have unit processing times. This is in stark contrast to the fact that one can even compute an optimal schedule for this problem in polynomial time.

Contributions. In this chapter, we present an approach to derive weakly group-strategyproof mechanisms from approximation algorithms. More precisely, we show how a ρ-approximation algorithm for the underlying optimization problem of a cost sharing game can be turned into a generalized incremental mechanism that is ρ-budget balanced, and prove that this mechanism is weakly group-strategyproof. Our construction uses the approximation algorithm as a black-box. The basic idea is very simple: According to the order on the set of players that are remaining

Problem	Generalized incremental mechanisms	Lower bounds for Moulin mechanisms
$P\|p_i = 1\|C_{\max}$	$(1, \log n)$	2
$P\| \|C_{\max}$	$\frac{4}{3} - \frac{1}{3m}$	$\frac{2m}{m+1}$ [10]
$P\| \|\sum C_i$	(1, 2)	$\frac{n+1}{2}$ [16]
$P\| \|\sum w_i C_i$	(1.21, 2.42)	$\frac{n+1}{2}$ [16]
$1\|r_i, pmtn\|\sum C_i$	(1, 4)	$\frac{n+1}{2}$ [16]
$P\|r_i, pmtn\|\sum_i C_i$	(1.25, 5)	$\frac{n+1}{2}$ [16]
$1\|r_i, pmtn\|\sum F_i$	1	$\frac{n+1}{2}$ [16]

Table 4.1: Summary of the approximation guarantees obtained by generalized incremental mechanisms in this chapter. (β, α) denotes the budget balance (β) and social cost (α) approximation factors; all other entries refer to budget balance factors. The respective lower bounds on the budget balance factors of Moulin mechanisms are given in the right column.

in the game (which may change during the course of the mechanism), we charge every player her incremental cost with respect to the approximate cost function induced by the approximation algorithm.

A difficulty that arises is that in general the resulting mechanism per-se does not fulfill the *no positive transfer property* (which requires that all cost shares are non-negative). While there are different ways to overcome this problem, we identify a *consistency* property on the order in which players are considered and argue that the mechanism guarantees no positive transfers whenever its approximate cost is non-decreasing according to this consistent order. We provide examples that illustrate that several (approximation) algorithms naturally induce a consistent order on the players such that its cost is non-decreasing.

Exploiting the consistency property, we also provide some general means that facilitate proving social cost approximation factors of our mechanisms. Essentially, we determine a *weak monotonicity* property that, if satisfied by the incremental approximate cost shares defined by the mechanism, enables us to simplify bounding its social cost approximation guarantee.

We demonstrate the applicability of our techniques by developing weakly group-strategyproof mechanisms for completion time (and flow time) scheduling problems with and without release dates and preemption. Our techniques turn out to be particularly effective in this scheduling context. The results are summarized in

Table 4.1. Our mechanisms outperform the strong lower bounds of $(n+1)/2$ on the budget balance factor of Moulin mechanisms for all completion time related objectives presented in Chapter 3 [16]. We emphasize that these are the first cost sharing mechanisms that are weakly group-strategyproof and achieve constant approximation guarantees with respect to both budget balance and social cost.

Independently of the above framework, we define a 1-budget balanced $\log n$-approximate acyclic cost sharing mechanism for the minimum makespan scheduling problem with unit processing times $P|p_i = 1|\max C_i$. This improves over the lower bound of 2 for the budget balance factor of Moulin mechanisms for the problem.

Consequences and Further Implications. While previously, most cost sharing mechanisms were developed in case-by-case studies, this is the first characterization of sufficient conditions that allow to derive mechanisms from existing approximation algorithms, thereby exploiting their full strength. Our results can therefore be seen as an important first step towards quantifying the gap between the best possible approximation guarantees achievable by approximation algorithms and weakly group-strategyproof mechanisms.

Using our approach, deriving a weakly group-strategyproof mechanism essentially boils down to identifying a consistent order on the players such that the cost of the (approximation) algorithm is non-decreasing, which is an easy task in many cases. Moreover, the resulting mechanism inherits the approximation guarantee as budget balance factor. Hence, all that is left is to bound its social cost approximation guarantee, for which we provide some general techniques. Our approach thus simplifies the process of developing weakly group-strategyproof cost sharing mechanisms significantly.

While group-strategyproof mechanisms with good performance guarantees have been developed for many cost sharing games with subadditive cost functions, the common approaches for developing such mechanisms inevitably fail for strongly superadditive cost functions (compare Section 3.4). However, our scheduling examples demonstrate that generalized incremental mechanisms excel precisely for these types of problems. As a consequence, generalized incremental mechanisms form a good counterpart to Moulin mechanisms in the search for well-performing mechanisms for cost sharing problems with different types of cost functions.

Besides the game-theoretical insights that we gain, our results have some further implications in the scheduling context: Every mechanism that approximates the social cost objective is at the same time an approximation algorithm for the *price-collecting* variant of the underlying optimization problem. In the scheduling context, these problems are also called *scheduling problems with rejection* (formal definitions are given in Section 2.5.1). As a by-product, we therefore obtain constant factor approximation algorithms for several machine scheduling problems with rejection. This might be of independent interest.

Our generalized incremental mechanisms belong to the class of acyclic mechanisms; indeed, we first encountered these mechanisms when studying the framework of acyclic mechanisms (see also the exposition in [18]). We explain how in this framework generalized incremental mechanisms can be viewed as complementary to Moulin mechanisms regarding the degree of freedom that the mechanism has for ordering its price proposals to players.

Interestingly, the mechanisms proposed in this chapter correspond to generalized incremental mechanisms as introduced by Moulin [73] (see Section 2.4.5). Our work therefore reveals that these mechanisms naturally arise as weakly group-strategyproof mechanisms with attractive budget balance and social cost approximation guarantees for several fundamental cost sharing games for which Moulin mechanisms are doomed to fail.

Organization of Chapter. In Section 4.2, we present our general framework for deriving generalized incremental mechanisms from approximation algorithms. This framework is used in Section 4.3 to derive weakly group-strategyproof mechanisms for a series of fundamental cost sharing games from the domains of network design and scheduling. Section 4.4 provides a method for proving approximate social cost for generalized incremental mechanisms. This method is used in Section 4.5 to prove constant social cost approximation factors of generalized incremental mechanisms for several completion time scheduling problems. In Section 4.6, we draw the connection between incremental mechanisms and acyclic mechanisms and sketch the implications of our results in the area of scheduling with rejection. Section 4.7 presents our optimal weakly group-strategyproof mechanism for makespan scheduling with unit processing times. We give some concluding remarks in Section 4.8.

4.2 Generalized Incremental Mechanisms

In the following, we describe our approach for converting approximation algorithms into weakly group-strategyproof cost sharing mechanisms. We call the resulting mechanisms *generalized incremental mechanisms* due to their affinity to Moulin's mechanisms in [73].

4.2.1 Construction and Basic Properties

Suppose we are given a ρ-approximation algorithm ALG for the underlying optimization problem $\mathcal{P}$. Let $\bar{C}$ denote the cost function induced by ALG, i.e., $\bar{C}(S)$

Algorithm 3: Generalized incremental mechanism $M(\text{ALG}, \tau)$ induced by ALG and τ.

```
Input: Set of players $U$ and bid vector $\boldsymbol{b} = (b_i)_{i \in U}$
Output: Allocation vector $\boldsymbol{x} = (x_i)_{i \in U}$ and payment vector $\boldsymbol{x} = (p_i)_{i \in U}$
1 Initialize $A := \emptyset$, $R := U$.
2 while $A \neq R$ do
3     Among all players $i \in R \setminus A$, let $i^*$ be the one with minimum $\tau(i, R)$.
4     Define $x_{i^*} := \bar{C}(A \cup \{i^*\}) - \bar{C}(A)$.
5     if $b_{i^*} \geq x_{i^*}$ then set $A := A \cup \{i^*\}$;
6     else set $R := R \setminus \{i^*\}$.
7 end
8 Output the characteristic vector $\boldsymbol{x}$ of $A$ and payments $\boldsymbol{x}$.
```

is the cost of the solution computed by ALG for player set $S \subseteq U$. Without loss of generality, we assume that $\bar{C}(\emptyset) = 0$. Besides the approximation algorithm ALG, the main ingredient for our framework is an injective *order function* $\tau : U \times 2^U \to \mathbb{R}^+$ which defines a permutation for every subset $S \subseteq U$ by ordering the elements in S with respect to increasing τ-values.

The *generalized incremental mechanism* $M(\text{ALG}, \tau)$ induced by ALG and τ receives the bid vector $\boldsymbol{b}$ as input and proceeds as indicated in Algorithm 3. Throughout its execution, R refers to the set of players who currently remain in the game, and A denotes the set of players who have been accepted so far. The mechanism starts with the entire player set $R = U$ and initializes $A = \emptyset$. In every iteration, it picks the player i^* from $R \setminus A$ with the smallest τ-value, and computes her incremental approximate cost share p_{i^*}, defined as the increase in the approximate cost $\bar{C}$ when player i^* is added to A. If player i^* accepts this cost share, she is added to the set A of accepted players; otherwise, she is removed from R and hence rejected from the game. The mechanism continues like this until eventually all remaining players have been accepted. It outputs the characteristic vector $\boldsymbol{x}$ of the accepted players A and the corresponding payments $\boldsymbol{p}$ (where we implicitly set $p_i = 0$ for all $i \notin A$).

We remark that the cost shares assigned to the served players depend on the cost function $\bar{C}$ induced by the approximation algorithm ALG and are not necessarily non-negative. Thus, our generalized incremental mechanism does not necessarily satisfy the *no positive transfer* property. We address this issue in Section 4.2.2 below.

It is straightforward to see that the generalized incremental mechanism inherits its budget balance factor from the input approximation algorithm:

Lemma 4.1. *The generalized incremental mechanism $M(\text{ALG}, \tau)$ is ρ-budget bal-*

anced.

Proof. In every iteration of the mechanism, we have $\sum_{i\in A} x_i = \bar{C}(A)$, since every accepted player pays exactly the incremental approximate cost of adding her to the current set A. In particular, this is true for the output set S^M. Since ALG is a ρ-approximation algorithm, we obtain

$$\bar{C}(S^M) = \sum_{i\in S^M} x_i \leq \rho \cdot C(S^M),$$

which proves ρ-budget balance. □

We next prove that the generalized incremental mechanism is weakly group-strategyproof.

Lemma 4.2. *The generalized incremental mechanism $M(\text{ALG}, \tau)$ is weakly group-strategyproof.*

Proof. Fix a coalition $T \subseteq U$ and a bid vector $\boldsymbol{b}$ with $b_i = v_i$ for all $i \in T$. Assume for contradiction that all members of the coalition can increase their utilities by changing their bids to $\boldsymbol{b'}$ (while $b_i = b'_i$ for all $i \notin T$). The runs of the generalized incremental mechanism on $\boldsymbol{b}$ and $\boldsymbol{b'}$ are identical until the first member of T, say j, is offered a cost share. Since the cost share offered to her depends only on the set A of previously accepted players, which coincides in both runs, the utility of j is maximized when bidding v_j and cannot be influenced by other members of T. Hence j cannot increase her utility by joining the coalition. □

The following theorem follows from Lemmas 4.1 and 4.2.

Theorem 4.3. *Let τ be an order function and let ALG be a ρ-approximate algorithm for an optimization problem $\mathcal{P}$. The generalized incremental mechanism $M(\text{ALG}, \tau)$ is a weakly group-strategyproof and ρ-budget balanced cost sharing mechanism for $\mathcal{P}$, which does not necessarily satisfy the no positive transfer property.*

The following example shows that in general, generalized incremental mechanisms are not group-strategyproof.

Example 4.4. *We define an instance of a cost sharing game on $n = 2$ players with valuations $v_1 = 1$ and $v_2 = 2$. Let $\bar{C}(\{1\}) = \bar{C}(\{2\}) = 1$ and $\bar{C}(\{1,2\}) = 3$. This cost function is for instance realized by an optimal algorithm for the completion time scheduling problem on one machine with two jobs of unit processing* *times. Let τ be the offer function which orders players by their index. The induced generalized incremental mechanism accepts both players and yields utilities $u_1(v) = u_2(v) = 0$. Consider the forming of a coalition with bids $b_1 = 0$ and $b_2 = 2$. In this case, player 1 rejects, and so $u_1(b) = 0$ as before, but $u_2(b) = 2 - 1 = 1$. Hence, this coalition breaks the property of group-strategyproofness.*

4.2.2 No Positive Transfer

As we have mentioned above, our generalized incremental mechanism does not guarantee the *no positive transfer* property. This property is easily seen to be fulfilled if the *approximate* cost function $\bar{C}$ induced by the approximation algorithm ALG is non-decreasing, i.e., $\bar{C}(S) \leq \bar{C}(T)$ for all $S \subseteq T \subseteq U$.

Moreover, the following adaptation can be used to realize the no positive transfer property if the *optimal* cost function C is non-decreasing, i.e., $C(S) \leq C(T)$ for all $S \subseteq T \subseteq U$. We redefine the incremental approximate cost share in Line 4 of Algorithm 3 as

$$x_{i^*} := \max\left\{0,\ \bar{C}(A \cup \{i^*\}) - \sum_{i \in A} p_i\right\}.$$

That is, we simply charge a player zero if the incremental approximate cost of adding this player is negative.

Lemma 4.5. *The adapted generalized incremental mechanism $M(\text{ALG}, \tau)$ is ρ-budget balanced if the optimal cost function C is non-decreasing.*

Proof. The lemma can be shown by induction. Whenever a player i^* is added to the current set A, we have

$$\sum_{i \in A \cup \{i^*\}} x_i = p_{i^*} + \sum_{i \in A} p_i \geq \bar{C}(A \cup \{i^*\}).$$

If $p_{i^*} > 0$ the above inequality is an equality, and it holds that $\bar{C}(A \cup \{i^*\}) \leq \rho \cdot C(A \cup \{i^*\})$ since ALG is a ρ-approximation algorithm. Otherwise, $p_{i^*} = 0$ and we have $\sum_{i \in A} p_i \leq \rho \cdot C(A)$ by the induction hypothesis and $C(A) \leq C(A \cup \{i^*\})$ since C is non-decreasing. □

Although this adaptation works for several natural optimization problems (and indeed also for the scheduling problems considered in this chapter), we propose a different approach here. The reason for this is twofold:

1. The above adaptation may fail even for very simple optimization problems for which the underlying cost function is not monotone (as illustrated by the minimum spanning tree game below).

2. Our approach will enable us to derive attractive approximation factors with respect to both budget balance and social cost for completion time scheduling problems (see Section 4.5).

To illustrate the difficulties mentioned above, we introduce the minimum spanning tree game. In the *minimum spanning tree game*, we are given an undirected graph $G = (V \cup \{r\}, E)$ with edge weights $w_e \geq 0$ for all edges $e \in E$ and a designated root vertex r. We assume that there is an edge between r and every vertex in V. Every player $i \in U$ is associated with a unique vertex in V and the goal of player i is to connect her vertex to the root r. A *minimum spanning tree* for a given subset $S \subseteq U$ of players is a tree $\mathcal{T}$ in the subgraph induced by $S \cup \{r\}$ that spans all vertices in $S \cup \{r\}$ and minimizes the total weight $w(\mathcal{T}) := \sum_{e \in \mathcal{T}} w_e$. The cost $C(S)$ to connect a player set $S \subseteq U$ is defined as the weight of a minimum spanning tree for S. The following example shows that the cost function defined by the minimum spanning tree game is in general not non-decreasing.

Example 4.6. *Consider the instance of the minimum spanning tree game with player set $U = \{1, 2, 3\}$ depicted in Figure 4.1. All edges incident to vertex 3 have weight $(1 + \varepsilon)$ with $0 < \varepsilon < \frac{1}{3}$; all other edges have weight 2 as indicated. We have $C(\{1\}) = C(\{2\}) = 2$, $C(\{3\}) = 1+\varepsilon$, $C(\{1, 2\}) = 4$, $C(\{1, 3\}) = C(\{2, 3\}) = 2 + 2\varepsilon$ and $C(U) = 3 + 3\varepsilon$. Note that C is not non-decreasing since $C(\{1, 2\}) = 4 > 3 + 3\varepsilon = C(\{1, 2, 3\})$ by choice of ε.*

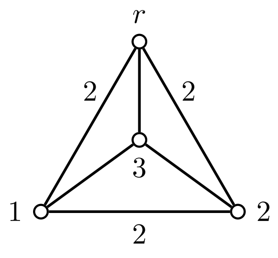

Figure 4.1

Let ALG *refer to an optimal minimum spanning tree algorithm and suppose that τ orders players by increasing index. Then, the resulting generalized incremental mechanism does not satisfy the no positive transfer property since if players 1 and 2 accept, the cost share of player 3 is $(3+3\varepsilon)-4 < 0$. However, if we choose the order function τ that orders players by decreasing index, the resulting mechanism does satisfy the no positive transfer property.*

The above example already illustrates some finer points of our approach. Essentially, we will see that many approximation algorithms exhibit a natural order function that ensures that the resulting incremental approximate cost shares are non-negative. Hence, a much weaker restriction than requiring monotonicity of the approximate cost function $\bar{C}$ (or the optimal cost function C) suffices if we define the order function τ appropriately.

4.2.3 Consistency

We now define a property of order functions that will allow us to define a much weaker requirement than monotonicity from the approximate cost function $\bar{C}$. This restriction will enable us later to find clever and well-performing combinations of approximation algorithms and order functions.

Consider a set $S \subseteq U$ and order the players in S according to increasing $\tau(\cdot, S)$ values, i.e.

$$S = \{i_1, \dots, i_p\} \quad \text{such that} \quad \tau(i_k, S) < \tau(i_l, S) \quad \forall 1 \le k < l \le p.$$

We also say S *is ordered by* τ. We denote by $S_k := \{i_1, \dots, i_k\} \subseteq S$ the set of the first $1 \le k \le p$ players in S ordered by τ and define $S_0 := \emptyset$.

Definition 4.7 (Consistent Order Function)**.** *An order function $\tau : U \times 2^U \to \mathbb{R}^+$ is* consistent *if for all subsets $S \subseteq T \subseteq U$, ordered by τ as $S = \{i_1, i_2, \dots, i_p\}$ and $T = \{j_1, j_2, \dots, j_q\}$, the following holds: If k is minimal with $j_k \in T \setminus S$, then $i_l = j_l$ for all $l < k$.*

Figure 4.2 illustrates the restriction imposed by the consistency property for two sets $S \subseteq T := \{1, \dots, 9\}$. Both sets are ordered according to τ, and elements belonging to S are depicted in gray. Intuitively, consistency requires that S has to be ordered like T up to the first element j_k of T that is missing in S (indicated in bold).

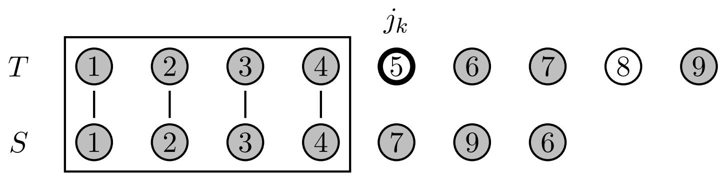

Figure 4.2: Illustration of the consistency property.

The simplest example of a consistent order function is one in which the order of every subset of players is induced by a fixed global order on U. To give an example for a more elaborate consistent order function, we reconsider the minimum spanning tree game introduced above.

Example 4.8. *Prim's algorithm (*PRIM*) [85] solves the minimum spanning tree problem optimally and proceeds as follows: It starts with the root vertex r as the initial connected component and then iteratively picks a minimum weight edge that connects a new vertex to the current component until all vertices are connected; if* *there are several minimum weight edges that might be chosen, we assume that an arbitrary but consistent tie breaking rule is used. For a subset $S \subseteq U$ of vertices, let $\tau(\cdot, S)$ be the order in which* PRIM *adds the vertices to the connected component in the run on S. It is not hard to verify that τ is a consistent order function:*

Let $S \subseteq T \subseteq U$ be two subsets ordered by τ as $S = \{i_1, i_2, \dots, i_p\}$ and $T = \{j_1, j_2, \dots, j_q\}$. Let k be minimal with $j_k \in T \setminus S$. We need to argue that the order in which PRIM *picks the first $k-1$ vertices in the run on S is the same as in the run on T, i.e. $i_l = j_l$ for all $l < k$. The proof is by induction on the first $1 \leq l < k$ vertices added by* PRIM *in the run on T. The claim clearly holds for $l = 1$, since* PRIM *starts with the same vertex in both runs. Suppose the claim is true for the first $l-1$ vertices and consider the iteration in the run on T in which vertex j_l, $1 < l < k$, is added. Then j_l is a vertex that is closest to the current connected component in the run on T. By the choice of k, j_l is also contained in S. Moreover, by the induction hypothesis, the $l-1$ vertices that have previously been chosen in the run on T and in the run on S are the same. Note that j_l must also be closest to the current connected component in the run on S because $S \subseteq T$. Thus $i_l = j_l$ (assuming a consistent tie-breaking rule). However, note that the order in which the vertices in $S \setminus \{j_1, \dots, j_{k-1}\}$ are considered might differ from the respective order in T due to the absence of j_k. Our consistency property reflects exactly this.*

We next show that a consistent order function τ in combination with a certain monotonicity property of the algorithm ALG guarantees that the resulting generalized incremental algorithm $M(\text{ALG}, \tau)$ (as defined in Algorithm 3) satisfies the no positive transfer property. Consider the execution of the generalized incremental mechanism $M(\text{ALG}, \tau)$ induced by ALG and a consistent order function τ. Recall that R refers to the set of players that are currently remaining in the game. Note that the order in which the players in $R = \{i_1, \dots, i_{|R|}\}$ are considered remains the same until the first player, say i_k, is dropped. The consistency of τ now ensures that the ordered sets $R' = R \setminus \{i_k\}$ and R agree on the first $k-1$ players. Said differently, only the order of the players succeeding i_k in R can change in R'. Hence, the first $k-1$ players correspond to the set A of currently accepted players. We prove this formally in the next lemma.

Lemma 4.9. *At the beginning of every iteration of $M(\text{ALG}, \tau)$, we have $R_{|A|} = A$.*

Proof. We prove the lemma by induction on the number of iterations. In the first iteration, $R_{|A|} = R_0 = \emptyset = A$. For the induction step, assume that $R_{|A|} = A$ at the beginning of some iteration. Let i^* be the player that is picked in this iteration, i.e., by the induction hypothesis, i^* is the $(|A|+1)$st player in the order on R. Let R' and A' denote the updated sets at the end of this iteration. There are two cases: (i) If i^* accepts, then $R' = R$ and $A' = A \cup \{i^*\}$. Hence, we can conclude that $R'_{|A'|} = R_{|A|+1} = A \cup \{i^*\} = A'$. (ii) On the other hand, if i^* rejects, then $A' = A$ and $R' = R \setminus \{i^*\}$. Note that i^* is the first element in the order on R which is not in R', and so by consistency of τ, we have $R'_{|A'|} = R_{|A|} = A = A'$. □

We can use this lemma to prove that the order in which players are added to the set A during the course of the generalized incremental mechanism $M(\text{ALG}, \tau)$ coincides with the order induced by τ on the final output set S^M.

Lemma 4.10. *Let S^M be the set of players output by $M(\text{ALG}, \tau)$. During the course of $M(\text{ALG}, \tau)$, players are added to A by increasing $\tau(\cdot, S^M)$-values.*

Proof. Suppose S^M is ordered by τ as $S^M = \{i_1, \ldots, i_p\}$. Let i be the kth player that is added to A in the execution of $M(\text{ALG}, \tau)$. We need to show that $i = i_k$.

Consider the iteration in which player i is added to A. Let R and A be the sets of remaining and accepted players at the beginning of the iteration, respectively, and let R' and A' be the respective sets at the end of the iteration. We have $R' = R$, $A' = A \cup \{i\}$ and $|A'| = k$. By Lemma 4.9, $R_{k-1} = A$ and $R'_k = R_k = A'$.

Note that all players in $R_{k-1} = A$ are contained in S^M, so by consistency of τ, at least the first $k-1$ elements of S^M coincide with those of R, i.e. $R_{k-1} = S^M_{k-1}$. By the same argument, we have $R'_k = R_k = S^M_k$. We conclude that $S^M_k \setminus S^M_{k-1} = R_k \setminus R_{k-1}$ and thus $i = i_k$. □

Lemma 4.10 has an important consequence: If the cost function $\bar{C}$ of the approximation algorithm ALG is non-decreasing as players are added to S^M one by one (in the order of τ), then the final payments charged by the generalized incremental mechanism $M(\text{ALG}, \tau)$ to the players in S^M are non-negative. Since potentially every subset $S \subseteq U$ might be chosen as the final output set, we require that the approximation algorithm satisfies this property for every subset of players:

Definition 4.11 (τ-Increasing)**.** *Let* ALG *be a ρ-approximate algorithm for an optimization problem $\mathcal{P}$. We say that* ALG *is* τ-increasing *if for every $S \subseteq U$ and every $1 \leq k \leq |S|$, we have $\bar{C}(S_k) \geq \bar{C}(S_{k-1})$.*

The following theorem now follows directly from Lemma 4.10.

Theorem 4.12. *Let τ be a consistent order function and let* ALG *be a τ-increasing ρ-approximate algorithm for an optimization problem $\mathcal{P}$. The generalized incremental mechanism $M(\text{ALG}, \tau)$ is a weakly group-strategyproof and ρ-budget balanced cost sharing mechanism for $\mathcal{P}$, which satisfies the no positive transfer property.*

We revisit the minimum spanning tree game to argue that Prim's algorithm is τ-increasing for the consistent order function defined in Example 4.8.

Example 4.13. *We argued above that Prim's algorithm induces a consistent order function τ. It is not hard to verify that* PRIM *is τ-increasing: Given an arbitrary subset $S = \{i_1, \dots, i_p\}$ ordered by τ,* PRIM *adds the vertices $i_1, \dots, i_p$ one by one to the current component. The cost $C(S_k) - C(S_{k-1})$ of adding a vertex i_k, $1 \leq k \leq p$, is equal to the weight of the edge that is added to connect i_k to the tree. Since edge weights are non-negative, it follows that* PRIM *is τ-increasing.*

Recall from Example 4.6 that PRIM *is not τ-increasing for every possible consistent order function, e.g. for the one that orders vertices by increasing index.*

In Section 4.3 we provide several additional examples of approximation algorithms that are τ-increasing with respect to natural consistent order functions.

4.3 Applications

We demonstrate the applicability of our approach by deriving generalized incremental mechanisms for a series of fundamental cost sharing games. For some of the examples given here, better cost sharing mechanisms (with respect to social cost approximation) exist in the literature. However, the main purpose of this section is to show that our mechanisms can easily be derived from existing approximation algorithms.

4.3.1 Network Design Applications

We first present examples from the field of network design.

Spanning Tree, Steiner Tree and TSP.

Reconsider the *minimum spanning tree problem (MST)* defined in Section 4.2.2. We showed in Examples 4.8 and 4.13 that the order $\tau(\cdot, S)$ in which PRIM adds the vertices to the connected component if run on S is a consistent order function and that PRIM is τ-increasing. We define $M^{\text{PRIM}} := M(\text{PRIM}, \tau)$ as the generalized incremental mechanism induced by PRIM and τ. Theorem 4.12 yields the following corollary.

Corollary 4.14. *The generalized incremental mechanism M^{PRIM} induced by Prim's algorithm and τ is weakly group-strategyproof and budget balanced for the minimum* *spanning tree problem.*

We can use Prim's algorithm to obtain 2-budget balanced cost sharing methods for the Steiner tree and traveling salesman problems. The *Steiner tree problem* asks for a minimum weight tree that spans a subset of prespecified terminal vertices; the tree may contain some non-terminal vertices. In the *traveling salesman problem*, the goal is to determine a minimum weight tour through all vertices such that every vertex is visited exactly once. Both problems admit a simple approximation algorithm that constructs a 2-approximate solution from a minimum spanning tree (see e.g. [98]). We can therefore use the cost sharing mechanism based on Prim's algorithm to obtain 2-budget balanced generalized incremental mechanisms for these problems.

Corollary 4.15. *Prim's algorithm yields a 2-budget balanced and weakly group-strategyproof generalized incremental mechanism for the Steiner tree problem.*

Proof. Run M^{PRIM} on the set of terminals. Let $\mathcal{T}$ be the MST computed for the final output set S^M and let p_i^{PRIM} denote the cost share of player $i \in S^M$. We output $\mathcal{T}$ and charge every player $i \in S^M$ a cost share of $p_i = p_i^{\text{PRIM}}$. The sum of the cost shares collected equals the weight $w(\mathcal{T})$ of the MST.

Let $\mathcal{T}^*$ be a minimum weight Steiner tree on S^M. Double every edge of $\mathcal{T}^*$ to obtain an Euler tour on the terminals S^M. By traversing this Euler tour and shortcutting all Steiner vertices, we obtain a spanning tree $\mathcal{T}'$ on S^M whose weight is at most $2w(\mathcal{T}^*)$; note that we can assume without loss of generality that the edge weights satisfy the triangle inequality. Since $\mathcal{T}$ is an optimal MST, we conclude

$$w(\mathcal{T}) \leq w(\mathcal{T}') \leq 2w(\mathcal{T}^*),$$

which proves 2-budget balance. □

Corollary 4.16. *Prim's algorithm yields a 2-budget balanced and weakly group-strategyproof generalized incremental mechanism for the traveling salesman problem.*

Proof. Run M^{PRIM} on the set of vertices. Let $\mathcal{T}$ be the MST computed for the final output set S^M and let p_i^{PRIM} be the cost share of player i. Double every edge of $\mathcal{T}$ to obtain an Euler tour on the vertices in S^M. By traversing this Euler tour and shortcutting vertices that have been visited before, we obtain a traveling salesman tour $\mathcal{T}'$ whose weight is at most $2w(\mathcal{T})$. We return $\mathcal{T}'$ and charge every player $i \in S^M$ a cost share of $p_i := 2p_i^{\text{PRIM}}$. Note that $\sum_{i \in S^M} p_i = 2w(\mathcal{T})$.

Let $\mathcal{T}^*$ be a minimum weight traveling salesman tour on S^M. By deleting an arbitrary edge, we obtain a tree spanning all vertices in S^M and thus $w(\mathcal{T}) \leq$

$w(\mathcal{T}^*)$. We have

$$w(\mathcal{T}') \leq 2w(\mathcal{T}) \leq 2w(\mathcal{T}^*),$$

which concludes the proof. □

Our generalized mechanisms for the above problems match the budget balance factors of previously known cost sharing mechanisms [54, 59, 68], but are inferior in terms of social cost approximation. While the known mechanisms achieve polylogarithmic social cost approximation factors (see [89]), the situation is much worse for our generalized incremental mechanisms: It is not hard to see that the minimum spanning tree game contains the public excludable good problem as a special case and thus inherits the social cost inapproximability of n (see Example 4.20 below). However, our generalized incremental mechanisms stand out due to their simplicity.

4.3.2 Scheduling Applications

Next, we state three examples from parallel machine scheduling.

Makespan Scheduling.

In the minimum makespan scheduling problem $P| |C_{\max}$, a set of jobs U is to be scheduled on a set of identical parallel machines to minimize the latest completion time of a job, also called the *makespan*. The problem is NP-complete (see [36]). Graham's *largest processing time* (LPT) algorithm [41] is a 4/3-approximation. LPT is a *list scheduling* algorithm: It first orders the jobs by non-increasing processing times and then adds jobs one by one (according to this order) to the current schedule. Every new job is assigned to the machine which currently has the least amount of processing time assigned to it.

We use Graham's LPT algorithm to obtain a generalized incremental mechanism which beats the corresponding lower bound of essentially 2 for Moulin mechanisms [10]. Let $M^{\text{LPT}} := M(\text{LPT}, \tau)$ be the generalized incremental mechanism induced by LPT and the order function τ which sorts the jobs in LPT's list scheduling order.

Corollary 4.17. *The generalized incremental mechanism M^{LPT} induced by Graham's* LPT *algorithm and τ is weakly group-strategyproof and 4/3-budget balanced for the makespan scheduling problem $P| |C_{\max}$.*

Proof. The order function τ is induced by a global order and thus consistent. Given a subset S of jobs, LPT adds the jobs in S one by one according to the order τ to the existing schedule. The incremental approximate cost of adding a new player in every step therefore equals the increase in the makespan of the resulting schedule. Hence, Graham's algorithm is τ-increasing. Theorem 4.12 now yields the claim. $\square$

The makespan scheduling problem contains the public excludable good problem as a special case and thus M^{LPT} suffers from the same inefficiency as outlined in Example 4.20. Bleischwitz et al. [12] give a 4/3-budget balanced and $O(\log n)$-approximate weakly group-strategyproof mechanism for the makespan problem, which outperforms our mechanism in terms of social cost approximation.

Weighted Completion Time Scheduling without Preemption.

The weighted completion time scheduling problem $P|\,|\sum w_iC_i$ asks to schedule a set U of n jobs with non-negative weights w_i on m parallel machines such that the total weighted completion time is minimized. Smith's list scheduling algorithm (SM) [96] orders the jobs by non-increasing weight per processing time ratios w_i/p_i and iteratively assigns each job to a machine with smallest total load. It is optimal on a single machine and $(1+\sqrt{2})/2 \approx 1.21$-approximate in the general case [58]. In the unweighted setting, i.e. when $w_i = 1$ for all $i \in U$, it reduces to the *shortest processing time* policy and also delivers an optimal schedule. Even in the unweighted case, no Moulin mechanism can achieve a budget balance factor better than $(n+1)/2$ [16]. Using generalized incremental mechanisms, we can heavily improve upon this. We define the generalized incremental mechanism $M^{\text{SM}} := M(\text{SM}, \tau)$ induced by Smith's rule and the order function τ which orders all jobs in the list scheduling order.

Corollary 4.18. *The generalized incremental mechanism M^{SM} induced by Smith's algorithm and τ is weakly group-strategyproof and budget balanced for $P|\,|\sum C_i$ and $1|\,|\sum w_iC_i$, and 1.21-budget balanced for $P|\,|\sum w_iC_i$.*

Proof. The order function τ is induced by the global order used by Smith's rule and is therefore consistent. Given a set of jobs S, SM adds the jobs in S one by one to the existing schedule, in the order of τ. Hence, the incremental approximate cost of adding job i is precisely its (weighted) completion time in the produced schedule. Thus, SM is τ-increasing. It follows from Theorem 4.12 that M^{SM} is weakly group-strategyproof and ρ^{SM}-budget balanced, where ρ^{SM} is the approximation guarantee of Smith's rule for the respective scheduling problem. $\square$

We will show in Section 4.5 that M^{SM} achieves constant social cost approximation guarantees for these problems.

Min-Sum Scheduling with Release Dates and Preemption.

In the completion time scheduling problem with release dates and preemption $P|r_i, pmtn|\sum_i C_i$, the goal is to schedule a set U of n jobs on m parallel machines such that the total completion time is minimized. Each job $i \in U$ becomes available at its release date r_i and jobs can be preempted. The *shortest remaining processing time* (SRPT) policy is a good approximation algorithm for this problem. At any point of time, SRPT executes the m available jobs with the smallest remaining processing times. SRPT computes an optimal schedule for the total completion time and flow time objectives in the single-machine case [93]. Sitters [95] very recently showed that SRPT achieves an approximation factor of 1.25 for the parallel machine case $P|r_i, pmtn|\sum_i C_i$.

Moulin mechanisms cannot achieve a budget balance factor better than $\Omega(n)$ for these problems [16]. We obtain the following strongly superior results. For a given subset of jobs $S \subseteq U$, let $\tau(\cdot, S)$ be the order induced by increasing completion times in the SRPT schedule for S; if two jobs are completed at the same time, we assume an arbitrary but consistent tie breaking rule. Let $M^{\text{SRPT}} := M(\text{SRPT}, \tau)$ be the generalized incremental mechanism induced by the SRPT algorithm and τ. We prove in Section 4.5 that τ is consistent and that SRPT is τ-increasing. Using Theorem 4.12, we thus obtain the following theorem.

Corollary 4.19. *The generalized incremental mechanism M^{SRPT} induced by* SRPT *and τ is weakly group-strategyproof and budget balanced for $1|r_i, pmtn|\sum_i F_i$ and $1|r_i, pmtn|\sum_i C_i$, and 1.25-budget balanced for $P|r_i, pmtn|\sum_i C_i$.*

We will prove in Section 4.5 that this mechanism also achieves constant social cost approximation guarantees for these problems.

4.4 Bounding Social Cost

In this section, we derive some general techniques that are helpful in proving social cost approximation guarantees of generalized incremental mechanisms. These techniques seem particularly applicable if the underlying approximate cost function is superadditive.

The following example shows that generalized incremental mechanisms cannot be expected to achieve attractive social cost approximation factors if the underlying cost function is subadditive. In the *public excludable good problem*, the serving cost is $C(S) = 1$ for every non-empty subset of players $\emptyset \neq S \subseteq U$, and $C(\emptyset) = 0$. Dobzinski et al. [30] showed that for this problem no constantly budget balanced and truthful mechanism can approximate social cost by less than a logarithmic factor, even if only strategyproofness is required. The situation is worse for generalized incremental mechanisms.

Example 4.20. *Consider an instance of the public excludable good problem with n players. Set the valuation of each player $i \in U$ to $v_i = 1-\epsilon$ for an arbitrarily small constant $\epsilon > 0$. Assuming truthful bidding, any generalized incremental mechanism serves the empty set incurring a social cost of $\Pi(\emptyset) = (1-\epsilon)n$. On the other hand, serving the whole set induces a social cost of $\Pi(U) = 1$. We therefore obtain a lower bound on the social cost approximation factor of essentially n.*

Our bounding technique relies on a weak monotonicity property for the induced *cost sharing method* of a generalized incremental mechanism. A generalized incremental mechanism $M(\textsc{alg}, \tau)$ naturally induces the cost sharing method $\xi(\textsc{alg}, \tau)$ for which $\xi_i(S)$ is defined as the cost share x_i that player i has to pay when S is the output set of the mechanism. More formally, we define the cost sharing method $\xi : U \times 2^U \to \mathbb{R}$ induced by ALG and τ as follows. Let $S = \{i_1, \dots, i_p\} \subseteq U$ be an arbitrary subset of players ordered by τ. The incremental approximate cost share $\xi_i(S)$ of player $i = i_k$, $1 \leq k \leq p$, is defined as $\xi_i(S) := \bar{C}(S_k) - \bar{C}(S_{k-1})$; for every player $i \notin S$, we define $\xi_i(S) := 0$. Note that Definition 4.11 is equivalent to stating that for every subset $S \subseteq U$ and every player $i \in S$, $\xi_i(S)$ is non-negative.

We are now ready to state our weak monotonicity property, which is reminiscent of the inverse of the core property.

Definition 4.21. *Let ξ be the incremental approximate cost sharing method induced by an approximation algorithm* ALG *and an order function τ. We call ξ* weakly monotone *if for all subsets $S \subseteq T \subseteq U$, $\sum_{i \in S} \xi_i(T) \geq \bar{C}(S)$.*

For proving a social cost approximation guarantee for a mechanism M, we need to upper bound the ratio between the social cost of the set S^M chosen by the mechanism and the cost of a socially optimal set

$$S^* := \arg\min_{S \subseteq U} \left(C(S) + \sum_{i \notin S} v_i \right).$$

We obtain the following theorem for generalized incremental mechanisms that implement weakly monotone cost sharing methods.

Theorem 4.22. *Let τ be a consistent order function and* ALG *be a τ-increasing algorithm. Suppose that the incremental approximate cost sharing method ξ induced by* ALG *and τ is weakly monotone. Then, the generalized incremental mechanism*

$M(\text{ALG}, \tau)$ approximates social cost by a factor of α if

$$\frac{\bar{C}(S^M \cup S^*)}{C(S^*) + C(S^M \setminus S^*)} \leq \alpha.$$

Proof. We can bound the social cost approximation factor by

$$\begin{aligned}\frac{\Pi(S^M)}{\Pi^*} &= \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i + \sum_{i \notin S^M \cup S^*} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} v_i + \sum_{i \notin S^M \cup S^*} v_i} \leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} v_i} \\ &\leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} \xi_i(S^M)} \leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + C(S^M \setminus S^*)}.\end{aligned}$$

Here, the first inequality follows from the fact that $\frac{a}{b} \leq \frac{a-c}{b-c}$ for arbitrary real numbers $a \geq b > c \geq 0$. The second inequality holds because $v_i \geq \xi_i(S^M)$ for every player $i \in S^M$, since i accepted and we assume truthful bidding. The last inequality follows from weak monotonicity of ξ and the fact that $\bar{C}(S) \geq C(S)$ for every set $S \subseteq U$.

We conclude the proof by showing that

$$\sum_{i \in S^* \setminus S^M} v_i \leq \bar{C}(S^M \cup S^*) - \bar{C}(S^M).$$

Without loss of generality, number the players in $S^* \setminus S^M$ in the order in which they were rejected by M, i.e., $S^* \setminus S^M =: \{1, \ldots, \ell\}$. Fix a player $i \in S^* \setminus S^M$ and consider the iteration in which player i was removed. Let R and A be the sets of remaining and accepted players at the beginning of this iteration, respectively. Define R^i as the subset of players in $S^* \cup S^M$ that were still remaining in the game when i was picked, i.e., $R^i := S^M \cup \{i, i+1, \ldots, \ell\}$. Let $k := |A|$. By Lemma 4.9, we have $R_k = A$. Moreover, since i is chosen, we have $R_{k+1} = A \cup \{i\}$. Note that $A \cup \{i\}$ is a subset of R^i. By consistency of τ, the first $k+1$ elements of R^i and R must coincide and we thus have $A \cup \{i\} = R_{k+1} = R^i_{k+1}$. The same argument also yields that $A = R_k = R^i_k$; see Figure 4.3 for an illustration. Therefore,

$$p_i = \bar{C}(A \cup \{i\}) - \bar{C}(A) = \bar{C}(R^i_{k+1}) - \bar{C}(R^i_k) = \xi_i(R^i).$$

Since i rejected, we have $v_i < p_i = \xi_i(R^i)$. Note that $R^i = R^{i+1} \cup \{i\}$. Exploiting that ξ is weakly monotone, we obtain that

$$\bar{C}(R^i) = \sum_{j \in R^i} \xi_j(R^i) = \xi_i(R^i) + \sum_{j \in R^{i+1}} \xi_j(R^i) \geq \xi_i(R^i) + \bar{C}(R^{i+1}).$$

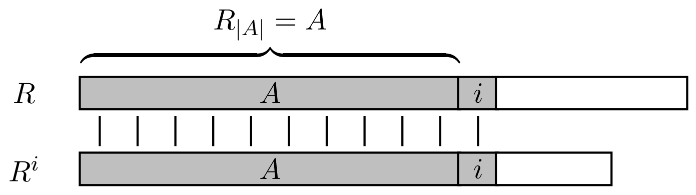

Figure 4.3: Illustration of the consistency property as used in the proof of Theorem 4.22.

Summing over all $i \in \{1, \dots, \ell\}$ yields

$$\sum_{i \in S^* \setminus S^M} v_i < \sum_{i=1}^{\ell} \xi_i(R^i) \leq \sum_{i=1}^{\ell} \big(\bar{C}(R^i) - \bar{C}(R^{i+1})\big) = \bar{C}(S^M \cup S^*) - \bar{C}(S^M).$$

□

We use Theorem 4.22 in Section 4.5 to prove constant social cost approximation factors for our generalized incremental mechanisms for scheduling problems with completion time objectives.

4.5 Completion Time Scheduling

In this section, we study the performance of generalized incremental mechanisms for parallel machine scheduling problems with total completion time objectives, also taking into account their social cost approximation guarantees. We distinguish between the model with weights in which all jobs arrive at time zero and no preemption is allowed, and the model in which jobs have release dates and may be preempted.

4.5.1 Weighted Completion Time

We reconsider the (weighted) completion time scheduling problems introduced in Section 4.3.2. We already showed that the generalized incremental mechanism $M^{\text{SM}} := M(\text{SM}, \tau)$ induced by Smith's rule and the offer function τ defined by non-increasing weight per processing time is weakly group-strategyproof and achieves ρ^{SM}-budget balance, where ρ^{SM} is the approximation guarantee of Smith's rule. In this section, we show that M^{SM} also achieves a surprisingly small social cost approximation factor.

Theorem 4.23. *The generalized incremental mechanism M^{SM} induced by Smith's algorithm and τ is weakly group-strategyproof, ρ^{SM}-budget balanced, and $2\rho^{\text{SM}}$-approximate for the respective (weighted) completion time scheduling problem.*

We first prove the following lemma.

Lemma 4.24. *Let* ALG *be an algorithm for $P||\sum w_iC_i$ with cost function $\bar{C}$. Let X and Y be two disjoint sets of jobs. Then, the cost of an optimal schedule for $X \cup Y$ can be bounded by $C(X \cup Y) \leq 2(\bar{C}(X) + \bar{C}(Y))$.*

Proof. We prove the inequality individually for each machine $\hat{M}$. Consider the jobs $\hat{X} \subseteq X$ and $\hat{Y} \subseteq Y$ scheduled on $\hat{M}$ in the runs of ALG on X and Y, respectively. We denote by c_i the completion time of job i in its respective schedule, i.e. $c_i := \bar{C}_i(X)$ for all $i \in \hat{X}$ and $c_i := \bar{C}_i(Y)$ for all $i \in \hat{Y}$.

Consider the schedule which processes all jobs in $\hat{X} \cup \hat{Y}$ on $\hat{M}$ according to non-decreasing c_i. The completion time of a job $i \in \hat{X}$ in this schedule is $c_i + c_{i^*}$, where i^* denotes the last job in $\hat{Y}$ that is processed before i. Since i^* is processed before i, we have $c_i + c_{i^*} \leq 2c_i$. By exchanging the roles of X and Y, we can show the same for the completion time of every job $i \in \hat{Y}$.

Since the cost of an optimal schedule for $X \cup Y$ is at most that of the schedule produced by repeating the above procedure for each machine, we have

$$C(X \cup Y) \leq \sum_{i \in X \cup Y} w_i \cdot 2c_i = 2\Big(\sum_{i \in X} w_ic_i + \sum_{i \in Y} w_ic_i\Big) = 2\big(\bar{C}(X) + \bar{C}(Y)\big).$$

□

We can now prove Theorem 4.23.

Proof of Theorem 4.23. It follows from Corollary 4.18 that M^{SM} is weakly group-strategyproof and ρ^{SM}-budget balanced. In order to obtain the social cost approximation guarantee, we show that the induced cost sharing method ξ is weakly monotone. Consider an arbitrary subset $S \subseteq U$ of jobs. Note that the incremental approximate cost share $\xi_i(S)$ of a player $i \in S$ with respect to S equals her completion time in the schedule output by Smith's rule for S. It is not hard to see that $C_i(T) \geq C_i(S)$ for every $i \in S \subseteq T$. Hence, $\sum_{i \in S} \xi_i(T) \geq \sum_{i \in S} \xi_i(S) = \bar{C}(S)$. The social cost approximation factor now follows from Lemma 4.24 and Theorem 4.22. □

The following example shows that our social cost analysis is tight, even in the unweighted single machine case.

Example 4.25. *Consider an instance of $1|p_i = 1|\sum C_i$ on an even number of n jobs with valuations $v_i = i$ for all $i \in [n]$. Assume that M^{SM} orders the jobs* *according to increasing valuations (note that we can easily enforce this by slightly perturbing the processing times) and thus accepts all jobs. Consequently,* $\Pi(S^M) = \bar{C}([n]) = n(n+1)/2$. *However, if we exclude the first* $n/2$ *jobs from the scheduled set, we obtain a social cost of*

$$C\Big(\Big[\frac{n}{2}\Big]\Big) + \sum_{i=1}^{n/2} v_i = 2 \cdot \Big(\frac{n}{4}\Big(\frac{n}{2}+1\Big)\Big) = n(n+2)/4 \geq \Pi^*,$$

yielding a social cost approximation ratio that approaches 2.

4.5.2 Completion Time with Release Dates and Preemption

We reconsider the preemptive completion (and flow) time scheduling problems introduced in Section 4.3.2. We prove the following result:

Theorem 4.26. *The generalized incremental mechanism* M^{SRPT} *induced by the* SRPT *algorithm and* τ *is weakly group-strategyproof,* ρ^{SRPT}*-budget balanced, and* $4\rho^{\text{SRPT}}$*-approximate for the respective completion time scheduling problem with release dates and preemption.*

For the sake of clarity, we first consider the single machine case and comment on the extension of the results given below to the parallel machine case at the end of this section.

Single Machine Case. Consider the problem $1|r_i, pmtn|\sum_i C_i$ of scheduling a set of jobs U on a single machine to minimize the total completion time. The *shortest remaining processing time* (SRPT) policy solves this problem to optimality [93]. Throughout this section, we denote by $C_i(S)$ the completion time of job $i \in S$ in the SRPT schedule for $S \subseteq U$. Note that by optimality of SRPT, we have $\bar{C}(S) = C(S) = \sum_{i \in S} C_i(S)$. As in Section 4.3.2, we define $\tau(\cdot, S)$ to be the order induced by increasing completion times in the SRPT schedule, i.e. $\tau(i,S) := C_i(S)$ for all $i \in S$, and let $M^{\text{SRPT}} := M(\text{SRPT}, \tau)$ be the generalized incremental mechanism induced by SRPT and τ.

The proof of Theorem 4.26 relies on Lemmas 4.27 and 4.28 below. The most work goes into showing that the order function τ is consistent and that SRPT is τ-increasing. However, we defer this part of the proof to the end of this section. Lemma 4.28 is used to prove the social cost approximation factor.

Lemma 4.27. *The order function* τ *is consistent. Moreover,* SRPT *is* τ*-increasing.*

Lemma 4.28. *Let* ALG *be an algorithm for $P|r_i, pmtn|\sum_i C_i$ with cost function $\bar{C}$. Let X and Y be two disjoint sets of jobs. Then, the cost of an optimal schedule for $X \cup Y$ can be bounded by $C(X \cup Y) \le 4(\bar{C}(X) + \bar{C}(Y))$.*

Proof. Phillips et al. [82] prove that any preemptive schedule for $P|r_i, pmtn|\sum_i C_i$ can be turned into a non-preemptive schedule NP with at most twice the cost. With Lemma 4.24, we obtain $C(X \cup Y) \le 2(C^{\text{NP}}(X \cup Y)) \le 4(\bar{C}(X) + \bar{C}(Y))$. □

Assuming that Lemma 4.27 holds, we can now prove Theorem 4.26.

Proof of Theorem 4.26. Lemma 4.27 together with Theorem 4.3 imply that M^{SRPT} is weakly group-strategyproof and budget balanced. To prove that M^{SRPT} approximates social cost, we first show that ξ is weakly monotone. Fix some set T and let $S \subseteq T$. Consider the SRPT schedule for T. By removing all jobs in $T \setminus S$ from this schedule, we obtain a feasible schedule for S of cost at most $\sum_{i \in S} C_i(T)$, hence $\sum_{i \in S} C_i(T) \ge C(S)$. Subsequently, it will become clear that the incremental cost share $\xi_i(T)$ of a job $i \in T$ with respect to T is equal to its completion time $C_i(T)$. We conclude that ξ is weakly monotone. Now, the bound on the social cost approximation factor follows from Lemma 4.28 (letting ALG refer to an optimal algorithm) and Theorem 4.22. □

It remains to show that the order function τ induced by increasing completion times in the SRPT schedule is consistent and that SRPT is τ-increasing. To this end, we study the effect of removing a single job from the SRPT schedule. We claim the following:

Lemma 4.29. *Let $T \subseteq U$. Suppose we remove an arbitrary job j from T. Define $S := T \setminus \{j\}$ as the set of remaining jobs. Let $C_i(S)$ and $C_i(T)$ denote the completion times of job $i \in S$ in the* SRPT *schedules for S and T, respectively. Then*

1. *$C_i(S) = C_i(T)$ for every job $i \in S$ with $C_i(T) < C_j(T)$; and*
2. *$C_i(S) \ge C_j(T)$ for every job $i \in S$ with $C_i(T) > C_j(T)$.*

Suppose this lemma is true. We can then prove that τ is consistent and that SRPT is τ-increasing:

Proof of Lemma 4.27. We first prove consistency. Let $S \subseteq T \subseteq U$ be two subsets ordered by τ as $S = \{i_1, i_2, \dots, i_p\}$ and $T = \{j_1, j_2, \dots, j_q\}$. Let k be minimal with $j_k \in T \setminus S$. Define $j := j_k$ to simplify notation. By definition of τ, for every job $i = j_l$ with $1 \le l < k$, we have $C_i(T) < C_j(T)$. Also, for every job $i = j_r$ with $k < r \le q$, we have $C_i(T) > C_j(T)$. Thus, by removing job j from T we obtain a new set $T' = T \setminus \{j\}$ such that $C_i(T') = C_i(T)$ for all $i = j_l$ with $1 \leq l < k$ and $C_i(T') \geq C_j(T)$ for all $i = j_r$ with $k < r \leq q$. Repeating the above procedure (with T' instead of T), we eventually remove all jobs in $T \setminus S$ from T and conclude that $i_l = j_l$ for all $1 \leq l < k$.

It remains to prove that SRPT is τ-increasing. Consider an arbitrary subset $S \subseteq U$ of jobs and suppose S is ordered by τ as $S = \{i_1, \dots, i_p\}$. We need to argue that $\bar{C}(S_k) \geq \bar{C}(S_{k-1})$ for every $1 \leq k \leq p$. The proof is by induction on k. For $k = p$ the claim follows since we remove a job $j = i_p$ with $C_j(S) > C_i(S)$ for all $i \in S \setminus \{j\}$ and by Lemma 4.29, the completion times of all remaining jobs remain the same. Thus $\bar{C}(S_p) - \bar{C}(S_{p-1}) = C_j(S_p) = C_j(S) \geq 0$. Suppose the claim holds true for all $k + 1 \geq \ell$ for some $1 < \ell \leq p$. We show that it remains true for k. Let $j = i_k$. We have $C_j(S) > C_i(S)$ for all $i \in S_{k-1}$. The consistency of τ implies that $C_j(S_k) > C_i(S_k)$ for all $i \in S_{k-1}$. Thus, by Lemma 4.29, the completion times of all jobs $i \in S_{k-1}$ remain the same if we remove job j from the SRPT schedule for S_k. We conclude that the incremental cost share of player j is exactly its completion time, i.e., $\bar{C}(S_k) - \bar{C}(S_{k-1}) = C_j(S_k) \geq 0$. ☐

Intuitively, it is relatively easy to verify Lemma 4.29: During the lifetime (i.e. between release and completion time) of job j in the SRPT schedule for T, job j prevents some jobs, call them *losing jobs*, to be executed (because they have a larger remaining processing time) while some other jobs, call them *winning jobs*, prevent j from being executed (because they have a smaller remaining processing time). Clearly, every losing job has a larger completion time than j, while every winning job has a smaller completion time than j. Now suppose we remove job j from the input set and consider the resulting SRPT schedule. There are two crucial insights: (i) nothing changes for the winning jobs, and (ii) whenever j was processed in the SRPT schedule for T, a losing job might now be processed in the SRPT schedule for S; however, this losing job will not be completed before time $C_j(T)$. See Figure 4.4 for an illustration.

In order to turn this intuition into a formal proof, we first introduce some more notation. Let $e_i(t)$ be the amount of time that has been spent on processing job i up to time t. The *remaining processing* time $x_i(t)$ of job i at time t is $x_i(t) := p_i - e_i(t)$. We call a job i *active* at time t if it has been released but not yet completed at this time, i.e., $r_i \leq t < C_i$. Let $A(t)$ be the set of jobs that are active at time t. SRPT works as follows: At any time $t \geq 0$, SRPT schedules an active job $i \in A(t)$ with minimum remaining processing time, i.e., $x_i(t) \leq x_k(t)$ for all $k \in A(t)$. We assume that SRPT uses a consistent tie breaking rule, e.g., if $x_i(t) = x_k(t)$ for two different jobs i and k, then schedule the one with smaller index.

Consider the SRPT schedule for a set $T \subseteq U$. Let $i, j \in A(t)$ be two jobs that are active at time t. We define $i \prec_t j$ iff either $x_i(t) < x_j(t)$ or $x_i(t) = x_j(t)$ and $i \leq j$. Note that at any point of time t, SRPT schedules the job $i \in A(t)$ with

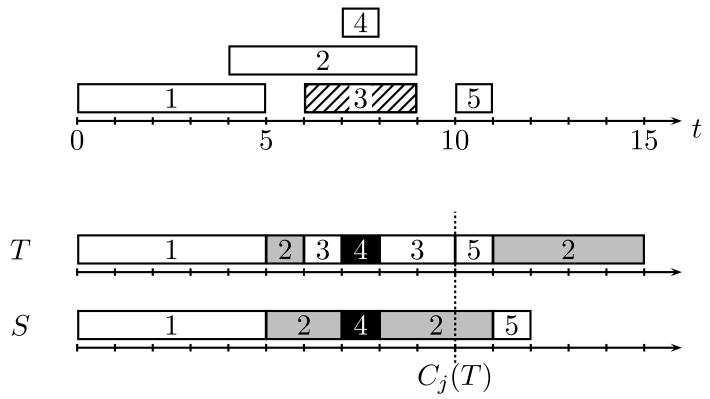

Figure 4.4: The effect of removing a single job $j = 3$ from the SRPT schedule on $T = \{1, \dots, 5\}$. The upper part represents the input instance for T; jobs are numbered by increasing release times. The lower part shows the two SRPT schedules for T and $S := T \setminus \{j\}$. The winning and losing jobs are indicated in black and gray, respectively.

$i \prec_t j$ for all $j \in A(t)$. Thus, if $i \prec_t j$ for some t, then $i \prec_{t'} j$ for all $t' \in [t, C_i)$. We therefore simply write $i \prec j$ iff there exists a time t with $i \prec_t j$. Let $\sigma(t)$ denote the job that is executed at time t in the SRPT schedule for T; we define $\sigma(t) = \emptyset$ if $A(t) = \emptyset$.

Let $j \in T$ be an arbitrary job and consider the time interval $[r_j, C_j)$. We define the set $\mathcal{C}_j$ of jobs that are *competing* with j as $\mathcal{C}_j := \{i \in T \setminus \{j\} : [r_i, C_i) \cap [r_j, C_j) \neq \emptyset\}$. Note that $j \notin \mathcal{C}_j$. We partition the jobs in $\mathcal{C}_j$ into a set $\mathcal{W}_j$ of *winning jobs* and a set $\mathcal{L}_j$ of *losing jobs* with respect to j: $\mathcal{W}_j := \{i \in \mathcal{C}_j : i \prec j\}$ and $\mathcal{L}_j := \mathcal{C}_j \setminus \mathcal{W}_j$. Intuitively, suppose i and j are both active at some time t. If i is a winning job, then i prevents j from being executed by SRPT. On the other hand, if i is a losing job, then j prevents i from being executed.

We next investigate the effect of removing a job j from T. We use the superscript S if we refer to the SRPT schedule for $S := T \setminus \{j\}$.

Lemma 4.30. *Consider the two* SRPT *schedules on job sets T and $S := T \setminus \{j\}$. For every job $i \in \mathcal{C}_j$ that is active at time $t \in [r_j, C_j)$,*

$$x_i^S(t) = x_i(t) \text{ if } i \in \mathcal{W}_j \quad \text{and} \quad x_i^S(t) \geq x_j(t) \text{ if } i \in \mathcal{L}_j.$$

Proof. We partition the time interval $[r_j, C_j)$ into a sequence of maximal subintervals $I_1, I_2, \dots, I_f$ such that the set of active jobs remains the same within every subinterval $I_\ell := [s_\ell, e_\ell)$. We prove by induction over ℓ that the claim holds for every $t \in [r_j, e_\ell)$.

Note that both schedules are identical up to time $r_j = s_1$. If $\sigma(s_1) \neq j$, then both schedules process the same job during I_1 and the claim follows. Suppose

$\sigma(s_1) = j$. This implies that $A(s_1) \cap \mathcal{W}_j = \emptyset$ and thus all jobs in $A(s_1) \setminus \{j\} = A^S(s_1)$ are losing jobs. If $A^S(s_1) = \emptyset$, the claim follows. Otherwise, let $k := \sigma^S(s_1)$ be the job that is processed in the schedule for S. Since k is a losing job, we have $x_k^S(s_1) = x_k(s_1) \geq x_j(s_1)$. Since k and j receive the same processing time during I_1 in their respective schedules, the claim holds for all $t \in [r_j, e_1)$.

Now, assume that the claim is true for every $t \in [r_j, e_{\ell-1})$ for some $\ell > 1$. We show that it remains true during the time interval I_ℓ. By the induction hypothesis, $x_i^S(t) = x_i(t)$ for every job $i \in \mathcal{W}_j$ that is active at time $t \in [r_j, e_{\ell-1})$. This implies that a job $j \in \mathcal{W}_i$ is executed at time $t \in [r_j, e_{\ell-1})$ in the schedule for T iff it is executed at time t in the schedule for S. We thus have $A^S(s_\ell) \cap \mathcal{W}_j = A(s_\ell) \cap \mathcal{W}_j$. Moreover, $x_i^S(t) \geq x_j(t)$ for every job $i \in \mathcal{L}_j$ that is active at time $t \in [r_j, e_{\ell-1})$. Since $x_j(t) > 0$ for every $t \in [r_j, C_j)$, every job $i \in \mathcal{L}_j$ that is active at time $t \in [r_j, e_{\ell-1})$ in the schedule for T must also be active at time t in the schedule for S. Thus, $A^S(s_\ell) \cap \mathcal{L}_j = A(s_\ell) \cap \mathcal{L}_j$. We now distinguish two cases:

(i) First, assume $\sigma(s_\ell) =: k \in \mathcal{W}_j$. Job k then has smallest remaining processing time, i.e., $x_k(s_\ell) \leq x_i(s_\ell)$ for all $i \in A(s_\ell)$. We conclude that

$$x_k^S(s_\ell) = x_k(s_\ell) \leq x_i(s_\ell) = x_i^S(s_\ell) \quad \forall i \in A(s_\ell) \cap \mathcal{W}_j = A^S(s_\ell) \cap \mathcal{W}_j$$
$$x_k^S(s_\ell) = x_k(s_\ell) \leq x_j(s_\ell) \leq x_i^S(s_\ell) \quad \forall i \in A(s_\ell) \cap \mathcal{L}_j = A^S(s_\ell) \cap \mathcal{L}_j.$$

Since we assume that SRPT uses a consistent tie breaking rule, this implies that $\sigma^S(s_\ell) = k$ and the claim follows.

(ii) Now, suppose $\sigma(s_\ell) = j$. (Note that $\sigma(s_\ell) \in \mathcal{L}_j$ is impossible.) Then $x_j(s_\ell) \leq x_i(s_\ell)$ for every $i \in A(s_\ell)$ and $A(s_\ell) \cap \mathcal{W}_j = \emptyset$. But then we also have $A^S(s_\ell) \cap \mathcal{W}_j = \emptyset$ and thus $A^S(s_\ell) \subseteq \mathcal{L}_j$. If $A^S(s_\ell) = \emptyset$, the claim follows. Otherwise, let $k := \sigma^S(s_\ell) \in \mathcal{L}_j$ be the job that is executed at time s_ℓ in the schedule for S. Since $x_k^S(s_\ell) \geq x_j(s_\ell)$ and the remaining processing times of k and j in their respective schedules reduce by the same amount during I_ℓ, the claim follows. □

Using Lemma 4.30, we can now easily prove Lemma 4.29.

Proof of Lemma 4.29. Let $i \in S$ be a job with $C_i(T) < C_j(T)$. If i is not competing with j, then $r_j \geq C_i$ and thus removing j from the schedule does not change the completion time of i, i.e., $C_i(S) = C_i(T)$. Otherwise, i is competing with j, but since $C_j(T) > C_i(T)$, i is a winning job with respect to j. By Lemma 4.30, job i is completed at the same time in the SRPT schedules for S and for T and thus $C_i(S) = C_i(T)$.

Next, consider a job $i \in S$ with $C_i(T) > C_j(T)$. The claim clearly holds if $r_i \geq C_j(T)$ since $C_i(S) \geq r_i$. Assume $r_i < C_j(T)$. Then i is competing with j and i is a losing job with respect to j. By Lemma 4.30, job i cannot be completed before time $C_j(T)$ in the SRPT schedule for S. Thus $C_i(S) \geq C_j(T)$. □

Parallel Machine Case. The crucial insight in the single machine case is Lemma 4.29. The same property holds in the parallel machine case if we assume a consistent tie breaking rule between jobs with equal remaining processing times. Showing that the computed output set is $4\rho^{\text{SRPT}}$-approximate proceeds exactly along the same lines as in Theorem 4.26 (in fact, Lemma 4.28 is formulated for the multiple machine case). The only difference is that SRPT produces a schedule whose total completion time is at most 1.25 times the optimum [95].

Flow Time Scheduling.

The following example shows that a variant of the above mechanism for flow time objectives does not achieve similar results. Even for the single machine problem $1|r_i, pmtn|\sum F_i$, no cost sharing mechanism in which every player pays her own flow time, i.e. $\xi_i(S) := F_i(S) = C_i(S) - r_i$, can approximate social cost by a factor smaller than $n/4$.

Example 4.31. *Define two sets A and B, each containing $k = \frac{n}{2}$ jobs of unit processing time. At each integral point of time, one job from each set is released. All jobs in A have valuation n, whereas all jobs in B have valuation 1.*

Now, at each integral point of time, the SRPT algorithm may choose to schedule the job from B that has just been released. (Note that at any integral point of time, any job in the system has a remaining processing time of 1.) If this happens, no job in A is executed before time $t = \frac{n}{2}$, thus the jobs in A experience an average delay of $\frac{n}{2}$ each. However, due to their high valuation, all jobs in A still accept, and we obtain a total flow time of

$$\bar{C}(S^M) = \bar{C}(A \cup B) = k + k + k^2 = k^2 + 2k.$$

However, the set A only has social cost $\Pi(A) = \bar{C}(A) + v(B) = k + k = 2k$, hence the approximation factor for this example is more than $\frac{n}{4}$:

$$\frac{\Pi(S^M)}{\Pi^*} \geq \frac{\Pi(A \cup B)}{\Pi(A)} = \frac{k^2 + 2k}{2k} = \frac{k+2}{2} = \frac{n+4}{4} > \frac{n}{4}.$$

4.6 Connections to Other Frameworks

4.6.1 Acyclic Mechanisms

Our generalized incremental mechanisms are a subclass of *acyclic mechanisms* introduced by Mehta, Roughgarden, and Sundararajan [69] (see Section 2.4.4). As we explain in the following, they can be viewed as being complementary to Moulin mechanisms in the scope of acyclic mechanisms.

Our research presented in this chapter was initiated by the following simple observations. Consider the offer function τ of an acyclic mechanism. For a given set of players $S \subseteq U$, τ divides S into subsets of players with equal offer times $\tau(\cdot, S)$. We like to think about acyclic mechanisms in terms of such maximal player sets with equal offer times, and call them *clusters*. Depending on the size of these clusters, we can illustrate the landscape of acyclic mechanisms as follows:

- Towards one end, assume that every set S consists of one big cluster that contains all players in S. Then, the Definition 2.36 reduces to (P2), which is equivalent to the definition of cross-monotonicity. Hence, acyclic mechanisms with maximum cluster size are Moulin mechanisms.

- Towards the other end, consider an acyclic mechanism for which all clusters are singletons, i.e. in every set S, every player has a unique offer time. In this case, Definition 2.36 reduces to (P1) and once a cost share is announced to a player, it can never be changed again.

Following these observations, we defined *order functions* to be offer functions that produce only singleton clusters, i.e. offer functions $\tau(i, S)$ in which each $i \in S$ receives a distinct offer time with respect to S. We call the subclass of acyclic mechanisms that are induced by order functions *singleton mechanisms* (see also [18]). In this chapter, we study the subclass of singleton mechanisms in which every player is charged the *incremental* cost of adding her to the current solution. It can easily be verified that consistent order functions are valid for the induced incremental cost sharing methods defined in this chapter. Intuitively, the reason is that the cost share of a player only depends on the set of players that precede her in the order of τ. As a consequence, generalized incremental mechanisms fulfill all conditions of Theorem 2.37 and thus belong to the class of acyclic mechanisms. Bleischwitz et al. [12] showed that acyclic mechanisms are *weakly group-strategyproof against collectors*. As a consequence, our generalized incremental mechanisms also satisfy this slightly stronger truthfulness notion.

4.6.2 Scheduling with Rejection

It is easy to verify that every cost sharing mechanism that approximates social cost by a factor of α defines an α-approximate algorithm for the underlying optimization problem with rejection. Along with our results in mechanism design, we therefore obtain several approximation algorithms for scheduling problems with rejection.

Let $\mathcal{P}$ be an arbitrary scheduling problem. For every job $i \in U$, let z_i be the rejection penalty for the price-collecting variant of $\mathcal{P}$. We define a cost sharing game on $\mathcal{P}$ by identifying every player's valuation with the penalty of her job, i.e. $v_i := z_i$ for all $i \in U$. An α-approximate mechanism for this cost sharing game outputs a served set of players S^M and a feasible solution of cost $\bar{C}(S^M)$ for this set with social cost

$$\bar{C}(S^M) + \sum_{i \notin S^M} v_i \leq \alpha \cdot \min_{S \subseteq U} \left(C(S) + \sum_{i \notin S} v_i \right) .$$

Now, it is easy to see that the algorithm that schedules S^M and rejects all other jobs outputs an α-approximate solution to the scheduling problem with rejection.

We therefore obtain the following theorem:

Theorem 4.32. *Let M be a mechanism that approximates social cost by a factor α for a scheduling problem $\mathcal{P}$. Then, M defines an α-approximation algorithm for the respective scheduling problem $\mathcal{P}$ with rejection.*

Thus, the following results are immediate consequences of our mechanisms presented in Section 4.5.

Corollary 4.33. *The generalized incremental mechanism M^{SM} induced by Smith's rule defines a 2.42-approximate algorithm for the weighted completion time scheduling problem $P| \,| \sum w_i C_i$ with rejection. The algorithm is 2-approximate in both the single machine case and the unweighted case.*

Corollary 4.34. *The generalized incremental mechanism M^{SRPT} based on the* SRPT *policy defines a 5-approximate algorithm for the completion time scheduling problem $P|r_i, pmtn| \sum C_i$ with rejection. The algorithm is 4-approximate in the single machine case.*

4.7 Makespan Scheduling with Unit Processing Times

In this section, we consider the minimum makespan scheduling problem m parallel machines with unit processing times $P|p_i = 1|C_{\max}$. Any list scheduling algorithm is optimal for this problem. We obtain a weakly group-strategyproof cost sharing mechanism that is budget balanced and $(H_n + 1)$-approximate. The social cost guarantee of our mechanism matches the the lower bound proved in [68], which applies since we are dealing with a public excludable good problem in the special case where $m = n$.

Assuming a fixed order on U, let $S = \{1, 2, \ldots, |S|\}$ denote the elements in $S \subseteq U$ according to this order. We partition every set S into $\lceil \frac{n}{m} \rceil$ clusters of size (at most) m by defining the offer function as

$$\tau(i,S) := \lfloor \tfrac{i}{m} \rfloor .$$

Further, we let every player pay an equal share of the incremental cost incurred by her cluster, i.e.

$$\xi_i(S) := \frac{1}{|E(i,S)|}.$$

Let M^{um} denote the acyclic mechanism induced by τ and ξ.

Theorem 4.35. *The acyclic cost sharing mechanism M^{um} is budget balanced and approximates social cost by a factor of $(H_n + 1)$.*

Proof. We first prove that τ is valid for ξ. Property (P1) in Definition 2.36 is fulfilled since the index of a job does not change when jobs with higher indices leave the set. Just as well, $|E(i,S)|$ can only decrease when a subset of $G(i,S) \cup (E(i,S) \setminus \{i\})$ is removed, which ensures Property (P2).

Assume that the list schedule for a set S schedules the jobs in our given order $\{1, 2, \ldots, |S|\}$. By definition of τ, each cluster thus causes an objective function increase of 1. Since $\sum_{i \in E(i,S)} \xi_i(S) = 1$, the sum of the cost shares equals exactly the cost of the produced schedule. Hence, 1-budget balance follows from optimality of the list schedule.

It remains to show that M^{um} approximates social cost by a factor of $H_n + 1$. Let $\ell := |S^M|$ and let S^* be a player set with optimal social welfare. First, consider the case where $|S^*| \geq 1$. We have $\sum_{i \in S^* \setminus S^M} v_i \leq H_{n-\ell}$, because the cost share of the job that is asked in the very last iteration is bounded by 1, that in the second to last iteration is bounded by $1/2$, and so on. Hence, the valuations of the rejected players are upper bounded by these values. With this,

$$\begin{aligned}\frac{\Pi(S^M)}{\Pi^*} &= \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i + \sum_{i \notin S^M \cup S^*} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} v_i + \sum_{i \notin S^M \cup S^*} v_i} \leq \frac{\bar{C}(S^M) + \sum_{i \in S^* \setminus S^M} v_i}{C(S^*) + \sum_{i \in S^M \setminus S^*} v_i} \\ &\leq \frac{\lceil \frac{\ell}{m} \rceil + H_{n-\ell}}{1} \leq \tfrac{\ell}{m} + 1 + H_{n-\ell} \leq H_n + 1.\end{aligned}$$

Here, as in the proof of Theorem 4.22, the first inequality follows from the fact that $\frac{a}{b} \leq \frac{a-c}{b-c}$ for arbitrary real numbers $a \geq b > c \geq 0$.

Otherwise, if $S^* = \emptyset$, we have

$$\frac{\Pi(S^M)}{\Pi^*} = \frac{\bar{C}(S^M) + \sum_{i \notin S^M} v_i}{\sum_{i \in S^M} v_i + \sum_{i \notin S^M} v_i} \leq \frac{\bar{C}(S^M)}{\sum_{i \in S^M} v_i} \leq \frac{\lceil \frac{\ell}{m} \rceil}{\frac{\ell}{m}} \leq 2,$$

assuming that $\ell \geq 1$ (otherwise $S^M = S^*$). This concludes the proof. $\square$

The best-possible budget balance factor achievable by cross-monotonic cost sharing methods is $2 - 1/m$ for this problem. Hence, the above result proves once more that acyclic mechanisms allow for better budget balance factors in scheduling than Moulin mechanisms.

4.8 Conclusion

We presented a general approach to derive weakly group-strategyproof mechanisms from approximation algorithms, therefore allowing to benefit from the enormous theory on approximation algorithms. The approach is applicable whenever the approximation algorithm exhibits a consistent order function with respect to which the approximate cost is monotonically increasing. We provided a series of examples showing that many approximation algorithms naturally give rise to such order functions. It turned out that our mechanisms are particularly efficient for completion time scheduling problems. We are confident that our approach can be applied to various other combinatorial optimization problems. It would be interesting to see more examples for which good social cost approximation guarantees can be proven. The most promising problems in this context seem to be ones with superadditive cost functions, i.e. where congestion effects occur. Concurrently, these are problems for which Moulin mechanisms usually perform only poorly.

Our generalized incremental mechanisms belong to the class of *singleton mechanisms* that constitutes a subclass of acyclic mechanisms which can be seen as being complementary to Moulin mechanisms. While different cost share definitions are conceivable, we concentrated on singleton mechanisms with incremental approximate cost shares. We showed that these types of mechanisms are sufficient to exploit the full strength of existing approximation algorithms for completion time scheduling problems and allow to derive mechanisms with attractive budget balance and social cost approximation guarantees. It would be interesting to understand the limitations of singleton mechanisms in general. Moreover, stepping back to the full generality of acyclic mechanisms, some of the most intriguing open problems are to find a general way to construct acyclic mechanisms from approximation algorithms and to find a general property for proving approximate social cost, alike the summability property for Moulin mechanisms.

5

ONLINE COST SHARING

The problem of sharing the cost of a common infrastructure among a set of strategic and cooperating players has been the subject of intensive research in recent years. However, most of these studies consider cooperative cost sharing games in an *offline* setting, i.e. the mechanism knows all players and their respective input data in advance. In this chapter, we study cooperative cost sharing games in an *online* setting: Upon the arrival of a new player, the mechanism has to take instantaneous and irreversible decisions without any knowledge about players that arrive in the future. We propose an online model for general demand cost sharing games and give a complete characterization of both weakly group-strategyproof and group-strategyproof online cost sharing mechanisms in this model. Moreover, we present a simple method to derive incremental online cost sharing mechanisms from online algorithms such that the competitive ratio is preserved. Based on our general results, we develop online cost sharing mechanisms for several binary demand and general demand cost sharing games.

Publication Note. The results contained in this chapter have been presented at the *7th International Conference on Algorithms and Complexity (CIAC 10)* [19].

5.1 Introduction

The pivotal point in mechanism design is to achieve a global objective even though part of the input information is owned by selfish players. In cost sharing, the aim is to share the cost of a common service in a fair manner while the players' valuations for the service are private information. Based on the declared bids of the players, a cost sharing mechanism determines a service allocation and distributes the incurred cost among the served players. In many cost sharing games, the common service is represented by a combinatorial optimization problem like minimum Steiner tree, machine scheduling, etc., which defines a cost for every possible service allocation. We consider *cooperative* cost sharing games, i.e. players may form coalitions to coordinate their bidding strategies.

During the last decade, there has been substantial research on *binary demand* cost sharing games, where a service allocation defines simply whether or not a player is served. In this chapter, we consider the *general demand* setting introduced in Section 2.3.4. In this setting, players require not only one but several *levels* of service, and the mechanism determines which service level is granted to each player and at what price. We assume that players are concerned only about the *quantity* of service levels they obtain, e.g. the number of distinct connections to a source, executions of their job, etc. Moreover, once a player's request for a certain service level was refused, she will not be granted a higher level. This general demand cost sharing model has recently been investigated quite intensively; see [14, 29, 69, 73].

To the best of our knowledge, all previous works on cooperative cost sharing consider *offline* settings, where the entire instance is known in advance. Hence, when determining the allocation and payment scheme, the mechanism can take into account all input data associated with every player (bids for different service levels and other relevant player characteristics). However, many natural cost sharing games inherently bear an *online* characteristic in the sense that players arrive over time and reveal their input data only at their arrival. In such settings, the mechanism needs to take instantaneous and irreversible decisions with respect to the assigned service level and payment of the player without any knowledge about players that arrive in the future. Problems in which the input data is revealed gradually and irreversible decisions have to be taken without knowledge of future requests are the subject of *online computation* [15]. The standard yardstick to assess the quality of an online algorithm is by means of its *competitive ratio*, i.e. the worst case ratio of the cost of the solution produced by the online algorithm compared to the cost of an optimal offline algorithm that knows the entire input data in advance.

Our Contributions. The main contributions of this chapter are as follows:

1. We propose the first online model for general demand cost sharing games: In its most general form, every player arrives several times to request an additional service level. Upon the arrival of a player, the online mechanism immediately determines a price for her new request. We require that at each point of time, the sum of the collected payments approximates the cost of the (optimal offline) solution for the current allocation.

2. We completely characterize weakly group-strategyproof and group-strategyproof online mechanisms for general demand cost sharing games: We show that online cost sharing mechanisms are inherently weakly group-strategyproof for binary demand games. In the general demand case, this is true if the marginal costs of the underlying cost function are increasing. Moreover, we prove necessary and sufficient conditions for group-strategyproofness of online cost sharing mechanisms.

3. We present a simple yet effective method to derive online cost sharing mechanisms from competitive online algorithms: Given a ρ-competitive algorithm for the underlying problem, we show that the induced *incremental* online mechanism is ρ-budget balanced at all times. Using the above characterization, this enables us to derive incentive compatible online mechanisms for several binary demand and general demand cost sharing games for network design and scheduling problems. For example, we obtain an $O(\log^2 |V|)$-budget balanced group-strategyproof online mechanism for the online binary demand Steiner forest cost sharing game, where V denotes the set of vertices of the underlying graph.

Related Work. Immorlica et al. [53] partially characterized group-strategyproof cost sharing mechanisms in the offline case. They state that *upper-continuous* group-strategyproof β-budget balanced binary demand cost sharing mechanisms correspond to *cross-monotonic* cost sharing schemes. Juarez [56] very recently gave a similar characterization for mechanisms fulfilling the MAX property, meaning that indifferent players are always accepted.[1] He also showed that group-strategyproof cost sharing mechanisms correspond to feasible *sequential mechanisms* if indifferent players are always rejected. A sequential mechanism offers players service one after another according to an order that may change with previous decisions. The player is added to the set of selected players if her bid is larger than her payment (see Section 2.4.5 for more details).

Moulin [73] introduced *incremental cost sharing mechanisms* in the offline setting. An incremental mechanism is a sequential mechanism in which the payment offered to a player is equal to her *incremental cost*, i.e. the increase in cost caused

[1]A player is said to be *indifferent* if her bid is equal to her requested payment.

by adding her to the set of previously selected players. He claimed that for supermodular cost functions, incremental mechanisms are group-strategyproof and budget balanced. However, this statement is flawed (as indicated in [56]) and holds only under the assumption that players are never indifferent.

We extend the characterizations for group-strategyproof mechanisms to the general demand online setting. The mechanisms in our online model always accept indifferent players and thus fulfill Juarez' MAX property. This allows us to guarantee group-strategyproofness for all incremental mechanisms derived from *submodular* cost functions. Moreover, we achieve weak group-strategyproofness for the whole class of games with increasing marginal cost functions.

Organization of Chapter. In Section 5.2, we present our online model for general demand cost sharing. Section 5.3 contains the characterizations of weakly strategyproof, group-strategyproof and group-strategyproof mechanisms in this model. In Section 5.4, we show how to derive incremental online mechanisms from competitive algorithms and provide several examples. We conclude with Section 5.5

5.2 Online General Demand Cost Sharing

We extend general demand cost sharing games as defined in Section 2.3.4 to an *online* scenario [15]. Many cost sharing games studied in the literature are derived from combinatorial optimization problems. This motivates us to define *online cost sharing games* very generally depending on the varying online characteristics inherited from different online optimization problems.

The most important characteristic of our model is that an online mechanism must immediately fix the payment for a requested service at the point of time when it is revealed, without any knowledge about future requests. As in the offline setting, we assume that an online mechanism never accepts further requests from players that have previously been rejected. For cost sharing games that are derived from combinatorial optimization problems, the mechanism has to maintain a (possibly suboptimal) feasible solution for the current service allocation. The feasible modifications of this current solution are inherited from the underlying online optimization problem.

We use the *online list* model by Borodin et al. [15] to describe the proceeding of an online mechanism: Service requests (i, l) arrive according to an online order. (For certain problems like online scheduling, jobs may have release dates which are then treated as arrival times of the respective requests.) Upon arrival, the player reveals the characteristics of her new request (i.e. the input information for the

Algorithm 4: Online general demand cost sharing mechanism.

```
Input: online cost sharing game
Output: allocation vector $\boldsymbol{x} = (x_i)_{i \in U}$, payment vector $\boldsymbol{\phi} = (\phi_{i,l})_{i \in U, l \leq L}$
1 Initialize $\boldsymbol{x}^0 = \mathbf{0}$
2 forall requests $t \in T$ do
3     Read out input data and bid $b_{i,l}$ of newly arrived request $t =: (i, l)$.
4     Determine payment $p$ for new request.
5     if $b_{i,l} \geq p$ then set $\boldsymbol{x}^t = \boldsymbol{x}^{t-1} + \boldsymbol{e}_i$ and $\phi_{i,l} = p$
6     else set $\boldsymbol{x}^t = \boldsymbol{x}^{t-1}$ and $\phi_{i,l} = 0$ and delete all further appearances of
      player $i$.
7 end
8 Output allocation vector $\boldsymbol{x}$ and payments $\boldsymbol{\phi}$
```

underlying combinatorial optimization problem) and her bid $b_{i,l}$. The mechanism immediately offers her the additional service level at a price p that may depend on previous inputs and decisions only. If her bid $b_{i,l}$ is larger or equal to this price, the request is accepted and added to the current allocation. Otherwise, the request is rejected and all further appearances of player i are removed from the online list (formally, we may set $p = \infty$ for all subsequent requests of player i). A more formal description is given in Algorithm 4. We denote by $\boldsymbol{e}_i \in \mathbb{N}_0^U$ the ith unit vector. We sometimes write $\boldsymbol{x}(\boldsymbol{b})$ and $\phi(\boldsymbol{b})$ to refer to the outcome resulting from bid vector $\boldsymbol{b}$.

Let $\boldsymbol{x}^t$ denote the current allocation after processing request $t \in T = \{1, 2, \ldots\}$. Let $\bar{C}(\boldsymbol{x}^t)$ denote the cost of the actually computed solution for $\boldsymbol{x}^t$. We call an online cost-sharing mechanism *β-budget balanced at all times* for some $\beta \geq 1$ if for all requests $t \in T$:

$$\bar{C}(\boldsymbol{x}^t) \leq \sum_{i \in U} \sum_{l=1}^{x_i^t} \phi_{i,l} \leq \beta \cdot C(\boldsymbol{x}^t).$$

The conditions of individual rationality and no positive transfer as well as the different forms of incentive compatibility transfer in a straightforward way.

5.3 Incentive Compatibility

The following characterizations hold for all online mechanisms in our framework. Note that the requirements for group-strategyproofness highly depend on the fact that requests are accepted if the announced bid is equal to the offered price.

5.3.1 STRATEGYPROOFNESS

To achieve strategyproofness, we need to bound the increase in marginal valuations of individual players. As expressed by Fact 5.3 below, this is essential to prevent players from overbidding for some level to obtain positive utility for higher levels. In previous works on general demand cost sharing [14, 69], players' valuations were assumed to be non-increasing. However, we can slightly relax this condition by introducing a positive factor λ:

Definition 5.1 (λ-Decreasing Valuations)**.** *A valuation vector* $v_i \in \mathbb{R}^L$ *is* λ-decreasing *if for all* $1 < l \leq L$*,*

$$v_{i,l} \leq \lambda \cdot v_{i,l-1}.$$

Given λ-decreasing valuations for all players, an online mechanism is guaranteed to be weakly group-strategyproof if and only if the induced cost shares grow faster than the valuations (the proof is given in Section 5.3.2):

Definition 5.2 (λ-Increasing Prices)**.** *A cost sharing mechanism has* λ-increasing prices *if for every bid vector* $\boldsymbol{b}$ *and player* $i \in U$*, the price for any service level* $1 < l \leq x_i(\boldsymbol{b})$ *is at least* λ *times the price for the previous service level, i.e.*

$$\phi_{i,l}(\boldsymbol{b}) \geq \lambda \cdot \phi_{i,l-1}(\boldsymbol{b}).$$

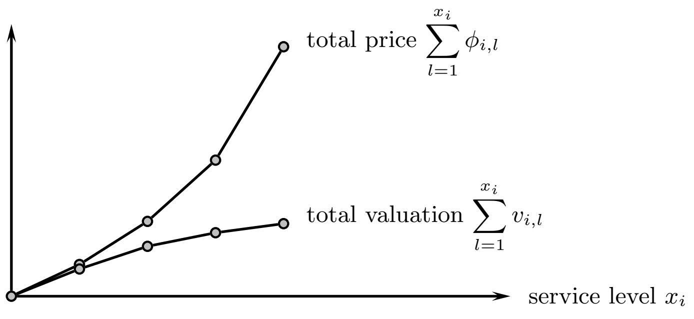

Figure 5.1: Example of λ-decreasing valuations and λ-increasing prices for $\lambda = 1$

The above conditions can be further generalized by letting λ vary for every player (and/or level) or by adding constant terms to the right hand sides. However, the following fact emphasizes that a set of conditions similar to the above are indeed necessary to achieve strategyproofness. We omit the proof due to space restrictions.

Fact 5.3. *A general demand online mechanism is not strategyproof if cost shares do not increase by more than valuations per service level.*

Proof. We assume for simplicity that there is only one player. Further, assume that there is a bid vector $\boldsymbol{b}$ such that for some service level $l \leq x(\boldsymbol{b})$, we have $\phi_l(\boldsymbol{b}) < \lambda \cdot \phi_{l-1}(\boldsymbol{b})$; say $\phi_l(\boldsymbol{b}) = \lambda \cdot \phi_{l-1}(\boldsymbol{b}) - \delta$ for some $\delta > 0$. We define the player's valuations as $v_{l-1} = \phi_{l-1}(\boldsymbol{b}) - \epsilon$, $v_l = \lambda \cdot v_{l-1}$ and $v_k = \phi_k(\boldsymbol{b})$ for $k < l-1$. Thus, the valuation increases by a factor λ when going from level $l-1$ to level l, whereas the prices increase by less than a factor λ.

With this valuation vector, the player obtains positive utility in the run on $\boldsymbol{b}$: $u(\boldsymbol{b}) = 0 + u_{l-1}(\boldsymbol{b}) + u_l(\boldsymbol{b}) = -\epsilon + \lambda(\phi_{l-1}(\boldsymbol{b}) - \epsilon) - (\lambda\phi_{l-1}(\boldsymbol{b}) - \delta) = \delta - (\lambda+1)\epsilon > 0$ for sufficiently small ϵ. On the other hand, she gets zero utility if she bids truthfully since she gets rejected at service level $l-1$. Hence, the mechanism is not strategyproof. The same argumentation can be pursued with additive instead of multiplicative increase. □

5.3.2 Weak Group-Strategyproofness

We prove now that under the above conditions, an online mechanism is in fact weakly group-strategyproof.

Theorem 5.4. *If valuations are λ-decreasing, a general demand online cost sharing mechanism with λ-increasing prices is weakly group-strategyproof.*

Proof. Fix a coalition $S \subseteq U$ and a bid vector $\boldsymbol{b}$ with $b_i = v_i$ for all $i \in S$. Assume for contradiction that all members of the coalition can strictly increase their utilities by changing their bids to $\boldsymbol{b'}$ (while $b_i = b'_i$ for all $i \notin S$). Let (i, l) be the first request for which the mechanism makes different decisions in the runs on $\boldsymbol{b}$ and $\boldsymbol{b'}$. By the online character of the mechanism, the price offered for request (i, l) only depends on previous decisions and is thus equal in both runs. Let p denote this offer price. There are two possible cases:

1. $v_{i,l} < p \leq b'_{i,l}$. Then, $\phi_{i,l}(\boldsymbol{b'}) = p$, and λ-decreasing valuations and λ-increasing prices yield $\ldots \leq \lambda^{-2}v_{i,l+2} \leq \lambda^{-1}v_{i,l+1} \leq v_{i,l} < \phi_{i,l}(\boldsymbol{b'}) \leq \lambda^{-1}\phi_{i,l+1}(\boldsymbol{b'}) \leq \lambda^{-2}\phi_{i,l+2}(\boldsymbol{b'}) \leq \ldots$. Hence, player i has negative utility for service levels l and higher in the run on $\boldsymbol{b'}$, whereas when bidding truthfully, the utility for each level is non-negative.

2. $b'_{i,l} < p \leq v_{i,l}$. Then, player i obtains only $l-1$ levels of service in the run on $\boldsymbol{b'}$, whereas she may get additional utility by accepting level l in the run on $\boldsymbol{b}$.

Consequently, player i gets less or equal utility in the run on $\boldsymbol{b}'$, a contradiction to the assumption. □

For binary demand cost sharing games, both Definitions 5.1 and 5.2 are always fulfilled since there is only one service level. Hence, quite remarkably, binary demand online cost sharing mechanisms are inherently weakly group-strategyproof.

5.3.3 Group-Strategyproofness

In order to ensure the stronger notion of group-strategyproofness, we need to prevent that *dropping out*, i.e. underbidding in case of indifference, can help subsequent players. Towards this end, we introduce the following generalization of the well-known notion of cross-monotonicity for binary demand cost sharing games [73].

Consider a fixed instance of an online cost sharing game and let $\phi_{i,l}(\boldsymbol{b})$ denote the price that player i is offered for service level l when $\boldsymbol{b}$ is the bid vector input to the mechanism. Throughout this section, we assume λ-decreasing valuations and λ-increasing prices.

Definition 5.5 (Cross-monotonicity)**.** *An online mechanism is* cross-monotonic *if for every player $i \in U$ and service level l, the offered price does not decrease when a subset of requests are accepted in previous iterations, i.e.*

$$\phi_{i,l}(\boldsymbol{b}') \geq \phi_{i,l}(\boldsymbol{b})$$

for all bid vectors $\boldsymbol{b}, \boldsymbol{b}'$ such that $x^{t-1}(\boldsymbol{b}') \leq x^{t-1}(\boldsymbol{b})$, where (i,l) is request t.

This condition is sufficient for an online cost sharing mechanism to be group-strategyproof. The proof contains two main ideas: First, dropping out can never help others since it only increases cost shares of subsequent bidders. Second, the first member of a coalition who overbids for an additional level of service can only decrease her utility by doing this, since prices increase more than valuations in terms of service levels.

Theorem 5.6. *If valuations are λ-decreasing, an online cost sharing mechanism with λ-increasing prices is group-strategyproof if it is cross-monotonic.*

Proof. Fix a coalition $S \subseteq U$ and a bid vector $\boldsymbol{b}$ with $b_i = v_i$ for all $i \in S$. Assume that every member of the coalition increases or maintains her utility when the coalition changes their bids to $\boldsymbol{b}'$ (while $b_i = b_i'$ for all $i \notin S$).

We first prove that $\boldsymbol{x}^t(\boldsymbol{b}') \leq \boldsymbol{x}^t(\boldsymbol{b})$ for all $t \in T$. Assume for contradiction that there is a request which is accepted in the run on $\boldsymbol{b}'$ but not in the run on $\boldsymbol{b}$. Let

(i,l) be the earliest such request, say request t. That is, $\boldsymbol{x}^\tau(\boldsymbol{b}') \leq \boldsymbol{x}^\tau(\boldsymbol{b})$ for all $\tau < t$. By cross-monotonicity, we have $\phi_{i,l}(\boldsymbol{b}') \geq \phi_{i,l}(\boldsymbol{b})$. Since players outside the coalition submit the same bids in both runs, player i must be a member of the coalition to gain service in the run on $\boldsymbol{b}'$. But then, $\phi_{i,l}(\boldsymbol{b}') \geq \phi_{i,l}(\boldsymbol{b}) > b_{i,l} = v_{i,l}$ and hence by λ-decreasing valuations and λ-increasing prices, player i has negative utility for service levels l and higher in the run on $\boldsymbol{b}'$. Since $\boldsymbol{x}^\tau(\boldsymbol{b}') \leq \boldsymbol{x}^\tau(\boldsymbol{b})$ for all $\tau < t$, by cross-monotonicity $\phi_{i,k}(\boldsymbol{b}') \geq \phi_{i,k}(\boldsymbol{b})$ for all $k < l$ as well, and therefore $u_i(\boldsymbol{b}') < u_i(\boldsymbol{b})$, contradicting the first assumption.

We can conclude that $\boldsymbol{x}^t(\boldsymbol{b}') \leq \boldsymbol{x}^t(\boldsymbol{b})$ for all $t \in T$. Hence, $\phi_{i,l}(\boldsymbol{b}') \geq \phi_{i,l}(\boldsymbol{b})$ for all i, l by cross-monotonicity. This means that

$$u_i(\boldsymbol{b}') = \sum_{l=1}^{x_i(\boldsymbol{b}')} (v_{i,l} - \phi_{i,l}(\boldsymbol{b}')) \leq \sum_{l=1}^{x_i(\boldsymbol{b})} (v_{i,l} - \phi_{i,l}(\boldsymbol{b})) = u_i(\boldsymbol{b})$$

for all i and l, hence we obtain group-strategyproofness. □

We prove next that the conditions in Theorem 5.6 are not only sufficient but also necessary, even in the binary demand case.

Theorem 5.7. *A binary demand online mechanism is not group-strategyproof if it is not cross-monotonic.*

Proof. Consider an online mechanism that is not cross-monotonic; let $L = 1$. That is, there are bid vectors $\boldsymbol{b}, \boldsymbol{b}'$ with $\boldsymbol{x}^{t-1}(\boldsymbol{b}') \leq \boldsymbol{x}^{t-1}(\boldsymbol{b})$ and $\phi_i(\boldsymbol{b}') < \phi_i(\boldsymbol{b})$ for some player i. For simplicity, assume that i is the last player in the online instance. Since the mechanism is online, $\phi_i(\boldsymbol{b}')$ does not depend on b'_i, so we can assume that $b'_i = \phi_i(\boldsymbol{b})$. We will define valuations such that there is a coalition S which has an incentive to misreport their valuations.

Define $S := \{j \in U \mid b_j \neq b'_j\} \cup \{i\}$. Assume that all $j \in U \setminus S$ bid $b_j = b'_j$. Now, define $v_j := \phi_j(\boldsymbol{b})$ for all $j \in S$. Observe that if all players in S bid truthfully, the outcome of the mechanism is the same as for bid vector $\boldsymbol{b}$. Now, if the coalition changes their bids to $\boldsymbol{b}'$, some players $j \in S \setminus \{i\}$ lose service but all retain their previous utility of zero. Meanwhile, player i increases her utility from zero to $\phi_i(\boldsymbol{b}) - \phi_i(\boldsymbol{b}') > 0$. Hence, the mechanism is not group-strategyproof. □

5.4 Incremental Online Mechanisms

We now describe a generic method to turn competitive online algorithms into online cost sharing mechanisms. Given a ρ-competitive online algorithm ALG for a combinatorial optimization problem $\mathcal{P}$, we define an *incremental* online mechanism for the corresponding cost sharing game, which is ρ-budget balanced at all times. The mechanism is weakly group-strategyproof if the algorithm's marginal costs are increasing, which is gratuitous in the binary demand case.

Let ALG be a ρ-competitive algorithm for an online combinatorial optimization problem $\mathcal{P}$. Consider an instance $\mathcal{I}$ of $\mathcal{P}$. The incremental online mechanism induced by ALG works as follows: Requests arrive according to $\mathcal{I}$. Each time a new request arrives, we simulate ALG on the online instance induced by the requests that have previously been accepted plus the new item. The price p for the additional service level is set to be the incremental cost caused by the update in the competitive algorithm. We call an online algorithm ALG *cross-monotonic* if the induced incremental online mechanism is cross-monotonic. It is straightforward to see that the budget balance factor of an incremental online mechanism is inherited from the competitive ratio of the input algorithm:

Lemma 5.8. *The incremental online mechanism is ρ-budget balanced at all times.*

Proof. In every iteration t of the mechanism, we have

$$\sum_{i \in U} \sum_{l=1}^{x_i^t} \phi_{i,l} = \bar{C}(\boldsymbol{x}^t),$$

since every accepted player pays exactly her incremental cost. Since ALG is ρ-competitive, we obtain $\bar{C}(\boldsymbol{x}^t) = \sum_{i \in U} \sum_{l=1}^{x_i^t} \phi_{i,l} \leq \rho \cdot C(\boldsymbol{x}^t)$. □

5.4.1 Binary Demand Examples

To demonstate the applicability of our framework, we now apply it to competitive online algorithms for a number of combinatorial optimization problems. In this section, we give examples for *binary demand* cost sharing games, i.e. the maximum service level is $L = 1$ and every player has only one request.

Online Scheduling.

Consider the parallel machine scheduling problem with the objective of minimizing the makespan. Any list scheduling algorithm has an approximation factor of at most 2 for this problem. Hence, the online algorithm that adds each arriving job to the machine with the currently least load is 2-competitive. Unfortunately, it is not cross-monotonic as deleting jobs can cause higher or lower completion times for subsequent jobs. Nonetheless, our framework leads to a 2-budget balanced, weakly group-strategyproof online mechanism. Note that in this scenario, jobs do not have release dates and so the online order is not coupled with scheduling time.

Corollary 5.9. *There is a 2-budget balanced weakly group-strategyproof incremental online mechanism for the minimum makespan scheduling problem on parallel machines* $P||C_{\max}$.

Online Steiner Tree and Forest.

Given an undirected graph G with edge costs, connection requests arrive online. In the Steiner forest problem, each request consists of a pair of terminals s_i, t_i; in the Steiner tree problem, all requests have one vertex in common, i.e. $s_i = s_j$ for all $i, j \in U$. The goal is to select a minimum cost set of edges such that each terminal pair is connected by a path. Let n denote the number of players (i.e., terminal pairs).

The online greedy Steiner tree algorithm picks the shortest path to the current tree each time a new terminal pair arrives. It has a competitive ratio of $\log n$, while the competitive ratio of any online algorithm is shown to be at least $1/2 \log n$ [51]. Hence, our framework gives a weakly group-strategyproof $\Theta(\log n)$-budget balanced online cost sharing mechanism for the Steiner tree problem, which is asymptotically best possible. The greedy algorithm for the online Steiner forest problem achieves an approximation ratio of $O(\log^2 n)$.

Corollary 5.10. *There is an* $O(\log^2 n)$*-budget balanced weakly group-strategyproof incremental online mechanism for the Steiner forest game. This mechanism is* $(\log n)$*-budget balanced for the Steiner tree game.*

Unfortunately, the greedy algorithm is not cross-monotonic, as the removal of some players may have the effect that some other players switch their paths, which in turn can have arbitrary effects on the costs incurred by subsequent players. This issue can be overcome if paths are unambiguous; e.g. if $G = (V, E)$ is a forest, the above mechanisms are group-strategyproof. Pushing this insight even further, we obtain an $O(\log |V|)$-budget balanced group-strategyproof mechanism for the Steiner forest game if the underlying graph is known in advance: We use the *oblivious* online Steiner forest algorithm proposed by Gupta et al. [44], which essentially works as follows: Given the underlying graph, the algorithm precomputes a collection of paths. When a new terminal pair arrives, it simply connects it by one of the predefined paths. The authors show that one can identify a collection of paths such that the resulting algorithm is $O(\log |V|)$-competitive. Since the used paths are defined in advance, a player can only benefit from the presence of other players, who might pay for parts of her designated path. Hence, we obtain cross-monotonicity without losing much in terms of the budget balance guarantee.

Corollary 5.11. *There is an* $O(\log^2 |V|)$*-budget balanced group-strategyproof in-*

cremental online mechanism for the Steiner forest game, where V is the vertex set of the underlying graph.

We believe that such “universal” algorithms that determine generic approximate solutions without knowing the upcoming instance will also yield group-strategyproof incremental online mechanisms for several other interesting problems such as e.g. the traveling salesman problem.

5.4.2 General Demand Examples

In this section, we exploit the whole range of our framework by deriving incremental mechanisms for *general demand* cost sharing games. In the first example, we assume that players arrive only once with the complete list of their requests, while in the second example, the arrival sequence is mixed, i.e. players can take turns announcing additional requests.

Online Preemptive Scheduling.

A common problem in preemptive scheduling is the parallel machine setting in which each job has a release date. The cost of a solution is given by the sum of all completion times. The single machine case is solved optimally by the *shortest remaining processing time* (SRPT) algorithm [93]. SRPT is a 1.25-approximation for the parallel machine case [95].

In the corresponding cost sharing game, we treat the release date of a job as its arrival time. Upon arrival, each player may request multiple executions of her job. In scheduling terms, each player owns a set of jobs which all have the same release date and processing time. E.g. consider a student who asks a copy shop to print and bind several copies of his thesis, or a joinery is asked to produce a few of the same individual piece of furniture. In such scenarios, it is very natural to assume that the marginal valuation for each additional copy is decreasing, i.e. $v_{i,l} \geq v_{i,l+1}$ for all i, l.

Before subsequent players arrive, SRPT schedules all of player i's jobs subsequently. Hence, each of them delays the same number of jobs, and later copies have larger completion times. Therefore, the general demand incremental online mechanism induced by SRPT has increasing marginal prices.

Corollary 5.12. *There is a 1.25-budget balanced weakly group-strategyproof general demand incremental online mechanism for the preemptive scheduling problem with release dates $P|r_i, pmtn|\sum C_i$. This mechanism is budget balanced in the single machine case.*

Online Multicommodity Routing.

In an online multicommodity routing problem, we are given a directed graph with monotonically increasing cost functions on each arc. Commodities arrive online and request routing of l units of capacity from some vertex to another. We assume that the routing is splittable in integer units. The greedy algorithm which routes each unit of flow separately in an optimal way is $(3 + 2\sqrt{2})$-competitive for this problem [47]. It is clear that marginal costs are increasing, since the cost functions on each arc grow with increasing traffic. This is true even when players arrive in a mixed order and request to route additional units between their source-destination pair. However, this is a congestion-type game (the more players in the game, the higher the costs per request), and so we cannot expect group-strategyproofness.

Corollary 5.13. *There is a $(3+2\sqrt{2})$-budget balanced weakly group-strategyproof incremental online mechanism for the online multicommodity routing problem in which each player arrives multiple times.*

5.5 Conclusion

We proposed a new framework for online general demand cost sharing games and characterized strategyproof, weak group-strategyproof and group-strategyproof mechanisms in this framework. Quite surprisingly, weak group-strategyproofness comes for free for binary demand problems; for general demands, cost shares for subsequent service levels must increase faster than valuations. In both cases, online mechanisms are group-strategyproof if and only if *dropping out* cannot help subsequent players. Consequently, we cannot expect incremental cost sharing mechanisms for problems with congestion effects like e.g. scheduling games to be group-strategyproof, while this seems easier for network design problems.

In the offline setting, finding an appropriate order in which players are considered is the key to derive cost sharing mechanisms with attractive budget balance guarantees (see [18]). In the online case, this order is determined by an adversary and thus not under the control of the mechanism designer, which strongly constrains the possibilities of designing valuable cost sharing mechanisms. However, our results prove that binary demand problems, there is no gap between the best possible competitive ratio of an online algorithm and the best possible budget balance factor of a weakly group-strategyproof online cost sharing mechanism.

We consider this work as a very natural and general starting point to exploit the possibilities and limits of cooperative cost sharing in different online contexts. It would be interesting to see more applications to our framework, with and without usage of the direct derivation of incremental mechanisms from competitive algorithms. Our model restricts feasible allocations to a continuous sequence of accepts for each player, starting with their first request. This feature of the model enhances truthfulness as it prevents players from underbidding to reject some service request and then obtain it later for a cheaper price. One interesting line of research would be to allow for more general mechanisms which might accept further requests of players even after a request has been rejected.

6

MECHANISM DESIGN WITH CONGESTION

We study mechanisms for resource allocation problems with congestion externalities. Our model is based on standard models from the area of *congestion games.* On top of this, we develop a natural definition for players' utility functions in the presence of congestion, thereby combining the two fields of mechanism design and congestion games. In this chapter, we assume that the utility of a player depends only on the most congested resource she uses, as in so-called *bottleneck* congestion games. Our main result is a generic mechanism for symmetric bottleneck congestion games which is truthful and approximates social welfare. Our approach is to reduce any given instance to an instance with singleton strategies only, which we then solve optimally using a dynamic program. The resulting mechanism applies to various resource allocation settings and achieves different approximation guarantees that we prove using duality in hypergraphs. We show that our mechanism is optimal for problems evolving from single-commodity network congestion games. We complement our results by identifying several special cases of the problem that are hard to approximate.

6.1 Introduction

Resource allocation has been a central topic in computer science ever since it exists. In a resource allocation problem, resources are requested by different users. A resource may be allocated to multiple users and shared among them. For example, consider a network link: Several computers may be assigned to route their data using the same link. Similar and more complex situations arise in diverse applications ranging from job scheduling to sponsored search (where the search result page is shared among different advertisers). In such settings, the utility of a user is not only determined by whether she is allocated a resource but is also influenced by other users who are allocated the same resource: A page with numerous advertisements may not have the same impact as one with only few. Similarly, an overloaded network link may have bad performance. This phenomenon in general is known as *congestion*.

Upon the introduction of *algorithmic game theory* [76, 81], the resource allocation problem has been revisited from various economic viewpoints. These approaches are motivated by computer systems that are used by players with diverse and selfish interests. The main challenge in such settings is that the players' interests are usually not aligned with the desired global performance measure. This measure may depend on data part of which is only available to the users. For instance, in packet routing, only the sender knows how important/urgent her request is. A strategic user may not want to reveal her true data if she prefers the outcome of the algorithm fed with a false report. As described in Section 2.2, the area of *mechanism design* aims at designing efficient mechanisms that elicit the true data from selfish players and yet optimize a global objective.

One natural approach to model resource allocation in a game theoretic context is via *congestion games*, which we introduced in Section 2.6. In a congestion game, we are given a set of resources and a set of players. Each player has a set of strategies. A strategy corresponds to a subset of the resources, e.g. a route between a source and a destination in a given network. In a classical congestion game, it is assumed that each player chooses one of her strategies to optimize her payoff, e.g. minimize the total delay of her route. The cost depends on the congestion of a resource, i.e. the number of players using this resource. A congestion game is called *symmetric* if all players have the same strategy set. Many works employ Nash equilibria to predict the outcome of congestion games. In particular, the price of anarchy and the price of stability [90] are used to quantify the degradation of the solution in the worst and the best Nash equilibria compared to the optimal allocation, respectively.

Contributions. In this chapter, we take a mechanism design approach to resource allocation problems inspired by congestion games. We model a mechanism design problem based on a given congestion game as follows: For a given set of strategies and set of players, we assume that each player assigns a private valuation if she is assigned one of the strategies in her strategy set. The resources can be shared among the players; however, like in congestion games, there is a loss caused by resource congestion. We model this loss by decreasing the player's valuation for a resource through multiplying it with a decreasing *throughput function* that depends on the number of players using the resource. In general, an allocation causes players to have different congestion on different resources. Although our model in general captures different metrics for computing players' utility functions (compare Section 2.6), all our results in this chapter concern the case that players experience the loss of their *most* congested resource (also called *bottleneck*). Our goal is to design *efficient* mechanisms in the sense that they maximize social welfare.

We design a generic truthful mechanism for maximizing social welfare of symmetric bottleneck congestion games. Our mechanism is based on reducing the problem to an instance with singleton strategies by using duality in hypergraphs. Our mechanism applies to various settings and achieves different approximation guarantees. As an example, it computes optimal allocations for single-commodity network congestion games. In order to prove truthfulness, we extend the notion of monotone mechanisms and show that this extension is sufficient for incentive compatibility in our setting. We also identify several special cases of congestion games in which the optimal social welfare is hard to approximate.

Related Work. There is a significant amount of work in the field of congestion games, most of which studies several variations of the price of anarchy or stability; see Section 2.6 for an overview of related work in that area.

Chakrabarty, Mehta, Nagarajan and Vazirani [24] study congestion games from a coordinational point of view. They give a polynomial-time algorithm for computing the social optimum of single-commodity network congestion games with linear edge-dependent cost functions and unweighted players. They also present efficient algorithms based on dynamic programming for finding the social optimum of singleton congestion games for different scenarios. Meyers [70] studies the computational complexity of finding socially optimal allocations for different classes of congestion games. However, in both works, players sum up the latencies over all resources they use, i.e. they do not consider the bottleneck case. Also, neither considers truthful implementations of their allocation algorithms.

Our work generally belongs to the area of *auctions with externalities* [57, 20, 55] in which different types of negative externalities among the winners of an auction are studied. Ghosh and Mahdian [38] recently examined the computational challenges of allocation problems with externalities and showed that the problem is inapproximable in some general settings. Our problem, however, is much simpler because it is "anonymous": Our negative externalities only depend on the number of winners and not on their identities or characteristics. Salek and Kempe [92] study auctions with congestion externalities. In their model, bidders can share items and each bidder has a "share-averseness" function that reflects her disutility with respect to sharing. They derive partial characterizations for revenue maximization in the Bayesian setting. Moreover, they obtain tight approximations for maximizing social welfare in the case of single-minded bidders.

Organization of Chapter. In Section 6.2, we present our model for congestion games in a mechanism design context. Section 6.3 develops necessary conditions for truthful mechanisms in this model. We present our generic mechanism in Section 6.4, and give several examples for applications and the corresponding approximation guarantees in Section 6.5. These results are complemented with several hardness results in Section 6.6. We conclude with some final remarks in Section 6.7.

6.2 Model

We now introduce our model for mechanism design with congestion externalities. The setting largely corresponds to the mechanism design setting with *single parameter domains* described in Section 2.2.5. However, to incorporate the externalities caused by congestion effects, we perturb the players' utility functions by a parameter g that depends on the congestion experienced by a player. We model the congestion along the lines of standard models in the classical *congestion games* literature [87, 72]. See Section 2.6 for an introduction to congestion games.

Let R be a set of m resources and let U be a set of n potential users of the resources, also called players. Every player $i \in U$ has a *strategy set* $\mathcal{S}_i \subseteq \mathcal{P}(R)$ and is interested in using one of the resource sets contained in her strategy set. For intuition, consider the example of *network congestion games*:

Example 6.1 (Network Congestion Game)**.** *In a network congestion game, the set of resources corresponds to the set of edges in a given graph. Each player wishes to route one unit of demand from her specified source to her sink vertex. The strategy set of a player is thus defined as the set of all paths between these two vertices.*

An outcome of the corresponding mechanism design problem is defined by an *allocation* of resources to players. We denote an allocation by a matrix $\boldsymbol{x} \in \{0,1\}^{n\times m}$, where $x_{ir} = 1$ if player i is allocated resource r and $x_{ir} = 0$ otherwise.

We denote by $x_r := \sum_{i \in U} x_{ir}$ the total congestion on resource r. We call an allocation *feasible* if every player is either allocated one of her strategies or nothing. More precisely, an allocation is feasible if either $x_{ir} = 0$ for all $r \in R$ (the player loses), or there is a strategy $S \in \mathcal{S}_i$ such that $x_{ir} = 1$ for all $r \in S$ and $x_{ir} = 0$ for all $r \notin S$ (she wins and is assigned strategy S). We define $x_{iS} = 1$ if player i is assigned strategy $S \in \mathcal{S}$ and $x_{iS} = 0$ otherwise.

The *throughput* of a resource is defined by a function $g : \mathbb{N} \to [0, 1]$, where $g(k)$ denotes the throughput of resource $r \in R$ when this resource is used by k players. We assume w.l.o.g. that g is scaled such that $\max_{k \leq n} g(k) = 1$. Unless otherwise stated, we require g to be non-increasing to account for the *negative* externalities caused by congestion.

The valuation function of a player depends on the throughput of the most congested resource that she uses, which we call her *bottleneck*. Formally, player i's *valuation function* is defined as

$$w_i(\boldsymbol{x}) := \begin{cases} v_i \cdot g(c_i(\boldsymbol{x})) & \text{if } i \text{ wins,} \\ 0 & \text{if } i \text{ loses.} \end{cases}$$

Here,

$$c_i(\boldsymbol{x}) := \max\{x_r \mid r \in R \text{ and } x_{ir} > 0\}$$

denotes the maximum congestion that player i experiences (her *bottleneck congestion*), and $v_i \in \mathbb{R}^+$ is her private type. Intuitively, v_i corresponds to player i's congestion-free value of winning. Note that v_i is only known to player i, while the throughput function g is publicly known.

Accordingly, when $\boldsymbol{p} = (p_1, \ldots, p_n)$ is the vector of prices requested from the players, the *utility* of player $i \in U$ is defined as

$$u_i(\boldsymbol{x}, \boldsymbol{p}) := \begin{cases} v_i \cdot g(c_i(\boldsymbol{x})) - p_i & \text{if } i \text{ wins,} \\ 0 & \text{otherwise.} \end{cases}$$

We assume that each player aims at maximizing her own utility. Players are strategic, i.e. each player tries to influence the outcome by reporting a possibly false *bid* b_i instead of her true value v_i.

A *direct revelation mechanism* $M = (\boldsymbol{x}, \boldsymbol{p})$ consists of an *allocation rule* $\boldsymbol{x} : \mathbb{R}^n_+ \to \{0,1\}^{n \times m}$ and a payment rule $\boldsymbol{p} : \mathbb{R}^n_+ \to \mathbb{R}^n_+$. Given the bid vector $\boldsymbol{b} = (b_1, \ldots, b_n)$, a it selects a feasible allocation $\boldsymbol{x}(\boldsymbol{b})$ and requests a non-negative payment $p_i(\boldsymbol{b})$ from every player $i \in U$. We require that $p_i = 0$ for all losing players.

We are interested in strategyproof mechanisms that maximize the (utilitarian) social welfare. The *social welfare* of an allocation $\boldsymbol{x}$ is defined as

$$W(\boldsymbol{x}) := \sum_{i \in U} w_i(\boldsymbol{x}).$$

As defined in Section 2.2.1, a mechanism is *strategyproof* if no player can increase her utility by reporting a false bid $b_i \neq v_i$.

The results presented in this chapter concern the important special case of *symmetric* congestion games. A congestion game is called symmetric if all players have the same strategy set, i.e. $\mathcal{S}_i = \mathcal{S}$ for all $i \in U$. We denote by $\Gamma = (U, R, \mathcal{S})$ the symmetric bottleneck congestion game induced by U, R and $\mathcal{S}$.

We call a congestion game in which all strategies are singletons, i.e. $|S| = 1$ for all $S \in \mathcal{S}$, a *singleton* congestion game. In the singleton case, the bottleneck measure coincides with other standard measures used for congestion games, and so we no longer distiguish between bottleneck congestion games and congestion games. We denote by $\Gamma_a = (U, [a], [a])$ a symmetric singleton congestion game on a resources with $\mathcal{S} = [a]$. We speak of cardinality two strategies if $|S| = 2$ for all $S \in \mathcal{S}$.

6.3 Conditions for Truthfulness

As we stated in Section 2.2.3, a social choice function is truthfully implementable if it is weakly monotone. For binary demand games (where players either "lose" or "win"), weak monotonicity reduces to the property that a winner continues to win if she raises her bid. For mechanisms that provide different levels of service, the condition is more subtle. Below, we prove that a condition we call *smooth monotonicity* gives rise to truthful mechanisms in our setting. Intuitively, smooth monotonicity requires that when a winner raises her bid, this must not degrade her quality of service, i.e. she can only experience less congestion.

Definition 6.2 (Smooth Monotonicity). *An allocation rule $\boldsymbol{x}(\boldsymbol{b})$ is* smoothly monotone *if for every winner i in $\boldsymbol{x}(\boldsymbol{v})$, if i raises her bid to $b_i > v_i$ then she still wins and her bottleneck congestion does not increase, i.e.*

$$c_i(\boldsymbol{x}(b_i, v_{-i})) \leq c_i(\boldsymbol{x}(\boldsymbol{v})).$$

Since the types in our mechanism design setting are real numbers v_i, the type domains are convex and thus Theorem 2.13 applies.[1] However, to make this thesis as self-contained as possible, we directly prove that smoothly monotone allocation rules are implementable. Furthermore, we explicitly provide a pricing

[1]In our model, using the above notation, the weak monotocity condition in Definition 2.12 translates to $v_i g(c_i(\boldsymbol{x}(\boldsymbol{v}))) - v_i g(c_i(\boldsymbol{x}(b_i, v_{-i}))) \leq b_i g(c_i(\boldsymbol{x}(\boldsymbol{v}))) - b_i g(c_i(\boldsymbol{x}(b_i, v_{-i})))$. Rearranging yields $g(c_i(\boldsymbol{x}(b_i, v_{-i}))) \geq g(c_i(\boldsymbol{x}(\boldsymbol{v})))$ for the case that $b_i > v_i$, w.l.o.g. Hence, smooth monotonicity is just a slightly stronger condition than weak monotonicity in our setting (while both conditions coincide if g is strictly decreasing).

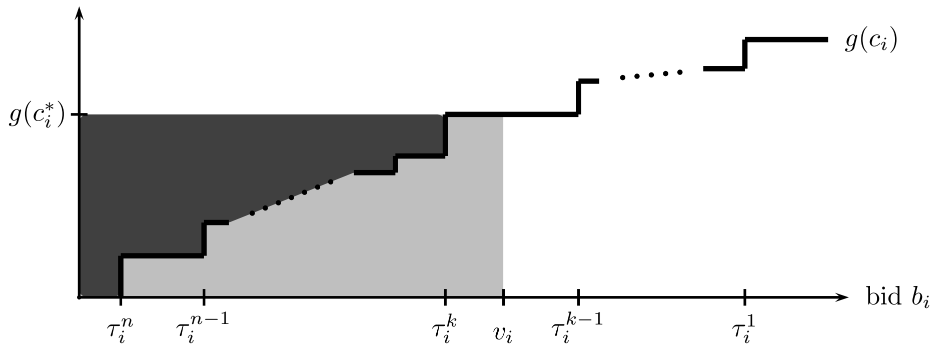

Figure 6.1: The troughput of player i as a function of her bid b_i given a fixed vector b_{-i}

scheme combined with which a smoothly monotone allocation rule yields a truthful mechanism. This also enables us to show that both the allocation rule and the payment scheme can be computed in polynomial time, given that the currency of prices and bids exhibits a smallest unit.

Let $\boldsymbol{x}(\boldsymbol{b})$ be a smoothly monotone allocation function. Consider a player i and fix arbitrary bids b_{-i} for all other players. We define the k^{th} *critical bid* $\tau_i^k(b_{-i})$ of player i as the infimum over all b_i such that her congestion $c_i(\boldsymbol{x}(\boldsymbol{b}))$ is at most k. Player i loses if she bids less than τ_i^n. The property of smooth monotonicity implies that $\tau_i^n \leq \ldots \leq \tau_i^2 \leq \tau_i^1$. Given all critical bids, we charge player i an amount of

$$p_i := g(k)\tau_i^k - \sum_{j=k+1}^{n} (\tau_i^{j-1} - \tau_i^j)g(j)$$

if her bid b_i lies in the interval $(\tau_i^k, \tau_i^{k-1}]$.

Figure 6.3 illustrates the pricing scheme for player i given a fixed vector b_{-i} of bids submitted by other players. On the x-axis, the critical bids $\tau_i^k(b_{-i})$ for player i are marked. The bold curve depicts her bottleneck troughput $g(c_i(\boldsymbol{x}(\boldsymbol{b}))$ as a function of her bid b_i. The area of the light gray rectangle corresponds to her valuation $w_i = v_i \cdot g(c_i^*)$ if she receives congestion c_i^* by bidding her true value v_i. The dark gray area marks the payment p_i she has to make when she bids v_i or any value in the interval $(\tau_i^k, \tau_i^{k-1}]$.

As we have stated above, the necessary critical bids can be computed in polynomial time if there is a smallest currency unit: For every player and congestion level, we make a binary search to find her respective critical bid while holding all other players' bids fixed. We thus perform a polynomial number of binary searches, each of which takes $O(\log b_i)$ time, which is equal to the encoding size of player i's bid b_i, assuming binary encoding.

The following theorem states that the above payment scheme guarantees truthfulness.

Theorem 6.3. *Given a smoothly monotone allocation rule $\boldsymbol{x}$, the mechanism $M = (\boldsymbol{x}, \boldsymbol{p})$ induced by $\boldsymbol{x}$ and the above payment scheme is strategyproof.*

Proof. Fix a player $i \in U$. Let $(\boldsymbol{x}, \boldsymbol{p})$ denote the outcome when all players bid according to $\boldsymbol{v}$, and $(\boldsymbol{x}', \boldsymbol{p}')$ denote the outcome when player i unilaterally switches to $b_i \neq v_i$ while all other bidders stick to $\boldsymbol{v}$. Let c and c' be the bottleneck congestion values that player i experiences, respectively. We prove the claim by doing a case distinction. For a better readability, we sometimes drop the index i from τ_i and v_i in this proof.

First, assume that player i loses in $\boldsymbol{x}$ and hence $u_i(\boldsymbol{x}, \boldsymbol{p}) = 0$. She can only improve her utility by submitting a higher bid $b_i \geq \tau_i^n \geq v_i$ to become a winner in $\boldsymbol{x}'$. However, by doing this, she obtains non-positive utility:

$$\begin{aligned} u_i(\boldsymbol{x}', \boldsymbol{p}') = vg(c') - p' &= (v - \tau^{c'})g(c') + \sum_{k=c'+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\ &\leq (v - \tau^{c'})g(c') + \sum_{k=c'+1}^{n} (\tau^{k-1} - \tau^k)g(c') \\ &= (v - \tau^n)g(c') \\ &\leq 0 \end{aligned}$$

Next, assume that player i wins in $\boldsymbol{x}$ and loses in $\boldsymbol{x}'$. Then by definition, her utility is zero in $\boldsymbol{x}'$. However, she obtains non-negative utility by bidding truthfully:

$$u_i(\boldsymbol{x}, \boldsymbol{p}) = vg(c) - p = (v - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \geq 0$$

since $v > \tau^c$ and thus both summands are non-negative.

Finally, assume that player i wins both in $\boldsymbol{x}$ and $\boldsymbol{x}'$. If this does not change her bottleneck congestion, her utility remains equal. We distinguish the remaining two cases:

Case 1. Player i underbids and obtains a larger congestion $c' > c$. This means that $\tau^{c'} < \tau^c < v_i$, and we can conclude (using monotonicity of g) that

$$\begin{aligned}
u_i(\boldsymbol{x}, \boldsymbol{p}) &= (v - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&\geq (v - \tau^c)g(c') + \sum_{k=c+1}^{c'} (\tau^{k-1} - \tau^k)g(c') + \sum_{k=c'+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= (v - \tau^c)g(c') + (\tau^c - \tau^{c'})g(c') + \sum_{k=c'+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= (v - \tau^{c'})g(c') + \sum_{k=c'+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= u_i(\boldsymbol{x}', \boldsymbol{p}').
\end{aligned}$$

Case 2. Player i overbids to obtain a smaller congestion $c' < c$. In this case, we have $\tau^c < v_i \leq \tau^{c-1} \leq \tau^{c'}$ and obtain (again using monotonicity of g) that

$$\begin{aligned}
u_i(\boldsymbol{x}', \boldsymbol{p}') &= (v - \tau^{c'})g(c') + \sum_{k=c'+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&\leq (v - \tau^{c'})g(c') + \sum_{k=c'+1}^{c-1} (\tau^{k-1} - \tau^k)g(c') \\
&\qquad + (\tau^{c-1} - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= (v - \tau^{c'})g(c') + (\tau^{c'} - \tau^{c-1})g(c') \\
&\qquad + (\tau^{c-1} - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= (v - \tau^{c-1})g(c') + (\tau^{c-1} - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&\leq (v - \tau^{c-1})g(c) + (\tau^{c-1} - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= (v - \tau^c)g(c) + \sum_{k=c+1}^{n} (\tau^{k-1} - \tau^k)g(k) \\
&= u_i(\boldsymbol{x}, \boldsymbol{p})
\end{aligned}$$

The second inequality in the above equation is true because by monotonicity of g and since $v \leq \tau^{c-1}$, we have $(v - \tau^{c-1})g(c') \leq (v - \tau^{c-1})g(c)$.

Concluding, player i cannot improve her utility by misreporting her valuation. This completes the proof. $\square$

6.4 Approximation via Disjoint Strategies

Section 6.3 reduces the problem of designing truthful mechanisms for bottleneck congestion games to finding smoothly monotone allocation functions. Since our goal is to design truthful mechanisms that maximize social welfare, we seek for allocation functions that approximate this objective function as well as possible. In this section, we define a generic mechanism for symmetric bottleneck congestion games. Our idea is to restrict the problem to a set of disjoint strategies for which we calculate an optimal allocation using a dynamic program. The social welfare approximation guarantee of our mechanism can be bounded using duality in hypergraphs.

In order to state our main theorem, we first define the notions of a *hitting set* and an *independent set*. A *hitting set* for the strategy set $\mathcal{S}$ is a set $H \subseteq R$ of resources such that every strategy contains at least one element of the hitting set, i.e. $S \cap H \neq \emptyset$ for all $S \in \mathcal{S}$. A *minimum* hitting set is a hitting set of minimum cardinality; we denote the size of a minimum hitting set by $\eta(\mathcal{S})$. An *independent set* I of strategies in $\mathcal{S}$ is a subset of strategies which are pairwise disjoint. A maximum independent set is one of maximum cardinality; we denote this maximum cardinality by $\iota(\mathcal{S})$. It is easy to see that $\eta \geq \iota$: Any hitting set must contain at least one resource of every strategy contained in an independent set I.

Our main result is the following theorem, which we prove in Section 6.4.1.

Theorem 6.4. *Given a c-approximation algorithm for the independent set problem, and provided that $\eta/\iota \leq r$, we derive a truthful rc-approximation for the optimal social welfare of a symmetric bottleneck congestion game with non-increasing throughput function g.*

Our generic mechanism roughly works as follows: Let $\Gamma = (U, R, \mathcal{S})$ be an instance of a symmetric bottleneck congestion game. First, we identify a maximum independent set of strategies in $\mathcal{S}$ using an approximation algorithm for the independent set problem. Let a be the cardinality of this independent set. We derive an instance $\Gamma_a = (U, [a], [a])$ of the *singleton* congestion game on a resources by keeping the same player set and introducing a disjoint strategies that each contain a single resource. We then compute an optimal allocation on Γ_a which we transfer back to the disjoint strategies in our original independent set.

6.4.1 Reduction

The proof of Theorem 6.4 relies on the following three lemmas and is presented at the end of this section.

Lemma 6.5. *Any feasible allocation for the singleton congestion game $\Gamma_{\iota(\mathcal{S})}$ on $\iota(\mathcal{S})$ resources yields a welfare equivalent allocation for the corresponding symmetric bottleneck congestion game Γ with strategy set $\mathcal{S}$.*

Proof. Let I be a maximum independent subset of $\mathcal{S}$. Consider an allocation $\boldsymbol{x}'$ for the singleton congestion game $\Gamma_{\iota(\mathcal{S})}$ on $\iota(\mathcal{S}) = |I|$ resources. We construct a welfare equivalent allocation $\boldsymbol{x}$ for Γ as follows: Identify each resource in $\Gamma_{\iota(\mathcal{S})}$ with a strategy in I in a one-to-one fashion. For all $r = 1, \ldots, \iota$, assign every player who uses resource r in $\boldsymbol{x}'$ to the corresponding strategy in I. Since all strategies in I are disjoint, the bottleneck congestion of a player in $\boldsymbol{x}$ equals her congestion in the original allocation $\boldsymbol{x}'$. Hence, the social welfare is equal in both allocations. □

The following lemma states that any allocation for a singleton congestion game yields an allocation for the corresponding singleton congestion game on fewer resources such that the social welfare only decreases by the ratio of the two different numbers of resources.

Lemma 6.6. *Let U be a set of players and let $0 < a \leq b$ be integers. Any allocation for the singleton congestion game $\Gamma_b = (U, [b], [b])$ yields an allocation for the singleton congestion game $\Gamma_a = (U, [a], [a])$ which achieves at least an a/b fraction of the social welfare of the former.*

Proof. Pick an allocation $\boldsymbol{x}$ for Γ_b. Choose the subset of a resources which contribute the most welfare. We define an allocation $\boldsymbol{x}'$ for Γ_a by assigning the same players to these resources as before, and not serving the rest of the players. By construction, we have $W(\boldsymbol{x}') \geq a/b \cdot W(\boldsymbol{x})$. □

Lemma 6.7. *Given that g is non-increasing, any feasible allocation to a symmetric bottleneck congestion game Γ yields a feasible allocation with better or equal social welfare for the corresponding singleton congestion game $\Gamma_{\eta(\mathcal{S})}$ on $\eta(\mathcal{S})$ resources.*

Proof. Let $H \subseteq R$ be a minimum hitting set for the strategy set $\mathcal{S}$. For every player $i \in U$ and strategy $S \in \mathcal{S}$, we choose an arbitrary resource $r_i(S) \in S \cap H$ and call it the *representative* of S for player i. This is possible since H is a hitting set for $\mathcal{S}$ and thus $S \cap H \neq \emptyset$ for all $S \in \mathcal{S}$. For every resource $r \in H$ and player

$i \in U$, let $P_i(r) := \{S \in \mathcal{S} \mid r = r_i(S)\}$ denote the set of strategies for which r is the representative for player i.

Consider a feasible allocation $\boldsymbol{x}$ for Γ. We define an allocation $\boldsymbol{x}'$ for the singleton congestion game $\Gamma_\eta(\mathcal{S})$ on $\eta(\mathcal{S}) = |H|$ resources which achieves better or equal social welfare. Identify the resources of Γ_η with those in the hitting set H in a one-to-one fashion. We define $\boldsymbol{x}'$ by assigning every player who uses a strategy in $\boldsymbol{x}$ to her representative resource of this strategy in Γ_η. More formally, for every $i \in U$ and $r \in H$, we define $x'_{ir} := \sum_{S \in P_i(r)} x_{iS}$. Note that we have

$$x'_{ir} = \sum_{S \in P_i(r)} x_{iS} \leq \sum_{S \in \mathcal{S}: r \in S} x_{iS} = x_{ir}.$$

Thus $x'_r \leq x_r$ for every $r \in H$. The bottleneck congestion of player i in $\boldsymbol{x}'$ is

$$c_i(\boldsymbol{x}') = \max_{r \in H: x'_{ir} > 0} x'_r \leq \max_{r \in H: x'_{ir} > 0} x_r \leq \max_{r \in H: x_{ir} > 0} x_r \leq \max_{r \in R: x_{ir} > 0} x_r = c_i(\boldsymbol{x}).$$

Since g is non-increasing, we conclude that $\sum_{i \in U} w_i(\boldsymbol{x}') \geq \sum_{i \in U} w_i(\boldsymbol{x})$. □

We are now ready to prove Theorem 6.4.

Proof of Theorem 6.4. We compute an approximately maximum independent set I of the strategy set $\mathcal{S}$ with the given c-approximation algorithm. Let $a := |I|$ denote the cardinality of I. By the approximation factor and since $\eta/\iota \leq r$ by assumption, we have $a \geq \iota/c \geq \eta/(rc)$.

We compute an optimal allocation $\boldsymbol{x^a}$ to the corresponding singleton congestion game Γ_a on a resources using Algorithm 5 presented in Section 6.4.2. We transfer this solution to the original problem Γ by assigning every player to the corresponding strategy in I as in the proof of Lemma 6.5. Let $\boldsymbol{x}$ be the resulting allocation. For any congestion game Γ, we denote by $W_{\text{OPT}}(\Gamma)$ the maximum social welfare obtainable for Γ.

By Lemma 6.5 and optimality of $\boldsymbol{x^a}$, we have

$$W(\boldsymbol{x}) = W(\boldsymbol{x^a}) = W_{\text{OPT}}(\Gamma_a).$$

Applying Lemma 6.6 with $b := \eta \geq \iota \geq a$ and plugging in $a/\eta \geq 1/(rc)$ as derived above implies that

$$W_{\text{OPT}}(\Gamma_a) \geq \frac{a}{\eta} \cdot W_{\text{OPT}}(\Gamma_\eta) \geq \frac{1}{rc} \cdot W_{\text{OPT}}(\Gamma_\eta)$$

for the optimal social welfare $W_{\text{OPT}}(\Gamma_\eta)$ obtainable in the corresponding singleton game on η resources.

Putting both equations together and employing Lemma 6.7 yields

$$W(\boldsymbol{x}) \geq \frac{1}{rc} \cdot W_{\text{OPT}}(\Gamma_\eta) \geq \frac{1}{rc} \cdot W_{\text{OPT}}(\Gamma).$$

Hence, $\boldsymbol{x}$ is an rc-approximation for social welfare of the original problem Γ.

To verify that this mechanism is strategyproof, note that the choice of the independent set I does not depend on the bids reported by the players. In fact, the bids are only taken into account by Algorithm 5. Further, the proof of Lemma 6.5 shows that the congestion experienced by a player, and thus her utility, does not change when the allocation is transferred to the original instance. Hence, truthfulness is inherited from that of Algorithm 5, which we prove in Lemma 6.9. □

6.4.2 Optimal Mechanism for Singleton Congestion Games

In this section, we define a smoothly monotone allocation rule for singleton congestion games that maximizes social welfare. The allocation is computed using a dynamic program. The basic idea for our approach relies on the following simple observation.

Consider a singleton congestion game $\Gamma_m = (U, [m], [m])$ with a set of players U and m resources. W.l.o.g. we assume that the players are numbered such that $v_1 \geq v_2 \geq \ldots \geq v_n$. Now, assume that we are given the congestion values $x_1 \leq \ldots \leq x_m$ of all resources corresponding to an optimal allocation of players in U. By monotonicity of g, we know that the least congested resource has the most throughput and so on. Hence, since all players have unit demands and are thus indistinguishable except for their valuations, it is easy to reconstruct an optimal allocation from the given values $x_1 \leq \ldots \leq x_m$. We simply allocate the highest x_1 bidders to the least congested resource, the next x_2 highest bidders to the second least congested resource, and so on, until all resources have their designated congestion values.

Let $\textsc{opt}(k, l)$ denote the optimal social welfare obtainable by assigning the k highest-valued bidders to l resources. Our dynamic program determines the value $\textsc{opt}(k, l)$ by optimizing over the number of players assigned to the lth strategy. Given this number j, $\textsc{opt}(k, l)$ is easily calculated as the optimal welfare $\textsc{opt}(k - j, l - 1)$ of assigning the first $k - j$ players to $l - 1$ strategies plus the welfare contributed by strategy l when used by players $k - j + 1$ through k. Given all values of $\textsc{opt}(k, l)$ for $k \leq n$ and $l \leq m$, the optimal social welfare of the singleton congestion game Γ_m is the largest number in the table. Given this number, the algorithm recapitulates which congestion values led to the optimal social welfare and assigns the players as described above. A formal definition of our dynamic program is given in Algorithm 5.

Lemma 6.8. *Algorithm 5 maximizes social welfare.*

Proof. We already argued that a vector of congestion values for all resources de-

Algorithm 5: Dynamic Program for Singleton Congestion Games.

```
Input: $n$ players with values $v_1 \geq v_2 \geq \ldots \geq v_n$, $m$ resources
Output: optimal allocation $\boldsymbol{x}$
1  Initialize $\textsc{opt}(k,1) := \sum_{i=1}^{k} g(k)v_i$, $\boldsymbol{x} = \mathbf{0}$
2  for $l = 2$ to $m$ do
3      for $k = l$ to $n$ do
4          $\textsc{opt}(k,l) := \max_{j<k}(\textsc{opt}(k-j, l-1) + \sum_{i=k-j+1}^{k} g(j)v_i)$
5          $\textsc{pred}(k,l) := \arg\max_{j<k}(\textsc{opt}(k-j, l-1) + \sum_{i=k-j+1}^{k} g(j)v_i)$
6      end
7  end
8  Let $(n^*, m^*) \in \{\arg\max_{k \leq n, l \leq m} \textsc{opt}(k,l)\}$ with $(n^*, m^*)$ lexicographically
   minimal.
9  Set $k := n^*$
10 for $l = m^*$ to 1 do
11     $x_l := \textsc{pred}(k,l)$
12     $x_{il} := 1$ for all players $i = k - x_l + 1, \ldots, k$
13     $k = k - x_l$
14 end
15 Return allocation $\boldsymbol{x}$
```

fines an optimal allocation by allocating the highest bidders to the least congested resources. Algorithm 5 optimizes over all such allocations defined by possible congestion vectors and thus returns an allocation of maximum social welfare. ☐

Lemma 6.9. *Algorithm 5 is smoothly monotone.*

Proof. Let $\boldsymbol{x} := \boldsymbol{x}(\boldsymbol{v})$ be the allocation output by Algorithm 5 when it receives the valuations $\boldsymbol{v}$ as input. We need to show that no winner in $\boldsymbol{x}$ can experience more congestion or even lose by raising her bid. Assume for contradiction that player i achieves this by reporting a valuation of $v_i + \epsilon$ for some $\epsilon > 0$. Let $\boldsymbol{x}' := \boldsymbol{x}(v_i + \epsilon, v_{-i})$ denote the resulting allocation output by Algorithm 5.

For ease of notation, we denote by $W(\boldsymbol{y}, t)$ the social welfare of allocation $\boldsymbol{y} \in \{\boldsymbol{x}, \boldsymbol{x}'\}$ when player i's true value is t, i.e. $W(\boldsymbol{y}, t) := \sum_{j \neq i} w_j(\boldsymbol{y}) + tg(c_i(\boldsymbol{y}))$ if she wins and $W(\boldsymbol{y}, t) := \sum_{j \neq i} w_j(\boldsymbol{y})$ if she loses in $\boldsymbol{y}$.

First, assume that player i loses in $\boldsymbol{x}'$. By optimality of $\boldsymbol{x}$, we have $W(\boldsymbol{x}, v_i) \geq W(\boldsymbol{x}', v_i)$. Further, $W(\boldsymbol{x}', v_i + \epsilon) = W(\boldsymbol{x}', v_i)$ since i loses in $\boldsymbol{x}'$ and $W(\boldsymbol{x}, v_i + \epsilon) > W(\boldsymbol{x}, v_i)$ since i wins in $\boldsymbol{x}$ and $g(c_i(\boldsymbol{x})) > 0$ by minimality of n^* in Line 8 of Algorithm 5. Together, we obtain

$$W(\boldsymbol{x}, v_i + \epsilon) > W(\boldsymbol{x}, v_i) \geq W(\boldsymbol{x}', v_i) = W(\boldsymbol{x}', v_i + \epsilon),$$

a contradiction to the optimality of $\boldsymbol{x}'$.

Next, assume that player i experiences more congestion in $\boldsymbol{x}'$. Let $c := c_i(\boldsymbol{x}) < c' := c_i(\boldsymbol{x}')$ be her respective experienced bottleneck congestions. As before, $W(\boldsymbol{x}, v_i) \geq W(\boldsymbol{x}', v_i)$ by optimality of $\boldsymbol{x}$. Further, $W(\boldsymbol{x}, v_i + \epsilon) = \epsilon g(c) + W(\boldsymbol{x}, v_i)$ and $W(\boldsymbol{x}', v_i + \epsilon) = \epsilon g(c') + W(\boldsymbol{x}', v_i)$. By monotonicity of g, we obtain

$$W(\boldsymbol{x}, v_i + \epsilon) = \epsilon g(c) + W(\boldsymbol{x}, v_i) \geq \epsilon g(c') + W(\boldsymbol{x}', v_i) = W(\boldsymbol{x}', v_i + \epsilon),$$

which contradicts the optimality of $\boldsymbol{x}'$ unless $W(\boldsymbol{x}, v_i + \epsilon) = W(\boldsymbol{x}', v_i + \epsilon)$. In this case, we can conclude from lexicographical minimality of (n^*, m^*) that $\boldsymbol{x}' = \boldsymbol{x}$ and thus $c = c'$. Hence, player i cannot increase her congestion by overbidding. □

6.5 Implications and Applications

6.5.1 Network Congestion Games

Our first implied result is a truthful polynomial-time approximation mechanism for maximizing the social welfare of *single-commodity network congestion games* with the bottleneck metric. In the network setting, resources are the edges of a given directed graph $G = (V, E)$. Each agent wishes to route one unit of demand along a path between a source and a sink. In the *symmetric* (or *single-commodity*) case, all players have the same common source s and sink t. Accordingly, the set of strategies $\mathcal{S}$ is defined as the set of all s, t-paths in G.

Due to the min-cut max-flow equality (see e.g. [61]), the results of Section 6.4 yield particularly nice results which we summarize in the following theorem.

Theorem 6.10. *Algorithm 5 defines a truthful mechanism for single-commodity network congestion games which maximizes social welfare in the bottleneck setting.*

Proof. We apply Theorem 6.4. By the min-cut max-flow equality, we have $r = 1$. Furthermore, the maximum independent set problem corresponds to the single-commodity maximum flow problem which can be solved in polynomial time, thus $c = 1$. □

6.5.2 Hypergraph Models

We can model the strategy set $\mathcal{S}$ by a hypergraph in two different ways, and obtain different upper bounds the ratio between η and ι.

1. In the *vertex-strategy model*, every strategy is represented by a vertex, and

for every resource $r \in R$, we introduce a hyperedge which contains all vertices corresponding to strategies containing r.

2. Controversely, in the *edge-strategy model*, we identify with every resource $r \in R$ a vertex, and define a hyperedge $e = S$ for every strategy $S \in \mathcal{S}$.

In the following, we discuss some bounds that we can obtain from existing literature on hypergraphs for different classes of congestion games. Using Theorem 6.4, these bounds yield strategyproof mechanisms that approximate the social cost for the respective classes of congestion games.

The Vertex-Strategy Model.

In the vertex-strategy model, we make use of the following theorem by Berger and Ziv [8] to upper bound the size of a minimum hitting set of resources:

Theorem 6.11 ([8]). *Consider a hypergraph with m edges whose maximal edge cardinality (i.e.* rank*) is at most $r \geq 3$. Let α denote the maximal cardinality of an independent set of vertices. Then, the size of a minimum edge cover is at most*

$$\rho \leq \frac{(r-2)m + \alpha}{r-1}.$$

Here, the rank of the modeled hypergraph is the maximal cardinality of an edge, i.e. a resource. Hence, the theorem conditions on the fact that the maximum *frequency* of a resource is upper bounded by r, and the number of resources is m. Further, an edge cover corresponds to a hitting set of resources, and an independent set of vertices corresponds to an independent set of strategies. Hence, the theorem gives us that

$$\eta \leq \frac{(r-2)m + \iota}{r-1}.$$

Especially, if every resource is constrained to belong to at most 3 strategies (i.e. $r = 3$), we obtain $\eta \leq \frac{m+\iota}{2}$.

The Edge-Strategy Model.

In the edge-strategy model, a minimum hitting set corresponds to a minimum vertex cover, and a maximum independent set corresponds to a maximum matching of hyperedges.

It is well known that in case every edge has cardinality two, i.e. when the hypergraph is a *graph*, the ratio between the cardinalities of a maximum matching and a minimum vertex cover is at most two. Thus, we have $\eta/\iota \leq 2$ if every strategy has cardinality two. Since there exist polynomial-time algorithms for finding a maximum matching in a graph, Theorem 6.4 together with Lemma 6.6 yield the following corollary:

Corollary 6.12. *There is a truthful* 2*-approximate mechanism for symmetric cardinality two bottleneck congestion games with monotone throughput function* g*.*

Further results on the maximum ratio between the cardinalities of a maximum matching and a minimum vertex cover in hypergraphs for many different graph classes can be found in the graph theory literature.

6.6 Hardness Results

6.6.1 Symmetric Bottleneck Congestion Games

Finding the social optimum of symmetric bottleneck congestion games with monotone throughput functions is NP-complete, even when every resource is contained in exactly two strategies. The proof is by reduction from the Independent Set problem for graphs.

In the Independent Set problem for graphs, we are given a graph $G = (V, E)$ and a positive integer K. A set of vertices $V' \subseteq V$ is called an *independent set* if for every pair of vertices $u, v \in V'$, the edge (u, v) is *not* in E. The question is whether G contains an independent set V' of cardinality at least K. Independent Set is known to be NP-complete (see [37]).

Theorem 6.13. *Finding the social optimum of symmetric bottleneck congestion games is NP-complete, even if every resource is contained in exactly two strategies.*

Proof. We reduce Independent Set to our setting. Let $G = (V, E)$ be the graph in the given independent set instance. Define a resource for every edge $e \in E$, and a strategy for every vertex $v \in V$ containing exactly those resources that are identified with an edge $e \in \delta(v)$, as in the vertex-strategy model introduced in Section 6.5.2. Introduce $|V|$ players with equal private values $v_i = 1$ and define $g(1) = 1$ and $g(k) = 0$ for all $k \geq 2$. The maximum social welfare obtainable in this congestion game instance equals the number of disjoint strategies, which correspond to independent vertices in G. Thus, an allocation of social welfare K or greater immediately reveals an independent set of the same size in G and vice versa. Thus, NP-completeness follows from that of Independent Set. $\square$

Since the above reduction conserves objective function values, it also leads to various inapproximability results as a consequence of results for independent set problems in graphs or hypergraphs (replace "edge" by "hyperedge" in the proof of Theorem 6.13 to obtain the reduction for hypergraphs). We give two examples below.

In the MAX CLIQUE problem, we are given a graph $G = (V, E)$. A set of vertices $V' \subseteq V$ is called a *clique* if for every pair of vertices $u, v \in V'$, the edge (u, v) is in E. The problem is to find the largest integer K such that G contains a clique of cardinality K. Note that a clique in $G = (V, E)$ is an independent set in the complementary graph $\overline{G} = (V, \overline{E})$, where $\overline{E} := (V \times V) \setminus E$ contains an edge between each pair of vertices that is not adjacent in G. For general graphs, the MAX CLIQUE problem is thus equivalent to the maximum independent set problem in graphs.

Proposition 6.14. *It is NP-hard to approximate the social welfare of symmetric bottleneck congestion games by a factor of*

- $|\mathcal{S}|^{1-\epsilon}$ *by hardness of approximation of the max clique problem in graphs [31].*
- $B/2^{O(\sqrt{\log B})}$ *if the cardinality of each strategy is at most B [97].*

6.6.2 Matroid Bottleneck Congestion Games

For the much more restricted class of matroid congestion games, the problem of finding a social optimum remains NP-hard at least when one refrains from restricting the throughput function g to be monotone. The proof is in the spirit of a similar proof in [1] for classical congestion games, where the payoff of a player is defined as the sum of payoffs over all resources that she uses.

In a matroid congestion game, the strategy set of a player is defined as the set of bases of a matroid. See Section 2.6 for a formal definition of matroids and matroid congestion games. In the special case of a *cycle* matroid, the resource set is the set of edges of a given graph G, and an independent set is a forest in G. In the corresponding congestion game, the strategy set of each player is thus the set of all spanning trees of G.

Theorem 6.15. *Finding the social optimum of symmetric bottleneck congestion games with arbitrary throughput functions whose strategy space is the set of bases of a cyclic matroid is NP-complete.*

Proof. The proof is by reduction from the HAMILTONIAN CYCLE problem, in which we are given a graph $G = (V, E)$ and the question is whether G contains a simple cycle which contains all vertices in V. HAMILTONIAN CYCLE is NP-complete as shown in [37].

We define a cycle matroid congestion game on G by introducing $n = |V|$ players with equal private values $v_i = 1$ and setting $g(n-1) := 1$ and $g(k) := 0$ for all $k \neq n-1$. Observe that there is an allocation with social welfare n if the graph has a Hamiltonian cycle: Assign each player to one of the trees obtained by deleting one edge of the Hamiltonian cycle. On the other hand, if there is an allocation of social welfare n, every player's throughput must be 1, so every used edge needs to be used by exactly $n-1$ players. Then, as argued in [1], the allocation defines a Hamiltonian cycle in G. $\square$

6.7 Conclusion

In this chapter, we studied a new model for incorporating negative externalities caused by congestion of resources in a mechanism design setting. We achieved this by employing standard models used in the area of *congestion games*. Instead of allowing each player to choose her strategy, we requested a bid from every player and studied the problem of finding socially optimal allocations that give each player the incentive to report her true valuation and thus support the imposed allocation.

Externalities describe an important effect that arises in many real world applications and is not yet considered sufficiently in game theoretic models such as mechanism design. We describe a very natural model for incorporating externalities which is relevant for many applications. Our model entails numerous intriguing research questions, among which are to extend the model to cooperative settings and/or cost sharing problems. Our contributions in this chapter form a starting point for this promising line of research.

BIBLIOGRAPHY

[1] H. Ackermann, H. Röglin, and B. Vöcking. On the impact of combinatorial structure on congestion games. *Journal of the ACM*, 55(6):1–22, 2008. 37, 118, 119

[2] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Symposium on the Foundations of Computer Science*, pages 32–43, 1999. 33

[3] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and collusion in multicast cost sharing. *Games and Economic Behavior*, 47(1):36–71, 2004. 27

[4] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd Symposium on the Foundations of Computer Science*, pages 482–491. IEEE Computer Society, 2001. 40

[5] R. Banner and A. Orda. Bottleneck routing games in communication networks. *IEEE Journal on Selected Areas in Communications*, 25(6):1173–1179, 2007. 37

[6] N. Bansal, A. Blum, S. Chawla, and K. Dhamdhere. Scheduling for flow-time with admission control. In *In Proceedings of the 11th Annual European Symposium on Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 43–54. Springer, 2003. 34

[7] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, and L. Stougie. Multiprocessor scheduling with rejection. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 95–103, 1996. 33

[8] E. Berger and R. Ziv. A note on the edge cover number and independence number in hypergraphs. *Discrete Mathematics*, 308(12):2649–2654, 2008. 116

[9] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006. 13

[10] Y. Bleischwitz and B. Monien. Fair cost-sharing methods for scheduling jobs on parallel machines. In *Proceedings of the 6th International Conference on Algorithms and Complexity*, volume 3998 of *Lecture Notes in Computer Science*, pages 175–186, 2006. 4, 27, 28, 44, 45, 46, 56, 57, 69

[11] Y. Bleischwitz and B. Monien. Fair cost-sharing methods for scheduling jobs on parallel machines. *Journal of Discrete Algorithms*, 7(3):280–290, 2009. 41, 47

[12] Y. Bleischwitz, B. Monien, and F. Schoppmann. To be or not to be (served). In *Proceedings of the 3rd International Workshop on Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*, pages 515–528, 2007. 27, 30, 56, 70, 82

[13] Y. Bleischwitz, B. Monien, F. Schoppmann, and K. Tiemann. The power of two prices: Beyond cross-monotonicity. In *Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science*, volume 4708 of *Lecture Notes in Computer Science*, pages 657–668, 2007. 28

[14] Y. Bleischwitz and F. Schoppmann. Group-strategyproof cost sharing for metric fault tolerant facility location. In *Proceedings of the 1st International Symposium on Algorithmic Game Theory*, volume 4997 of *Lecture Notes in Computer Science*, pages 350–361, 2008. 22, 88, 92

[15] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998. 88, 90

[16] J. Brenner and G. Schäfer. Cost sharing methods for makespan and completion time scheduling. In *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science*, volume 4393 of *Lecture Notes in Computer Science*, pages 670–681, 2007. 4, 28, 39, 56, 57, 58, 70, 71

[17] J. Brenner and G. Schäfer. Group-strategyproof cost sharing mechanisms for makespan and other scheduling problems. *Theoretical Computer Science*, 401(1–3):96–106, 2008. 39

[18] J. Brenner and G. Schäfer. Singleton acyclic mechanisms and their applications to scheduling problems. In *Proceedings of the 1st International Symposium on Algorithmic Game Theory*, volume 4997 of *Lecture Notes in Computer Science*, pages 315–326, 2008. 55, 59, 82, 99

[19] J. Brenner and G. Schäfer. Online cooperative cost sharing. In *Proceedings of the 7th International Conference on Algorithms and Complexity*, volume 6078 of *Lecture Notes in Computer Science*, pages 252–263, 2010. 87

[20] I. Brocas. Auctions with type dependent and negative externalities: the optimal mechanism, November 2007. IEPR working paper. 103

[21] P. Brucker. *Scheduling Algorithms*. Springer, New York, USA, 2004. 33

[22] D. Bunde. Scheduling on a single machine to minimize total flow time with job rejections. In *Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory & Applications*, pages 562–572, 2005. 34

[23] C. Busch and M. Magdon-Ismail. Atomic routing games on maximum congestion. In *Proceedings of the 2nd International Conference on Algorithmic Aspects in Information and Management*, volume 4041 of *Lecture Notes in Computer Science*, pages 79–91, 2006. 37

[24] D. Chakrabarty, A. Mehta, V. Nagarajan, and V. Vazirani. Fairness and optimality in congestion games. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005. 103

[25] S. Chawla, T. Roughgarden, and M. Sundararajan. Optimal cost-sharing mechanisms for Steiner forest problems. In *Proceedings of the 2nd International Workshop on Internet and Network Economics*, pages 112–123, 2006. 4, 28, 53, 56

[26] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971. 14

[27] R. Cole, Y. Dodis, and T. Roughgarden. Bottleneck links, variable demand and the tragedy of the commons. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 2006. 35, 36

[28] N. Devanur, M. Mihail, and V. Vazirani. Strategyproof cost-sharing mechanisms for set cover and facility location games. In *Proceedings of the ACM Conference on Electronic Commerce*, 2003. 4, 21

[29] N. Devanur, M. Mihail, and V. Vazirani. Strategyproof cost-sharing mechanisms for set cover and facility location games. *Decision Support Systems*, 39(1):11–22, 2005. 22, 88

[30] S. Dobzinski, A. Mehta, T. Roughgarden, and M. Sundararajan. Is Shapley cost sharing optimal? In *Proceedings of the 1st International Symposium on Algorithmic Game Theory*, volume 4997 of *Lecture Notes in Computer Science*, pages 327–336, 2008. 22, 71

[31] L. Engebretsen and J. Holmerin. Clique is hard to approximate within $n^{1-o(1)}$. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 2–12. Springer-Verlag, 2000. 118

[32] D. Engels, D. Karger, S. Kolliopoulos, S. Sengupta, R. Uma, and J. Wein. Techniques for scheduling with rejection. *Journal of Algorithms*, 49:175–191, 2003. 33

[33] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Hardness results for multicast cost-sharing. *Theoretical Computer Science*, 304:215–236, 2003. 21, 27

[34] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001. 21, 27

[35] M. Gairing and F. Schoppmann. Total latency in singleton congestion games. In *Proceedings of the 3rd International Workshop on Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*, pages 381–387, 2007. 36

[36] M. R. Garey and D. S. Johnson. Strong NP-completeness results: Motivation, examples and implications. *Journal of the ACM*, 25(3):499–508, 1978. 33, 69

[37] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979. 34, 35, 117, 118

[38] A. Ghosh and M. Mahdian. Externalities in online advertising. In *17th international World Wide Web Conference*, 2008. 103

[39] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973. 12

[40] R. Graham, E. Lawler, J. Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979. 32

[41] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969. 33, 69

[42] J. Green, E. Kohlberg, and J. J. Laffont. Partial equilibrium approach to the free rider problem. *Journal of Public Economics*, 6:375–394, 1976. 21

[43] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973. 14

[44] A. Gupta, M. Hajiaghayi, and H. Räcke. Oblivious network design. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 970–979, 2006. 97

[45] A. Gupta, J. Könemann, S. Leonardi, R. Ravi, and G. Schäfer. An efficient cost-sharing mechanism for the prize-collecting Steiner forest problem. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1153–1162, 2007. 4, 27, 28, 53

[46] A. Gupta, A. Srinivasan, and É. Tardos. Cost-sharing mechanisms for network design. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2004. 4, 27

[47] T. Harks, S. Heinz, and M. Pfetsch. Competitive online multicommodity routing. *Theory of Computing Systems*, 2008. 99

[48] B. Heydenreich, R. Müller, and M. Uetz. Decentralization and mechanism design for online machine scheduling. In *Proceedings of the 1st International Workshop on Computational Social Choice*, pages 136–147. Springer, 2006. 40

[49] D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1997. 47

[50] D. Hochbaum and D. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987. 33

[51] M. Imase and B. Waxman. Dynamic Steiner tree problems. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991. 97

[52] N. Immorlica, M. Mahdian, and V. S. Mirrokni. Limitations of cross-monotonic cost sharing schemes. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 602–611, 2005. 4, 27, 28, 56

[53] N. Immorlica, M. Mahdian, and V. S. Mirrokni. Limitations of cross-monotonic cost-sharing schemes. *ACM Transactions on Algorithms*, 4(2):1–25, 2008. 89

[54] K. Jain and V. Vazirani. Applications of approximation algorithms to cooperative games. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 364–372, 2001. 4, 26, 27, 69

[55] P. Jehiel and B. Moldovanu. Efficient design with interdependent valuations. *Econometrica*, 69(5):1237–59, 2001. 103

[56] R. Juarez. Group strategyproof cost sharing. In *Unpublished*, 2008. 24, 28, 30, 31, 89, 90

[57] M. Kamien, S. Oren, and Y. Tauman. Optimal licensing of cost reducing innovation. *Journal of Mathematical Economics 21*, pages 483–508, 1992. 103

[58] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow time problem. *SIAM Journal on Computing*, 15(4):1119–1129, 1986. 70

[59] K. Kent and D. Skorin-Kapov. Population monotonic cost allocations on MSTs. In *Proceedings of the 6th International Conference on Operational Research*, pages 43–48. Croatian Operations Research Society, Zagreb, 1996. 27, 69

[60] J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam. From primal-dual to cost shares and back: a stronger LP relaxation for the Steiner forest problem. In *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 930–942. Springer, 2005. 4, 27, 28, 56

[61] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer, 2000. 34, 35, 37, 115

[62] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999. 36, 37

[63] A. Kovacs. Fast monotone 3-approximation algorithm for scheduling related machines. In *Proceedings of the 13th Annual European Symposium on Algorithms*, Lecture Notes in Computer Science, pages 616–627. Springer, 2005. 40

[64] S. Leonardi and G. Schäfer. Cross-monotonic cost sharing methods for connected facility location games. *Theoretical Computer Science*, 326(1-3):431–442, 2004. 4, 27

[65] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic theory.* Oxford University Press, 1995. 8, 12

[66] V. Mazalov, B. Monien, F. Schoppmann, and K. Tiemann. Wardrop equilibria and price of stability for bottleneck games with splittable traffic. In *Proceedings of the 2nd International Workshop on Internet and Network Economics*, volume 4286 of *Lecture Notes in Computer Science*, pages 331–342, 2006. 37

[67] R. McNaughton. Scheduling with deadlines and loss functions. *Management Sciences*, 6:1–12, 1959. 33

[68] A. Mehta, T. Roughgarden, and M. Sundararajan. Beyond Moulin mechanisms. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, 2007. 21, 69, 83

[69] A. Mehta, T. Roughgarden, and M. Sundararajan. Beyond Moulin mechanisms. *Games and Economic Behavior*, 67(1):125–155, 2009. 4, 22, 24, 28, 29, 30, 55, 81, 88, 92

[70] C. Meyers. *Network Flow Problems and Congestion Games: Complexity and Approximation Results*. PhD thesis, Massachusetts Institute of Technology, 2006. 103

[71] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996. 36

[72] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996. 36, 104

[73] H. Moulin. Incremental cost sharing: Characterization by coalition strategy-proofness. *Social Choice and Welfare*, 16:279–320, 1999. 3, 22, 24, 28, 30, 55, 59, 88, 89, 94

[74] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Economic Theory*, 18(3):511–533, 2001. 25, 26, 27, 40

[75] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981. 13

[76] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 129–140, 1999. 102

[77] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games Economic Behavior*, 35(1-2):166–196, 2001. 40

[78] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007. 8, 10, 11, 17

[79] M. Osborne and A. Rubinstein, editors. *A course in game theory*. MIT Press, 1994. 17, 18, 35

[80] M. Pál and É. Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Symposium on Foundations of Computer Science*, pages 584–593, 2003. 4, 27

[81] C. Papadimitriou. Algorithms, games and the internet. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 749–752, 2001. 102

[82] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199–223, 1998. 33, 77

[83] R. Porter. Mechanism design for online real-time scheduling. In *Proceedings of the ACM Conference on Electronic Commerce*. ACM Press, 2004. 40

[84] E. Pountourakis and A. Vidali. A complete characterization of group-strategyproof mechanisms of cost-sharing. In *Unpublished*, 2010. 28

[85] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957. 64

[86] K. Roberts. The characterization of implementable choice rules. In J. J. Laffont, editor, *Aggregation and Revelation of Preferences*. North-Holland, 1979. 15, 21

[87] R. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973. 35, 104

[88] T. Roughgarden and M. Sundararajan. New trade-offs in cost-sharing mechanisms. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 79–88, 2006. 4, 21, 22, 27, 28, 40, 41, 42, 43, 53, 56

[89] T. Roughgarden and M. Sundararajan. Optimal efficiency guarantees for network design mechanisms. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization*, pages 469–483, 2007. 4, 28, 43, 53, 56, 69

[90] T. Roughgarden and E. Tardos. How bad is selfish routing? In *Proceedings of the 41st Symposium on Foundations of Computer Science*, 2000. 37, 102

[91] M. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 286–293, New York, NY, USA, 2005. ACM. 13

[92] M. Salek and D. Kempe. Auctions for share-averse bidders. In *Procedings of the 4th International Workshop on Internet and Network Economics*, volume 5385 of *Lecture Notes in Computer Science*, pages 609–620, 2008. 104

[93] L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968. 33, 71, 76, 98

[94] B. Simons. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM Journal on Computing*, 12(2):294–299, 1983. 33

[95] R. Sitters. Efficient algorithms for average completion time scheduling. In *Proceedings of the 14th International Conference on Integer Programming and Combinatorial Optimization*, volume 6080 of *Lecture Notes in Computer Science*, pages 411–423, 2010. 33, 71, 81, 98

[96] W. Smith. Various optimizers for single-stage production. In *Naval Research Logistics Quarterly*, volume 3, pages 59–66, 1956. 33, 50, 70

[97] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 453–461, 2001. 118

[98] V. Vazirani. *Approximation Algorithms.* Springer, 2004. 68

[99] W. Vickrey. Counterspeculations, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961. 14

[100] T. Voice, M. Polukarov, A. Byde, and N. R. Jennings. On the impact of strategy and utility structures on congestion-averse games. In *Procedings of the 5th International Workshop on Internet and Network Economics*, pages 600–607, 2009. 37