Graphs, Mechanisms and Scheduling

This book was typeset by the author using LaTeX.

Graphs, Mechanisms and Scheduling

Printed by PrintPartners Ipskamp
Cover design by Logowerk Berlin

Graphs, Mechanisms and Scheduling

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit Maastricht,
op gezag van de Rector Magnificus, Prof. mr. G.P.M.F.Mols,
volgens het besluit van het College van Decanen,
in het openbaar te verdedigen
op woensdag 22 april 2009 om 16.00 uur

door

Birgit Heydenreich

Promotores:
Prof. dr. R.J. Müller
Prof. dr. M.J. Uetz (Universiteit Twente)

Beoordelingscommissie:
Prof. dr. H.J.M. Peters (voorzitter)
Prof. dr. B. Moldovanu (University of Bonn)
Dr. T. Vredeveld

Het in dit proefschrift beschreven onderzoek werd financieel mede mogelijk gemaakt door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO).

Contents

Chapter 1

Introduction

Traditional optimization deals with the efficient selection of an optimal solution to a minimization or maximization problem. It is usually assumed that a central planner has full access to the model and all the involved data. For example, he has to schedule a number of tasks on several machines in a production process such that the time needed to finish all the tasks is minimized. Or he has to determine the best way to transport a number of goods from different locations to different destinations along a road network. In contrast, in distributed settings, there are several agents, possibly equipped with private information that is not publicly known, and these agents need to interact in order to derive a solution to the problem. This happens e.g. on the internet, where various selfish users send their data packages along the different communication links in the network. Or in daily traffic, when each driver tries to find the fastest route for himself and does not care about the other drivers. In such settings, the agents have incentives for strategic behavior, possibly leading to sub-optimal system performance. The analysis of such distributed settings requires techniques from classical optimization as well as techniques from game theory and economic theory.

Motivated by this insight, more and more researchers from the "classical optimization communities" as computer science and operations research have recently become interested in game theory and economic theory. In fact, numerous conferences and publications are devoted to the intersection of the mentioned disciplines. Contributions are in both directions. While the consideration of strategic behavior of selfishly behaving agents has been introduced into classical optimization problems, notions as efficient computation or the price of anarchy have found their way into game theory.

For illustration, consider the task scheduling model regarded in Nisan and Ronen (2001). In the classical optimization setting, there are a number of machines that

have to process a set of tasks. The processing time of each of those tasks may differ among the machines. The goal is to assign the tasks to the machines in order to minimize the makespan, i.e., the completion time of the last task. A typical result from computer science says that the problem is computationally hard, i.e., finding an optimal schedule for the problem may cost a huge amount of computation time. A traditional computer science answer to this challenge would be an approximation algorithm, that is, an algorithm that efficiently finds a solution that is not too far away from the optimum. In a distributed setting, machines might be selfish agents who are privately informed about the processing time they need for each task and they might be interested to get as few workload as possible. This gives rise to incentives and strategic behavior, since machines can manipulate the resulting schedule in their favor by misreporting about their private information. Mechanism design deals with this kind of problems. Usually, strategic payments are introduced that make it beneficial for the machines to truthfully report about their private information. A question that comes up in the new research field at the border between computer science and mechanism design is how close one can come to the optimal makespan under the additional constraint that machines must not have incentives to misreport about their private information. An answer given in Nisan and Ronen (2001) is that there is no such truthful mechanism that can guarantee a makespan that is at most twice the optimal makespan.

Another illustrative example can be found in the selfish routing model considered in Koutsoupias and Papadimitriou (1999). Selfish agents have to route a particular amount of traffic along parallel links in a congested network. The encountered delay depends on the congestion of the used network links. In classical optimization, a central planner would assign the traffic to the links, therefore obtaining the solution with the least maximum delay. In game theory, however, selfish agents are often assumed to behave according to a Nash equilibrium. In such a Nash equilibrium, the resulting traffic might have a maximum delay that is larger than the one that the central planner would have derived. A question that arises is how much one looses due to the absence of central coordination. Koutsoupias and Papadimitriou (1999) answer this question by introducing the price of anarchy, i.e., the ratio of the selfish solution in the worst Nash equilibrium over the centrally coordinated optimum solution. For the simplest model with two identical links, the price of anarchy can be quantified as 3/2. The price of anarchy parallels e.g. the approximation ratio from optimization: here the comparison is between the solution obtained by a certain efficient algorithm to the optimal solution to the optimization problem.

This thesis contributes to the intersection of optimization and game theory/ economic theory in several ways. We address classical questions in mechanism design

by means of graphs in Part I. Chapter 2 contains a new characterization of revenue equivalence. In order to understand what revenue equivalence means, consider an auction setting, where a single good has to be sold to one of several bidders. Each bidder has private knowledge about how much the good is worth to him, but he is uncertain about the other bidders' valuations. Suppose, the goal was to allocate the good to the bidder who values it most and at the same time to charge a price that gives no bidder an incentive to bid anything else than his true valuation. In such a setting, the expected revenue of the seller will always be the same, no matter how the actual auction format looks like (Myerson 1981). This fact is referred to as revenue equivalence. Revenue equivalence results can be considered for much more general settings than auctions. We regard such a general setting and prove a characterization of revenue equivalence via elementary graph theory. Our theorem applies in settings where all previous revenue equivalence results fail to apply and has the additional advantage of being simple and elementary.

In the next chapter, Chapter 3, we regard optimal mechanisms for a scheduling setting. Let us again employ the auction example to get an idea of what an optimal mechanism could be. In the single good auction described above, an optimal auction mechanism is one that maximizes the revenue to the seller. According to the famous result in Myerson (1981), the optimal auction does not always give the good to the bidder who values it most, but the seller might e.g. keep the item if all bids are below a certain reserve price. Again, optimal mechanisms are not only interesting in auction settings, but in any game with payments. We regard a simple scheduling application and derive optimal mechanisms. We also see that even the simplest of all scheduling models is substantially more complicated with respect to determining an optimal mechanism than the auction setting.

Part II of the thesis is devoted to the application of mechanism design and game theoretic frameworks to multiple machine scheduling applications. In multiple machine settings, many different models and questions can arise in the context of game theoretic settings. Therefore, Chapter 4 contains an overview about the most interesting and most recent topics in this field. It includes the price of anarchy as well as approximation algorithms for optimization problems in distributed settings.

In the last chapter, Chapter 5, we regard an online scheduling model from a mechanism design point of view. The additional complication in the online situation is that the tasks that have to be scheduled on machines become only known over time. Decisions have to be made already before the entire set of tasks is known. Additionally, we require a decentralized setting, i.e., we want the tasks to have the power to select their machine themselves. We suggest a new equilibrium concept appropriate for this situation and provide a mechanism for decentralized online

scheduling. Moreover, our mechanism behaves well with respect to the approximation of a certain objective function.

In order to give a more precise description about the individual chapters of this thesis, we have to define some of the necessary terms used throughout the thesis. In the following two sections, we sketch the most important concepts in mechanism design and machine scheduling, respectively. Thereafter, we give a more detailed overview over the topics covered in the individual chapters of the thesis.

1.1 Mechanism Design

We start with the definition of mechanism design from Sandholm (2003).

> Mechanism design is the art of designing the rules of the game so that the desirable outcome is reached despite the fact that each agent acts in his own self-interest.

Thus, mechanism design is about *games*, i.e., situations in which several *agents* interact to obtain a common *outcome*. Agents have preferences over outcomes, which are expressed by an agent's *valuation* function. Agents can influence the outcome by choosing one of several *actions*. A *mechanism* defines the rules of the game and consists of two parts: an *allocation rule* and a *payment scheme*. Depending on the chosen actions of all agents, the allocation rule chooses an outcome, while the payment scheme determines a payment to or from every agent. The overall *utility* of an agent depends on his valuation for the chosen outcome and the payment. Throughout the thesis, we will assume *quasi-linear* utilities, meaning that the utility of an agent is computed as his valuation minus the payment he has to make. Usually in mechanism design, each agent has some piece of private information only known to the agent himself. We refer to this information as the agent's *type*. The valuation of an agent depends on his type as well, therefore being a function of outcome and type. We assume that agents are *rational*, i.e., that they strive to maximize their utility.

A well-studied kind of mechanisms are *direct revelation mechanisms*. Here, the only action of an agent is to announce his type. In many situations one can restrict oneself to analyzing direct revelation mechanisms without loss of generality. The justification for that is the *revelation principle* by Myerson (1981). However, in Chapter 5, we regard a setting where the revelation principle is not applicable and where we therefore model agents' actions explicitly.

Given type reports of all agents, the *efficient* allocation rule chooses an outcome that maximizes the total valuation of all agents. We also say that it maximizes the *social welfare*.

A *strategy* of an agent is a mapping that assigns an action to every possible type of an agent. A *dominant strategy* is one that maximizes the utility of the agent for every combination of actions of the other agents. We say that a direct revelation mechanism is *dominant strategy incentive compatible* or *truthful*, if reporting the true type is a dominant strategy for every agent. An allocation rule that can be complemented by a payment scheme to a truthful mechanism is also called (*dominant strategy*) *implementable*. The concept of dominant strategy incentive compatibility is predominantly used in Chapters 2, 4 and 5. In Chapter 3 we regard a Bayes-Nash setting, i.e., uncertainties with respect to other agents' types are represented by commonly known probability distributions. Here, agents aim to maximize their expected utilities and a mechanism is called *Bayes-Nash incentive compatible*, if truthful reporting maximizes the expected utility of every agent, given that all other agents report truthfully.

Besides (dominant strategy or Bayes-Nash) incentive compatibility, we usually require a mechanism to satisfy *individual rationality*. That is, a truth-telling agent must not receive a negative (expected) utility. This makes sense in settings, where agents have the choice not to participate in the mechanism and hence have an *outside option* which gives them utility zero. On the other hand, there are settings, in which such an outside option does not exist (see Chapter 5).

We conclude this section by sketching the probably most famous mechanism design setting, the single item auction. Here, one indivisible good is for sale and agents have a certain valuation for possessing the good. Their valuation is zero if they do not get the good. In the direct revelation version, the *sealed-bid* auction, agents submit their *bids* for the good e.g. in a sealed envelope. The highest bidder receives the good. In a *first price* auction, the winner has to pay his own bid, whereas he pays the second highest bid in the *second price* auction. The latter is also known as the famous Vickrey auction.

The idea of the Vickrey auction can be generalized to other than auction settings. Observe that the payment for the winner x in the second price auction is essentially the "disutility imposed on the other agents", i.e., the total welfare loss for the other agents due to the presence of x. The generalized Vickrey-Clarke-Groves (VCG) mechanism adopts this idea. The allocation rule in the VCG mechanism is the efficient one and every agent pays the disutility he imposes on the other agents. In Chapter 4, we restate the proof that the VCG mechanism is indeed truthful in more general settings.

For a more extensive introduction to mechanism design, see Mas-Colell, Whinston, and Green (1995).

1.2 Machine Scheduling

Although we give a more detailed description of scheduling models in the introduction to Chapter 4, we sketch the most important concepts already at this point. In the machine scheduling models considered in this thesis, a set of *jobs* has to be processed on a set of *machines*. Each machine can handle one job at a time and for the main part of the thesis, jobs may not be interrupted once started, i.e., we regard *non-preemptive* scheduling. Jobs are characterized by a *processing time* and a *weight*. The latter can be seen as the waiting cost per time unit incurred by the job-owner.

A *schedule* is a plan that specifies which job is processed on what machine and at which time. While in Chapter 3 a schedule on a single machine can be simply identified with the permutation of jobs corresponding to the order in which they are processed, a schedule in Chapters 4 and 5 refers to a specification of a time slot and a machine for every job. The length of the time slot should equal the job's (reported) processing time. Each job has a *start time* and a *completion time* in a specific schedule. The most important *objective functions* that we aim to minimize in models considered in this thesis are

1. the weighted total completion time or start time of all jobs,

2. the *makespan*, i.e., the maximum completion time of any job.

Given a set of jobs with weights and processing times, the difference between the weighted total completion time and the weighted total start time is constant over all schedules. Therefore any schedule that minimizes one, also minimizes the other objective function. The difference becomes important, when we regard approximation algorithms. Therefore we stick to the more standard objective to minimize the total weighted completion time in Chapter 5, where we regard approximations, while it is more convenient to use the total weighted start time as an objective in Chapter 3.

Most scheduling problems that we address in Part II of this thesis are *NP-hard*. Generally speaking, that means that the regarded objective function cannot be optimized by an algorithm whose running time is polynomial in the input length of the problem if $P \neq NP$ - an assumption that among researchers is widely believed to be true. One way to deal with this problem is the design of *approximation algorithms*. An approximation algorithm with *approximation factor* ϱ for a minimization problem is a polynomial time algorithm that guarantees the ratio of the obtained solution and the optimal solution to be at most ϱ. For a more comprehensive treatment of complexity theory and approximation algorithms we refer to Garey and Johnson (1979) and Papadimitriou (1994).

In Chapter 5 and a part of Chapter 4, we regard *online scheduling problems*. Here, jobs arrive over time, each job at his *release date*. An online algorithm has to make decisions, before the entire set of jobs to be scheduled is known. In the regarded settings, no *online algorithm* can guarantee to find an optimal solution to the problem due to the online nature of the situation, i.e., regardless of the $P = NP$ question. An online algorithm for a minimization problem is said to be ϱ-competitive if the ratio of a solution obtained by the algorithm and the optimal solution is at most ϱ. For an overview about online scheduling problems, see Pruhs, Sgall, and Torng (2004).

The wealth of different scheduling models can certainly not be covered in this thesis. We briefly reflect on other scheduling models in the introduction to Chapter 4 and refer to Leung (2004) for further reading.

1.3 Outline of the Thesis

The chapters of this thesis are to a large extent self-contained. Necessary notation and background are given in the respective introductions.

Part I deals with typical mechanism design questions. Important tools used in this part are the *type graph* and the *allocation graph* of an allocation rule. Basic theorems from graph theory enable us to view payment schemes implementing an allocation rule as *node potentials* in the mentioned graphs.

In Chapter 2, we give a new characterization of *revenue equivalence*. Revenue equivalence is the property of an allocation rule to be implementable by a uniquely defined payment scheme. Via the analogy to node potentials in the graphs mentioned above, we obtain our characterization in a simple and elementary way. In contrast to most previous work about revenue equivalence, we do not have to assume any differentiability conditions of the allocation rule or the valuation functions. Our characterization implies many of the existing results and we demonstrate by means of an economic application that it can be used to identify revenue equivalence where known results fail to apply.

In Chapter 3, we regard optimal mechanism design for a scheduling setting on a single machine. Our mechanisms are optimal in the sense that they minimize the total payment made to job-agents in order to reimburse them for their waiting time. The results are comparable to those in Myerson (1981) and Malakhov and Vohra (2007) for auction settings. We derive optimal mechanisms for the one-dimensional case, i.e., when only the weight of a job is private information. In contrast to Myerson (1981), we regard discrete type spaces as well. The type graph mentioned above is used to derive the optimal payment scheme for a given allocation rule. For

the two-dimensional setting, i.e., when additionally the processing time is private, we show that classical approaches must fail and that optimal mechanism design for the scheduling setting must be more difficult than for the classical auction settings.

In Part II, we study applications of mechanism design to multiple machine scheduling problems.

Chapter 4 contains a survey of recent literature about applications of mechanism design on the one hand and the analysis of full information games arising in machine scheduling problems on the other. The purpose of this chapter is to give an overview and to illustrate techniques by giving alternative proofs for known results, rather than obtaining new ones.

In Chapter 5, we study mechanism design for a decentralized online scheduling model on parallel machines. Here, the type of each job agent consists of his weight, processing time and release date. We define the concept of a decentralized online scheduling mechanism, which accounts for the requirement that job-agents select a machine themselves rather than being assigned by a central coordination authority. Furthermore, we introduce the concept of a myopic best response equilibrium that we find appropriate for online situations. The main contribution of this chapter is a 3.28-competitive decentralized online scheduling mechanism, where truthful reporting about private information is a myopic best response equilibrium.

1.4 Publications Underlying this Thesis

- B.Heydenreich, R.Müller, M.Uetz and R.Vohra, "Characterization of Revenue Equivalence", *Econometrica 77*(1), p.307-316, 2009
- B.Heydenreich, D.Mishra, R.Müller and M.Uetz, "Optimal Mechanisms for Single Machine Scheduling", in *Internet and Network Economics - WINE 2008*, C. Papadimitriou and S. Zhang (eds.), *Lecture Notes in Computer Science 5385*, p.414-425, 2008, Springer
- B.Heydenreich, R.Müller and M.Uetz, "Games and Mechanism Design in Machine Scheduling - An Introduction", *Production and Operations Management 16*(4), p.437-454, 2007
- B.Heydenreich, R.Müller and M.Uetz, "Mechanism Design for Decentralized Online Machine Scheduling", *Operations Research*, to appear

Part I

A Graph Theoretic Perspective on Mechanism Design

Chapter 2

Characterization of Revenue Equivalence

The property of an allocation rule to be implementable in dominant strategies by a unique payment scheme is called *revenue equivalence*. In this chapter we give a characterization of revenue equivalence based on a graph theoretic interpretation of the incentive compatibility constraints. The characterization holds for any (possibly infinite) outcome space and many of the known results are immediate consequences. Moreover, revenue equivalence can be identified in cases where existing theorems are silent.[1]

2.1 Introduction

One of the most important results of auction theory is the Revenue Equivalence Theorem. Subject to certain reasonable assumptions, it concludes that a variety of different auctions generate the same expected revenue for the seller. Klemperer (1999) writes that "much of auction theory can be understood in terms of this theorem.....". Hence the long line of papers that have attempted to relax the sufficient conditions under which revenue equivalence holds. In this chapter, we provide necessary and sufficient conditions on the underlying primitives for revenue equivalence to hold.

We consider direct revelation mechanisms for agents with multidimensional types. Such mechanisms consist of an allocation rule and a payment scheme. The allocation rule selects an outcome depending on the agents' reported types, whereas the

[1]Part of the results of this chapter were published in Heydenreich, Müller, Uetz, and Vohra (2009).

payment scheme assigns a payment to every agent. We focus attention on allocation rules that are implementable in dominant strategies. Hereafter we refer to such rules as implementable. In this environment we characterize the uniqueness of the relevant payment scheme in terms of conditions that are easily verified in potential applications. The property of an allocation rule to be implementable in dominant strategies by a unique payment scheme is called *revenue equivalence*. Our characterization of revenue equivalence is based on a graph theoretic interpretation of the incentive compatibility constraints. This interpretation has been used before by Rochet (1987), Gui, Müller, and Vohra (2004), as well as Saks and Yu (2005) to identify allocation rules that are implementable in dominant strategies. Müller, Perea, and Wolf (2007) used it to identify Bayes-Nash implementable allocation rules. With this graph theoretic interpretation, our characterizing condition for revenue equivalence is almost self-evident and the proof writes itself. The characterization holds for any (possibly infinite) outcome space. Many of the known results about revenue equivalence are immediate consequences of our characterization. More importantly, with this characterization revenue equivalence can be identified in cases where existing theorems are silent.

Related Work. The bulk of prior work on revenue equivalence has been devoted to identifying sufficient conditions on the type space that ensure all allocation rules from a given class satisfy revenue equivalence. The papers by Green and Laffont (1977) and Holmström (1979) restrict attention to allocation rules called 'utilitarian maximizers', that is, allocation rules that maximize the sum of the valuations of all agents. Holmström (1979), generalizing the paper by Green and Laffont (1977), shows that when the type space is smoothly path connected then utilitarian maximizers satisfy revenue equivalence.

Myerson (1981) shows that revenue equivalence holds for every implementable rule in a single item auction setting where the type space is an interval of the real line and an agents valuation for an outcome is linear in his type.

Krishna and Maenner (2001) derive revenue equivalence under two different hypotheses. In the first, agents' type spaces must be convex and the valuation function of an agent is a convex function of the type of the agent. Under these conditions they show that every implementable rule satisfies revenue equivalence. The second hypothesis requires the allocation rule to satisfy certain differentiability and continuity conditions and the outcome space to be a subset of the Euclidean space. Furthermore, the valuation functions must be regular Lipschitzian and monotonically increasing in all arguments.

Milgrom and Segal (2002) show that revenue equivalence is a consequence of a particular envelope theorem in a setting where the type spaces are one-dimensional

and the outcome space is arbitrary. An agent's valuation function is assumed differentiable and absolutely continuous in the type of the agent and the partial derivative of the valuation function with respect to the type must satisfy a certain integrability condition. Their result can be applied to multi-dimensional type spaces as well. In this case the type spaces must be smoothly connected and the valuation functions must be differentiable with bounded gradient.

We know of only two papers that identify necessary as well as sufficient conditions – i.e. *characterizing* conditions – for revenue equivalence to hold. If the outcome space is finite, Suijs (1996) characterizes type spaces and valuation functions for which utilitarian maximizers satisfy revenue equivalence. Chung and Olszewski (2007) characterize type spaces and valuation functions for which *every* implementable allocation rule satisfies revenue equivalence, again under the assumption of a finite outcome space. Furthermore, they derive sufficient conditions on the type spaces and valuation functions that generalize known results when the outcome space is countable or a probability distribution over a finite set of outcomes. In particular, they can show that some of the previously known conditions can be weakened for countable outcome spaces.

Our Contribution. Our characterization differs from prior work in an important way. We identify a condition on the type spaces, the valuation functions *and* the implementable allocation rule together that characterize revenue equivalence. In other words, we prove that a particular allocation rule satisfies revenue equivalence if and only if this condition is satisfied. Our characterization differs from the one by Chung and Olszewski (2007) in three ways. First, ours holds for general outcome spaces. Second, our result implies revenue equivalence in cases where their result does not apply. In fact, given agents' type spaces and valuation functions, several allocation rules may be implementable in dominant strategies, some of which satisfy revenue equivalence and some do not. In this case, the conditions on the type space and valuation functions from their paper obviously cannot hold. However, our characterization can be used to determine which of the allocation rules do satisfy revenue equivalence. We give an example in Section 2.5.4. Third, the characterization in Chung and Olszewski (2007) can be seen as a corollary of our result, in the sense that their necessary and sufficient condition is naturally related to our graph theoretic interpretation of revenue equivalence. We refer to Section 2.5.3 for details. Moreover, our characterization yields elementary and direct alternative proofs for their sufficient conditions for countable infinite outcome spaces.

As in Chung and Olszewski (2007), a sufficient condition derived from the characterization yields a number of the earlier results as immediate consequences We list some of them below.

1. By restricting attention to countable outcome spaces we can relax the smooth connectedness condition on the type space invoked in Holmström (1979) to (topological) connectedness. In addition, our sufficient condition applies to any allocation that can be implemented in dominant strategies rather than just utilitarian maximizers.

2. The sufficient condition that Suijs (1996) derives from his characterization follows as a special case.

3. The sufficient condition of Krishna and Maenner (2001) under their first hypothesis when the outcome space is countable follows as a special case.

4. The sufficient condition of Milgrom and Segal (2002) when the outcome space is countable follows as a special case.

Organization. The remainder of the chapter is organized as follows. In Section 2.2 we introduce notation and basic definitions. In Section 2.3, we prove some graph theoretic results. In Section 2.4, we derive our graph-theoretic characterization of revenue equivalence. In Section 2.5 we give four applications of the new characterization. The first two are simple and elementary proofs for sufficient conditions for revenue equivalence in settings with finite and countable outcome spaces, respectively. Third, we show how the characterization from Chung and Olszewski (2007) can be obtained as a consequence of ours. Finally, we give an example of an economic setting where our characterization can be used to identify revenue equivalence, whereas all known previous results are not applicable. We conclude with extensions to other notions of incentive compatibility in Section 2.6.

2.2 Setting and Basic Concepts

Denote by $\{1,\dots,n\}$ the set of *agents* and let $\mathcal{A}$ be the set of possible *outcomes*. Outcome space $\mathcal{A}$ is allowed to have infinitely many, even uncountably many, elements. Denote the *type* of agent $i \in \{1,\dots,n\}$ by t_i. Let T_i be the *type space* of agent i. Type spaces T_i can be arbitrary sets. Agent i's preferences over outcomes are modeled by the *valuation function* $v_i\colon \mathcal{A}\times T_i \to \mathbb{R}$, where $v_i(a,t_i)$ is the valuation of agent i for outcome a when he has type t_i.

A *mechanism* (f,π) consists of an *allocation rule* f and a *payment scheme* π. In a *direct revelation mechanism*, the allocation rule $f\colon \Pi_{i=1}^n T_i \to \mathcal{A}$ chooses for a vector t of aggregate type reports of all agents an outcome $f(t)$, whereas the payment scheme $\pi\colon \Pi_{i=1}^n T_i \to \mathbb{R}^n$ assigns a payment $\pi_i(t)$ to each agent i. Let the vector (t_i,t_{-i}) denote the aggregate type report vector when i reports t_i and the other

agents' reports are represented by t_{-i}. We assume *quasi-linear utilities*, that is, the utility of agent i when the aggregate report vector is (t_i, t_{-i}) is $v_i(f(t_i, t_{-i}), t_i) - \pi_i(t_i, t_{-i})$.

Definition 2.1. *A direct revelation mechanism (f, π) is called* dominant strategy incentive compatible *if for every agent i, every type $t_i \in T_i$, all aggregate type vectors t_{-i} that the other agents could report and every type $s_i \in T_i$ that i could report instead of t_i:*

$$v_i(f(t_i, t_{-i}), t_i) - \pi_i(t_i, t_{-i}) \geq v_i(f(s_i, t_{-i}), t_i) - \pi_i(s_i, t_{-i}).$$

If for allocation rule f there exists a payment scheme π such that (f, π) is a dominant strategy incentive compatible mechanism, then f is called implementable in dominant strategies*, in short* implementable*.*

In this chapter we assume that the allocation rule is implementable in dominant strategies and study the uniqueness of the corresponding payment scheme. We refer to the latter as revenue equivalence.[2]

Definition 2.2. *An allocation rule f implementable in dominant strategies satisfies the* revenue equivalence *property if for any two dominant strategy incentive compatible mechanisms (f, π) and (f, π') and any agent i there exists a function h_i that only depends on the reported types of the other agents t_{-i} such that*

$$\forall t_i \in T_i : \pi_i(t_i, t_{-i}) = \pi'_i(t_i, t_{-i}) + h_i(t_{-i}).$$

2.3 Unique Node Potentials in Directed Graphs

In this section, we prove two theorems about node potentials in directed graphs. The first theorem yields a necessary and sufficient condition for a graph to have a node potential that is uniquely defined up to a constant. The second theorem provides another sufficient condition for uniqueness of the node potential up to a constant. In the following sections we will make use of these results to obtain necessary and sufficient conditions for revenue equivalence to hold.

[2] We choose the term revenue equivalence in accordance with Krishna (2002). In our setting it is equivalent to payoff equivalence as used in Krishna and Maenner (2001). See Milgrom (2004), Section 4.3.1. for settings where it is not equivalent.

Let $G = (V, E)$ be a directed graph with node set V and arc set E. V is allowed to be infinite. By ℓ_{ab} we denote the (finite) length of the arc (a, b) from node a to node b. A *path* from node a to node b in G, or short (a, b)-path, is defined as $\mathfrak{p} = (a = a_0, a_1, \ldots, a_k = b)$ such that $(a_{i-1}, a_i) \in E$ for $i = 1, \ldots, k$. Denote by $length(\mathfrak{p})$ the length of this path. A *cycle* is a path with $a = b$. For any a, we regard the path from a to a without any arcs as (a, a)-path as well and define its length as 0. We assume that G is *strongly connected*, that is, between any two nodes $a, b \in V$, there exists an (a, b)-path and a (b, a)-path. Define $\mathcal{P}(a, b)$ to be the set of all (a, b)-paths.

If G does not contain a negative cycle, we say that it satisfies the *nonnegative cycle property*. In the following assume that G satisfies the nonnegative cycle property. Let

$$dist_G(a, b) = \inf_{\mathfrak{p} \in \mathcal{P}(a,b)} length(\mathfrak{p}).$$

If V is a finite set, then $dist_G(a, b)$ simply equals the length of a shortest path from a to b in G. For infinite V, such a shortest path may not exist. Nevertheless, $dist_G(a, b)$ is finite, since we assume that G does not have any negative cycle. In fact, fix some (b, a)-path $\mathfrak{p}_{ba}$, then $length(\mathfrak{p}) \geq -length(\mathfrak{p}_{ba})$ holds for every (a, b)-path $\mathfrak{p}$ and the infimum is finite.

Definition 2.3. *A* node potential *p is a function $p\colon V \to \mathbb{R}$ such that for all arcs $(x, y) \in E$, $p(y) \leq p(x) + \ell_{xy}$.*

Lemma 2.4. *A graph G has a node potential if and only if it has no cycle of negative length.*

Proof. Proofs can be found e.g. in Schrijver (2003) for finite V and in Rochet (1987) for infinite V. For completeness, we give a simple proof. If G has no negative cycle, then for any $a \in V$, $dist_G(a, \cdot)$ is well-defined, i.e. takes only finite values. The distances $dist_G(a, \cdot)$ define a node potential, because $dist_G(a, x) \leq dist_G(a, y) + \ell_{yx}$ for all $x, y \in V$. On the other hand, given a node potential p, add up the inequalities $p(y) - p(x) \leq \ell_{xy}$ for all arcs (x, y) on a cycle to prove that the cycle has nonnegative length. □

The next theorem is concerned with the uniqueness of a node potential in a graph without negative cycles.

Theorem 2.5. *Let $G = (V, E)$ be a strongly connected directed graph that satisfies the nonnegative cycle property. Then the following statements are equivalent.*

1. *Any two node potentials in G differ only by a constant.*

2. Distances are anti-symmetric, i.e., $dist_G(a,b) = -dist_G(b,a)$ for all $a, b \in V$.

Proof. [1 ⇒ 2] As observed earlier, $dist_G(a,\cdot)$ is a node potential in G. Similarly, $dist_G(b,\cdot)$ is a node potential. As any two node potentials differ only by a constant, we have that $dist_G(a,\cdot) - dist_G(b,\cdot)$ is a constant function. Especially, for a and b we get that $dist_G(a,a) - dist_G(b,a) = dist_G(a,b) - dist_G(b,b)$. Clearly, $dist_G(a,a) = dist_G(b,b) = 0$ and hence $dist_G(a,b) = -dist_G(b,a)$.

[2 ⇒ 1] Let $a, b \in V$. Let $\mathfrak{p}_{ab}$ be an (a,b)-path with nodes $a = a_0, a_1, \ldots, a_k = b$. For any node potential p we have that

$$\begin{aligned} p(a_1) - p(a) &\leq \ell_{aa_1} \\ p(a_2) - p(a_1) &\leq \ell_{a_1a_2} \\ &\vdots \\ p(b) - p(a_{k-1}) &\leq \ell_{a_{k-1}b} \end{aligned}$$

and consequently $p(b) - p(a) \leq length(\mathfrak{p}_{ab})$. Therefore,

$$p(b) - p(a) \leq \inf_{\mathfrak{p} \in \mathcal{P}(a,b)} length(\mathfrak{p}) = dist_G(a,b).$$

Similarly, $p(a) - p(b) \leq dist_G(b,a)$. Therefore, $-dist_G(b,a) \leq p(b) - p(a) \leq dist_G(a,b)$. Since $dist_G(a,b) = -dist_G(b,a)$, $p(b) - p(a) = dist_G(a,b)$ for any node potential p. Hence, any potential is completely defined, once $p(a)$ has been chosen for some outcome a. Thus, any two node potentials can only differ by a constant. □

Next, we define a property of the graph G that is sufficient (though not necessary) for uniqueness of node potentials up to a constant.

Definition 2.6. *A graph with node set V and arc lengths ℓ is called* two-cycle connected *if for every partition $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $V_1, V_2 \neq \emptyset$, there are $a_1 \in V_1$ and $a_2 \in V_2$ with $\ell_{a_1a_2} + \ell_{a_2a_1} = 0$.*

Theorem 2.7. *Let G be a directed graph that satisfies the nonnegative cycle property. If G is two-cycle connected then its node potential is uniquely defined up to a constant.*

Proof. First, we show that if G is two-cycle connected, then any two nodes $a, b \in V$ are connected in G by a finite path with nodes $a = a_0, a_1, \ldots, a_k = b$ such that $\ell_{a_ia_{i+1}} + \ell_{a_{i+1}a_i} = 0$ for $i = 0, \ldots, k-1$. Call such a path a zero-path. Suppose to

the contrary, that there is a node $a \in V$ that is not connected to all nodes in G by a zero-path. Define V_1 to be the set containing all nodes b that can be reached from a by a zero-path. Let $V_2 = V \setminus V_1$. By assumption $V_2 \neq \emptyset$. Then, as G is two-cycle connected, there is an $a_1 \in V_1$ and $a_2 \in V_2$ with $\ell_{a_1 a_2} + \ell_{a_2 a_1} = 0$ contradicting the assumption that $a_2 \in V_2$.

Consider $a, b \in V$ and a zero-path $\mathfrak{p}_{ab} = (a_0, a_1, \ldots, a_k)$. Then $\mathfrak{p}_{ab}$ together with the (b,a)-path $\mathfrak{p}_{ba} = (a_k, \ldots, a_1, a_0)$ form a cycle of length 0. Note, that between any two nodes c, d on a cycle of length 0, the path from c to d on the cycle must be a shortest path, as otherwise, we could construct a negative cycle by substituting this path by a shorter one. Therefore, $dist_G(a,b) + dist_G(b,a) = length(\mathfrak{p}_{ab}) + length(\mathfrak{p}_{ba}) = 0$. Hence, any two node potentials in G differ only by a constant due to Theorem 2.5. □

To see that two-cycle connectedness is not necessary for the uniqueness of the node potential, consider the following example.

Example 2.8. *Consider graph G in Figure 2.1. The graph satisfies the nonnegative cycle property, and for every two nodes $u, v \in V$, $dist_G(u,v) + dist_G(v,u) = 0$. Hence, the node potential is uniquely defined up to a constant according to Theorem 2.5. Notice, however, that G is not two-cycle connected, as the partition $(\{a,c\},\{b\})$ violates the condition of Definition 2.6.*

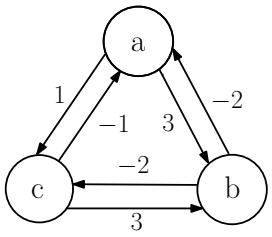

Figure 2.1: Graph G satisfies anti-symmetric distances but not two-cycle connectedness

2.4 Characterization of Revenue Equivalence

We give a necessary and sufficient condition for revenue equivalence with the aid of a graph theoretic interpretation used to characterize implementable allocation rules by Rochet (1987), Gui, Müller, and Vohra (2004) and Saks and Yu (2005). We also adopt some of their notation.

Fix agent i and the reports, t_{-i}, of the other agents. For simplicity of notation we write T and v instead of T_i and v_i. Similarly, for any mechanism (f, π), we regard f and π as functions of i's type alone, i.e. $f\colon T \to \mathcal{A}$ and $\pi\colon T \to \mathbb{R}$. If (f, π) is dominant strategy incentive compatible, it is easy to see that for any pair of types $s, t \in T$ such that $f(t) = f(s) = a$ for some $a \in \mathcal{A}$, the payments must be equal, i.e. $\pi(t) = \pi(s) =: \pi_a$. Hence, the payment of agent i is completely defined if the numbers π_a are defined for all outcomes $a \in \mathcal{A}$ such that $f^{-1}(a)$ is nonempty. For ease of notation, we let $\mathcal{A}$ denote the set of "achievable" outcomes, i.e., the set of outcomes a such that there exists some type $t \in T$ of agent i such that $f(t) = a$. For an allocation rule f, let us define two different kinds of graphs. The *type graph* T_f has node set T and contains an arc from any node s to any other node t of length[3]

$$\ell_{st} = v(f(t), t) - v(f(s), t).$$

Here, ℓ_{st} represents the gain in valuation for agent i truthfully reporting type t instead of lying type s. This could be positive or negative. The *allocation graph* G_f has node set $\mathcal{A}$. Between any two nodes $a, b \in \mathcal{A}$, there is a directed arc with length[3]

$$\ell_{ab} = \inf_{t \in f^{-1}(b)} (v(b, t) - v(a, t)).$$

The arc lengths ℓ_{ab} in the allocation graph represent the least gain in valuation for agent i with *any* type $t \in f^{-1}(b)$ for reporting truthfully, instead of misreporting so as to get outcome a (instead of b). The type graphs and allocation graphs are complete, directed, and possibly infinite graphs[4]. Note that type and allocation graphs are strongly connected, since they are complete graphs. We introduce our main results in terms of allocation graphs. Analogous results hold for type graphs as well.

Observation 2.9. *Let f be an allocation rule. Payment schemes π such that (f, π) is a dominant strategy incentive compatible mechanism, exactly correspond to node potentials in each of the allocation graphs G_f that are obtained from a combination of an agent and a report vector of the other agents.*

Proof. Assume f is implementable. Fix agent i and the reports t_{-i} of the other

[3]We assume that arc lengths are strictly larger than $-\infty$. For allocation rules implementable in dominant strategies this is no restriction, as the incentive compatibility constraints imply finiteness of the arc lengths.

[4]Clearly, type and allocation graph depend on the agent i and reports t_{-i} of the other agents. In order to keep notation simple, we suppress the dependence on i and t_{-i} and will simply write T_f and G_f.

agents. Consider the corresponding allocation graph G_f. For any pair of types $s, t \in T$ such that $f(t) = f(s) = a$ for some $a \in \mathcal{A}$, the payments must be equal, i.e. $\pi(t) = \pi(s) = \pi_a$. Therefore, π assigns a real number to every node in the graph. Incentive compatibility implies for any two outcomes $a, b \in \mathcal{A}$ and all $t \in f^{-1}(b)$ that $v(b,t) - \pi_b \geq v(a,t) - \pi_a$, hence, $\pi_b \leq \pi_a + \ell_{ab}$.

For the other direction, define the payment π for agent i as follows. For any report vector of the other agents t_{-i}, consider the corresponding allocation graph G_f and fix a node potential p. At aggregate report vector (t_i, t_{-i}) with outcome $a = f(t_i, t_{-i})$, let the payment be $\pi_a := p(a)$. Incentive compatibility now follows from the fact that p is a node potential in G_f, similarly to the above. □

Clearly, the allocation graphs can be defined for any allocation rule such that all arc lengths are finite. Observation 2.9 together with Lemma 2.4 therefore yields a characterization of allocation rules that are implementable in dominant strategies (see also e.g. Rochet (1987)).

Observation 2.10. *The allocation rule f is implementable in dominant strategies if and only if all allocation graphs G_f obtained from a combination of an agent and a report vector of the other agents satisfy the nonnegative cycle property.*

From Lemma 2.4 and Observations 2.9 and 2.10 it follows that for any allocation rule f implementable in dominant strategies, there exist node potentials in all allocation graphs G_f. The allocation rule f satisfies revenue equivalence if and only if in each allocation graph G_f, the node potential is uniquely defined up to a constant. Combining this with Theorem 2.5 yields our main result.

Theorem 2.11. *Let f be an allocation rule that is implementable in dominant strategies. Then f satisfies revenue equivalence if and only if in all allocation graphs G_f obtained from a combination of an agent and a report vector of the other agents, distances are anti-symmetric, i.e., $dist_{G_f}(a,b) = -dist_{G_f}(b,a)$ for all $a, b \in \mathcal{A}$.*

An analogous characterization holds for type graphs as well. One can check that all previous arguments still apply when using type graphs. On the other hand, note the following relation of node potentials in G_f and node potentials in T_f. Given a node potential p^G in G_f, we can define a node potential p^T in T_f by letting $p^T(t) := p^G(f(t))$ for any $t \in T$. In fact, let ℓ^G and ℓ^T denote the arc lengths in G_f and T_f respectively and observe that $\ell^G_{ab} = \inf\{\ell^T_{st} \mid s \in f^{-1}(a),\ t \in f^{-1}(b)\}$. Then, for any $s, t \in T$,

$$p^T(t) = p^G(f(t)) \leq p^G(f(s)) + \ell^G_{f(s)f(t)} \leq p^T(s) + \ell^T_{st}$$

and p^T is a node potential. On the other hand, given a node potential p^T in T_f, let $p^G(a) := p^T(s)$ for any $s \in f^{-1}(a)$. Note that p^G is well-defined as $f(s) = f(t) = a$ implies $\ell^T_{st} = 0$ and hence $p^T(s) = p^T(t)$. Furthermore, for any $a, b \in \mathcal{A}$ and any $s \in f^{-1}(a), t \in f^{-1}(b)$,

$$p^G(a) = p^T(s) \leq p^T(t) + \ell^T_{ts} = p^G(b) + \ell^T_{ts}.$$

Hence, $p^G(a) \leq p^G(b) + \ell^G_{ba}$ and p^G is a node potential in G_f. Consequently, there is a one-to-one relationship between node potentials in G_f and node potentials in T_f. This insight yields the following corollary.

Corollary 2.12. *Let f be an allocation rule that is implementable in dominant strategies. Then f satisfies revenue equivalence if and only if in all type graphs T_f obtained from a combination of an agent and a report vector of the other agents, distances are anti-symmetric, i.e., $dist_{T_f}(s,t) = -dist_{T_f}(t,s)$ for all $s, t \in T$.*

2.5 Applications

In this section we give four applications of our main result. The first two results show how Theorem 2.11 yields simple, transparent proofs that all implementable f satisfy revenue equivalence in settings where the outcome space is finite or countably infinite, respectively. The third shows that the characterization by Chung and Olszewski (2007) can be derived from Theorem 2.11. The fourth describes an economic environment with uncountable outcome space where Theorem 2.11 can be used to identify revenue equivalence, but where the existing theorems are silent.

2.5.1 Finite Outcome Spaces

In this section, we prove revenue equivalence for finite outcome spaces when type spaces and valuation functions satisfy very weak assumptions. From now on, we assume that agents' type spaces are (arbitrary) topological spaces. Recall that a subset T of a topological space is *connected* if it cannot be covered non-trivially by the disjoint union of two open sets. That is, there exist no open sets T_1, T_2 with $T \subseteq T_1 \cup T_2$, $T_1 \cap T_2 = \emptyset$, $T \cap T_1 \neq \emptyset$ and $T \cap T_2 \neq \emptyset$. We prove the following.

Theorem 2.13. *Let $\mathcal{A}$ be a finite outcome space. Let each agent $i \in \{1, \ldots, n\}$ have types from the (topologically) connected type space T_i. Let each agent's valuation function $v_i(a, \cdot)$ be a continuous function in the type of the agent for every $a \in \mathcal{A}$.*

Then, every allocation rule $f\colon \Pi_{i=1}^n T_i \to \mathcal{A}$ that is implementable in dominant strategies satisfies revenue equivalence.

For the proof, we need the following fact from topology that can be found e.g. in Munkres (2000).

Fact 2.14. *Let T be a connected subset of a topological space. Then any partition of T into subsets $T_1, T_2 \neq \emptyset$, $T_1 \cup T_2 = T$, $T_1 \cap T_2 = \emptyset$ satisfies $\overline{T}_1 \cap \overline{T}_2 \neq \emptyset$, where $\overline{T}_i$ is the closure of T_i in T.*

Now, we are able to prove the theorem.

Proof (*Theorem 2.13*). Consider a single agent with type space T and valuation function v. Regard f as a function on T as before. Let $\mathcal{A}_1 \cup \mathcal{A}_2 = \mathcal{A}$, $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$, $\mathcal{A}_1, \mathcal{A}_2 \neq \emptyset$ be a partition of $\mathcal{A}$. Then, $T = f^{-1}(\mathcal{A}_1) \cup f^{-1}(\mathcal{A}_2)$, $f^{-1}(\mathcal{A}_1) \cap f^{-1}(\mathcal{A}_2) = \emptyset$ is a partition of T and $f^{-1}(\mathcal{A}_1), f^{-1}(\mathcal{A}_2) \neq \emptyset$, since f is onto. According to the fact above, there exists $t \in \overline{f^{-1}(\mathcal{A}_1)} \cap \overline{f^{-1}(\mathcal{A}_2)}$. Hence, there are sequences $(t_1^n) \subseteq f^{-1}(\mathcal{A}_1)$ and $(t_2^n) \subseteq f^{-1}(\mathcal{A}_2)$ with $\lim_{n\to\infty} t_1^n = \lim_{n\to\infty} t_2^n = t$. As $\mathcal{A}$ is finite, there must be $a_1 \in \mathcal{A}_1$ and $a_2 \in \mathcal{A}_2$ and subsequences $(t_1^{n_k}) \subseteq (t_1^n)$ and $(t_2^{n_m}) \subseteq (t_2^n)$ with $f(t_1^{n_k}) = a_1$ for all k and $f(t_2^{n_m}) = a_2$ for all m. Since v is continuous in the type,

$$\begin{aligned} 0 &= v(a_2,t) - v(a_1,t) + v(a_1,t) - v(a_2,t) \\ &= \lim_{n\to\infty} (v(a_2,t_2^{n_m}) - v(a_1,t_2^{n_m}) + v(a_1,t_1^{n_k}) - v(a_2,t_1^{n_k})). \end{aligned}$$

According to the definition of the arc length in G_f, the latter can be bounded from below as follows.

$$\lim_{n\to\infty} (v(a_2,t_2^{n_m}) - v(a_1,t_2^{n_m}) + v(a_1,t_1^{n_k}) - v(a_2,t_1^{n_k})) \geq \ell_{a_1a_2} + \ell_{a_2a_1} \geq 0.$$

The last inequality is true, since G_f has no negative cycles. Hence, all inequalities are equalities and $\ell_{a_1a_2} + \ell_{a_2a_1} = 0$. Consequently, G_f is two-cycle connected. The claim follows from Theorem 2.7 and Observation 2.9. □

Notice that we cannot omit the continuity assumption, as the following example demonstrates.

Example 2.15. *Let there be one agent with type $t \in [0,1]$ and two outcomes $\mathcal{A} = \{a,b\}$. Let the agent's valuation be $v(a,t) = 1$, if $t < 1/2$ and $v(a,t) = 0$, if $t \geq 1/2$. Let $v(b,t) = 1/2$ for all t. That is, $v(a,\cdot)$ is discontinuous at $t = 1/2$. Let the*

allocation rule be the efficient one, i.e., $f(t) = a$ *for* $t < 1/2$ *and* $f(t) = b$ *otherwise. Then dominant strategy incentive compatibility is equivalent to* $1 - \pi_a \geq 1/2 - \pi_b$ *and* $1/2 - \pi_b \geq -\pi_a$*, which is satisfied whenever* $|\pi_a - \pi_b| \leq 1/2$*. For instance,* $\pi_a = \pi_b = 0$ *or* $\pi'_a = 1/2$*,* $\pi'_b = 0$ *are two payment schemes that make* f *truthful, but* π *and* π' *do not differ by a constant.*

If the type space is not connected, there are examples of the same flavor as Example 2.15, where the payment scheme is not unique. However, even if the valuation function is not continuous everywhere, sufficient conditions for revenue equivalence can be proven using Theorem 2.7. Indeed, if the valuation functions are continuous at particular type vectors as they are constructed in the proof, two-cycle connectedness of the graphs G_f follows. Also, if T is path connected and between any two types there exists a path such that the valuation functions are continuous along this path, we can make use of Theorem 2.7.

2.5.2 Countable Outcome Spaces

We investigate the case of countably infinite outcome spaces in this section. Theorem 2.11 yields an elementary proof for revenue equivalence under weak assumptions. One of these assumptions is equicontinuity of the valuation functions.

Definition 2.16. *The family of functions* $\{v(a,\cdot)\}_{a\in\mathcal{A}}$ *is equicontinuous if for all* $\varepsilon > 0$ *and* $t \in T \subseteq \mathbb{R}^k$ *there is* $\delta > 0$ *such that for all outcomes* $a \in \mathcal{A}$ *and all* $s \in T$ *with* $\|t - s\| < \delta$ *it holds that* $|v(a,t) - v(a,s)| < \varepsilon$*. Especially,* δ *must not depend on* a*. Here,* $\|\cdot\|$ *denotes the Euclidean norm on* $\mathbb{R}^k$*.*

Theorem 2.17. *Let* $\mathcal{A}$ *be a countable outcome space. Let each agent* $i \in \{1,\dots,n\}$ *have types from the (topologically) connected type space*[5] $T_i \subseteq \mathbb{R}^{k_i}$*. Let the family of valuation functions* $\{v_i(a,\cdot)\}_{a\in\mathcal{A}}$ *be equicontinuous for every agent* i*. Then, every allocation rule* $f\colon \Pi_{i=1}^n T_i \to \mathcal{A}$ *implementable in dominant strategies satisfies revenue equivalence.*

Note that Theorem 2.17 generalizes Theorem 2.13 if type spaces are subsets of the Euclidian space, since equicontinuity of $\{v(a,\cdot)\}_{a\in\mathcal{A}}$ for finite $\mathcal{A}$ is equivalent to the requirement that $v_i(a,\cdot)$ be continuous for all $a \in \mathcal{A}$. While the proof of Theorem 2.13 follows easily from Theorem 2.7, this approach fails in the infinite

[5] It is not possible to formulate the theorem for general topological type spaces any more due to the equicontinuity assumption. However, any metric space rather than the Euclidean space used here would have been sufficient.

case, and in fact, our proof of Theorem 2.17 is based on Theorem 2.5 rather than 2.7.

Proof. Assume an allocation rule f implementable in dominant strategies that does not satisfy revenue equivalence. We show this implies that there is an agent i whose type space is not connected. According to Theorem 2.11, there is an agent i and a report vector of the other agents t_{-i} with corresponding allocation graph G_f and $a^*, b^* \in A$ such that $dist_{G_f}(a^*, b^*) + dist_{G_f}(b^*, a^*) > 0$. Let T be the type space of i and regard f as a function on T as before. As $\mathcal{A}$ is countable, the set $\{dist_{G_f}(a^*, x) + dist_{G_f}(x, a^*) \,|\, x \in \mathcal{A}\}$ contains only countably many values. Hence, there exists a $z > 0$ such that the sets $\mathcal{A}_1 = \{x \,|\, dist_{G_f}(a^*, x) + dist_{G_f}(x, a^*) < z\}$ and $\mathcal{A}_2 = \{x \,|\, dist_{G_f}(a^*, x) + dist_{G_f}(x, a^*) > z\}$ are both non-empty and together yield a partition of $\mathcal{A}$. Clearly, $a^* \in \mathcal{A}_1$. Then, the sets $T_1 = f^{-1}(\mathcal{A}_1)$ and $T_2 = f^{-1}(\mathcal{A}_2)$ are non-empty and yield a partition of the type space T. T_1 is a proper subset of T. We show that T_1 is open and closed, implying that T is not connected.

T_1 is open: Let $t \in T_1$. Let $f(t) = x \in \mathcal{A}_1$. Then, $dist_{G_f}(a^*, x) + dist_{G_f}(x, a^*) = z - \varepsilon$ for some $\varepsilon = \varepsilon(x) > 0$. As the $v(a, \cdot)$, $a \in \mathcal{A}$, are equicontinuous, there is a $\delta_{\varepsilon/2}$ such that $|v(a, t) - v(a, s)| < \varepsilon/2$ for all $a \in \mathcal{A}$ and $s \in T$ with $\|s - t\| < \delta_{\varepsilon/2}$. Let $s \in T$ such that $\|s - t\| < \delta_{\varepsilon/2}$ and let $y = f(s)$. Then the following is true:

$$\begin{aligned} dist_{G_f}(a^*, y) + dist_{G_f}(y, a^*) &\leq dist_{G_f}(a^*, x) + \ell_{xy} + \ell_{yx} + dist_{G_f}(x, a^*) \\ &= z - \varepsilon + \ell_{xy} + \ell_{yx} \\ &\leq z - \varepsilon + |v(y, s) - v(y, t)| + |v(x, t) - v(x, s)| \\ &< z \end{aligned}$$

Hence $dist_{G_f}(a^*, y) + dist_{G_f}(y, a^*) < z$. Thus $y \in \mathcal{A}_1$ and $s \in T_1$ for any s in the $\delta_{\varepsilon/2}$-ball around t. Consequently, T_1 is open.

T_1 is closed: Let $(t^n)_{n \in \mathbb{N}}$ be a sequence in T_1 that converges to $t \in T$. Suppose for contradiction that $t \in T_2$. Let $x = f(t)$. Then $dist_{G_f}(a^*, x) + dist_{G_f}(x, a^*) = z + \varepsilon$ for some $\varepsilon = \varepsilon(x) > 0$. By equicontinuity, there is a $\delta_{\varepsilon/2}$ such that $|v(a, t) - v(a, s)| < \varepsilon/2$ for all $a \in \mathcal{A}$ and $s \in T$ with $\|s - t\| < \delta_{\varepsilon/2}$. Choose n_0 such that $\|t^{n_0} - t\| < \delta_{\varepsilon/2}$. Let $y = f(t^{n_0})$. As $t^{n_0} \in T_1$, $dist_{G_f}(a^*, y) + dist_{G_f}(y, a^*) < z$. Then the following

holds:

$$\begin{aligned} z+\varepsilon &= dist_{G_f}(a^*,x) + dist_{G_f}(x,a^*) \\ &\leq dist_{G_f}(a^*,y) + \ell_{yx} + \ell_{xy} + dist_{G_f}(y,a^*) \\ &< z + \ell_{yx} + \ell_{xy} \\ &\leq z + |v(x,t) - v(x,t^{n_0})| + |v(y,t^{n_0}) - v(y,t)| \\ &< z+\varepsilon, \end{aligned}$$

a contradiction. □

An example in Section 2.5.4 demonstrates that Theorem 2.17 cannot be generalized to uncountable outcome spaces.

A different way of modeling valuations for countable outcome spaces is to identify types with vectors in $\mathbb{R}^{\mathcal{A}}$, such that $v_i(a,t) = t_a$. On $\mathbb{R}^{\mathcal{A}}$ we can define the *sup-topology* by defining for $\varepsilon > 0$, and $t \in \mathbb{R}^{\mathcal{A}}$ the ε-ball around t as $B_\varepsilon(t) = \{s \in \mathbb{R}^{\mathcal{A}} \mid \sup_{a\in\mathcal{A}} |t_a - s_a| < \varepsilon\}$. Using almost the same proof as for Theorem 2.17, one can show the following theorem, which can also be found in Chung and Olszewski (2007).

Theorem 2.18. *Let $\mathcal{A}$ be a countable outcome space. Let the type space $T_i \subseteq \mathbb{R}^{\mathcal{A}}$ for every agent i be (topologically) connected with respect to the sup-topology on $\mathbb{R}^{\mathcal{A}}$. Then, every allocation rule $f\colon \Pi_{i=1}^n T_i \to \mathcal{A}$ that is implementable also satisfies revenue equivalence.*

Note that if for every agent i there are no two different types that have the same valuation for all outcomes $a \in \mathcal{A}$, then Theorem 2.17 follows easily from Theorem 2.18. That is because the function $v_i\colon T_i \to \mathbb{R}^{\mathcal{A}}$ with $v_i(t) := (t_{a_1}, t_{a_2}, \dots)$, where $\mathcal{A} = (a_1, a_2, \dots)$, is one-to-one. Then an implementable allocation rule f in the sense of Theorem 2.17 canonically translates into an implementable allocation rule $\tilde{f}$ in the sense of Theorem 2.18 by setting $\tilde{f}(v(t_1), \dots, v(t_n)) = f(t_1, \dots, t_n)$. Moreover, the $v_i(T_i)$ are connected for connected T_i and equicontinuous $\{v_i(a,\cdot)\}_{a\in\mathcal{A}}$. Revenue equivalence of $\tilde{f}$ then implies revenue equivalence of f.

On the other hand, if there are types $s, t \in T_i$ for some agent i with $v_i(a,s) = v_i(a,t)$ for all $a \in \mathcal{A}$, it is not clear, how to derive Theorem 2.17 from Theorem 2.18.

2.5.3 The Characterization by Chung and Olszewski

In this section, we show how the characterization by Chung and Olszewski (2007) follows from our main result and comment briefly on the results in other literature on

revenue equivalence. First, we introduce the notation used by Chung and Olszewski (2007) and restate their characterization theorem.

Let $\mathcal{A}$ be countable. As before, regard everything from the perspective of a single agent. Let $\mathcal{A}_1, \mathcal{A}_2$ be disjoint subsets of $\mathcal{A}$ and $r : \mathcal{A}_1 \cup \mathcal{A}_2 \to \mathbb{R}$. For every $\varepsilon > 0$, let

$$\mathcal{T}_1(\varepsilon) = \bigcup_{a_1 \in \mathcal{A}_1} \{t \in T \,|\, \forall\, a_2 \in \mathcal{A}_2 : v(a_1, t) - v(a_2, t) > r(a_1) - r(a_2) + \varepsilon\}$$

and

$$\mathcal{T}_2(\varepsilon) = \bigcup_{a_2 \in \mathcal{A}_2} \{t \in T \,|\, \forall\, a_1 \in \mathcal{A}_1 : v(a_1, t) - v(a_2, t) < r(a_1) - r(a_2) - \varepsilon\}.$$

Finally, let $\mathcal{T}_i = \cup_{\varepsilon > 0} \mathcal{T}_i(\varepsilon)$, $i = 1, 2$. Observe that $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$. Call the type space T *splittable* if there are $\mathcal{A}_1, \mathcal{A}_2$ and r such that $T = \mathcal{T}_1 \cup \mathcal{T}_2$ and $\mathcal{T}_i \neq \emptyset$ for $i = 1, 2$. Note that, if T is splittable, T is disconnected with respect to the sup-topology on $\mathbb{R}^{\mathcal{A}}$ defined above, as $\mathcal{T}_1$ and $\mathcal{T}_2$ are open sets. T being topologically disconnected, does not imply that T is splittable.

Theorem 2.19 (Chung and Olszewski 2007)**.** *If $\mathcal{A}$ is finite then the following two statements are equivalent.*

(*i*) *All f that are implementable in dominant strategies satisfy revenue equivalence.*

(*ii*) *For all agents, T_i is not splittable.*

If $\mathcal{A}$ is not finite, but countable, (ii) implies (i).

Notice that in Theorem 2.11 no assumption on the cardinality of $\mathcal{A}$ is made, whereas in Theorem 2.19, $\mathcal{A}$ is assumed finite or countable, respectively. On the other hand, Theorem 2.11 imposes a condition on the allocation rule, whereas Theorem 2.19 characterizes T and v such that *all* allocation rules that are implementable in dominant strategies satisfy revenue equivalence. We elaborate on this difference in Section 2.5.4.

In order to show that (*ii*) in Theorem 2.19 is a necessary condition for revenue equivalence in the case of finite $\mathcal{A}$, one can directly construct an allocation rule and two payment schemes that do not differ by a constant from the assumption that T is splittable. This is done in the paper by Chung and Olszewski. We give an

alternative proof for the fact that (ii) is a sufficient condition for countable $\mathcal{A}$. Our proof establishes a connection to the allocation graph defined in Section 2.4.

Proof. $[ii \Rightarrow i]$ Suppose an allocation rule f that is implementable in dominant strategies but does not satisfy revenue equivalence. Since f is implementable, the allocation graphs satisfy the non-negative cycle property. Since revenue equivalence is violated, Theorem 2.11 implies that there is an agent i and reports of the other agents t_{-i} such that in the corresponding allocation graph G_f, $dist_{G_f}(a^*, b^*) + dist_{G_f}(b^*, a^*) > 0$ for some $a^*, b^* \in \mathcal{A}$. Assume the perspective of agent i. In the following, we write $dist(\cdot, \cdot)$ instead of $dist_{G_f}(\cdot, \cdot)$ and T instead of T_i for ease of notation. We show the above assumptions imply that T is splittable.

Define $d(a) = dist(a^*, a) + dist(a, a^*)$ for all $a \in \mathcal{A}$. Since the function d takes only countably many values, there exists $z \in \mathbb{R}$ such that the following sets form a non-trivial partition of $\mathcal{A}$: $\mathcal{A}_1 = \{a \in \mathcal{A} \mid d(a) > z\}$, $\mathcal{A}_2 = \{a \in \mathcal{A} \mid d(a) < z\}$. Observe that for every $a_1 \in \mathcal{A}_1$, there exists $\varepsilon(a_1) > 0$ such that $d(a_1) > z + \varepsilon(a_1)$. Similarly, for every $a_2 \in \mathcal{A}_2$, there exists $\varepsilon(a_2) > 0$ such that $d(a_2) < z - \varepsilon(a_2)$.

For $a_1 \in \mathcal{A}_1$, let $r(a_1) = -dist(a_1, a^*)$. For $a_2 \in \mathcal{A}_2$, let $r(a_2) = dist(a^*, a_2) - z$. Now let $t \in T$ such that $f(t) = a_1 \in \mathcal{A}_1$. We claim that for all $a_2 \in \mathcal{A}_2$ it holds that $v(a_1, t) - v(a_2, t) > r(a_1) - r(a_2) + \varepsilon(a_1)$, which proves $t \in \mathcal{T}_1(\varepsilon(a_1))$. Indeed, $v(a_1, t) - v(a_2, t) \geq \ell(a_2, a_1)$. The claim then follows from:

$$\begin{aligned} dist(a^*, a_2) + \ell(a_2, a_1) + dist(a_1, a^*) &\geq dist(a^*, a_1) + dist(a_1, a^*) \\ &> z + \varepsilon(a_1). \end{aligned}$$

Next, let $t \in T$ such that $f(t) = a_2 \in \mathcal{A}_2$. We claim that for all $a_1 \in \mathcal{A}_1$ it holds that $v(a_1, t) - v(a_2, t) < r(a_1) - r(a_2) - \varepsilon(a_2)$, which proves $t \in \mathcal{T}_2(\varepsilon(a_2))$. Again, $v(a_1, t) - v(a_2, t) \leq -\ell(a_1, a_2)$, and the claim follows from:

$$\begin{aligned} & dist(a^*, a_2) + dist(a_1, a^*) - \ell(a_1, a_2) \\ \leq\ & dist(a^*, a_2) + dist(a_1, a^*) - dist(a_1, a^*) + dist(a_2, a^*) \\ =\ & dist(a^*, a_2) + dist(a_2, a^*) \\ <\ & z - \varepsilon(a_2). \end{aligned}$$

□

Using Theorem 2.19, Chung and Olszewski derive Theorem 2.18 for the case of countable outcome spaces and show how the results of prior work mentioned in the introduction (Green and Laffont 1977, Holmström 1979, Krishna and Maenner 2001, Milgrom and Segal 2002) follow from that theorem. In a similar way, those results are implied by Theorem 2.17. We therefore refer to Chung and Ol-

szewski (2007) for a detailed discussion of the mentioned literature. Furthermore, the paper by Suijs (1996) is not mentioned in Chung and Olszewski (2007). Next to proving a characterization theorem for revenue equivalence for the setting with a finite outcome space and the allocation rule being the utilitarian maximizer, Suijs relaxes the smooth path-connectedness condition from Holmström (1979) for this setting. He shows that path-connectedness of type spaces is sufficient. This result follows directly from Corollary 2.13 and the fact that path-connectedness implies connectedness.

2.5.4 A Setting with an Uncountable Outcome Space

In this section, we give an example for an economic setting where Theorem 2.11 can be used to identify revenue equivalence, while all previous results are not applicable.

Cachon and Lariviere (1999) consider demand rationing problems where agents have to share a divisible good. We consider the following variant of the problem. A supplier has one unit of a perfectly divisible good that has to be distributed among n retailers (agents). The type of agent i is his demand $t_i \in (0,1]$. Given the reports $t \in (0,1]^n$ of all agents, an allocation rule $f\colon (0,1]^n \to [0,1]^n$ assigns a fraction of the good to every agent such that $\sum_{i=1}^n f_i(t) \leq 1$. If an agent's demand is met, he incurs a disutility of 0, otherwise his disutility is linear in the amount of unmet demand. More precisely, agent i's valuation[6] if he is assigned quantity q_i is

$$v_i(q_i, t_i) = \begin{cases} 0, & \text{if } q_i \geq t_i, \\ q_i - t_i, & \text{if } q_i < t_i. \end{cases}$$

In this context, payments are reimbursements by the supplier for unmet demand.

Let us call an allocation rule f *dictatorial*, if there is an agent i that always gets precisely his demanded quantity, $f_i(t_i, t_{-i}) = t_i$ for all t_{-i}. We show that any dictatorial rule violates revenue equivalence.

Theorem 2.20. *For the above demand rationing problem, let $f_1(t) = t_1$ and $f_i(t) = (1 - t_1)/(n-1)$. Then f is implementable but does not satisfy revenue equivalence.*

Theorem 2.20 is formulated for the allocation rule that splits the remaining supply equally among agents 2 to n. However, from the proof it can be easily seen

[6] A similar valuation function appears in Holmström (1979) as an example to demonstrate that his smooth path-connectedness assumption cannot be weakened. Likewise, the example can be used to show that the convexity assumption of the valuation function in Krishna and Maenner (2001) cannot be relaxed.

that the conclusion of Theorem 2.20 holds for all other dictatorial rules that are implementable.

Proof. Note that for agents $2, \dots, n$ their assigned quantity does not depend on their report. Therefore, truthful reporting is a (weakly) dominant strategy for those agents. Regard the type graph T_f for agent 1 and note that it does not depend on the report of the other agents. For simplicity we use v, t and f instead of v_1, t_1 and f_1. Let $s, t \in (0, 1]$ with $s < t$. We call (s, t) a *forward arc* and (t, s) a *backward arc*. Then

$$\begin{aligned} \ell_{st} &= v(f(t), t) - v(f(s), t) = v(t, t) - v(s, t) = t - s > 0, \text{ and} \\ \ell_{ts} &= v(f(s), s) - v(f(t), s) = v(s, s) - v(t, s) = 0. \end{aligned}$$

As all arcs have non-negative length, there is no negative cycle and f is implementable. Furthermore, $dist_{T_f}(t, s) = \ell_{ts} = 0$ for $t > s$. We claim that $dist_{T_f}(s, t) = t - s > 0$. To that end, note that all paths from s to t that use only forward arcs have the same length $t - s$. If a path from s to t contains a backward arc, the forward arcs of that path have a total length more than $t - s$. Hence, for all $s < t$ we have $dist_{T_f}(s, t) + dist_{T_f}(t, s) > 0$. Therefore revenue equivalence does not hold according to Theorem 2.11. In fact, $\pi_1(t) = 0$ for all $t \in (0, 1]$ and $\pi_1(t) = t - 1$ for all $t \in (0, 1]$ are two payment schemes for agent 1 that make f dominant strategy incentive compatible. (For the other agents, pick any constant payment scheme.) □

Theorem 2.20 implies that in this setting not all implementable f satisfy revenue equivalence. A theorem describing sufficient conditions for *all* implementable f to satisfy revenue equivalence is necessarily silent here. Nevertheless, we can use Theorem 2.11 to identify properties of allocation rules that guarantee revenue equivalence in this setting. We state such properties formally in Theorem 2.21 below, and then show that for instance the *proportional allocation rule* with $f_i(t) = t_i / \sum_{j=1}^{n} t_j$ satisfies these properties.

Theorem 2.21. *Let $f\colon (0, 1]^n \to [0, 1]^n$ be an allocation rule. If for every agent i and every report t_{-i} of the other agents, f satisfies any of the conditions below, then f is implementable and satisfies revenue equivalence.*

(*i*) *$f_i(t_i, t_{-i})$ is continuous in t_i and $f_i(t_i, t_{-i}) < t_i$ for all $t_i \in (0, 1]$,*

(*ii*) *$f_i(t_i, t_{-i})$ is continuous in t_i and $f_i(t_i, t_{-i}) > t_i$ for all $t_i \in (0, 1)$, $f_i(1, t_{-i}) = 1$,*

(*iii*) $f_i(t_i, t_{-i})$ *is continuous and increasing in* t_i *and has exactly one fixed point* $x \in (0,1]$ *with* $f_i(x, t_{-i}) = x$, $f_i(t_i, t_{-i}) > t_i$ *for all* $t_i \in (0, x)$, $f_i(t_i, t_{-i}) < t_i$ *for all* $t_i \in (x, 1]$.

Proof. Again, fix agent i, t_{-i} and use v, t and f instead of v_i, t_i and f_i. Furthermore, regard f as a function of i's type only. For all three cases, assume that f is continuous. Regard the corresponding type graph T_f.

(*i*) We have $f(t) < t$ for $t \in (0,1]$. Let $s < t$. Using $f(s) < s < t$, the arc lengths in T_f are as follows:

$$\begin{aligned} \ell_{st} &= f(t) - f(s) \text{ and} \\ \ell_{ts} &= \begin{cases} f(s) - f(t), & \text{if } f(t) < s, \\ f(s) - s, & \text{if } f(t) \geq s. \end{cases} \end{aligned}$$

To verify implementability, we prove that any cycle in the type graph has non-negative length. Note that for $s < t$, $\ell_{st} = \ell_{sx} + \ell_{xt}$ for any $s < x < t$. We say that we *split* arc (s,t) at x if we replace (s,t) by arcs (s,x) and (x,t) in a path or cycle. Note that splitting forward arcs does not change the length of the path or cycle. Consider any finite cycle c with nodes c_1 to c_k and rename the nodes such that $c_1 < c_2 < \cdots < c_k$. Split every forward arc (c_u, c_v) with $u < v$ at all intermediate nodes $c_{u+1}, \ldots, c_{v-1}$ and call the resulting cycle c'; the cycle length remains the same. Consider some backward arc (c_v, c_u) with $u < v$. As c' is a cycle, it contains all forward arcs $(c_u, c_{u+1}), \ldots, (c_{v-1}, c_v)$. The length of the sub-cycle $(c_u, c_{u+1}, \ldots, c_{v-1}, c_v, c_u)$ is equal to

$$\ell_{c_u c_{u+1}} + \ldots + \ell_{c_{v-1} c_v} + \ell_{c_v c_u} = \ell_{c_u c_v} + \ell_{c_v c_u} = \begin{cases} f(c_v) - c_u, & \text{if } f(c_v) \geq c_u, \\ 0, & \text{if } c_u > f(c_v), \end{cases}$$

and hence is non-negative. Removing the arcs of the sub-cycle $(c_u, c_{u+1}, \ldots, c_v, c_u)$ from c' leaves a new cycle c'' of smaller or equal length. As c'' is still a cycle we can repeat the argument finitely many times and finally arrive at the empty cycle with length 0. Hence, c has non-negative length and f is implementable.

In order to verify that f satisfies revenue equivalence, we compute distances $dist_{T_f}(s,t)$ and $dist_{T_f}(t,s)$ for $s < t$. Note that $dist_{T_f}(s,t) \leq \ell_{st} = f(t) - f(s)$. We claim that also $dist_{T_f}(t,s) \leq f(s) - f(t)$, so $dist_{T_f}(s,t) + dist_{T_f}(t,s) \leq 0$. Together with implementability we get $dist_{T_f}(s,t) + dist_{T_f}(t,s) = 0$, and revenue equivalence holds by Corollary 2.12. To prove the claim we consider two cases. If $f(t) < s$, $dist_{T_f}(t,s) \leq \ell_{ts} = f(s) - f(t)$ and we are done. If $f(t) \geq s$, consider

the sequence $(x_n)_{n=0}^{\infty} = (t, f(t), f(f(t)), \dots)$. It is monotonically decreasing and bounded and so converges. The sequence $(f(x_n))_{n=0}^{\infty} = (f(t), f(f(t)), f^3(t), \dots)$ converges for the same reason and $\lim_{n\to\infty} x_n = \lim_{n\to\infty} f(x_n) =: x$. As f is continuous, we have $x = f(x)$ if x is in $(0, 1]$. As there is no fixed point in $(0, 1]$, we conclude $x = 0$. Therefore, there exists some smallest index K such that $x_K < s$. Now consider the path $\mathfrak{p}_K$ defined as $(t = x_0, \dots, x_K, s)$. Path $\mathfrak{p}_K$ has length $f^2(t) - f(t) + f^3(t) - f^2(t) + \cdots + f^{K+1}(t) - f^K(t) + f(s) - f^{K+1}(t) = f(s) - f(t)$. We conclude $dist_{T_f}(t, s) \leq length(\mathfrak{p}_K) = f(s) - f(t)$, and we are done.

(ii) We have $f(t) > t$ for $t \in (0, 1)$ and $f(1) = 1$. Let $s < t$. Using $f(t) \geq t > s$, the arc lengths in T_f can be computed as follows:

$$\begin{aligned} \ell_{st} &= \begin{cases} 0, & \text{if } f(s) \geq t; \\ t - f(s), & \text{if } f(s) < t \end{cases} \quad \text{and} \\ \ell_{ts} &= 0. \end{aligned}$$

As all arc lengths are non-negative, all cycles have non-negative length and f is implementable.

Since all paths have non-negative length, we know that $dist_{T_f}(t, s) = \ell_{ts} = 0$. We claim that also $dist_{T_f}(s, t) = 0$, hence revenue equivalence holds by Corollary 2.12. If $f(s) \geq t$, $dist_{T_f}(s, t) = \ell_{st} = 0$ and we are done. If $f(s) < t$, regard the sequence $(x_n)_{n=0}^{\infty} = (s, f(s), f(f(s)), \dots)$. This sequence converges to fixed point $x = 1$ by the same arguments as in *(i)*. Let us regard paths $\mathfrak{p}_k$ defined as $\mathfrak{p}_k = (s = x_0, \dots, x_k, 1)$. As $\ell_{x_i, f(x_i)} = 0$ for $i = 0, \dots, k-1$ and $\ell_{x_k, 1} = 1 - f(x_k)$, the length of $\mathfrak{p}_k$ is $1 - f(x_k)$, which converges to 0 as k increases. Thus, $dist_{T_f}(s, 1) = 0$. With $\ell_{1t} = 0$ it follows that $dist_{T_f}(s, t) = 0$.

(iii) We have that f is increasing on $(0, 1]$ and there is $x \in (0, 1)$ such that $f(x) = x$, $f(t) > t$ for $t \in (0, x)$, and $f(t) < t$ for $t \in (x, 1]$. There are six types of arcs, whose lengths are as follows.

If $s < t \leq x$:	$\ell_{st} = \begin{cases} 0, & \text{if } f(s) \geq t; \\ t - f(s), & \text{if } t > f(s); \end{cases}$	$\ell_{ts} = 0;$
If $s \leq x < t$:	$\ell_{st} = f(t) - f(s);$	$\ell_{ts} = 0;$
If $x < s < t$:	$\ell_{st} = f(t) - f(s);$	$\ell_{ts} = \begin{cases} f(s) - f(t), & \text{if } f(t) < s; \\ f(s) - s, & \text{if } s \leq f(t). \end{cases}$

Implementability can be verified in a manner similar to case *(i)*. It can be checked that splitting forward arcs can only decrease the cycle length. When removing subcycles consisting of a backward arc (t, s), for $s < t$, together with the corresponding

forward arcs, there are different cases. If $x < s < t$, the cycle length is non-negative by arguments similar to case (i). If $s < x < t$ or $s < t < x$, the length of the backward arc is 0 and all forward arcs have non-negative length. Thus, such a sub-cycle has non-negative length and the argument can be continued as in (i).

For revenue equivalence, we consider two cases. If $s < t \leq x$ or $x < s < t$, both s and t lie on a 0-length cycle just like in cases (i) and (ii). Hence, $dist_{T_f}(s,t) + dist_{T_f}(t,s) = 0$. If $s \leq x < t$, an analogue argument as in (ii) yields $dist_{T_f}(s,x) = 0$. With $\ell_{xt} = f(t) - f(x)$, we get $dist_{T_f}(s,t) \leq f(t) - f(x)$. Similarly as in (i), we can show that $dist_{T_f}(t,x) \leq f(x) - f(t)$. Consequently, and since $\ell_{xs} = 0$, we have $dist_{T_f}(t,s) \leq dist_{T_f}(t,x) + \ell_{xs} \leq f(x) - f(t)$. Thus, $dist_{T_f}(s,t) + dist_{T_f}(t,s) \leq 0$ and by implementability, $dist_{T_f}(s,t) + dist_{T_f}(t,s) = 0$. Hence revenue equivalence holds. □

Corollary 2.22. *For the demand rationing problem, the proportional allocation rule f with $f_i(t) = t_i / \sum_{j=1}^{n} t_j$ for $i = 1, \ldots, n$ is implementable and satisfies revenue equivalence.*

Proof. For every agent i and every report of the other agents t_{-i}, the function $f_i(t_i, t_{-i}) = t_i / \sum_{j=1}^{n} t_j$ satisfies the assumptions of either case (i) or (iii) in Theorem 2.21. □

The proportional allocation rule completely distributes the supply among the retailers. In particular, a retailer can get more than his demanded quantity. In an economy without free disposal this may not be desired. In this case, the proportional allocation rule becomes $f_i(t) = \min\{t_i, t_i / \sum_{j=1}^{n} t_j\}$ for $i = 1, \ldots, n$, and the supplier keeps part of the good if the supply exceeds the total demand. We next show that this allocation rule does not satisfy revenue equivalence. The result follows from the following theorem.

Theorem 2.23. *Suppose in the demand rationing problem, f is implementable and there is one agent i and reports t_{-i} with $f_i(\cdot, t_{-i})$ increasing and continuous such that there are $x_1, x_2 \in [0,1]$ with*

$$\begin{aligned} f_i(t_i, t_{-i}) &> t_i \text{ if } t_i < x_1 \\ f_i(t_i, t_{-i}) &= t_i \text{ if } x_1 \leq t_i \leq x_2 \\ f_i(t_i, t_{-i}) &< t_i \text{ if } x_2 < t_i. \end{aligned}$$

Then f does not satisfy revenue equivalence.

Proof. Fix agent i and report vector t_{-i} of the other agents satisfying the assumptions of the Theorem. Regard the corresponding type graph T_f. There are six types of arcs (s,t) with $s < t$ depending on the position of s and t with respect to x_1 and x_2. Similarly, there are six types of arcs (t,s). It can be checked that the arc lengths are as follows.

$s < t \leq x_1$:	$\ell_{st} = \begin{cases} 0, & \text{if } f(s) \geq t; \\ t - f(s), & \text{if } t > f(s); \end{cases}$	$\ell_{ts} = 0$
$x_2 \leq s < t$:	$\ell_{st} = f(t) - f(s)$	$\ell_{ts} = \begin{cases} f(s) - f(t), & \text{if } f(t) < s; \\ f(s) - s, & \text{if } s \leq f(t). \end{cases}$
all other cases:	$\ell_{st} = f(t) - f(s)$	$\ell_{ts} = 0$

Note that the only arcs with negative lengths are (t,s)-arcs for $x_2 \leq s < t$.

That f is implementable can be verified in a manner similar to the other cases. Splitting forward arcs can only reduce the length of a cycle. In order to verify that the total length of a sub-cycle consisting of a backward arc together with the corresponding forward arcs is indeed non-negative, one can check the case $x_2 \leq s < t$ similar to case (i) of Theorem 2.21. In all other cases it is sufficient to note that the lengths of all involved forward and backward arcs are non-negative.

We show that f violates revenue equivalence by proving the existence of types s and t such that $dist_{T_f}(s,t) + dist_{T_f}(t,s) > 0$. In fact, this inequality holds for any $s,t \in [x_1,x_2]$. Let $s,t \in [x_1,x_2]$, $s < t$. Since $\ell_{ts} = 0$, we get $dist_{T_f}(t,s) \leq 0$. If $dist_{T_f}(t,s)$ was in fact smaller than 0, there would exist a path $\mathfrak{p}$ of negative length from t to s. Such a path must contain at least one (t',s')-arc with $x_2 \leq s' < t'$. In this case, the path must contain a sub-path from t to t'. Splitting forward arcs of this sub-path at s' (if necessary) yields a non-negative sub-cycle through s' and t'. The split does not increase the path length. Removing the sub-cycle can only reduce the length of $\mathfrak{p}$. This way, all negative arcs can be removed from $\mathfrak{p}$ without increasing its length. That contradicts the length of $\mathfrak{p}$ being negative. Hence, $dist_{T_f}(t,s) = 0$.

We claim that $dist_{T_f}(s,t) = t - s > 0$. To that end, note that any (s,t)-path that contains a backward arc, can be turned into a path of smaller or equal length without any backward arcs as follows. Suppose that backward arc (t',s') is contained in $\mathfrak{p}$, $s' < t'$. Because $\mathfrak{p}$ is an (s,t)-path for $t > s$, it has to contain forward arcs that completely "pass" the (t',s')-arc in forward direction. Splitting all forward arcs passing s' at s' and splitting all forward arcs passing t' at t' does not increase $\mathfrak{p}$'s length. Again, this splitting procedure results in a sub-cycle consisting of the (t',s')-arc and a forward path from s' to t'. This sub-cycle has non-negative length and can be removed without increasing the length of $\mathfrak{p}$. Hence, we can without loss of generality restrict our attention to (s,t)-paths that contain only forward arcs. Such paths can only contain nodes from $[s,t] \subseteq [x_1,x_2]$. Therefore, the length of all

such paths is equal to $f(t) - f(s) = t - s > 0$, which proves the claim. □

Corollary 2.24. *For the demand rationing problem, the allocation rule f with $f_i(t) = \min\{t_i, t_i / \sum_{j=1}^{n} t_j\}$ for $i = 1, \ldots, n$ is implementable, but does not satisfy revenue equivalence.*

Proof. For any agent i, and any t_{-i}, let $d_{-i} = \sum_{j \neq i} t_j$ be the total demand of the other agents. If $d_{-i} > 1$, f_i satisfies the assumptions of case (i) of Theorem 2.21. If $d_{-i} < 1$, f satisfies the assumptions of Theorem 2.23 for $x_1 = 0$ and $x_2 = 1 - d_{-i}$. Implementability follows from the mentioned theorems. However revenue equivalence does not hold according to Theorem 2.23. □

2.6 Discussion

Our results can be extended to other notions of incentive compatibility. We briefly discuss two of them.

Ex-post incentive compatibility with externalities. The notation used in this chapter is appropriate to model *allocational externalities* in auction settings, as an outcome $a \in \mathcal{A}$ can be used to represent the entire allocation of all the items to the various agents. In order to account for informational externalities in the case of ex-post incentive compatibility, the valuation function of agent i is also a function of the true types of the other agents. An ex-post equilibrium is a Nash-equilibrium where every agent knows the types of all other agents. Formally, truth-telling is an ex-post equilibrium for allocation rule f and payment scheme π if for all agents i, all types t_i and s_i of i and all types t_{-i} of the other agents

$$v_i(f(t_i, t_{-i}), t_i, t_{-i}) - \pi(t_i, t_{-i}) \geq v_i(f(s_i, t_{-i}), t_i, t_{-i}) - \pi(s_i, t_{-i}).$$

Again, incentive compatible payment schemes can be associated with node potentials in the allocation graphs. The arc lengths in G_f have to be defined as follows:

$$\ell_{ab} = \inf_{t_i \in f^{-1}(b)} (v_i(b, t_i, t_{-i}) - v_i(a, t_i, t_{-i})).$$

We get the following result.

Theorem 2.25. *An ex-post implementable allocation rule f satisfies revenue equivalence if and only if in every allocation graph G_f, for every two allocations a, b, $dist_{G_f}(a, b) = -dist_{G_f}(b, a)$.*

Bayes-Nash incentive compatibility. Müller, Perea, and Wolf (2007) already used type graphs for the case of Bayes-Nash incentive compatible allocation rules. An allocation rule f is said to be Bayes-Nash incentive compatible if there is a payment scheme π such that for all agents i and all types t_i and s_i of i the following is true

$$\mathbb{E}_{-i}[v_i(f(t_i, t_{-i}), t_i, t_{-i}) - \pi_i(t_i, t_{-i})] \geq \mathbb{E}_{-i}[v_i(f(s_i, t_{-i}), t_i, t_{-i}) - \pi_i(s_i, t_{-i})],$$

where the expected value is taken with respect to the types of all agents other than i. That is, truth-telling of every agent is a Nash equilibrium when agents try to maximize expected utilities. We use the type graph with arc lengths defined as follows:

$$\ell_{s_i t_i} = \mathbb{E}_{-i}[v_i(f(t_i, t_{-i}), t_i, t_{-i}) - v_i(f(s_i, t_{-i}), t_i, t_{-i})].$$

Note that by taking expectations, there is a single type graph for every agent. Here, we analyze expected payment schemes $\mathbb{E}_{-i}(\pi_i(\cdot, t_{-i}))$, and an allocation rule satisfies revenue equivalence if expected payments are unique up to a constant.

Theorem 2.26. *A Bayes-Nash implementable allocation rule f satisfies revenue equivalence with respect to expected payments if and only if in every type graph T_f for all types s and t of this agent, $dist_{T_f}(s,t) = -dist_{T_f}(t,s)$.*

Incentive compatibility, monotonicity and potential functions. In settings with multi-dimensional type spaces which are subsets of $\mathbb{R}^k$, finite $\mathcal{A}$, and valuation functions that are linear in types, it was established by several authors that implementability, and thus the non-negative cycle property, is sometimes implied by a monotonicity property, termed weak monotonicity in Bikhchandani, Chatterjee, Lavi, Mu'alem, Nisan, and Sen (2006). For dominant strategy incentive compatibility see Bikhchandani et al., Gui, Müller, and Vohra (2004), Saks and Yu (2005), and Monderer (2007), for Bayes-Nash incentive compatibility see Müller, Perea, and Wolf (2007). In these settings, implementability implies that the allocation rule f, viewed from a single agent perspective as a vector field that maps multi-dimensional types on (lotteries over) outcomes, has a potential function F. One can easily verify that this property has the following interpretation on type graphs: the length of a shortest path in T_f from some type s to some type t is upper bounded by a path integral of the vector field f, or equivalently $F(t) - F(s)$. From this it follows easily that $dist_{T_f}(s,t) = -dist_{T_f}(t,s)$, i.e., revenue equivalence holds. In particular, $dist_{T_f}(s,t) = F(t) - F(s)$ for any potential function F. This connection between implementability, potential functions and revenue equivalence is also established in

Jehiel, Moldovanu, and Stacchetti (1996), Jehiel, Moldovanu, and Stacchetti (1999), Jehiel and Moldovanu (2001) and Krishna and Maenner (2001).

Chapter 3

Optimal Mechanisms for Single Machine Scheduling

We study the design of optimal mechanisms in a setting where job-agents compete for being processed by a service provider that can handle one job at a time. Each job has a processing time and incurs a waiting cost. Jobs need to be compensated for waiting. We consider two models, one where only the waiting costs of jobs are private information (1-d), and another where both waiting costs and processing times are private (2-d). Probability distributions represent the public common belief about private information. We consider discrete and continuous distributions. In this setting, an optimal mechanism minimizes the total expected expenses to compensate all jobs, while it has to be Bayes-Nash incentive compatible. We derive closed formulae for the optimal mechanism in the 1-d case and show that it is efficient for symmetric jobs. For non-symmetric jobs, we show that efficient mechanisms perform arbitrarily bad. For the 2-d discrete case, we prove that the optimal mechanism in general does not even satisfy IIA, the 'independent of irrelevant alternatives' condition. Hence any attempt along the lines of the classical auction setting is doomed to fail. In the 2-d discrete case, we also show that the optimal mechanism is not even efficient for symmetric agents.[1]

[1]Part of the results of this chapter were published in Heydenreich, Mishra, Müller, and Uetz (2008).

3.1 Introduction

The design of optimal auctions is recognized as an intriguing issue in auction theory; first studied by Myerson (1981) for the case of single item auctions. In that setting, the goal is to maximize the seller's revenue. We study the design of optimal auctions (or more precisely, mechanisms) in a setting where job-agents compete for being processed by a service provider that can only handle one job at a time. No job can be interrupted once started, and each job is characterized by service time and weight, the latter representing its disutility for waiting per unit time. It is well known that the total disutility of the jobs is minimized by a scheduling policy known as Smith's rule: schedule jobs in order of non-increasing ratios of weight over service time (Smith 1956).

Our Contribution. We consider different cases. In the *one-dimensional* (1-d) case, jobs' processing times are public information and a job's weight is only known to the job itself. We further distinguish between the discrete and continuous case. Publicly known probability distributions over a finite set of possible weights represent common beliefs about the weights in the discrete case. For the continuous case, we regard continuous probability distributions. In the *two-dimensional* (2-d) case, both weights and processing times are private information of the jobs. For all different settings, we aim at finding Bayes-Nash incentive compatible mechanisms that minimize the expected expenses of the service provider. Given jobs' reports about their private information, a mechanism determines both an order in which jobs are served, and for each job a payment that the job receives. The payment can be seen as a compensation for waiting. By a graph theoretic interpretation of the incentive compatibility constraints - as used e.g. by Rochet (1987), Malakhov and Vohra (2007) and in Chapter 2 of this thesis - we show how to derive optimal mechanisms. For the one-dimensional discrete and continuous case, we obtain closed formulae for modified job weights, and show that serving the jobs in the order of non-increasing ratios of these modified weights over service times is optimal for the service provider, as long as a certain regularity condition is fulfilled. It turns out that the optimal mechanism is not necessarily efficient, i.e., in general it does not maximize total utility. But it does so if e.g. all jobs have identical weight distributions and equal processing times. We call such jobs symmetric. For non-symmetric jobs with discrete weights, we show by example that the cost can be arbitrarily far from optimal if we insist on efficiency. We also compare our optimal mechanism to the generalized VCG mechanism and see that for discrete weights, expected payments differ even for the case of symmetric jobs. For continuous weights, however, revenue equivalence applies (see Chapter 2) and the generalized VCG mechanism is an optimal mechanism for symmetric jobs. Furthermore, we analyze a mechanism

in the continuous setting that corresponds to the first price auction and we show that this yields another optimal mechanism. For the two-dimensional discrete case, our main result is that the optimal mechanism generally does not satisfy a property called IIA, 'independence of irrelevant alternatives'. From that we conclude that the optimal mechanism cannot be expressed in terms of modified weights along the lines of the 1-d case. In fact, any kind of priority based list scheduling algorithm where the priorities of a job depend only on the characteristics of that job itself cannot in general be an optimal mechanism. We conclude that optimal mechanism design for the two-dimensional case is substantially more involved than two-dimensional mechanism design for auction settings, as studied in Malakhov and Vohra (2007). We also show that even for symmetric jobs, in the 2-d case the optimal mechanism is not efficient.

Related Work. Optimal mechanism design goes back to Myerson (1981). He studies optimal mechanisms for single item auctions and continuous 1-dimensional type spaces. Here, optimal auctions are modifications of efficient auctions, more specifically, modifications of the Vickrey auction. When regarding the seller as additional agent who bids zero in the original auction, his modified bid might be non-zero in the optimal auction yielding a reservation price. Malakhov and Vohra (2007) regard optimal mechanisms for an auction setting with discrete 2-dimensional type spaces. The derived optimal mechanisms again employ the efficient allocation rule with modified bids. As Malakhov and Vohra (2007), we follow Myerson's approach and analyze in how far it also works in a simple scheduling setting. We observe similarities and differences, see Section 3.3. Especially, we show that for 2-dimensional type spaces the traditional approach must fail to determine an optimal auction. The fact that multi-dimensional optimal mechanism design is harder than that for 1-dimensional types, is well-known. For example, Armstrong (2000) studies a multi-object auction model where valuations are additive and drawn from a binary distribution (i.e., high or low). He gives optimal auctions under specific conditions that reduce the type graph. From this paper it becomes evident that optimal mechanism design with multi-dimensional discrete types is difficult. For our model, we formalize this difficulty by showing that traditional approaches inevitably yield IIA-mechanisms and that in some cases none of these is optimal. For details, we refer to Section 3.4. In Hartline and Karlin (2007), the authors give an introduction to optimal mechanism design with 1-dimensional continuous types under dominant strategy incentive compatibility. Both Myerson's and our optimal allocation rules turn out to be dominant strategy implementable as well, while they yield optimal mechanisms in the larger class of Bayes-Nash incentive compatible mechanisms. Other scheduling models have been looked at from a different angle

in the economic literature. See, e.g., Mitra (2001) for efficient and budget-balanced mechanism design in a 1-dimensional model and Moulin (2007) for mechanisms that prevent merging and splitting of jobs.

Organization. In Section 3.2, we study the 1-d discrete case and derive closed formulae for an optimal mechanism. We compare optimal to efficient mechanisms in Section 3.3. In Section 3.4, we study the 2-d discrete case and show that known approaches are doomed to fail here. The continuous case is studied in Section 3.5 and standard auction formats for the continuous scheduling model are analyzed in Section 3.6. We conclude with Section 3.7.

3.2 Optimal Mechanisms for the 1-Dimensional Setting

3.2.1 Setting and Preliminaries

Consider a single machine which can handle one job at a time. Let $J = \{1, \dots, n\}$ denote the set of jobs. We regard jobs as selfish agents that act strategically. Each job j has a processing time p_j and a weight w_j. While p_j is publicly known, the actual w_j is private information to job j. We refer to the private information of a job as its type. Jobs share common beliefs about other jobs' types in terms of probability distributions. We assume discrete distribution of weights, that is, agent j's weight w_j follows a probability distribution over the discrete set $W_j = \{w_j^1, \dots, w_j^{m_j}\} \subset \mathbb{R}$, where $w_j^1 < \dots < w_j^{m_j}$. Let ϕ_j be the probability distribution of w_j, that is, $\phi_j(w_j^i)$ denotes the probability associated with w_j^i for $i = 1, \dots, m_j$. Let $\Phi_j(w_j^i) = \sum_{k=1}^{i} \phi_j(w_j^k)$ be the cumulative probability up to w_j^i. Both ϕ_j and Φ_j are public information. We assume that jobs' weights are independently distributed. Let us denote by $W = \Pi_{j \in J} W_j$ the set of all type profiles. For any job j, let $W_{-j} = \Pi_{k \neq j} W_k$. Let ϕ be the joint probability distribution of $w = (w_1, \dots, w_n)$. Then $\phi(w) = \Pi_{j=1}^{n} \phi_j(w_j^{i_j})$ for $w = (w_1^{i_1}, \dots, w_n^{i_n}) \in W$. Let w_{-j} and ϕ_{-j} be defined analogously. For $w_j^i \in W_j$ and $w_{-j} \in W_{-j}$, we denote by (w_j^i, w_{-j}) the type profile where job j has type w_j^i and the types of all other jobs are w_{-j}.

A direct revelation mechanism consists of an allocation rule f and a payment scheme π. Jobs have to report their weights and they might report untruthfully if it suits them. Depending on those reports, the allocation rule selects a *schedule*, i.e., an order in which jobs are processed on the machine. The payment scheme assigns a payment that is made to jobs in order to reimburse them for their waiting cost.

Let $\mathfrak{S} = \{\sigma \mid \sigma \text{ is a permutation of } (1, \dots, n)\}$ denote the set of all feasible schedules. Then the allocation rule is a mapping $f \colon W \to \mathfrak{S}$. For any schedule $\sigma \in \mathfrak{S}$, let σ_j be the position of job j in the ordering of jobs in σ. Then, by $S_j(\sigma) = \sum_{\sigma_k < \sigma_j} p_k$,

we denote the start time or waiting time of job j in σ. If job j has waiting time S_j and actual weight w_j^i, it encounters a valuation of $-w_j^i S_j$. If j additionally receives payment π_j, its total utility is $\pi_j - w_j^i S_j$, i.e., we assume quasi-linear utilities. Let us denote by $ES_j(f, w_j^i) := \sum_{w_{-j} \in W_{-j}} S_j(f(w_j^i, w_{-j}))\phi_{-j}(w_{-j})$ the expected waiting time of job j if it reports weight w_j^i and allocation rule f is applied. Denote by $E\pi_j(w_j^i) := \sum_{w_{-j} \in W_{-j}} \pi_j(w_j^i, w_{-j})\phi_{-j}(w_{-j})$ the expected payment to j. We assume that jobs aim at maximizing their expected utility.

Definition 3.1. *A mechanism (f, π) is* Bayes-Nash incentive compatible *if for every agent j and every two types w_j^i,$w_j^k \in W_j$*

$$E\pi_j(w_j^i) - w_j^i ES_j(f, w_j^i) \geq E\pi_j(w_j^k) - w_j^i ES_j(f, w_j^k) \tag{3.1}$$

under the assumption that all agents apart from j report truthfully. If for allocation rule f there exists a payment scheme π such that (f, π) is Bayes-Nash incentive compatible, then f is called Bayes-Nash implementable. *The payment scheme π is referred to as an* incentive compatible *payment scheme.*

In order to account for individual rationality, we need to guarantee non-negative utilities for all agents that report their true weight. It will be convenient to ensure individual rationality by introducing a so-called dummy weight $w_j^{m_j+1}$, which we add to the type space W_j for every agent j. We assume $ES_j(f, w_j^{m_j+1}) = 0$ and $E\pi_j(w_j^{m_j+1}) = 0$ for all $j \in J$. Furthermore, we impose the incentive constraints $E\pi_j(w_j^i) - w_j^i ES_j(f, w_j^i) \geq E\pi_j(w_j^{m_j+1}) - w_j^i ES_j(f, w_j^{m_j+1})$, which imply that $E\pi_j(w_j^i) - w_j^i ES_j(f, w_j^i) \geq 0$ for any Bayes-Nash incentive compatible mechanism (f, π). Therefore, the dummy weights together with the mentioned assumptions guarantee that individual rationality is satisfied along with the incentive constraints. The dummy weight can be interpreted as an option for any job not to take part in the mechanism.

We next define the notion of monotonicity w.r.t. weights, which is easily shown to be a necessary condition for Bayes-Nash implementability. In our setting, it is even a sufficient condition.

Definition 3.2. *An allocation rule f satisfies* monotonicity w.r.t. weights *or short* monotonicity *if for every agent $j \in J$, $w_j^i < w_j^k$ implies that $ES_j(f, w_j^i) \geq ES_j(f, w_j^k)$.*

Theorem 3.3. *An allocation rule f is Bayes-Nash incentive compatible if and only if it satisfies monotonicity w.r.t. weights.*

Before we give a proof of Theorem 3.3, we introduce the type graph for the Bayes-Nash setting. T_f has node set W_j and contains an arc from any node w_j^i to any other node w_j^k of length

$$\ell_{ik} = w_j^i[ES_j(f, w_j^k) - ES_j(f, w_j^i)].$$

Here, ℓ_{ik} represents the gain in expected valuation for agent j by truthfully reporting type w_j^i instead of lying type w_j^k. The incentive constraints for a Bayes-Nash incentive compatible mechanism (f, π) and job j can be read as

$$E\pi_j(w_j^k) \leq E\pi_j(w_j^i) + w_j^i[ES_j(f, w_j^k) - ES_j(f, w_j^i)] = E\pi_j(w_j^i) + \ell_{ik}.$$

That is, the expected payments $E\pi_j(\cdot)$ constitute a node potential in T_f. According to Müller, Perea, and Wolf (2007) and similarly as in Chapter 2, Bayes-Nash implementability of an allocation rule f is equivalent to the non-negative cycle property of the type graph T_f for any agent j. Monotonicity is equivalent to the fact that there is no negative cycle consisting of only two arcs in T_f. We call this property the *non-negative two-cycle property*. It follows from

$$\begin{aligned} \ell_{ik} + \ell_{ki} &= w_j^i[ES_j(f, w_j^k) - ES_j(f, w_j^i)] + w_j^k[ES_j(f, w_j^i) - ES_j(f, w_j^k)] \\ &= (w_j^i - w_j^k)[ES_j(f, w_j^k) - ES_j(f, w_j^i)]. \end{aligned}$$

The last term is non-negative for all jobs j and any two types w_j^i and w_j^k if and only if monotonicity holds.

Proof (*Theorem 3.3*). All that remains to show is that the non-negative two-cycle property implies the non-negative cycle property. We first show that the arc lengths satisfy a property called *decomposition monotonicity*, i.e., whenever $i < k < l$ then $\ell_{ik} + \ell_{kl} \leq \ell_{il}$ and $\ell_{lk} + \ell_{ki} \leq \ell_{li}$. From that property follows that the length of any cycle can be lower bounded by the lengths of a number of two cycles, which proves the theorem.

Decomposition monotonicity follows from

$$\begin{aligned} \ell_{ik} + \ell_{kl} &= w_j^i[ES_j(f, w_j^k) - ES_j(f, w_j^i)] + w_j^k[ES_j(f, w_j^l) - ES_j(f, w_j^k)] \\ &\leq w_j^i[ES_j(f, w_j^k) - ES_j(f, w_j^i)] + w_j^i[ES_j(f, w_j^l) - ES_j(f, w_j^k)] \\ &= w_j^i[ES_j(f, w_j^l) - ES_j(f, w_j^i)] \\ &= \ell_{il}, \end{aligned}$$

where the inequality follows from monotonicity. Note that everything remains true

if the dummy type is involved, i.e., if $l = m_j + 1$. The inequality $\ell_{lk} + \ell_{ki} \leq \ell_{li}$ follows similarly.

In order to prove the second claim, consider a finite cycle c with nodes c_1 to c_k and rename the nodes such that $c_1 < c_2 < \cdots < c_k$. Replace every arc (c_u, c_v) with $u < v$ by arcs $(c_u, c_{u+1}), (c_{u+1}, c_{u+2}), \ldots, (c_{v-1}, c_v)$. Do the same for all arcs (c_v, c_u) with $u < v$. Call the resulting cycle c'. The cycle length of c' is less than or equal to the length of c, due to decomposition monotonicity. The new cycle c' consists only of two-cycles. Due to monotonicity, those have non-negative length. Hence, c has non-negative length as well. □

3.2.2 Optimal Mechanisms

Let us start by investigating the *efficient* allocation rule for the given setting, i.e., the allocation rule that maximizes the total valuation of agents. It is well known that scheduling in order of non-increasing weight over processing time ratios minimizes the sum of weighted start times $\sum_{j=1}^{n} w_j S_j(f(w))$ for any type profile $w \in W$, and therefore maximizes the total valuation of all agents. This allocation rule is known as Smith's rule (Smith 1956). The optimal mechanism that we derive deploys a slightly different allocation rule, namely Smith's rule with respect to certain modified weights.

Our goal is to set up a mechanism that is Bayes-Nash incentive compatible and among all such mechanisms minimizes the expected total payment that has to be made to the jobs. Given any Bayes-Nash incentive compatible mechanism (f, π), one can obviously substitute the payment scheme by its expected payment scheme yielding $(f, E\pi(\cdot))$ without loosing Bayes-Nash incentive compatibility. Moreover, the expected total payment to the agents remains unchanged under the substitution. Therefore, we restrict focus to mechanisms in which agents always receive a payment which is independent of the specific report of the other agents and of the actual allocation.

Note that, unlike e.g. in Myerson (1981), in the discrete setting considered here, revenue equivalence does not hold. Therefore, there are possibly multiple payment schemes that make an allocation rule incentive compatible. Let f be an allocation rule and let $\pi^f(\cdot)$ be a payment scheme that minimizes expected expenses for the machine among all payment schemes that make f Bayes-Nash incentive compatible. More specifically, $\pi_j^f(w_j^i)$ denotes the payment to agent j declaring weight w_j^i under this optimal payment scheme. Let $P^{min}(f) = \sum_{j \in J} \sum_{w_j^i \in W_j} \phi_j(w_j^i) \pi_j^f(w_j^i)$ be the minimum expected total expenses for allocation rule f. The following lemma specifies the optimal payment scheme for a given allocation rule.

Lemma 3.4. *For a Bayes-Nash implementable allocation rule f, the payment scheme defined by*

$$\pi_j^f(w_j^{m_j+1}) = 0, \quad \pi_j^f(w_j^i) = \sum_{k=i}^{m_j} w_j^k[ES_j(f, w_j^k) - ES_j(f, w_j^{k+1})] \text{ for } i = 1, \ldots, m_j$$

is incentive compatible, individually rational and minimizes the expected total payment made to agents. The corresponding expected total payment is given by

$$P^{min}(f) \quad = \quad \sum_{j \in J} \sum_{i=1}^{m_j} \phi_j(w_j^i) \overline{w}_j^i ES_j(f, w_j^i),$$

where the modified weights $\overline{w}_j$ are defined as follows

$$\overline{w}_j^1 = w_j^1, \quad \overline{w}_j^i = w_j^i + (w_j^i - w_j^{i-1}) \frac{\Phi_j(w_j^{i-1})}{\phi_j(w_j^i)} \text{ for } i = 2, \ldots, m_j.$$

Proof. Let $\mathfrak{p} = (w_j^i = a_0, a_1, \ldots, a_m = w_j^{m_j+1})$ denote a path from w_j^i to $w_j^{m_j+1}$ in the type graph T_f for agent j. Denote by $length(\mathfrak{p})$ the sum of its arc lengths. Let (f, π) be a Bayes-Nash incentive compatible mechanism. Adding up the incentive constraints

$$E\pi_j(a_i) \leq E\pi_j(a_{i-1}) + a_{i-1}[ES_j(f, a_i) - ES_j(f, a_{i-1})] = E\pi_j(a_{i-1}) + \ell_{a_{i-1}a_i}$$

for $i = 1, \ldots, m$ yields

$$E\pi_j(w_j^{m_j+1}) \leq E\pi_j(w_j^i) + length(\mathfrak{p}).$$

Assuming $E\pi_j(w_j^{m_j+1}) = 0$, this is equivalent to $-length(\mathfrak{p}) \leq E\pi_j(w_j^i)$. As f is Bayes-Nash implementable, T_f satisfies the non-negative cycle property. Consequently, we can compute shortest paths in T_f. With $dist(w_j^i, w_j^{m_j+1})$ being the length of a shortest path from w_j^i to $w_j^{m_j+1}$, the above yields $-dist(w_j^i, w_j^{m_j+1}) \leq E\pi_j(w_j^i)$. Therefore, $-dist(w_j^i, w_j^{m_j+1})$ is a lower bound on the expected payment for reporting w_j^i. On the other hand, since we have

$$dist(w_j^i, w_j^{m_j+1}) \leq \ell_{ik} + dist(w_j^k, w_j^{m_j+1})$$

for any two types w_j^i and w_j^k, it follows that

$$-dist(w_j^k, w_j^{m_j+1}) \leq -dist(w_j^i, w_j^{m_j+1}) + \ell_{ik}.$$

Consequently, $-dist(\cdot, w_j^{m_j+1})$ defines a node potential in T_f. Setting $\pi_j^f(w_j^i) = -dist(w_j^i, w_j^{m_j+1})$ therefore yields an incentive compatible payment scheme that minimizes the expected payment to every agent for any reported type of the agent. Consequently, this payment scheme also minimizes the expected total payment to agents. Recall that individual rationality is satisfied along with the incentive constraints.

Since arc lengths in T_f satisfy decomposition monotonicity, a shortest path from w_j^i to $w_j^{m_j+1}$ is the path that includes all intermediate nodes $w_j^{i+1}, \ldots, w^{m_j}$. Observing that $-dist(w_j^{m_j+1}, w_j^{m_j+1}) = 0$ and $-dist(w_j^i, w_j^{m_j+1}) = \sum_{k=i}^{m_j} w_j^k [ES_j(f, w_j^k) - ES_j(f, w_j^{k+1})]\, \forall w_j^i \in W_j \setminus \{w_j^{m_j+1}\}$ proves the first claim.

Next, we compute the minimum expected total payment for allocation rule f.

$$\begin{aligned}
P^{min}(f) &= \sum_{j\in J}\sum_{i=1}^{m_j} \phi_j(w_j^i)\pi_j^f(w_j^i) \\
&= \sum_{j\in J}\sum_{i=1}^{m_j} \phi_j(w_j^i) \sum_{k=i}^{m_j} w_j^k [ES_j(f, w_j^k) - ES_j(f, w_j^{k+1})] \\
&= \sum_{j\in J}\sum_{i=1}^{m_j} \phi_j(w_j^i) \left(\sum_{k=i}^{m_j} w_j^k ES_j(f, w_j^k) - \sum_{k=i+1}^{m_j} w_j^{k-1} ES_j(f, w_j^k) \right) \\
&= \sum_{j\in J}\sum_{i=1}^{m_j} \phi_j(w_j^i) \left(w_j^i ES_j(f, w_j^i) + \sum_{k=i+1}^{m_j} ES_j(f, w_j^k)(w_j^k - w_j^{k-1}) \right) \\
&= \sum_{j\in J} ES_j(f, w_j^1) w_j^1 \phi_j(w_j^1) \\
&\quad + \sum_{j\in J}\sum_{i=2}^{m_j} ES_j(f, w_j^i) \left(\phi_j(w_j^i) w_j^i + (w_j^i - w_j^{i-1}) \sum_{k=1}^{i-1} \phi_j(w_j^k) \right) \\
&= \sum_{j\in J} ES_j(f, w_j^1) w_j^1 \phi_j(w_j^1) \\
&\quad + \sum_{j\in J}\sum_{i=2}^{m_j} ES_j(f, w_j^i) \left(\Phi_j(w_j^i) w_j^i - \Phi_j(w_j^{i-1}) w_j^{i-1} \right)
\end{aligned}$$

Let us define modified weights $\overline{w}_j$ by setting $\overline{w}_j^1 = w_j^1$ and for $i = 2, \ldots, m_j$

$$\begin{aligned}\overline{w}_j^i &= \frac{w_j^i \Phi_j(w_j^i) - w_j^{i-1}\Phi_j(w_j^{i-1})}{\phi_j(w_j^i)} \\ &= \frac{w_j^i \phi_j(w_j^i) + w_j^i \Phi_j(w_j^{i-1}) - w_j^{i-1}\Phi_j(w_j^{i-1})}{\phi_j(w_j^i)} \\ &= w_j^i + (w_j^i - w_j^{i-1})\frac{\Phi_j(w_j^{i-1})}{\phi_j(w_j^i)}.\end{aligned}$$

This yields

$$P^{min}(f) = \sum_{j \in J} \sum_{i=1}^{m_j} \phi_j(w_j^i)\overline{w}_j^i ES_j(f, w_j^i).$$

□

Given the minimum payments per allocation rule, we want to specify the allocation rule f which minimizes $P^{min}(f)$ among all Bayes-Nash implementable allocation rules.

Definition 3.5. *If* $f \in \arg\min\{P^{min}(f) \mid f\colon W \to \mathfrak{S},\ f$ *Bayes-Nash implementable*$\}$*, then we call the mechanism* (f, π^f) *an* optimal mechanism.

We will need the following regularity condition that ensures Bayes-Nash implementability of the allocation rule in our optimal mechanism.

Definition 3.6. *We say that* regularity *is satisfied if for every agent* j *and* $i = 2, \ldots, m_j - 1$

$$w_j^i + (w_j^i - w_j^{i-1})\frac{\Phi_j(w_j^{i-1})}{\phi_j(w_j^i)} \leq w_j^{i+1} + (w_j^{i+1} - w_j^i)\frac{\Phi_j(w_j^i)}{\phi_j(w_j^{i+1})}.$$

This implies that $\overline{w}_j^i < \overline{w}_j^k$ *whenever* $w_j^i < w_j^k$.

Note that regularity is satisfied e.g. if the differences $w_j^i - w_j^{i-1}$ are constant and the distribution has a non-increasing reverse hazard rate[2].

Theorem 3.7. *Let the modified weights be defined as in Lemma 3.4. Let* f *be the allocation rule that schedules jobs in order of non-increasing ratios* $\overline{w}_j/p_j$*. If regularity holds, then* (f, π^f) *is an optimal mechanism.*

[2]The reverse hazard rate of the distribution with pdf ϕ and cdf Φ is defined as $\phi(x)/\Phi(x)$, see e.g. Krishna (2002).

Proof. We show that f is Bayes-Nash implementable and minimizes $P^{min}(f)$ among all Bayes-Nash implementable allocation rules. For any allocation rule f, we can rewrite $P^{min}(f)$ as follows, using independence of weight distributions. Let $W'_j = W_j \setminus \{w_j^{m_j+1}\}$ and $W' = \Pi_{j\in J} W'_j$.

$$\begin{aligned} P^{min}(f) &= \sum_{j\in J} \sum_{w_j^i \in W'_j} \phi_j(w_j^i)\overline{w}_j^i ES_j(f, w_j^i) \\ &= \sum_{j\in J} \sum_{w_j^i \in W'_j} \phi_j(w_j^i)\overline{w}_j^i \sum_{w_{-j}\in W_{-j}} S_j(f(w_j^i, w_{-j}))\phi_{-j}(w_{-j}) \\ &= \sum_{j\in J} \sum_{(w_j^i, w_{-j})\in W'} \phi(w_j^i,\, w_{-j})\overline{w}_j^i S_j(f(w_j^i, w_{-j})) \\ &= \sum_{w\in W'} \phi(w) \sum_{j\in J} \overline{w}_j S_j(f(w)). \end{aligned}$$

Thus, $P^{min}(f)$ can be minimized by minimizing $\sum_{j\in J} \overline{w}_j S_j(f(w))$ for every reported type profile w. This is achieved by using Smith's rule with respect to modified weights, i.e., scheduling in order of non-increasing ratios $\overline{w}_j/p_j$. Under Smith's rule, the expected start time $ES_j(w_j)$ is clearly non-increasing in the modified weight $\overline{w}_j$. The regularity condition ensures that it is non-increasing in the original weights w_j. Therefore, Smith's rule with respect to modified weights satisfies monotonicity and is hence Bayes-Nash implementable by Theorem 3.3. This completes the proof. □

3.3 Optimality versus Efficiency

For agents with identical weight distributions and equal processing times, the optimal and the efficient allocation coincide.

Corollary 3.8. *If agents are symmetric, i.e., $W_1 = \cdots = W_n$, $\phi_1 = \cdots = \phi_n$ and $p_1 = \cdots = p_n$ and if distributions are such that regularity holds, then the optimal mechanism is efficient.*

Proof. If $W_1 = \cdots = W_n = \{w^1, \ldots, w^m\}$ and $\phi_1 = \cdots = \phi_n$, then for any two agents j_1 and j_2, and $i = 1, \ldots, m$, the modified weights are equal, i.e. $\overline{w}_{j_1}^i = \overline{w}_{j_2}^i$. Since processing times are also equal and since regularity guarantees that modified weights are increasing in the original weights, scheduling jobs in order of their non-increasing ratios w_j/p_j is equivalent to scheduling them in order of their non-increasing ratios $\overline{w}_j/p_j$. That is, the efficient allocation rule and the allocation rule from the optimal mechanism in Theorem 3.7 coincide. □

If weight distributions differ among agents or if agents have different processing times, then the optimal mechanism is in general not efficient. In fact, when restricting to efficient mechanisms, the total expected payment can be arbitrarily bad in comparison to the optimal one. This is illustrated by the following two examples.

Example 3.9. *Let there be two jobs 1 and 2 with* $W_1 = \{M+1\}$ *and* $W_2 = \{1, M\}$ *for some constant* M*. Let* $\phi_2(1) = 1 - 1/M$*,* $\phi_2(M) = 1/M$ *and* $p_1 = p_2 = 1$*. Let Eff be the efficient and Opt be the optimal allocation rule. Then the ratio* $P^{min}(Eff)/P^{min}(Opt)$ *goes to infinity as* M *goes to infinity.*

Proof. The efficient allocation rule, Smith's rule, always allocates job 1 first. So the optimal payment for Smith's rule is to pay 0 to job 1 and to pay M to job 2, irrespective of its type. The minimum expected total payment is hence $P^{min}(Eff) = M$.

For the optimal allocation, we compute modified weights after Lemma 3.4: $\overline{w}_1^1 = w_1^1 = M+1$, $\overline{w}_2^1 = w_2^1 = 1$ and $\overline{w}_2^2 = M + (M-1)(1-1/M)/(1/M) = M^2 - M + 1$. The latter is larger than $M+1$ if $M > 2$. Therefore, job 2 is scheduled in front of job 1 if it has weight M and behind if it has weight 1. The expected start times for job 2 are $ES_2(Opt, 1) = 1$ and $ES_2(Opt, M) = 0$, respectively. Optimal payments according to Lemma 3.4 are $\pi_2^{Opt}(1) = 1$ and $\pi_2^{Opt}(M) = 0$. For job 1, the expected start time is $ES_1(Opt, M+1) = 1/M$ and the expected payment $\pi_1^{Opt}(M+1) = 1 + 1/M$. Hence, $P^{min}(Opt) = 1 + 1/M + 1 \cdot (1 - 1/M) = 2$.

Consequently, $P^{min}(Eff)/P^{min}(Opt) = M/2$, which tends to infinity if M goes to infinity. □

Remark 3.10. *In the above, the ratio of the expected payments of the efficient versus the optimal allocation rule is analyzed. Similarly, we can derive that the expected ratio of the payments tends to infinity as* M *approaches infinity. The latter is slightly more technical.*

Example 3.11. *Let there be two jobs 1 and 2 with the same weight distribution* $W_1 = W_2 = \{1, M\}$*,* $\phi_j(1) = 1 - 1/M$*,* $\phi_j(M) = 1/M$ *for* $j = 1, 2$*. Let* $p_1 = 1/2$ *and* $p_2 = M/2 + 1$*. Let Eff be the efficient and Opt be the optimal allocation rule. Then the ratio* $P^{min}(Eff)/P^{min}(Opt)$ *goes to infinity as* M *goes to infinity.*

Proof. The efficient allocation rule always schedules job 1 first, since $1/(1/2) = 2 > 2M/(M+2) = M/(M/2+1)$. Therefore, the expected start time of job 1 is 0 and that of job 2 is 1/2. Optimal payments according to Lemma 3.4 are $\pi_1^{Eff}(1) = \pi_1^{Eff}(M) = 0$ and $\pi_2^{Eff}(1) = \pi_2^{Eff}(1) = M/2$. Hence, $P^{min}(Eff) = M/2$.

For the optimal mechanism, we compute modified weights as $\overline{w}_1^1 = \overline{w}_2^1 = 1$ and $\overline{w}_1^2 = \overline{w}_2^2 = M^2 - M + 1$. Job 1 is scheduled first, whenever both jobs have the same weight or job 1 has a larger weight than job 2. In the case where job 1 has (modified) weight 1 and job 2 has modified weight $M^2 - M + 1$, job 2 is scheduled first for $M > 2$, since $1/(1/2) < (M^2 - M + 1)/(M/2 + 1)$. The resulting expected start times and payments are given below:

$$\begin{aligned}
&ES_1(Opt, 1) = 1/2 + 1/M && \pi_1^{Opt}(1) = 1/2 + 1/M \\
&ES_1(Opt, M) = 0 && \pi_1^{Opt}(M) = 0 \\
&ES_2(Opt, 1) = 1/2 && \pi_2^{Opt}(1) = 1 - 1/(2M) \\
&ES_2(Opt, M) = 1/(2M) && \pi_2^{Opt}(M) = 1/2.
\end{aligned}$$

Hence,

$$\begin{aligned}
P^{min}(Opt) &= (\frac{1}{2} + \frac{1}{M})(1 - \frac{1}{M}) + (1 - \frac{1}{2M})(1 - \frac{1}{M}) + \frac{1}{2} \cdot \frac{1}{M} \\
&= (1 - \frac{1}{M})(\frac{3}{2} + \frac{1}{2M}) + \frac{1}{2} \cdot \frac{1}{M}.
\end{aligned}$$

Thus, the ratio $P^{min}(Eff)/P^{min}(Opt)$ tends to infinity if M tends to infinity. □

Remark 3.12. *As in the first example, it can be shown that also the expected ratio of the payments tends to infinity as M approaches infinity.*

Comparison to Myerson's result. For the single item auction and continuous type spaces, Myerson (1981) has made similar observations: in his setting, the Vickrey auction is an efficient auction. The optimal auction can be seen as a modified Vickrey auction with the seller submitting a bit himself. In our setting also, the allocation in the optimal mechanism is equivalent to the efficient allocation rule with respect to modified data. Nevertheless, in Myerson (1981) the optimal and the efficient mechanism may differ. For the single item auction this can be due to the seller keeping the item (even in the symmetric case) or because a bidder that has not submitted the highest bid can get the item in the asymmetric case. In our setting, the optimal and the efficient mechanism can only differ if agents are asymmetric, see Corollary 3.8 and Examples 3.9 and 3.11.

On the generalized VCG Mechanism. The VCG mechanism is due to Vickrey (1961), Clarke (1971) and Groves (1973). The allocation rule is the efficient one. In our setting this means scheduling in order of non-increasing ratios w_j/p_j. The payment scheme pays to agent j an amount that is equal to an appropriate constant (possibly depending on other agents' types, but not on j's type) minus

the total loss in valuation of the other agents due to j's presence. For agent j with processing time p_j, the total loss in valuation of the other agents is equal to the product of p_j and the total weight of all agents processed after j. In order to ensure individual rationality, we have to add p_j times the total weight of all agents except j. Therefore, the resulting payment to j for reported type profile w and efficient schedule σ is equal to

$$\pi_j^{VCG}(w) = p_j \sum_{\substack{k \in J \\ \sigma_k < \sigma_j}} w_k.$$

As illustrated by examples 3.9 and 3.11, the allocation of the VCG mechanism can differ from the allocation of the optimal mechanism if agents are not symmetric. Moreover, if agents are symmetric, the VCG mechanism still can be non-optimal in terms of payments. This is illustrated by the following example.

Example 3.13. *There are two symmetric agents with $W_1 = W_2 = \{w^1, w^2\}$, $w^1 < w^2$, and $\phi_j(w^1) = \phi_j(w^2) = 1/2$ for $j = 1, 2$. Processing times are equal and without loss of generality $p_1 = p_2 = 1$. Then the expected expenses of the VCG mechanism are strictly higher than those of the optimal mechanism.*

Proof. Regularity is trivially satisfied and therefore the allocation of the optimal mechanism from Section 3.2 is efficient. There are four possible type profiles, each occurring with probability 1/4: (w^1, w^1), (w^1, w^2), (w^2, w^1), (w^2, w^2). The resulting schedules are the same for the VCG and the optimal mechanism and schedule the job with the higher weight first or randomize uniformly in the case of equal weights, respectively. Let us first compute the expected total payment for the VCG mechanism. The VCG mechanism pays to the job that is scheduled last the weight of the job that is scheduled before it. Thus, the VCG mechanism has to spend w^1 in the first case, and w^2 in the second, third and fourth case, respectively. The total expected payment of the VCG mechanism is hence $(3w^2+w^1)/4$. Let (f, π^f) denote the optimal mechanism from Section 3.2. In the optimal mechanism, the expected payment to a job with weight w^1 is equal to $E\pi_j^f(w^1) = w^1[ES_j(f, w^1) - ES_j(f, w^2)] + w^2 ES_j(f, w^2) = w^1[3/4 - 1/4] + w^2[1/4] = w^1/2 + w^2/4$. The expected payment to a job with weight w^2 is $E\pi_j^f(w^2) = w^2 ES_j(f, w^2) = w^2/4$. The total expected payment for the optimal mechanism is thus $2 \cdot 1/2 \cdot (w^1/2 + w^2/4 + w^2/4) = (w^1 + w^2)/2$. Since $w^2 > w^1$, the expected expenses of the VCG mechanism are strictly higher than those of the optimal mechanism. Therefore, the VCG mechanism is not optimal. □

3.4 The 2-Dimensional Setting

3.4.1 Setting and Notation

In contrast to the 1-dimensional setting, both weight and processing time of a job are now private information of the job. Hence j's type is the tuple (w_j, p_j). We restrict attention to discrete type spaces, i.e., $(w_j, p_j) \in W_j \times P_j$, where $W_j = \{w_j^1, \ldots, w_j^{m_j}\}$ with $w_j^1 \leq \cdots \leq w_j^{m_j}$ and $P_j = \{p_j^1, \ldots, p_j^{q_j}\}$ with $p_j^1 \leq \cdots \leq p_j^{q_j}$. Let ϕ_j be the probability distribution of j's type, that is, $\phi_j(w_j^i, p_j^k)$ denotes the probability associated with the type (w_j^i, p_j^k) for $i = 1, \ldots, m_j$ and $k = 1, \ldots, q_j$. Both ϕ_j and Φ_j are public. Distributions are independent between agents. Denote by $T = \Pi_{j \in J}(W_j \times P_j)$ the set of all type profiles. For any job j, let $T_{-j} = \Pi_{r \neq j}(W_r \times P_r)$ be the set of type profiles of all jobs except j. Let ϕ be the joint probability distribution of $(w_1, p_1, \ldots, w_n, p_n)$. Then for type profile $t = (w_1^{i_1}, p_1^{k_1}, \ldots, w_n^{i_n}, p_n^{k_n}) \in T$, $\phi(t) = \Pi_{j=1}^n \phi_j(w_j^{i_j}, p_j^{k_j})$. Let t_{-j} and ϕ_{-j} be defined analogously. For $(w_j^i, p_j^k) \in W_j \times P_j$ and $t_{-j} \in T_{-j}$, we denote by $((w_j^i, p_j^k), t_{-j})$ the type profile where job j has type (w_j^i, p_j^k) and the types of the other jobs are represented by t_{-j}. Denote by $ES_j(f, w_j^i, p_j^k) := \sum_{t_{-j} \in T_{-j}} S_j(f((w_j^i, p_j^k), t_{-j}))\phi_{-j}(t_{-j})$ the expected waiting time of job j if it reports type (w_j^i, p_j^k) and allocation rule f is applied. Denote by $E\pi_j(w_j^i, p_j^k) := \sum_{t_{-j} \in T_{-j}} \pi_j((w_j^i, p_j^k), t_{-j})\phi_{-j}(t_{-j})$ the expected payment to j.

We assume that a job can only report a processing time that is not lower than its true processing time and that a job is processed for its reported processing time. This is a natural assumption, since a job can add unnecessary work to achieve a longer processing time, but reporting a shorter processing time can easily be punished by preempting the job after the declared processing time (before it is actually finished).

Note that by regarding the processing time as private information, we introduce informational externalities: job j has a different valuation for a schedule if the processing time (and hence the type) of a job scheduled before j changes. In this regard, our model differs from the 2-dimensional auction model studied in Malakhov and Vohra (2007).

3.4.2 Bayes-Nash Implementability and the Type Graph

Definition 3.14. *A mechanism (f, π) is called* Bayes-Nash incentive compatible *if for every agent j and every two types $(w_j^{i_1}, p_j^{k_1})$ and $(w_j^{i_2}, p_j^{k_2})$ with $i_1, i_2 \in \{1, \ldots, m_j\}$, $k_1, k_2 \in \{1, \ldots, q_j\}$, $k_1 \leq k_2$,*

$$E\pi_j(w_j^{i_1}, p_j^{k_1}) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^{k_1}) \geq E\pi_j(w_j^{i_2}, p_j^{k_2}) - w_j^{i^1} ES_j(f, w_j^{i_2}, p_j^{k_2}) \quad (3.2)$$

under the assumption that all agents apart from j report truthfully.

Note that by defining the incentive constraints only for $k_1 \leq k_2$, we account for the fact that agents can only overstate their processing time, but cannot understate it.

In order to ensure individual rationality, again add a dummy type t_j^d to the type space for every agent j, and let $ES_j(f, t_j^d) = 0$ and $E\pi_j(t_j^d) = 0$ for all $j \in J$. As in the 1-dimensional case, the dummy types together with the mentioned extra incentive constraints guarantee that individual rationality is satisfied along with the incentive constraints. Sometimes, it will be convenient to write $(w_j^{m_j+1}, p_j^k)$ for some $k \in \{1, \ldots, q_j\}$ instead of t_j^d.

In the 2-dimensional setting, the type graph T_f of agent j has node set $W_j \times P_j$ and contains an arc from any node $(w_j^{i_1}, p_j^{k_1})$ to every other node $(w_j^{i_2}, p_j^{k_2})$ with $i_1 \in \{1, \ldots, m_j\}$, $i_2 \in \{1, \ldots, m_j + 1\}$, $k_1, k_2 \in \{1, \ldots, q_j\}$, $k_1 \leq k_2$ of length

$$\ell_{(i_1 k_1)(i_2 k_2)} = w_j^{i_1}[ES_j(f, w_j^{i_2}, p_j^{k_2}) - ES_j(f, w_j^{i_1}, p_j^{k_1})].$$

Note that we have arcs only in direction of increasing processing times, since agents can only overstate their processing time. Furthermore, every node has an arc to the dummy type, but there are no outgoing arcs from the dummy type.

Similar as in Malakhov and Vohra (2007), one can show that for monotonic allocation rules some arcs in the type graph are not necessary, since the corresponding incentive constraints are implied by others. We first give the definition of monotonicity in the 2-dimensional setting and then formulate a lemma which reduces the set of necessary incentive constraints.

Definition 3.15. *An allocation rule f satisfies* monotonicity w.r.t. weights *if for every agent $j \in J$ and fixed $p_j^k \in P_j$, $w_j^{i_1} < w_j^{i_2}$ implies that $ES_j(f, w_j^{i_1}, p_j^k) \geq ES_j(f, w_j^{i_2}, p_j^k)$.*

Lemma 3.16. *Let f be an allocation rule satisfying monotonicity w.r.t. weights. For any agent j, the following constraints imply all other incentive constraints:*

$$E\pi_j(w_j^i, p_j^k) - w_j^i ES_j(f, w_j^i, p_j^k) \geq E\pi_j(w_j^{i+1}, p_j^k) - w_j^i ES_j(f, w_j^{i+1}, p_j^k) \quad (3.3)$$
$$\text{for } i \in \{1, \ldots, m_j\}, k \in \{1, \ldots, q_j\},$$

$$E\pi_j(w_j^{i+1}, p_j^k) - w_j^{i+1} ES_j(f, w_j^{i+1}, p_j^k) \geq E\pi_j(w_j^i, p_j^k) - w_j^{i+1} ES_j(f, w_j^i, p_j^k) \quad (3.4)$$
$$\text{for } i \in \{1, \ldots, m_j - 1\}, k \in \{1, \ldots, q_j\},$$

$$E\pi_j(w_j^i, p_j^k) - w_j^i ES_j(f, w_j^i, p_j^k) \geq E\pi_j(w_j^i, p_j^{k+1}) - w_j^i ES_j(f, w_j^i, p_j^{k+1}) \quad (3.5)$$
$$\text{for } i \in \{1, \ldots, m_j\}, k \in \{1, \ldots, q_j - 1\}.$$

Proof. For any $i_1, i_2, i_3 \in \{1, \dots, m_j + 1\}$,$i_1 < i_2 < i_3$, and any $k \in \{1, \dots, q_j\}$ the constraint

$$E\pi_j(w_j^{i_1}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^k) \geq E\pi_j(w_j^{i_3}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_3}, p_j^k)$$

is implied by

$$E\pi_j(w_j^{i_1}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^k) \geq E\pi_j(w_j^{i_2}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^k)$$

and

$$E\pi_j(w_j^{i_2}, p_j^k) - w_j^{i_2} ES_j(f, w_j^{i_2}, p_j^k) \geq E\pi_j(w_j^{i_3}, p_j^k) - w_j^{i_2} ES_j(f, w_j^{i_3}, p_j^k).$$

In fact, adding up the latter two constraints yields

$$\begin{aligned}
& E\pi_j(w_j^{i_1}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^k) \\
\geq\ & E\pi_j(w_j^{i_3}, p_j^k) + w_j^{i_2}(ES_j(f, w_j^{i_2}, p_j^k) - ES_j(f, w_j^{i_3}, p_j^k)) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^k) \\
\geq\ & E\pi_j(w_j^{i_3}, p_j^k) + w_j^{i_1}(ES_j(f, w_j^{i_2}, p_j^k) - ES_j(f, w_j^{i_3}, p_j^k)) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^k) \\
=\ & E\pi_j(w_j^{i_3}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_3}, p_j^k),
\end{aligned}$$

where the second inequality follows from monotonicity and $w_j^{i_1} < w_j^{i_2}$. Note that everything remains true if the dummy type is involved, i.e., if $(w_j^{i_3}, p_j^k) = (w_j^{m_j+1}, p_j^k) = t_j^d$. These arguments imply that all constraints of the type

$$E\pi_j(w_j^{i_1}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^k) \geq E\pi_j(w_j^{i_2}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^k) \tag{3.6}$$

are implied by the subset of constraints where $i_2 = i_1 + 1$.

A similar effect can be shown for the "reverse" incentive constraints, i.e., the above constraints for $i_3 < i_2 < i_1$, where $i_1, i_2, i_3 \in \{1, \dots, m_j\}$. Again, out of all constraints of the type

$$E\pi_j(w_j^{i_1}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^k) \geq E\pi_j(w_j^{i_2}, p_j^k) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^k), \tag{3.7}$$

only those with $i_2 = i_1 - 1$ are necessary.

Similarly, out of all constraints of the type

$$E\pi_j(w_j^i, p_j^{k_1}) - w_j^i ES_j(f, w_j^i, p_j^{k_1}) \geq E\pi_j(w_j^i, p_j^{k_2}) - w_j^i ES_j(f, w_j^i, p_j^{k_2}), \tag{3.8}$$

for $i \in \{1, \dots, m_j\}$, $k_1, k_2 \in \{1, \dots, q_j\}$, $k_1 < k_2$ only those with $k_2 = k_1 + 1$ are necessary.

For any types $(w_j^{i_1}, p_j^{k_1})$,$(w_j^{i_2}, p_j^{k_2})$ with $i_1 < i_2$ and $k_1 < k_2$ the corresponding "diagonal" constraint

$$E\pi_j(w_j^{i_1}, p_j^{k_1}) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^{k_1}) \geq E\pi_j(w_j^{i_2}, p_j^{k_2}) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^{k_2})$$

follows by adding up the corresponding constraints of type (3.8) and (3.6)

$$E\pi_j(w_j^{i_1}, p_j^{k_1}) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^{k_1}) \geq E\pi_j(w_j^{i_1}, p_j^{k_2}) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^{k_2})$$

and

$$E\pi_j(w_j^{i_1}, p_j^{k_2}) - w_j^{i_1} ES_j(f, w_j^{i_1}, p_j^{k_2}) \geq E\pi_j(w_j^{i_2}, p_j^{k_2}) - w_j^{i_1} ES_j(f, w_j^{i_2}, p_j^{k_2}).$$

For any $(w_j^{i_1}, p_j^{k_1})$,$(w_j^{i_2}, p_j^{k_2})$ with $i_2 < i_1$ and $k_1 < k_2$, the corresponding "diagonal" constraint follows by adding up the appropriate constraints of type (3.8) and (3.7). □

Lemma 3.16 is in fact a generalization of decomposition monotonicity as discussed for the 1-dimensional case.

We define the reduced type graph of agent j, which contains only arcs that are necessary in the sense of Lemma 3.16. These arcs are:

- an arc from type (w_j^i, p_j^k) to (w_j^{i+1}, p_j^k) for all $i \in \{1, \dots, m_j\}$ and $k \in \{1, \dots, q_j\}$
- an arc from type (w_j^{i+1}, p_j^k) to (w_j^i, p_j^k) for all $i \in \{1, \dots, m_j - 1\}$ and $k \in \{1, \dots, q_j\}$
- an arc from type (w_j^i, p_j^k) to (w_j^i, p_j^{k+1}) for all $i \in \{1, \dots, m_j\}$ and $k \in \{1, \dots, q_j - 1\}$.

A sketch of the reduced type graph is given in Figure 3.1. Expected payments correspond to node potentials in the reduced type graph. Whenever we refer to the type graph T_f for a monotonic allocation rule f in the following, the reduced type graph is meant. The reduced type graph comes handy particularly when considering our (counter) examples in the next subsection.

We finally give the characterization of Bayes-Nash incentive compatible allocation rules for the 2-dimensional setting.

Theorem 3.17. *An allocation rule f is Bayes-Nash incentive compatible in the 2-dimensional setting if and only if it satisfies monotonicity w.r.t. weights.*

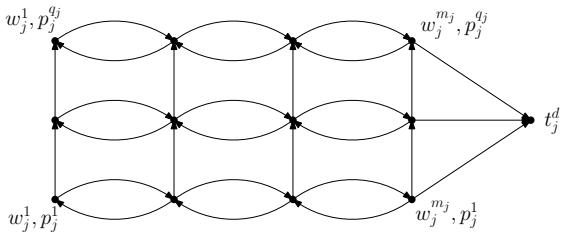

Figure 3.1: reduced type graph

Proof. Implementability implies monotonicity as before. The claim reduces to showing that in the (reduced) type graph of any agent j the non-negative cycle property is equivalent to the non-negative two-cycle property. After the reduction, every cycle in T_f consists of a finite number of two-cycles. Hence the non-negative cycle property is equivalent to the non-negative two-cycle property.

□

3.4.3 On Optimal Mechanisms

We start by reviewing an approach to two-dimensional optimal mechanism design studied in Malakhov and Vohra (2007). Here, the authors regard a multi-item auction, where each agent's type (i, j) is given by a marginal valuation i per item and a capacity j. Above that capacity, the agent has zero valuation for each additional item. Agents can only overstate their capacity. The goal is revenue maximization. Bayes-Nash implementability is equivalent to the expected amount of items allocated to an agent being monotone in his reported value for i. Malakhov and Vohra (2007) use the type graph approach as follows.

First, they regard a subset of all allocation rules - namely those that are monotone in j as well. It turns out that all those rules have the same shortest path tree, namely the "up-first-then-right" tree (see Figure 3.2 for a 3×3 example).

Second, the path lengths in this tree yield optimal payments to every job for every type. From that, the optimal revenue for a particular allocation rule is obtained as closed formula in terms of modified marginal valuations.

Third, the obtained expression for the revenue is maximized over *all* allocation rules. The resulting allocation rule is a modification of the efficient allocation rule. In addition, this rule turns out to be monotone in j, similar as in the proof of Theorem 3.7. Hence, its shortest path tree is the up-first-then-right tree.

In the last step, the monotonicity assumption in j is relaxed as follows. For any allocation rule – not necessarily monotone in j – the up-first-then-right tree

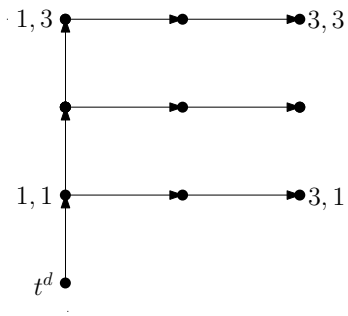

Figure 3.2: up-first-then-right tree

yields an individual upper bound on the revenue for that specific allocation rule. By maximizing the individual upper bounds over all allocation rules, a global upper bound for the revenue is achieved. But this upper bound is assumed by the modified efficient allocation rule derived before, which yields hence an optimal mechanism.

It turns out that the described approach is doomed to fail in our setting. Especially, one cannot find any tree $B \subseteq T_f$ – as e.g. the up-first-then-right tree above – such that the allocation rule optimizing the expected total payment computed on the basis of B in turn has B as a shortest path tree. Note that the approach described above and also our approach for the 1-dimensional setting focus on one agent and the corresponding type graph. Hence any allocation rule derived by the described approach is necessarily a modified Smith's rule with modified weights that can be computed from the characteristics (type report and distribution) of the agent itself similar as in Lemma 3.4. Such an allocation rule satisfies the following IIA property.

Definition 3.18. *We say that an allocation rule f satisfies* independence of irrelevant alternatives (*IIA*) *if the relative order of any two jobs j_1 and j_2 is the same in the schedules $f(t_1)$ and $f(t_2)$ for any two type profiles $t_1, t_2 \in T$ that differ only in the types of agents from $J \setminus \{j_1, j_2\}$.*

In other words, the relative order of two jobs is independent of all other jobs. For the 2-d setting, this is not necessarily the case for optimal mechanisms.

Theorem 3.19. *The optimal allocation rule for the 2-dimensional setting does in general not satisfy IIA.*

Proof. Consider the following instance with three jobs. Job 1 has type $(1, 1)$, job 2 has type $(2, 2)$ and job 3 has type space $\{1.9, 2\} \times \{1, 2\}$. The probabilities for job 3's

types are $\phi_3(1.9,1) = 0.8$, $\phi_3(2,2) = 0.2$ and $\phi_3(1.9,2) = \phi_3(2,1) = 0$, respectively. We will show that the best allocation rule that satisfies IIA achieves a minimum expected total payment of at least 5.6, whereas there exists an allocation rule – violating IIA – with an expected total payment of 4.88. The following argumentation would still work if we assumed small positive probabilities for types $(1.9,2)$ and $(2,1)$ as well, but everything would become much more technical.

There are six possible schedules for three jobs, where we denote e.g. by 312 the schedule where job 3 comes first and job 2 last. There are only two cases that occur with positive probability: job 3 has type $(1.9,1)$, which we refer to as case a, and job 3 has type $(2,2)$, which we refer to as case b. An allocation rule that satisfies IIA must schedule job 1 and 2 in the same relative order in case a and b. Therefore, any such rule must either choose a schedule from $\{123, 132, 312\}$ for case a and case b or it must choose a schedule from $\{213, 231, 321\}$ in case a and b. As an example, we compute a lower bound on the optimal payment $P^{min}(f)$ for the case where f chooses schedule 123 in case a and schedule 132 in case b. Since there is only one possible type for job 1 and 2, only individual rationality matters for the optimal payments to those jobs and hence $\pi_1^f(1,1) = 0$ and $\pi_2^f(2,2) = 2(0.8 \cdot 1 + 0.2 \cdot (1+2)) = 2.8$. For job 3, we take individual rationality into account as well as the incentive constraint $\pi_3^f(1.9,1) - 1.9 \cdot ES_3(1.9,1) \geq \pi_3^f(2,2) - 1.9 \cdot ES_3(2,2)$. While individual rationality requires $\pi_3^f(1.9,1) \geq 1.9 \cdot 3 = 5.7$ and $\pi_3^f(2,2) \geq 2$, the latter is equivalent to $\pi_3^f(1.9,1) \geq \pi_3^f(2,2) + 3.8$. Therefore, $\pi_3^f(2,2) \geq 2$ and $\pi_3^f(1.9,1) \geq 5.8$. Hence $P^{min}(f) \geq 2.8 + 0.8 \cdot 5.8 + 0.2 \cdot 2 = 7.84$. Note that this is only a lower bound, since for the exact value of $P^{min}(f)$, we must additionally consider the incentive constraints that result from the two types $(1.9,2)$ and $(2,1)$, which have zero probability, but are in the type space of job 3.

In total, there are 18 allocation rules that satisfy IIA. We list the corresponding lower bounds (LB) on $P^{min}(f)$ in the following table.

$f(a)$	$f(b)$	π_1^f	π_2^f	LB $\pi_3^f(1.9,1)$	LB $\pi_3^f(2,2)$	LB $P^{min}(f)$
123	123	0	2	6	6	8
123	132	0	2.8	5.8	2	7.84
123	312	0.4	2.8	5.7	0	7.76
132	123	0	3.6	2.2	6	6.56
132	132	0	4.4	2	2	6.4
132	312	0.4	4.4	1.9	0	6.32
312	123	0.8	3.6	0.3	6	5.84
312	132	0.8	4.4	0.1	2	5.68
312	312	1.2	4.4	0	0	5.6

213	213	2	0	6	6	8
213	231	2.4	0	5.9	4	7.92
213	321	2.4	0.8	5.7	0	7.76
231	213	2.8	0	4.1	6	7.28
231	231	3.2	0	4	4	7.2
231	321	3.2	0.8	3.8	0	7.04
321	213	2.8	1.6	0.3	6	5.84
321	231	3.2	1.6	0.2	4	5.76
321	321	3.2	2.4	0	0	5.6

Hence, 5.6 is a lower bound for the expected total payment made by any IIA mechanism. On the other hand, regard the allocation rule that chooses schedule 132 in case a and schedule 231 in case b. We extend the allocation rule to the zero probability type such that it chooses schedule 132 for type $(2, 1)$ and schedule 231 for type $(1.9, 2)$. Clearly, this allocation rule violates IIA. The optimal payments to job 1 and 2 are $\pi_1^f(1, 1) = 0.8$ and $\pi_2^f(2, 2) = 1.6$ respectively. For the optimal payment to job 3, we depict the type graph with associated arc lengths in Figure 3.3. The shortest path lengths from $(1.9, 1)$ and $(2, 2)$ to the dummy node are -2.1 and -4, respectively. Hence, $\pi_3^f(1.9, 1) = 2.1$ and $\pi_3^f(2, 2) = 4$. Consequently, $P^{min}(f) = 0.8 + 1.6 + 0.8 \cdot 2.1 + 0.2 \cdot 4 = 4.88$. This proves the claim. □

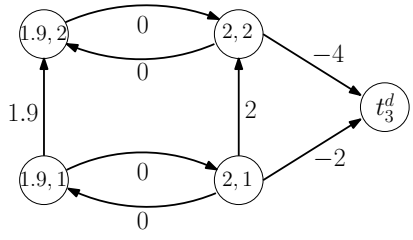

Figure 3.3: type graph job 3

Theorem 3.19 shows that any kind of priority based algorithm or list scheduling algorithm where the priority of a job can be computed from the characteristics of the job itself cannot be optimal in general. Moreover, the type graph approach must fail, since it focusses on a single agent. Hence, optimal mechanism design for our 2-dimensional setting is considerably more complicated than for the 1-dimensional setting and for traditional auction settings as described in Myerson

(1981) and Malakhov and Vohra (2007). One explanation for this complication may lie in the fact that the 2-d setting considered here in fact entails informational externalities, as opposed to the auction setting in (Malakhov and Vohra 2007). On the other hand, the informational externalities introduced by private processing times are not the only cause for complications in the 2-dimensional setting: Consider the 1-dimensional setting, where only the processing times are private, but the weights are public information. It turns out that all allocation rules are implementable, even when we allow that jobs understate their processing times. The optimal payment to a job j that reports processing time p_j^k is equal to $w_j ES_j(f, p_j^k)$, and therefore the total payment to jobs for allocation rule f is equal to $P^{min}(f) = \sum_{j \in J} \sum_{k=1}^{q_j} \phi_j(p_j^k) w_j ES_j(f, p_j^k)$. This is minimized by Smith's rule.

When there are only two agents present, then IIA is trivially satisfied. Recall that in the 1-dimensional case the optimal mechanism is efficient for symmetric agents and regular distributions and that the uniform distribution is regular. This is contrasted by the following theorem.

Theorem 3.20. *Even for two symmetric agents, 2×2-type spaces and uniform probability distributions, the optimal mechanism is not efficient.*

Proof. Consider the following example with two jobs, $W_1 = W_2 = \{1, 2\}$ and $P_1 = P_2 = \{1, 2\}$. We assume that $\phi_1(i, k) = \phi_2(i, k) = \frac{1}{4}$ for $i, k \in \{1, 2\}$. On one hand, consider the efficient allocation rule f_e, which schedules the job with higher weight over processing time ratio first. On the other hand, regard the so-called w-rule, f_w, that schedules the job with the higher weight first. In case of ties, both rules schedule job 1 first. The expected start times are listed below.

$$
\begin{aligned}
&ES_1(f_w, 1, 1) = ES_1(f_w, 1, 2) = 3/4 && ES_2(f_w, 1, 1) = ES_2(f_w, 1, 2) = 3/2 \\
&ES_1(f_w, 2, 1) = ES_1(f_w, 2, 2) = 0 && ES_2(f_w, 2, 1) = ES_2(f_w, 2, 2) = 3/4
\end{aligned}
$$

$$
\begin{aligned}
&ES_1(f_e, 1, 1) = ES_1(f_e, 2, 2) = 1/4, && ES_2(f_e, 1, 1) = ES_2(f_e, 2, 2) = 1, \\
&ES_1(f_e, 1, 2) = 1, && ES_2(f_e, 1, 2) = 3/2, \\
&ES_1(f_e, 2, 1) = 0, && ES_2(f_e, 2, 1) = 1/4.
\end{aligned}
$$

The type graphs corresponding to f_w for job 1 and 2 respectively are shown in Figure 3.4. From this, the optimal payments can be computed as:

Figure 3.4: type graphs for the w-rule for jobs 1 and 2

Figure 3.5: type graphs for the efficient rule for job 1 and 2

$$\begin{aligned} \pi_1^{f_w}(2,1) &= \pi_1^{f_w}(2,2) = 0, & \pi_2^{f_w}(2,1) &= \pi_2^{f_w}(2,2) = 3/2, \\ \pi_1^{f_w}(1,1) &= \pi_1^{f_w}(1,2) = 3/4, & \pi_2^{f_w}(1,1) &= \pi_2^{f_w}(1,2) = 9/4. \end{aligned}$$

Hence the (minimum) total expected payment for the w-rule is:

$$P^{min}(f_w) = \frac{1}{4}\sum_j \sum_{(i,k)} \pi_j^{f_w}(i,k) = 9/4.$$

The type graphs corresponding to f_e for agent 1 and 2 respectively are shown in Figure 3.5.
From this, the node potentials that minimize payment can be computed as:

$$\begin{aligned} \pi_1^{f_e}(1,1) &= \pi_1^{f_e}(2,2) = 1/2, & \pi_2^{f_e}(1,1) &= \pi_2^{f_e}(2,2) = 2, \\ \pi_1^{f_e}(2,1) &= 0 & \pi_2^{f_e}(2,1) &= 1/2, \\ \pi_1^{f_e}(1,2) &= 5/4, & \pi_2^{f_e}(1,2) &= 5/2. \end{aligned}$$

Hence the (minimum) total expected payment in the efficient rule is:

$$P^{min}(f_e) = \frac{1}{4} \sum_j \sum_{(i,k)} \pi^j(i,k) = 37/16.$$

Hence, $P^{min}(f_e) > P^{min}(f_w)$. This is even true if we break ties randomly. Thus, the efficient allocation is for some instances dominated by at least the w-rule and consequently is not part of an optimal mechanism even in the most symmetric case possible in this setting. □

3.5 Optimal Mechanisms for the Continuous Setting

For this section, we impose the following changes on the discrete setting described in the previous sections. For every job j, let the weight w_j be a continuous random variable with publicly known support $[m_j, M_j]$, probability density function ϕ_j, and cumulative distribution function Φ_j. Probability distributions are assumed to be independent between jobs. We will prove some results for general probability distributions and others for uniform distribution of weights. The latter has $\Phi_j(x) = (x-m_j)/(M_j-m_j)$ and $\phi_j(x) = 1/(M_j-m_j)$ for all $j \in J$ and $x \in [m_j, M_j]$. As in the 1-dimensional discrete case, the actual weight is private information of a job, whereas the processing time p_j of job j is fixed and common knowledge. We will refer to the definitions of Section 3.2, unless we give a new definition here.

In the following, we show that the characterization of Bayes-Nash implementable allocation rules from the previous section also applies to the continuous case. In addition, revenue equivalence holds. We show that Smith's rule with respect to certain modified weights and payments computed from the network approach is again an optimal mechanism under regularity. If the regularity condition is satisfied and agents are symmetric, then this mechanism is efficient, as before. The regularity condition is satisfied for instance by the uniform distribution. If $m_j = 0$ for $j = 1, \ldots, n$ and if the weights of all agents are distributed uniformly over their respective (not necessarily equal) intervals $[0, M_j]$, then this optimal mechanism is even efficient if the processing times differ among agents.

Hartline and Karlin (2007) discuss optimal mechanism design for a similar setting as the continuous setting at hand. They derive optimal mechanisms subject to dominant strategy implementability and thus mechanisms that are optimal in a more restricted class of mechanisms. The allocation rule of the optimal mechanism that we derive turns out to be dominant strategy implementable as well, but is optimal within the larger class of Bayes-Nash incentive compatible mechanisms. Strictly

speaking, our results are therefore not implied by the results in Hartline and Karlin (2007). On the other hand, looking at the techniques described in their paper, our optimal mechanism could be derived using these techniques, too. Although our optimal payments and regularity conditions differ from those in Hartline and Karlin (2007), these differences are completely due to the fact that in our case agents are paid by the mechanism and therefore individual rationality requires adding different constants.

3.5.1 Bayes-Nash Implementability and Revenue Equivalence

We make use of the type graph as before. Note that for continuous distribution of weights, the type graph has uncountably many nodes. We do not introduce an extra dummy node here, but we will account for individual rationality explicitly when deriving optimal mechanisms. In the continuous case, the following holds:

Theorem 3.21. *An allocation rule f is Bayes-Nash incentive compatible in the continuous setting if and only if it satisfies monotonicity.*

The proof of Theorem 3.21 is almost identical to the proof of Theorem 3.3. Note that even in an infinite type graph we only need to consider finite cycles. We do not repeat the proof here.

Theorem 3.22. *In the continuous setting, every Bayes-Nash implementable allocation rule f satisfies revenue equivalence.*

Proof. We use the characterization of revenue equivalence given in Chapter 2. Fix a Bayes-Nash implementable allocation rule f and agent j and consider the type graph T_f. Let $w, z \in [m_j, M_j]$ be two types of agent j. Using the same notation as before, we derive the following distance from w to z.

$$\begin{aligned} dist(w,z) &= \inf_{(w=a_0,\ldots,a_{k_\mathfrak{p}}=z)\in\mathcal{P}(w,z)} \sum_{i=0}^{k_\mathfrak{p}-1} \ell_{a_i a_{i+1}} \\ &= \inf_{(w=a_0<\cdots<a_{k_\mathfrak{p}}=z)\in\mathcal{P}(w,z)} \sum_{i=0}^{k_\mathfrak{p}-1} \ell_{a_i a_{i+1}} \\ &= \inf_{(w=a_0<\cdots<a_{k_\mathfrak{p}}=z)\in\mathcal{P}(w,z)} \sum_{i=0}^{k_\mathfrak{p}-1} a_i[ES_j(f,a_{i+1}) - ES_j(f,a_i)] \end{aligned}$$

$$= \inf_{(w=a_0<\cdots<a_{k_\mathfrak{p}}=z)\in\mathcal{P}(w,z)} \left(-wES_j(f,w) + zES_j(f,z) + \sum_{i=2}^{k_\mathfrak{p}} (a_{i-1}-a_i) ES_j(f,a_i) \right)$$

$$= -wES_j(f,w) + zES_j(f,z) - \int_w^z ES_j(f,x)dx.$$

Here, we use decomposition monotonicity and the nonnegative two-cycle property for the second equality. The last equality follows from decomposition monotonicity and the fact that $ES_j(f,\cdot)$ is a non-increasing function and therefore Riemann integrable. Similarly, we get

$$dist(z,w) = wES_j(f,w) - zES_j(f,z) - \int_z^w ES_j(f,x)dx,$$

and therefore $dist(w,z) = -dist(z,w)$. According to Theorem 2.11, f satisfies revenue equivalence. □

3.5.2 Optimal Mechanisms

As in the discrete case, we design a mechanism which assigns the payments to agents only on the basis of their reports, no matter what the announced types of the other agents are and no matter how therefore the actual allocation looks like. The goal is to minimize the expected total payment made to jobs.

The following lemma gives payments that minimize the expected total payment made to jobs for a given allocation rule.

Lemma 3.23. *For a Bayes-Nash implementable allocation rule f, the payment scheme*

$$\pi_j^f(w_j) = w_j ES_j(f,w_j) + \int_{w_j}^{M_j} ES_j(f,x)\,dx \text{ for } j\in J,\ w_j\in[m_j,M_j]$$

is incentive compatible, individual rational and minimizes the expected total payment made to agents. The expected total payment is then given by

$$P^{min}(f) = \sum_{j\in J} \int_W S_j(f(w))\overline{w}_j\phi(w)dw,$$

where the modified weights $\overline{w}_j$ are defined as

$$\overline{w}_j := w_j + \frac{\Phi_j(w_j)}{\phi_j(w_j)} \text{ for } w_j \in [m_j, M_j].$$

Proof. The given payment scheme is equal to

$$\pi_j^f(w_j) = -dist(w_j, M_j) + M_j ES_j(f, M_j) \text{ for } j \in J,\ w_j \in [m_j, M_j].$$

Similar to the previous section, it can easily be checked that this payment scheme satisfies the incentive constraints. For any allocation rule f, the expected payment to any agent $j \in J$ is fixed up to a constant due to Theorem 3.22. The constant must be chosen high enough such that individual rationality is satisfied, but also low enough, such that the total expected payment is minimized. Observe that the expected utility for type M_j is equal to $-M_j ES(f, M_j) + E\pi_j^f(M_j) = 0$, therefore adding a negative constant would violate individual rationality at type M_j. On the other hand, for any type $w_j \in [m_j, M_j]$, the expected utility is equal to $-w_j ES(f, w_j) + E\pi_j^f(w_j) = \int_{w_j}^{M_j} ES_j(f, x)\, dx \geq 0$, thus individual rationality is satisfied. Hence, adding a positive constant would make the expected payment non-minimum. Consequently, the above payment scheme is incentive compatible, individual rational and minimizes the expected payment for every type of every job. Hence, it also minimizes the expected total payment to agents.

Next, we derive an expression for the expected total payment.

$$\begin{aligned} P^{min}(f) &= \sum_{j \in J} \int_{m_j}^{M_j} \phi_j(w_j) \pi_j^f(w_j)\, dw_j \\ &= \sum_{j \in J} \int_{m_j}^{M_j} \phi_j(w_j) \left(w_j ES_j(f, w_j) + \int_{w_j}^{M_j} ES_j(f, x)\, dx \right) dw_j \\ &= \sum_{j \in J} \int_{m_j}^{M_j} w_j ES_j(f, w_j) \phi_j(w_j)\, dw_j + \sum_{j \in J} \int_{m_j}^{M_j} \int_{w_j}^{M_j} ES_j(f, x) \phi_j(w_j)\, dx\, dw_j \end{aligned}$$

Recall that $S_j(f(w_j, w_{-j}))$ denotes the start time of job j, when other jobs report

w_{-j} and that $W = \Pi_{i=1}^{n}[m_j, M_j]$. The summands of the first sum can be written as

$$\begin{aligned}
&\int_{m_j}^{M_j} w_j ES_j(f, w_j)\phi_j(w_j)\, dw_j \\
= &\int_W w_j S_j(f(w_j, w_{-j}))\phi_1(w_1)\dots\phi_j(w_j)\dots\phi_n(w_n)dw_1\dots dw_j\dots dw_n \\
= &\int_W w_j S_j(f(w))\phi(w)dw,
\end{aligned}$$

where $\phi(\cdot)$ is the joint distribution function of all agents. The summands in the second sum can be rewritten as follows

$$\begin{aligned}
&\int_{m_j}^{M_j}\int_{w_j}^{M_j} ES_j(f, x)\phi_j(w_j)\, dx\, dw_j \\
=&\int_{m_j}^{M_j}\int_{m_j}^{x} ES_j(f, x)\phi_j(w_j)\, dw_j\, dx \\
=&\int_{m_j}^{M_j} ES_j(f, x)\Phi_j(x)dx \\
=&\int_{m_j}^{M_j} ES_j(f, w_j)\Phi_j(w_j)dw_j \\
=&\int_W S_j(f(w_j, w_{-j}))\Phi_j(w_j)\phi_1(w_1)...\phi_{j-1}(w_{j-1})\phi_{j+1}(w_{j+1})...\phi_n(w_n)dw_1...dw_j...dw_n \\
=&\int_W S_j(f(w))\frac{\Phi_j(w_j)}{\phi_j(w_j)}\phi(w)dw.
\end{aligned}$$

Hence, we get for the total payment

$$\begin{aligned}
P^{min}(f) &= \sum_{j\in J}\int_W S_j(f(w))\left(w_j + \frac{\Phi_j(w_j)}{\phi_j(w_j)}\right)\phi(w)dw \\
&= \sum_{j\in J}\int_W S_j(f(w))\overline{w}_j\phi(w)dw,
\end{aligned}$$

where $\overline{w}_j := w_j + \Phi_j(w_j)/\phi_j(w_j)$ defines the modified weight for job j. □

As in the discrete case, $P^{min}(f)$ can be minimized for arbitrary distributions of weights by applying Smith's rule with respect to the modified weights. The resulting mechanism will be Bayes-Nash incentive compatible if the following regularity condition holds.

Definition 3.24. *The regularity condition holds in the continuous case if for $j \in J$*

and $w, z \in [m_j, M_j]$, $w < z$*:*

$$w + \frac{\Phi_j(w)}{\phi_j(w)} \leq z + \frac{\Phi_j(z)}{\phi_j(z)}.$$

We get the following result.

Theorem 3.25. *Let the modified weights and the payment scheme* π^f *be defined as in Lemma 3.23. Let* f *be the allocation rule that schedules jobs in order of non-increasing ratios* $\overline{w}_j/p_j$*. If regularity holds, then* (f, π^f) *is an optimal mechanism.*

Proof. As mentioned above, Smith's rule minimizes $\sum_{j \in J} S_j(f(w))\overline{w}_j$ for every type profile $w \in W$. Therefore, it also minimizes the total expected payment. As in the discrete case, the regularity condition ensures that modified weights be non-decreasing in the original weights. As $ES_j(w_j)$ is non-increasing in the modified weight $\overline{w}_j$ under Smith's rule with respect to modified weights, it is non-increasing in the original weight w_j for every $j \in J$ if regularity holds. Hence, under regularity weak monotonicity and consequently Bayes-Nash implementability is satisfied. □

The following theorem gives two important cases, when this optimal mechanism is efficient.

Theorem 3.26. *The optimal mechanism is efficient in the following two cases.*
1) Agents are symmetric, i.e., have identically distributed weights and equal processing times and the regularity condition holds for the distribution functions.
2) Agents' weights are distributed uniformly over $[0, M_j]$ *for* $j = 1, ..., n$*. Processing times can be arbitrary.*

Proof. 1) Smith's rule with respect to modified weights is equivalent to Smith's rule with respect to the original weights as in the discrete case. Regularity ensures weak monotonicity and hence Bayes-Nash incentive compatibility.

2) For the uniform distribution, we get for the virtual weights

$$w + \frac{\Phi_j(w)}{\phi_j(w)} = w + \frac{w/M_j}{1/M_j} = 2w,$$

which is increasing and linear in w and the linear relationship does not depend on the agent. Hence, Smith's rule with respect to virtual weights is equivalent to Smith's rule with respect to original weights, no matter what the processing times are. □

3.6 Optimal Mechanisms via Standard Auction Formats

After having derived an optimal mechanism for the continuous case, we are interested, whether standard auction formats also yield optimal mechanisms for our scheduling setting. We study the VCG mechanism and a mechanism that corresponds to the first price auction.

3.6.1 The Generalized VCG Mechanism

Recall that for the discrete setting, the generalized VCG mechanism was not optimal, even in cases when the optimal mechanism allocates efficiently. In the continuous setting, however, revenue equivalence implies that the expected payments to agent j in all Bayes-Nash incentive compatible mechanisms that allocate efficiently are the same up to a constant. As the optimal mechanism proposed in Section 3.5 allocates efficiently in the case of symmetric agents and regularity, also the VCG mechanism can be used in this case to derive an optimal mechanism by adding an appropriate constant to the payments of every agent.

Theorem 3.27. *For symmetric agents under regularity, the VCG mechanism with payments*

$$\pi_j^{VCG}(w) = p_j \sum_{\substack{k \in J \\ \sigma_k < \sigma_j}} w_k.$$

is optimal. Here, σ denotes the efficient schedule.

Proof. Assume symmetric agents with weights identically distributed over $[m, M]$ according to density function ϕ_1 and cumulative distribution function Φ_1. The distributions are assumed to satisfy regularity. Without loss of generality, let the processing times be equal to one. The result already follows from revenue equivalence and the fact that under the VCG mechanism, any job with type equal to its maximum possible type M has expected start time equal to zero and hence zero expected utility, just as in the optimal mechanism from Section 3.5.

Nevertheless, we check the equality of the expected payments under the VCG and the optimal mechanism explicitly for illustrative purposes. Since jobs have equal processing times, the VCG mechanism allocates in order of non-increasing weights. The payment made to an agent j under the VCG mechanism is the sum of the weights of all agents processed before j. To derive the expected payment to agent j announcing type w_j, we notice that any other agent k is scheduled before j if k's weight x is larger than w_j. In this case, x is paid to j. The expected payment at

type w_j is therefore

$$E\pi_j^{VCG}(w_j) = (n-1)\int_{w_j}^{M} x\phi_1(x)dx.$$

In the optimal mechanism proposed in Section 3.5, the payment for type w_j and any w_{-j} is equal to

$$\pi_j^f(w_j, w_{-j}) = E\pi_j^f(w_j) = w_j ES_j(f, w_j) + \int_{w_j}^{M} ES_j(f, x)dx.$$

The start time when announcing type w_j is a binomially distributed random variable with parameters $n-1$ and $1-\Phi_1(w_j)$, as the placement of any of the $n-1$ other jobs in front of j can be seen as a binomial trial with success probability $1-\Phi_1(w_j)$. The start time counts the number of "successes". Therefore, $ES_j(f, w_j) = (n-1)(1-\Phi_1(w_j))$. We get for the payments

$$\begin{aligned}
& E\pi_j^f(w_j) \\
= \;& w_j(n-1)(1-\Phi_1(w_j)) + (n-1)\int_{w_j}^{M}(1-\Phi_1(x))dx \\
= \;& w_j(n-1)(1-\Phi_1(w_j)) + (n-1)\int_{w_j}^{M}\int_{x}^{M}\phi_1(y)\,dy\,dx \\
= \;& w_j(n-1)(1-\Phi_1(w_j)) + (n-1)\int_{w_j}^{M}\int_{w_j}^{y}\phi_1(y)\,dx\,dy \\
= \;& w_j(n-1)(1-\Phi_1(w_j)) + (n-1)\int_{w_j}^{M}\phi_1(y)(y-w_j)\,dy \\
= \;& w_j(n-1)(1-\Phi_1(w_j)) - (n-1)w_j(1-\Phi_1(w_j)) + (n-1)\int_{w_j}^{M} y\phi_1(y)\,dy \\
= \;& (n-1)\int_{w_j}^{M} x\phi_1(x)\,dx \\
= \;& E\pi_j^{VCG}(w_j).
\end{aligned}$$

Hence, $E\pi_j^f(w_j) = E\pi_j^{VCG}(w_j)$ for all $j \in J$ and all types w_j. Therefore, the total expected payments of the optimal and the VCG mechanism are equal, too. Hence, the VCG mechanism is optimal. □

Remarkably, the payment under π^f depends only on the reported type of an agent and is constant over all reports of the other agents' and therefore over all

allocations. In contrast, π^{VCG} depends only on the allocation and not on the specific report of the agent. Nevertheless, both yield the same expected payments.

3.6.2 The First-Price Equivalent

In the first price auction, the highest bidder wins the object and has to pay the amount of his bid. In this auction, truthful reporting does not necessarily maximize a bidder's expected utility. On the other hand, there is a strictly increasing and differentiable bidding function β such that bidding according to β for all agents is a Bayes-Nash equilibrium. This result can e.g. be found in Myerson (1991). Especially, for uniformly distributed valuations for the object, the bidding function β scales the true valuation down by a factor of $(n-1)/n$.

We do a similar analysis for the continuous case of our scheduling problem. For symmetric agents, we derive a strictly increasing and differentiable function β yielding a symmetric Bayes-Nash equilibrium in which all agents report according to β. From that, it is easy to construct another optimal mechanism for the continuous case. We furthermore show that for two agents with different processing times, there is no such function β.

The Mechanism for the Symmetric Case. Suppose, the jobs in J are symmetric and their weights are drawn independently and identically distributed from the interval $[0, M]$ with probability density function ϕ_1 and cumulative distribution function Φ_1. Suppose $\phi_1(\cdot) > 0$ on $[0, M]$. Processing times are all equal to one. The proposed mechanism (f, π) works as follows. Schedule jobs in order of non-increasing weights and pay to each job an amount equal to its actual start time times its announced weight. A bidding function β is a function $\beta\colon [0, M] \to \mathbb{R}_+$. Recall the definition of a Bayes-Nash equilibrium.

Definition 3.28. *Reporting according to $\beta\colon [0, M] \to \mathbb{R}_+$ is a Bayes-Nash equilibrium if any job j with weight w_j maximizes its expected utility by reporting $\beta(w_j)$ given that all other jobs report according to β, too.*

Let us assume that there is a symmetric Bayes-Nash equilibrium in which agents report according to the same strictly increasing and differentiable bidding function β. We will first derive a functional form for β and then show that reporting according to β is a Bayes-Nash equilibrium.

Fix agent j with actual weight w_j and suppose that every other agent k with true weight w_k reports $\beta(w_k)$. Suppose, j reports some weight b_j. Then its expected utility is $(b_j - w_j)$ times its expected start time. If j bids $b_j \leq \beta(0)$ then it will get the last position with probability one and therefore has utility $(n-1)(b_j - w_j) \leq$

$(n-1)(\beta(0)-w_j)$. This utility is maximized at $\beta(0)$ and j will never bid strictly less than $\beta(0)$. Reporting more than $\beta(M)$ leads to an expected start time of 0 for agent j and hence to an expected utility of 0. Reporting any $w_j \leq b_j \leq \beta(M)$ leads to a non-negative expected utility. Hence we can assume $b_j \leq \beta(M)$ without loss of generality. Consequently, $\beta(0) \leq b_j \leq \beta(M)$. As β is continuous and strictly increasing, we can compute $\beta^{-1}(b_j) =: \tilde{w}_j$. Scheduling in order of non-increasing reports $\beta(w_k)$ is equivalent to scheduling in order of non-increasing reports w_k, as β is increasing. Therefore, j's start time when reporting $b_j = \beta(\tilde{w}_j)$ is again a binomially distributed random variable with parameters $n-1$ and $1-\Phi_1(\tilde{w}_j)$ and expected value

$$ES_j(f, b_j) = (n-1)(1-\Phi_1(\tilde{w}_j)).$$

Job j's expected utility is then equal to

$$(b_j - w_j)(n-1)(1-\Phi_1(\tilde{w}_j)) = (b_j - w_j)(n-1)(1-\Phi_1(\beta^{-1}(b_j))).$$

Differentiating with respect to b_j yields

$$(n-1)(1-\Phi_1(\beta^{-1}(b_j))) - (n-1)(b_j - w_j)\frac{\phi_1(\beta^{-1}(b_j))}{\beta'(\beta^{-1}(b_j))}.$$

The expected utility should be maximized at $b_j = \beta(w_j)$. We apply the first order condition.

$$\begin{aligned} (1-\Phi_1(w_j)) - (\beta(w_j) - w_j)\frac{\phi_1(w_j)}{\beta'(w_j)} &= 0 \\ \Leftrightarrow \beta'(w_j)(1-\Phi_1(w_j)) - \beta(w_j)\phi_1(w_j) &= -w_j\phi_1(w_j). \end{aligned}$$

This should be true for any true weight $x \in [0, M]$. Hence, we can write for $x \in [0, M]$

$$\frac{d}{dx}(\beta(x)(1-\Phi_1(x))) = -x\phi_1(x).$$

Integrating both sides from w_j to M yields

$$\beta(M)(1-\Phi_1(M)) - \beta(w_j)(1-\Phi_1(w_j)) = -\int_{w_j}^{M} x\phi_1(x)dx$$

$$\Leftrightarrow \beta(w_j) = \frac{1}{1-\Phi_1(w_j)}\int_{w_j}^{M} x\phi_1(x)dx.$$

The report $\beta(M)$ is obtained by taking the limit $\lim_{x\to M}\beta(x)$. Note that β is differentiable and strictly increasing if ϕ is strictly positive on $[0, M]$. Unlike in the first price auction, the bidding function is independent of the number of agents.

Next, we show that agents indeed maximize their expected utility by reporting according to β.

Theorem 3.29. *Let f be the allocation rule that schedules in order of non-increasing reported weights. Let π be such, that every agent gets a payment equal to its announced weight times its actual start time. Then, in the mechanism (f, π), reporting according to $\beta\colon [0, M] \to \mathbb{R}_+$, with*

$$\beta(w_j) = \frac{1}{1-\Phi_1(w_j)}\int_{w_j}^{M} x\phi_1(x)dx.$$

is a Bayes-Nash equilibrium.

Proof. Fix agent j with true weight w_j and suppose that all other agents report according to β. We show that reporting $\beta(w_j)$ indeed maximizes j's expected utility. Suppose, j reports b_j. Let $Eu_j(b_j, w_j)$ be the expected utility for j when reporting b_j while having actual weight w_j. As we already have seen, there is no loss of generality in assuming $\beta(0) \le b_j \le \beta(M)$. Hence, $\beta(\tilde{w}_j) = b_j$ for some $\tilde{w}_j \in [0, M]$. The expected utility from reporting $\beta(\tilde{w}_j)$ is equal to

$$\begin{aligned}
Eu_j(b_j, w_j) &= (\beta(\tilde{w}_j) - w_j)(n-1)(1-\Phi_1(\tilde{w}_j)) \\
\Leftrightarrow \frac{1}{n-1}Eu_j(b_j, w_j) &= \int_{\tilde{w}_j}^{M} x\phi_1(x)dx - w_j(1-\Phi_1(\tilde{w}_j)) \\
&= [x\Phi_1(x)]_{\tilde{w}_j}^{M} - \int_{\tilde{w}_j}^{M}\Phi_1(x)dx - w_j(1-\Phi_1(\tilde{w}_j)) \\
&= M - \tilde{w}_j\Phi_1(\tilde{w}_j) - \int_{\tilde{w}_j}^{M}\Phi_1(x)dx - w_j(1-\Phi_1(\tilde{w}_j)) \\
&= (M - w_j) + (w_j - \tilde{w}_j)\Phi_1(\tilde{w}_j) - \int_{\tilde{w}_j}^{M}\Phi_1(x)dx.
\end{aligned}$$

Hence,

$$\frac{1}{n-1}[Eu_j(\beta(w_j), w_j) - Eu_j(\beta(\tilde{w}_j), w_j)] = \int_{\tilde{w}_j}^{w_j}\Phi_1(x)dx - (w_j - \tilde{w}_j)\Phi_1(\tilde{w}_j) \ge 0.$$

This completes the proof. □

We give two examples of explicit bidding functions for the exponential and the uniform distribution.

Example 3.30 (Exponential distribution)**.** *Let* $\Phi_1(w) = 1 - e^{-\lambda w}$ *for some* $\lambda > 0$*. The interval is* $[0, \infty)$*. Then*

$$\begin{aligned}
\beta(w) &= \frac{1}{e^{-\lambda w}} \lambda \int_w^\infty x e^{-\lambda x} dx \\
&= \frac{1}{e^{-\lambda w}} \left([-x e^{-\lambda x}]_w^\infty + \int_w^\infty e^{-\lambda x} dx \right) \\
&= \frac{1}{e^{-\lambda w}} \left(w e^{-\lambda w} + \left[-\frac{e^{-\lambda x}}{\lambda} \right]_w^\infty \right) \\
&= w + \frac{1}{\lambda}.
\end{aligned}$$

Thus, if weights are exponentially distributed, an agent has to add the mean weight $1/\lambda$ to its actual weight in the equilibrium.

Example 3.31 (Uniform distribution)**.** *Let agents' weights be uniformly distributed over* $[0, M]$*. That is* $\phi_1(w) = 1/M$ *and* $\Phi_1(w) = w/M$*. Thus,*

$$\beta(w) = \frac{1}{1 - \frac{w}{M}} \int_w^M \frac{x}{M} dx = \frac{1}{M - w} \left[\frac{x^2}{2} \right]_w^M = \frac{M + w}{2}.$$

Taking the limit $w \to M$ *yields additionally* $\beta(M) = M$*.*

Hence, for uniform distributions, an agent reports the mean of its true weight and the maximum weight M.

From the above analysis, we get the following Bayes-Nash incentive compatible and optimal mechanism.

Theorem 3.32. *Allocating jobs in order of non-increasing reported weights and paying to job* j *with report* w_j *and realized start time* S_j *the payment* $S_j \beta(w_j)$ *is a Bayes-Nash incentive compatible and optimal mechanism.*

Proof. As β is increasing, scheduling jobs in order of non-increasing $\beta(w_j)$ is equivalent to scheduling in order of non-increasing w_j. If bidding according to β is a Bayes-Nash equilibrium in the mechanism where a job j bidding w_j is paid $S_j w_j$, then truthful bidding is a Bayes-Nash equilibrium in the mechanism where j is paid $S_j \beta(w_j)$. Therefore, the mechanism proposed in the theorem is Bayes-Nash incentive compatible. As the allocation is again efficient, expected payments for each

type coincide up to a constant with the expected payments of the VCG mechanism and with those of the optimal mechanism described in the previous two sections. The constant is zero, as also in this mechanism, a job with maximum weight M has zero expected start time, zero payment and hence zero utility, just as in the VCG mechanism and the optimal mechanism from Section 3.5. □

A Negative Result for Unequal Processing Times. In the case with two agents that have unequal processing times, there is no bidding function β according to which *both* agents report in a Bayes-Nash equilibrium.

Theorem 3.33. *Suppose, there are two agents, whose weights are continuous random variables with equal support $[0, M]$. If $p_1 \neq p_2$, then there is no continuous, non-decreasing bidding function $\beta\colon [0, M] \to \mathbb{R}_+$ according to which both agents report in a Bayes-Nash equilibrium. That is, there is no symmetric Bayes-Nash equilibrium.*

Note that the theorem holds for arbitrary continuous random weights with support $[0, M]$. Especially, we do not need to assume that weights are identically distributed.

Proof. Without loss of generality let $p_2 < p_1$. Assume β is a continuous, non-decreasing equilibrium bidding function. Let agent 2 bid according to β and look at agent 1. Let

$$b_1 = \min\left\{\left(\frac{1}{2} + \frac{p_1}{2p_2}\right)\beta(0), \beta(M)\right\},$$

then $b_1 \in [\beta(0), \beta(M)]$. There exists $w_1 \in [0, M]$ with $\beta(w_1) = b_1$, as β is continuous. Then

$$\begin{aligned} & \beta(w_1) = b_1 \leq \left(\frac{1}{2} + \frac{p_1}{2p_2}\right)\beta(0) \\ \Leftrightarrow \quad & p_2\beta(w_1) \leq \left(\frac{p_2}{2} + \frac{p_1}{2}\right)\beta(0) < p_1\beta(0) \\ \Leftrightarrow \quad & \frac{\beta(w_1)}{p_1} < \frac{\beta(0)}{p_2}. \end{aligned}$$

Bidding any b with $b/p_1 \leq \beta(0)/p_2$ results in an expected start time of p_2 for agent 1. The expected utility is then $(b - w_1)p_2$ which is strictly larger at $b = (p_1/p_2)\beta(0)$ than at $b = \beta(w_1) < (p_1/p_2)\beta(0)$. Thus, β does not maximize the expected utility at w_1. □

3.7 Discussion

We have seen that the graph theoretic approach is an intuitive tool for optimal mechanism design and yields a closed formula for the optimal mechanism in the 1-dimensional case. The results parallel Myerson's results for single item auctions; although there are differences. It is not hard to see that the optimal allocation rule – Smith's rule with respect to modified weights – is even dominant strategy implementable, with the same total expected payment for the mechanism. In order to obtain a dominant strategy incentive compatible mechanism, only the payment scheme has to be defined appropriately for each reported type profile.

In the discrete case, efficient mechanisms can be arbitrarily bad with respect to the total payment made to agents. For symmetric agents, however, the optimal mechanism is efficient. Even so, the payments of the generalized VCG mechanism can still be non-optimal. In the continuous case, revenue equivalence holds and the generalized VCG mechanism as well as a mechanism derived from the first price auction are optimal in those cases where the derived optimal mechanism allocates efficiently.

Moreover, we have seen that in the two-dimensional case the canonical approach does not work and that optimal mechanism design seems to be considerably more complicated than in the traditional auction models. We leave it as an open problem to identify (closed formulae for) optimal mechanisms for the 2-d case. It is conceivable, however, that closed formulae don't exist.

Part II

Strategic Multiple Machine Scheduling Models

Chapter 4

Overview of Problems and Models

While the focus in the first part of this thesis was on classical mechanism design questions and applications to a simple single machine scheduling model, we study (mostly) multiple machine models in this second part. In the presence of more than one machine, various models are conceivable. First of all, there is the issue whether the machines are identical, have different speeds or have completely independent processing times for each job. On the other hand, also the machines can be regarded as agents and they can have different kinds of private information, for example the speed.

In this chapter, we give an introduction to the literature that seems us suitable to illustrate some of the most interesting models and some of the techniques to tackle machine scheduling problems within a distributed setting. We introduce the most important of the underlying concepts, and give a selection of typical research questions and recent results. This includes the study of the so-called price of anarchy for settings where the agents do not possess private information, as well as the design and analysis of (truthful) mechanisms in settings where the agents do possess private information. Whenever it seems us appropriate, we give alternative proofs instead of the original ones. Therefor, we make use of the concepts and techniques introduced in the first part of this thesis. For example, we use monotonicity as introduced in Chapter 3 in order to give an alternative proof for the meanwhile famous result by Nisan and Ronen (2001) on the lower bound for the approximation ratio for a scheduling problem with machine agents. Moreover, we show how the type graph approach can be used to derive the payment scheme for a model considered

by Archer and Tardos (2001).[1]

4.1 Introduction

This chapter contains an – admittedly subjective – introduction into typical research questions that have recently been addressed in the literature on multiple machine scheduling in a distributed setting. From the application perspective, we focus on simple and classical scheduling and sequencing problems. In that perspective, the focus is not so much on actual applications in practice, but rather on the underlying game theoretic models and methodologies. We are aware that this focus is quite narrow, yet it serves well to exemplify the most important research questions that arise when addressing optimization problems from a distributed perspective. In particular, by keeping the focus narrow from the applications point of view, we think that we are able to better highlight the most important underlying theoretical challenges. For a more practical view point, we refer to the paper by Kreipl and Pinedo (2004).

In distributed settings, central coordination of a system is partially replaced by decisions and actions taken by agents that are assumed to act rationally on behalf of their own interest. It is generally assumed that their selfish behavior results in a situation that can be characterized by some sort of *system equilibrium*. From a global perspective, such an equilibrium may of course lead to suboptimal system performance. The following two issues arise in such settings and will be addressed in this chapter.

- Given a fixed distributed setting in which agents selfishly act on behalf of their own interest, try to *characterize and analyze* the quality of the resulting *system equilibria* from the perspective of the overall system performance.
- Try to *design* the distributed setting in such a way that selfish agents are induced to behavior that results in system equilibria that nevertheless exhibit a good overall system performance.

Moreover, both issues can be studied in settings where the individual agents *do* or *do not* have private information. The distinction between settings with or without private information leads to different challenges and related research questions. In fact, the chapter is structured along this distinction.

[1] The content of this chapter was first presented in Heydenreich, Müller, and Uetz (2007). Since then, the field has rapidly developed and there are plenty of new results. While we updated the results already contained in the mentioned paper, we did not extend its scope.

In settings *without private information*, also called *complete information* settings, the wealth of the literature is of a descriptive nature and addresses the issue to characterize and analyze equilibria of given systems. Only to a lesser extent the actual design of such settings is addressed. The analysis of system equilibria leads to the definition of the so-called *price of anarchy* or *coordination ratio*: Caused by selfish behavior of agents, by how much does the overall system performance deteriorate due to a lack of central coordination? In the literature on the price of anarchy, it is generally assumed that all data that describes the problem is publicly known. The 'only' complication is caused by the fact that agents act on their own behalf. The agents thus need to take into account the (strategic) behavior of other agents. The underlying equilibrium concept is the Nash equilibrium. Settings without private information and the analysis of the price of anarchy will be addressed in Section 4.3. The models and results discussed in Section 4.3 are mainly from the work of Koutsoupias and Papadimitriou (1999), Czumaj and Vöcking (2007), Christodoulou, Koutsoupias, and Nanavati (2004), Immorlica, Li, Mirrokni, and Schulz (2008) and Azar, Jain, and Mirrokni (2008). When it comes to the design of complete information settings, one is concerned with defining the rules within which the agents may interact. We give several examples of system designs for machine scheduling problems, and discuss the resulting price of anarchy.

In order to improve the quality of resulting equilibria in complete information settings, one can augment the system by introducing *payments*. In the context of a network routing problem, the issues that might arise with introducing payments have been addressed, for example, by Cole, Dodis, and Roughgarden (2006). Another option to improve the quality of system equilibria is to centrally control a certain fraction of the agents, leading to so-called *Stackelberg* games. Such a model was analyzed for example by Roughgarden (2004) in the context of scheduling. We refer to those papers for an introduction into these and related issues. In this chapter, we will not further elaborate on such extensions of complete information settings.

In settings *with private information*, we deal with *algorithmic mechanism design*; a term that was coined by Nisan and Ronen (2000). In these settings, the additional complication is that the agents own some piece of private information. In order to be able to run and evaluate the system, the agents need to reveal this private information to the system. Hence, as part of their (strategic) behavior within the system, agents might be tempted to falsely report their private information if it is beneficial for their own objectives. One important part of the design of such systems is therefore to induce the agents to *truthfully* report their private information; sometimes also called the design of *truthful mechanisms*. Notice that the equilibrium concepts in models with private information are more complex, because each agent is faced

with the additional uncertainty about the private information of the other agents. Algorithmic mechanism design problems are addressed in Section 4.4. The specific models and results discussed in Section 4.4 are based on the work of Archer and Tardos (2001), Nisan and Ronen (2000), and Porter (2004).

As mentioned before, the scope of this chapter is not to give an exhaustive overview of the field, but to highlight some typical research questions. Hence, we have chosen to discuss only a subjective selection of recent papers. Another reference from computer science not explicitly discussed here is, for example, Angel, Bampis, and Pascual (2005). Related problems are also studied in the literature on economic theory, addressing questions on the efficient organization of queues. There are, for example, papers on the existence of mechanisms with more properties than only truthfulness (Mitra 2001, 2005), or where queue disciplines are organized with the help of auctions (Kittsteiner and Moldovanu 2005).

The chapter is structured as follows. In Section 4.2.1, we introduce basic notation and terminology for the scheduling models that will be addressed. In Section 4.2.2 we give a survey of the most basic concepts and terminology in game theory and mechanism design that will be used throughout the chapter. Section 4.3 then addresses the analysis of the price of anarchy in settings without private information, and Section 4.4 addresses the design and analysis of (truthful) mechanisms in different settings with private information.

4.2 Concepts and Notation

4.2.1 Multiple Machine Scheduling Models

We consider machine scheduling problems with the following characteristics. There is a set of jobs $J = \{1, \ldots, n\}$, and each job has to be scheduled on any machine out of a set of machines $M = \{1, \ldots, m\}$. Unless explicitly stated otherwise, jobs must be scheduled *non-preemptively*, meaning that once the processing of a job has started, it cannot be interrupted until the job is completed. Regarding the machines we distinguish between three different models:

- In *parallel* machine scheduling, each job $j \in J$ has processing time $p_j > 0$, independent of the machine that processes the job.

- In *related* (or *uniform*) machine scheduling, each job j has processing time p_j (on a unit speed machine), each machine $i \in M$ has a speed $s_i > 0$, and the processing time of job j on machine i equals p_j/s_i.

- In *unrelated* machine scheduling, each job $j \in J$ has processing time $p_{ij} > 0$ when scheduled on machine $i \in M$.

In addition, jobs may have different characteristics depending on the specific model that is addressed. We only mention the two most important characteristics here. A *release date* $r_j \geq 0$ of job j is the time when job j comes into existence or is released for processing. In models with *deadlines*, each job j should be completed by its deadline d_j, and a job which is completed before or at its deadline is called *early*, otherwise a job is called *late*.

A feasible *schedule* is an assignment of jobs to machines, together with the specification of the time interval(s) when the job is processed. In non-preemptive settings, this reduces to specifying the machine and start time S_j for any job j. The precise definition of *feasibility* clearly depends on the particular model, but always comprises the requirement that each job must be completely processed and no machine can process more than one job at a time. If jobs have release dates, for example, no job must be started before its release date r_j.

With respect to the objective of scheduling, we address several classical objectives. Given a schedule, denote by S_j and C_j the start time and completion time of job j, respectively. Then the *makespan* of a schedule is the latest job completion time, denoted by $C_{\max} := \max_j C_j$. Jobs also might have *weights* $w_j \geq 0$, denoting a priority for being processed early. Then the *total weighted completion time* is $\sum_{j \in J} w_j C_j$. These job weights w_j could, for example, be deducted from an inventory value, and they can be interpreted as opportunity costs for delaying job j one unit of time.

Most models that we address represent NP-hard combinatorial optimization problems; for a survey and references, see, for example, the paper by Lawler, Lenstra, Rinnooy Kan, and Shmoys (1993). In addition, we address scheduling models that are *online*, thus the complete problem instance is not given at the outset, but only revealed gradually over time. For example, the existence of jobs might only become known upon their release dates r_j. For an introduction to online scheduling problems and models, see, for example, the paper by Pruhs, Sgall, and Torng (2004).

It should be mentioned that research in scheduling has addressed many more features and models than discussed here. For example, there might be precedence constraints between jobs, saying that the processing of job j may only start after another job k has been finished. Or the processing of jobs might need multiple resources, rather than one machine, and resources may be non-renewable. Also, there are other objectives than those considered here. We have decided to leave these models out of consideration, because – to the best of our knowledge – the combination of optimization and game theory has only been applied to machine

scheduling models.

4.2.2 Game Theory and Mechanism Design for Multiple Machine Scheduling Models

We next define some basic notation for game theoretic concepts used throughout this chapter. In addition, we introduce problem specific notation and concepts when needed, and indicate when we deviate from the game theoretic notation introduced here.

In the scheduling models we address, the *agents* will either be the set of jobs J or the set of machines M. Let us say we have l agents, then either $l = n$ or $l = m$. In some settings, an agent $k \in \{1, \ldots, l\}$ may own a piece of information that is not publicly known, its *type* t_k. Typical types are, for example, the speed s_i of a machine-agent $i \in M$, or the weight w_j of a job-agent $j \in J$. The possible types for agent k are denoted by T_k. Furthermore, let $T = T_1 \times \cdots \times T_l$ denote the type space of all agents. Next to the private information of agents, there is usually public information, as for example the number of machines or the type spaces of the agents (though not their actual types).

In a game, agents have to choose between several possible *actions*. An action could be that jobs have to select a machine on which they want to be processed, or that machines have to report their actual speed. We denote by A_k the possible actions of agent k and by $A = A_1 \times \cdots \times A_l$ the action space of all agents. The *outcome* of the game depends on the actions of all agents. In the games we consider, the outcome will always be a (feasible) schedule. Therefore, by $\mathfrak{S}$ we denote all (feasible) schedules.

Some care is required in order to translate 'problem instances' and 'algorithms' to a game theoretic setting. First, the term 'problem instance' that is used in optimization refers to both the public and the private information of a game. Let us denote by I the public information of a game. Then the equivalent of an algorithm is usually called an *allocation algorithm*, denoted by f; it computes an outcome (a schedule) on the basis of the public information I together with the actions of all agents. More precisely, $f : I \times A \to \mathfrak{S}$. Since there is hardly danger of ambiguity, we usually omit the public information I and write $f\colon A \to \mathfrak{S}$. To give an example, suppose that the jobs are agents and that their action is to select a machine. Then, $A = M^n$ is the action space, and the public information I consists of m, the number of machines, n, the number of jobs, as well as the set of processing times of all jobs $\{p_j \mid j \in J\}$. Assume that the allocation of jobs to time slots is defined by the *Local SPT rule*: Each machine processes its jobs in the order of non-decreasing processing times (SPT, shortest processing time first). For given actions $a = (a_1, \ldots, a_n)$ of

all n jobs, the allocation algorithm f thus assigns job k to machine a_k, in such a way that k is processed after all jobs j with $a_j = a_k$ and with $p_j < p_k$. (To make the game unambiguous, a tie breaking rule would be required for jobs with equal processing times assigned to the same machine. We assume that ties are broken in favor of jobs with smaller index j.) Notice that jobs are informed about the public information I, such as the number of available machines and the processing times of other jobs.

By $v_k(\sigma, t_k)$, we will denote agent k's valuation for a schedule $\sigma \in \mathfrak{S}$ when it has type t_k. The schedule $\sigma = f(a)$ depends on the actions a of all agents. If the allocation algorithm f is clear from the context, we also write $v_k(a, t_k)$, for convenience. A typical valuation of a job-agent j for a certain schedule might be $-C_j$, meaning that the job-agent wants to be finished as early as possible.

Given the public information, an agent's choice of an action depends on its type. Therefore, we need to define the *strategy* x_k of an agent k as a mapping from the agent's type space into its action space. Let $X_k = \{x_k \,|\, x_k \colon T_k \to A_k\}$ denote the strategy space of agent k and let $X = X_1 \times \cdots \times X_l$ be the possible strategies of all agents. For example, suppose a job j has to choose for being processed on one of two machines with different speeds, say machine 1 with speed $s_1 = 1$ and machine 2 with speed $s_2 = 2$. Suppose further that the job could be processed immediately on the slow machine 1, whereas it has to wait one time unit until the fast machine 2 becomes available. Assume that the type t_j of job j is just its processing time p_j (on a unit speed machine), and its valuation for an outcome (a schedule) is $-C_j$. Then the job's preferred strategy would be to choose the slow machine 1 if $p_j \leq 2$, but to wait for the fast machine 2 if $p_j > 2$.

As a central authority, we evaluate the overall quality of a schedule by the objective value that it achieves. In order to induce agents to choose their actions in a way that is favorable for the overall quality of a schedule, it is common to manipulate the agents by introducing *payments*. The payments depend on the actions of all agents and specify for each agent how much (money) is to be payed (or received) by that agent. Given the actions a of all agents, let $\pi_k(a)$ denote the required payment for agent k. This could be both positive or negative. The overall *payment scheme* π is then a mapping from the action space A to the space of all possible payments. Assuming we have l agents, we thus have $\pi : A \to \mathbb{R}^l$. (More precisely, we should write $\pi : I \times A \to \mathbb{R}^l$.)

In the models we address, we express the relation of valuations to payments by so-called *quasi-linear utilities*. That means that the *utility* u_k that an agent k receives from a schedule is just the valuation minus the payment. More precisely, if a schedule $\sigma = f(a)$ is computed by some allocation algorithm f on the basis

of actions a of all agents, with associated payments $\pi(a)$, then the utility of agent k (being of type t_k) is given by $u_k(f(a), t_k) = v_k(f(a), t_k) - \pi_k(a)$. Finally, notice that we assume that agents are *rational*; meaning that they aim at maximizing their utilities.

4.3 Models with Complete Information

When agents do not have any private information, we talk about *games with complete information*. In these settings, a strategy of an agent is simply the choice of an action, and it does not depend on any private information. Therefore, we can identify strategies X_k with actions A_k for every agent. (Recall that in models where agents have private types t_k, a strategy $x_k \in X_k$ maps possible types from T_k to actions in A_k.) As it is common practice in game theory, we will adopt the term strategy for the actions of agents, and we will use X_k instead of A_k. A game is then simply a mapping from the set of strategies of the agents to the set of schedules, coinciding with the allocation algorithm f defined earlier. An agent k's valuation for a schedule $\sigma \in \mathfrak{S}$ can be written simply as $v_k(\sigma)$, because it does not depend on a potential type t_k of that agent. Since the schedule σ only depends on the agent's strategies, the valuation can also be expressed as the valuation $v_k(x)$ for a certain strategy vector $x \in X$.

In a game with payments, we can compute the utility of an agent k from its valuation for a certain strategy vector x and its payment given that strategy vector x as $u_k(x) = v_k(x) - \pi_k(x)$. In a game without payments, an agent's utility equals its valuation; we use the term utility also in that case.

In general, agents are also allowed to play *mixed strategies*. A mixed strategy of an agent k is a probability distribution over the set of its *pure strategies* X_k. We denote the set of probability distributions over the pure strategy set X_k by $\Delta(X_k)$. For a given vector of mixed strategies, the utilities for the individual agents as well as the objective function value become random variables. A Nash equilibrium is then defined as follows.

Definition 4.1. *A strategy vector $x = (x_1, \dots, x_l) \in \Delta(X_1) \times \cdots \times \Delta(X_l)$ is called* Nash equilibrium *if for every agent $k = 1, \dots, l$*

$$\mathbb{E}[u_k(x)] \geq \mathbb{E}[u_k(x_1, \dots, x_{k-1}, x'_k, x_{k+1}, \dots, x_l)] \qquad \forall x'_k \in \Delta(X_k).$$

Here, $\mathbb{E}[\,\cdot\,]$ denotes the expectation. In a model where only pure strategies are allowed, this definition reduces to the following.

Definition 4.2. *A strategy vector* $x = (x_1, \ldots, x_l) \in X_1 \times \cdots \times X_l$ *is called* pure strategy Nash equilibrium *if for every agent* $k = 1, \ldots, l$

$$u_k(x) \geq u_k(x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_l) \qquad \forall x'_k \in X_k.$$

In general, Nash equilibria in pure strategies do not necessarily exist. Existence of Nash equilibria is only guaranteed if agents are allowed to play mixed strategies. Therefore, an interesting question is the existence of pure strategy Nash equilibria for a given problem. Moreover, one is interested in algorithms to compute pure or mixed strategy Nash equilibria (efficiently).

A third issue that is addressed in the literature that is specific to games with complete information is the following question. How does the objective value that results from a Nash equilibrium – thus a solution induced by utility maximizing selfish agents – compare to the optimal objective value. The latter might just be computed by some central authority. The extent to which the objective value deteriorates due to the *lack of central coordination* is called the *price of anarchy*. It can be defined for pure as well as for mixed strategy settings.

Definition 4.3. *For a minimization problem, let* V_{OPT} *be the optimal objective value and let* V_{NE} *be the worst possible objective value achieved by any (pure-strategy) Nash equilibrium. Then the* price of anarchy *(of pure Nash equilibria) is defined as*

$$POA = \frac{V_{NE}}{V_{OPT}}\,.$$

Accordingly, one defines the price of anarchy as V_{OPT}/V_{NE} in a maximization problem. The study of the price of anarchy as the worst case ratio between the objective value of a Nash equilibrium and that of the overall system optimum was initiated by Koutsoupias and Papadimitriou (1999). They were motivated by the fact that Nash equilibria in general do not optimize the overall performance of the system, the most prominent example being the Prisoner's Dilemma, see e.g. Owen (1995). In a part of the literature, the price of anarchy is also referred to as *coordination ratio*.

The following sections highlight a sample of different scheduling settings, their Nash equilibria, and the corresponding prices of anarchy.

4.3.1 The Price of Anarchy in Congestion Models

We first define the model as described and analyzed by Koutsoupias and Papadimitriou (1999). Consider n job-agents $j \in J = \{1, \ldots, n\}$ with processing times p_j that have to be processed on m machines $i \in M = \{1, \ldots, m\}$ with possibly different speeds $s_i > 0$. This is the related machine scheduling model, and if all speeds s_i are equal, the parallel machine scheduling model. Each pure strategy of an agent corresponds to the deterministic selection of one of the machines. A mixed strategy of agent j assigns a probability q_i^j to every machine i, such that $\sum_{i=1}^{m} q_i^j = 1$ for all j.

We call the model *congestion model* due to the following assumption. It is assumed that the valuation of any job for a given schedule is determined by the total processing time of the jobs assigned to the same machine. Stated otherwise, jobs are released from a machine only when the machine has finished *all* the jobs assigned to it. We therefore define the utility of job j with strategy $i \in M$ as follows. For any vector of pure strategies (i, i_{-j}),

$$u_j(i, i_{-j}) = -\frac{1}{s_i} \sum_{k:i_k=i} p_k.$$

The expected utility of agent j when the mixed strategy vector $(q^1, \ldots, q^n)$ is played is then

$$\mathbb{E}[u_j(q^1, \ldots, q^n)] = -\sum_{i=1}^{m} \frac{q_i^j}{s_i} \left(p_j + \sum_{k \neq j} q_i^k p_k \right).$$

The objective of the central authority is to minimize the makespan of the overall schedule, i.e.,

$$V_{OPT} = \min_{i_1, \ldots, i_n} \max_i \frac{1}{s_i} \sum_{j:i_j=i} p_j.$$

The model was originally motivated by regarding the machines as network links and the jobs as traffic that has to be routed via the links. The utility of each agent is then defined by the delay it experiences when being routed via a specific link, caused by the corresponding total congestion of that link. The utilities as defined above are therefore also called *linear cost functions*, as the congestion depends linearly on the total load assigned to that link.

For the case with two identical machines, i.e., machines with equal speeds, it was shown that the price of anarchy (for mixed strategies) is equal to 3/2 (Koutsoupias and Papadimitriou 1999). We present their example showing why the price of anarchy is at least 3/2.

Example 4.4. *Consider two jobs with $p_1 = p_2 = 1$ and two machines with unit speed. Then a mixed Nash equilibrium is the choice $q_i^j = 1/2$ for $i, j = 1, 2$. In that Nash equilibrium, both jobs choose the same machine with probability 1/2, resulting in makespan 2. With probability 1/2, the jobs are processed by different machines, which gives a makespan of 1. Therefore, the expected objective value is 3/2. The optimum is to assign both jobs to different machines, yielding an objective value of 1. Therefore the price of anarchy for minimizing the schedule makespan in the congestion model is at least 3/2.*

A matching upper bound of 3/2 for the price of anarchy can be derived as well.

Theorem 4.5 (Koutsoupias and Papadimitriou 1999)**.** *For $m = 2$ identical machines, the price of anarchy for minimizing the makespan in the congestion model is 3/2.*

Let us briefly summarize further (and more general) results by Koutsoupias and Papadimitriou (1999) and Czumaj and Vöcking (2007).

For an arbitrary number m of identical machines, Koutsoupias and Papadimitriou (1999) show that the POA is at least $\Omega(\log m/(\log\log m))$. This result goes back to the classical *bins-and-balls* result: When throwing m balls into m bins uniformly at random, then the expected maximum number of balls in any bin is $\Theta(\log m/(\log\log m))$. To translate this into the given setting, consider the case with m machines and m jobs with unit processing times. One can check that there is a Nash equilibrium where every job randomizes uniformly over all machines. In this Nash equilibrium, the expected makespan is $\Theta(\log m/(\log\log m))$, due to the bins-and-balls result. In the optimal solution, however, each machine processes exactly one job, yielding a makespan of 1. The claimed lower bound on the POA follows. Czumaj and Vöcking (2007) establish a matching upper bound of $\mathcal{O}(\log m/(\log\log m))$ for the POA on m machines; they even give an exact expression for the price of anarchy for that case.

For the case with two *related* machines (two machines with different speeds), the POA is equal to the golden ratio $\varphi \approx 1.618$ (for the lower bound see Koutsoupias and Papadimitriou 1999, the upper bound follows from Cho and Sahni 1980 and Schuurman and Vredeveld 2007 as described in Feldmann, Gairing, Lücking, Monien, and Rode 2003). For the more general case with m related machines, Czumaj and Vöcking (2007) show that the price of anarchy is in $\Theta(\log m/(\log\log\log m))$. This completes the picture for models with linear cost functions and identical or related machines. For other extensions (e.g., non-linear congestion models), we refer to the survey by Czumaj (2004).

Clearly, the price of anarchy when mixed strategies are allowed is at least as large as the price of anarchy when only pure Nash equilibria are considered. Pure strategy Nash equilibria are not analyzed by Koutsoupias and Papadimitriou (1999). We shortly elaborate here on the price of anarchy for pure Nash equilibria for the identical machine setting.

Theorem 4.6. *For $m = 2$ identical machines, the price of anarchy of pure Nash equilibria for minimizing the schedule makespan in the congestion model is 4/3.*

Proof. To see that the POA is at least 4/3, consider the following example. There are four jobs with $p_1 = p_2 = 1$ and $p_3 = p_4 = 2$. In an optimal solution, every machine processes one job of length 1 and one of length 2, yielding a makespan of 3. One pure Nash equilibrium is the strategy vector $(1, 1, 2, 2)$, i.e., the two short jobs go on the first machine, whereas the two long jobs go on the second machine. In that situation none of the jobs has an incentive to change the machine unilaterally. The makespan of this Nash equilibrium is 4, which proves that 4/3 is a lower bound on the price of anarchy of pure Nash equilibria.

To prove that the POA is at most 4/3, consider any schedule in (pure) Nash equilibrium. Denote by L_1 and L_2 the total loads of machines 1 and 2, respectively, and assume w.l.o.g. $L_2 \geq L_1$. Let $\delta = L_2 - L_1 \geq 0$. The makespan of the schedule in Nash equilibrium is hence $V_{NE} = L_2 = L_1 + \delta$. If there is only one job on machine 2, then no schedule can have a smaller makespan, and the schedule is optimal. Therefore, we assume that there are at least two jobs on machine 2. Any job on machine 2 must have a processing time at least δ, as any job with smaller processing time would have an incentive to change to machine 1. Therefore, $L_1 + \delta \geq 2\delta$ and thus $L_1 \geq \delta$. Since no schedule can do better than distributing the total processing time equally over both machines, $V_{OPT} \geq L_1 + \delta/2$. Thus we have

$$POA = \frac{V_{NE}}{V_{OPT}} \leq \frac{L_1 + \delta}{L_1 + \frac{\delta}{2}}.$$

This expression is maximized when L_1 is small. Using $L_1 \geq \delta$, we therefore get

$$POA \leq \frac{2\delta}{\frac{3}{2}\delta} = \frac{4}{3}.$$

□

In fact, the same proof technique works for an arbitrary number of machines m. One derives that the price of anarchy is at most $2 - 2/(m+1)$ (Finn and Horowitz 1979). A matching lower bound was given by Schuurman and Vredeveld (2007).

Theorem 4.7 (Finn and Horowitz 1979, Schuurman and Vredeveld 2007)**.** *For an arbitrary number of identical machines, the price of anarchy of pure Nash equilibria for minimizing the makespan in the congestion model is* $2 - 2/(m+1)$.

4.3.2 The Price of Anarchy in Sequencing Models

In the models of the previous section, the utility of any job assigned to a certain machine does only depend on the *total load* of that machine, but not on the *sequence* of the jobs on that machine. Next we discuss models where each job j's utility depends only on its own completion time C_j, and is independent of the processing that might occur later than C_j on the same machine.

Clearly, different local sequencing policies on the machines will yield different Nash equilibria, and the price of anarchy will depend on the employed local sequencing policies. The analysis of local sequencing policies in such settings was termed *coordination mechanisms* in the paper by Christodoulou, Koutsoupias, and Nanavati (2004). However, we prefer to not use this term in this context, as we reserve the term "mechanism" for problems where agents have private (type) information; this is not the case here.

In the following, we examine the price of anarchy in different scheduling models and with different local sequencing policies. As in the preceding section, the central authority objective is to minimize the makespan $C_{\max}$ of the overall schedule. Our aim is to only highlight a few phenomena and proof techniques rather than to give a complete survey of the known results. For a more comprehensive overview, we refer to the paper by Immorlica, Li, Mirrokni, and Schulz (2008).

Consider again the setting where n job-agents have to choose one out of m machines to be processed on, thus the jobs' actions are again $(i_1, \ldots, i_n) \in M^n$. Each job seeks to minimize its own completion time C_j, thus

$$u_j(i_1, \ldots, i_n) = -C_j(i_1, \ldots, i_n)\,,$$

where $C_j(i_1, \ldots, i_n)$ is the completion time of job j in dependence on the jobs' actions and the sequencing of the jobs per machine.

We will use the term *local sequencing policy* to denote the sequencing policies implemented locally by the machines. As it turns out, for some local sequencing policies, the schedules resulting from (pure strategy) Nash equilibria coincide with the

outcome of well-known, classical scheduling algorithms. In such cases, for analyzing the price of anarchy, we can just exploit well known results on the performance of those scheduling algorithms. To avoid confusion, note that we use the term 'policy' only for local sequencing policies, while we use the term 'algorithm' only for (centrally coordinated) scheduling algorithms.

We consider the most general of the three scheduling models, namely *unrelated* machine scheduling; thus if job j is scheduled on machine i, its processing time is p_{ij}. In the *Local SPT policy*, every machine processes the jobs that have selected that machine in order of non-decreasing processing times. As it turns out (Immorlica, Li, Mirrokni, and Schulz 2008), the pure strategy Nash equilibria of the Local SPT policy coincide with the schedules that result from the *Ibarra-Kim* algorithm (Ibarra and Kim 1977). Notice that we assume in both cases that ties between jobs with equal processing times on one machine are broken in favor of the job with smaller index.

> **Ibarra-Kim algorithm.** In each of the iterations $\tau = 1, \ldots, n$ of the algorithm, select a pair (j, i) where j is an unscheduled job and i is a machine. If $C_j^\tau(i)$ denotes the completion time of job j when scheduled after all jobs already assigned to machine i in iterations $1, \ldots, \tau - 1$, we select (j, i) as $\operatorname{argmin}_{i,j} C_j^\tau(i)$. We break ties by choosing a minimal j. In iteration τ, job j is then scheduled on machine i after all jobs already scheduled on i.

Theorem 4.8. *For unrelated (related, parallel) machines, the set of pure Nash equilibria for the Local SPT policy is precisely the set of solutions of the Ibarra-Kim algorithm.*

A proof of this result can be found in Immorlica, Li, Mirrokni, and Schulz (2008). However, when the paper corresponding to this chapter was published, this proof was not available yet. We present here our independent proof from Heydenreich, Müller, and Uetz (2007).

Proof. Consider any job j, and consider the iteration when the Ibarra-Kim algorithm places job j on a machine minimizing j's completion time. At that iteration, for all machines i, and all jobs k already scheduled on machine i, it holds that $p_{ik} < p_{ij}$ (or such job k has the same processing time but a smaller index than j). Thus, in the final schedule, assuming that machines implement the Local SPT policy, j cannot be processed before any of those jobs k either. Given this constraint, however, j is already sitting on a machine that minimizes its completion time. Thus, j cannot

improve its completion time by unilaterally changing to another machine. That means that the Ibarra-Kim schedule is a Nash equilibrium for the Local SPT policy.

Conversely, consider any schedule that is a pure strategy Nash equilibrium for the Local SPT policy. In that schedule, for any job j, denote by i_j the machine that hosts job j and let C_j^N be the completion time of job j (N for Nash equilibrium). Sort the jobs in order of non-decreasing completion times C_j^N. Note that jobs with equal completion times must be scheduled on different machines; let their respective order be chosen with respect to their index. We now schedule the jobs in this order on their respective machines i_j, and claim that this coincides with a run of the Ibarra-Kim algorithm. We need to show that whenever the τth job, say job j, is scheduled on its machine i_j, the combination (j, i_j) minimizes the completion time $C_k^\tau(i)$ among all combinations of unscheduled jobs k and machines i, where ties are broken by job index k. Suppose this is not the case and let (j, i_j) be the first job-machine pair for which the claim does not hold, j being the τth job in the given order. Then in iteration τ, there is a different job-machine pair (k, i) with $C_k^\tau(i) < C_j^\tau(i_j)$ or $C_k^\tau(i) = C_j^\tau(i_j)$ and $k < j$. Choose (k, i) such that $C_k^\tau(i)$ is minimum, and break ties according to smallest job index.

First, we argue that $C_k^\tau(i) < C_k^N$. Indeed, since

$$C_k^\tau(i) \leq C_j^\tau(i_j) = C_j^N \leq C_k^N \,, \tag{4.1}$$

we have $C_k^\tau(i) \leq C_k^N$. Assume that $C_k^\tau(i) = C_k^N$. Then we conclude from (4.1) that $C_k^\tau(i) = C_j^\tau(i_j)$, thus by the choice of k we must have $k < j$. But by (4.1) we also have that $C_j^N = C_k^N$. This, together with $k < j$, is a contradiction to the definition of our procedure, since we break ties according to smaller job index. Hence we must have $C_k^\tau(i) < C_k^N$.

Next, we claim that all jobs l that are hosted by machine i in the Nash-equilibrium, and that would precede k according to the Local SPT-policy if k chose machine i, are already present on machine i at iteration τ. To prove this claim, let l be a such a job. Since l would precede k, either $p_{il} < p_{ik}$, or $p_{il} = p_{ik}$, but $l < k$. In both cases, if l is not yet scheduled at iteration τ, its existence contradicts the choice of k.

This claim implies that in the Nash equilibrium, job k could improve its com-

pletion time C_k^N to $C_k^\tau(i) < C_k^N$ by choosing machine i, a contradiction. □

Utilizing this result, the price of anarchy of pure Nash equilibria for the Local SPT policy can be derived from known results on the performance of the Ibarra-Kim algorithm. Using such results by Graham (1966) for parallel machines, by Aspnes, Azar, Fiat, Plotkin, and Waarts (1997) and Immorlica, Li, Mirrokni, and Schulz (2008) for related machines, and by Ibarra and Kim (1977) and Azar, Jain, and Mirrokni (2008) for unrelated machines, one gets the following.

Theorem 4.9. *The price of anarchy for minimizing the makespan in the sequencing model, when using the Local SPT policy on each machine, is*

- $2 - 1/m$ *on parallel machines,*
- $\Theta(\log m)$ *on related machines, and*
- $\Theta(m)$ *on unrelated machines.*

For the sake of completeness, we mention that for the case of unrelated machines, the lower bound $\Omega(m)$ is not obtained by analyzing the Ibarra-Kim algorithm; the proof by Azar, Jain, and Mirrokni (2008) is based on other techniques. But due to Theorem 4.8, the authors simultaneously obtain a lower bound for the Ibarra-Kim algorithm as well, answering a question that has been open for a long time. In fact, they prove the more general result that every strongly local sequencing policy satisfying IIA results in a price of anarchy of $\Omega(m)$. Here, strongly local means that every machine uses only information on the processing times of the jobs on that machine to sequence the jobs. IIA is defined in a slightly different way than in Chapter 3, meaning that the relative order of two jobs must not depend on the *existence* of some other job.

For the case of parallel machines, the Ibarra-Kim algorithm is in fact equivalent to the classical greedy SPT algorithm.

> **Greedy SPT algorithm.** Whenever a machine becomes idle, start a job with the shortest processing time among all remaining unscheduled jobs.

Theorem 4.9 states that this algorithm yields a schedule with makespan no more than $2 - 1/m$ times the optimal makespan. However, for the parallel machine case, the LPT algorithm yields an even better performance bound of $4/3 - 1/(3m)$ (Graham 1969).

> **Greedy LPT algorithm.** Whenever a machine becomes idle, start a job with the longest processing time among all remaining unscheduled jobs.

This motivates the analysis of the Local LPT policy on parallel machines. Again, it can be shown that pure strategy Nash equilibria correspond to the output of the greedy LPT algorithm (Christodoulou, Koutsoupias, and Nanavati 2004). The well known analysis of the LPT-algorithm by Graham (1969) now yields the following.

Theorem 4.10 (Christodoulou, Koutsoupias, and Nanavati 2004)**.** *For the parallel machine setting, the price of anarchy for minimizing the makespan in the sequencing model when using the Local LPT policy on each machine, is* $4/3 - 1/(3m)$.

Clearly, from the above result it follows that the price of anarchy is at least $4/3 - 1/(3m)$ also for related (or unrelated machines). Immorlica, Li, Mirrokni, and Schulz (2008) obtain even constant bounds following from the work of Dobson (1984) and Friesen (1987). We mention the following result without a proof.

Theorem 4.11 (Immorlica, Li, Mirrokni, and Schulz 2008, Dobson 1984, Friesen 1987)**.** *For the related machine setting, the price of anarchy for minimizing the makespan in the sequencing model, when using the Local LPT policy on each machine, is bounded as follows:* $1.52 \leq POA \leq 1.59$.

Consider now the case of unrelated machines. In contrast to the price of anarchy of the Local SPT policy, which is in $\Theta(m)$ by Theorem 4.9, it is *unbounded* for the Local LPT policy. To that end, consider the following example.

Example 4.12. *Consider two machines* 1 *and* 2 *and two jobs* 1 *and* 2*. Let* $p_{11} = p_{22} = 1$ *and* $p_{12} = p_{21} = K$ *for some constant* $K > 0$*. Then in one Nash equilibrium, job 1 is processed by machine 1 and job 2 by machine 2. The makespan of the resulting schedule is 1. In the other Nash equilibrium, job 1 is processed on machine 2 and job 2 on machine 1. Because longer jobs are processed before shorter ones on every machine, unilaterally changing the machine is not beneficial for either job. The makespan is* K *in this case. Therefore, the price of anarchy is equal to* K *and is hence unbounded.*

The question of the *existence* of pure strategy Nash equilibria in the above mentioned settings is often answered by showing that a certain local sequencing policy constitutes a so-called *potential game*. This method is used by Immorlica,

Li, Mirrokni, and Schulz (2008) for analyzing the Local SPT policy for the case of related machines. Potential games have a *potential function* mapping strategy vectors to real numbers such that the potential function decreases whenever an agent unilaterally changes its strategy in such a way that its own utility increases. Minima of the potential function then correspond to pure strategy Nash equilibria of the game. Potential functions where first used by Rosenthal (1973) and formally introduced by Monderer and Shapley (1996). We refer to those references for further reading.

Notice that the link to potential games also establishes a close relationship to *local search algorithms* in optimization. One can define a *local search neighborhood* of a given strategy vector by considering all strategy vectors where only one agent has changed its strategy to the best response against the given strategies of the other agents. The potential function takes the role of the objective function of the local search. Local optima of those neighborhoods then correspond to pure strategy Nash equilibria. The analysis of the quality of local optima is therefore closely related to the analysis of (pure strategy) Nash equilibria. The quality of local optima of several neighborhoods in machine scheduling was analyzed, for example, by Schuurman and Vredeveld (2007).

In other settings, pure strategy Nash equilibria might not even exist. Azar, Jain, and Mirrokni (2008) illustrate this by giving the example of a local sequencing policy for unrelated machine scheduling where jobs are scheduled on each machine in order of their non-decreasing inefficiencies on that machine. The inefficiency of a job on a machine is defined as the ratio of the processing time on that machine over the smallest possible processing time on any other machine. This local policy is clearly not strongly local in the sense defined above, but it satisfies IIA and it is proved that the price of anarchy when using this policy is in $\Theta(\log m)$ beating the lower bound for strongly local policies. On the other hand, the authors construct an input instance such that the game resulting from using the inefficiency based local sequencing policy has no pure strategy Nash equilibrium.

4.3.3 The Price of Anarchy for Other Objective Functions

To our knowledge, the price of anarchy in scheduling has almost exclusively been studied with respect to the makespan objective. However, the strategic behavior of a job-agent that seeks to minimize its own completion time is not affected by the objective function of the central authority. Therefore, the Nash equilibria for the different models discussed in the previous section remain Nash equilibria if the central authorities' objective function is modified.

As an example, we consider a model that differs only slightly from the models

in the previous section. Each job-agent now has a weight w_j additionally to its processing time p_j, and as before, seeks to be finished as early as possible. The weight can be regarded as disutility per unit waiting time. Then, a job j's disutility for a schedule which gives it completion time C_j is $w_j C_j$. The strategy of each job remains the choice of a machine. As central objective, we consider the minimization of the total weighted completion time $\sum_{j \in J} w_j C_j$. This corresponds to maximizing the total social welfare, similarly as in Chapters 3 and 5.

The most natural (because optimal) local sequencing policy on the machines is then the well known Local WSPT policy, also known as Smith's rule: each machine processes its jobs in the order of non-increasing ratios w_j/p_j. For each machine individually, this yields the minimum total weighted completion time $\sum_{j \in J} w_j C_j$ (Smith 1956). Consider now the following algorithm.

> **(Ibarra-Kim version of) WSPT algorithm.** Sort the jobs in order of their weight over processing time ratios w_j/p_j, largest first. In that order, schedule each job on the machine that minimizes its completion time when scheduled after all jobs already scheduled on that machine.

Notice that for parallel machines, this algorithm reduces to the classical WSPT algorithm that just schedules the jobs during execution greedily in order of non-increasing ratios w_j/p_j. The worst case behavior of this algorithm has been analyzed by Kawaguchi and Kyan (1986). Similar to the proof of Theorem 4.8, one can show that the set of Nash equilibria of the Local WSPT policy is equal to the set of all possible outputs of this WSPT algorithm.

Theorem 4.13. *For related (parallel) machines, the set of pure Nash equilibria for the Local WSPT policy is precisely the set of solutions of the above WSPT algorithm.*

Consequently, the price of anarchy of the Local WSPT policy follows from the analysis of that algorithm. For parallel machines, the work of Kawaguchi and Kyan (1986) thus yields the following.

Theorem 4.14. *For parallel machines, the price of anarchy of the Local WSPT policy for minimizing $\sum w_j C_j$ in the sequencing model is $(\sqrt{2}+1)/2 \approx 1.207$.*

For the case of related machines, we are not aware of non-trivial bounds on the price of anarchy. For the unrelated machine case, it is not even clear whether or not a pure strategy Nash equilibrium exists. However, if all jobs have the same weights, then the Local WSPT policy is equivalent to the Local SPT policy. The existence of pure strategy Nash equilibria then follows from Theorem 4.8. Also for this case, however, we are not aware of any non-trivial bound on the price of anarchy.

4.4 Models with Private Information

In the previous section, we addressed the question what happens if in a scheduling application a part of the decisions is left to selfish job agents. Given various policies that determine how jobs are scheduled on the selected machines, we compared the objective in equilibrium with the objective of the optimal solution. In this section, we additionally assume that the agents own some private information, namely their types, and these are not publicly known. For any given agent, its type will influence its action in the game. Since agents do not know other agents' types, they do not know which actions are beneficial for the other agents and therefore which actions other agents are likely to chose. This additional uncertainty results in more complicated equilibrium concepts than in the previous section. We start with some general notation, then discuss general techniques and key results, and finally review a couple of interesting models related to scheduling.

4.4.1 Mechanism Design

Let us denote a mechanism with allocation algorithm f and payment scheme π by $\mu = (f, \pi)$. We will present several examples for mechanisms in the following sections. Next, we introduce the equilibrium that is most robust towards the information uncertainty described above and that is at the same time the one that is best studied in the algorithmically oriented literature in mechanism design.

Recall that for agent k, a strategy x_k is a mapping from types t_k to actions a_k. We denote by t_{-k}, x_{-k} and a_{-k} the vectors of types, strategies and actions respectively of all agents other than k. For the type, strategy and action vector of all the agents, we then write (t_k, t_{-k}), (x_k, x_{-k}), and (a_k, a_{-k}).

Definition 4.15. *Let $\mu = (f, \pi)$ be a mechanism. A strategy vector $x \in X$ is called a* dominant strategy equilibrium*, if for all agents k, for all types t_k of agent k, for all actions a_{-k} of the other agents and all alternative actions a_k of agent k it holds that*

$$v_k(f(x_k(t_k), a_{-k}), t_k) - \pi_k(x_k(t_k), a_{-k}) \geq v_k(f(a_k, a_{-k}), t_k) - \pi_k(a_k, a_{-k}).$$

Remarkably, this means that independent of which actions the other agents take, it never pays off for any agent k, to deviate from its strategy x_k.

In most cases, the revelation principle as mentioned in the introduction applies and we can restrict ourselves to the design of *direct revelation mechanisms*. Hence, this class of mechanisms receives great attention in the literature (Briest, Krysta,

and Vöcking 2005; Gui, Müller, and Vohra 2004; Bikhchandani, Chatterjee, Lavi, Mu'alem, Nisan, and Sen 2006; Saks and Yu 2005). In a direct revelation mechanism, the only action that an agent is required to take is reporting its type, thus $x_k \colon T_k \to T_k$. Assume we have a scheduling problem where part of the instance is private information of the agents. Given reports about that private information, we can define the allocation algorithm that merely chooses an optimal solution. (For the time being, we are not addressing the question *how* this optimal solution is derived.) Let us denote it by the *exact allocation algorithm*. Without payments, utility maximizing agents might misreport their private information in such settings in order to achieve a more favorable outcome. With well-designed payments, however, agents may get incentives to report their private information truthfully in the following sense.

Definition 4.16. *A direct revelation mechanism is called* dominant strategy incentive compatible*, or* truthful*, if the strategy vector x in which each agent truthfully reports its type, that is, $x_k = id$ for all k, is a dominant strategy equilibrium. An allocation algorithm f is said to be* truthfully implementable*, if we can find a payment rule π such that the mechanism $\mu = (f, \pi)$ is truthful.*

Given some optimization problem, if the exact allocation algorithm is truthfully implementable, there is of course still the issue whether the algorithm runs in polynomial time, and whether the payments can be computed in polynomial time. If we leave this algorithmic problem out of consideration, however, it is often surprisingly easy to provide a truthful implementation, because many optimization problems are special cases of the setting in which we can apply so-called Vickrey-Clarke-Groves (VCG) payments. In what follows we use the notation by Roberts (1979).

Definition 4.17. *Given l agents, their types $t_1, \ldots, t_l$, valuation functions $v_1, \ldots, v_l$, strictly positive weights $\gamma_1, \ldots, \gamma_l$, and constants β_y for every $\sigma \in \mathfrak{S}$, an allocation algorithm f is called an* affine maximizer*, if it chooses a schedule $\sigma \in \mathfrak{S}$ that maximizes $\beta_\sigma + \sum_{k=1}^{l} \gamma_k v_k(\sigma, t_k)$.*

The following theorem is in this generality due to Roberts (1979). For all weights equal to 1, it has been proven by Clarke (1971) and Groves (1973), while for the special case of single-item auctions it has been proven by Vickrey (1961).

Theorem 4.18. *Let an allocation algorithm f be an affine maximizer, and let for every agent k, h_k be an arbitrary function mapping type reports t_{-k} of the other*

agents to real numbers. Then the mechanism $\mu = (f, \pi)$ is truthful if the payments are defined as follows.

$$\pi_k(t) = h_k(t_{-k}) - \frac{1}{\gamma_k}\beta_{f(t)} - \sum_{k' \neq k} \frac{\gamma_{k'}}{\gamma_k}\, v_{k'}(f(t), t_{k'})\,.$$

It is intuitive to give the very short proof of this important theorem.

Proof. Let us assume that agent k reports $\hat{t}_k$ instead of its true type t_k, and let $\hat{t} = (\hat{t}_k, t_{-k})$. Since f is an affine maximizer we have:

$$\beta_{f(t)} + \sum_k \gamma_k v_k(f(t), t_k) \geq \beta_{f(\hat{t})} + \sum_k \gamma_k v_k(f(\hat{t}), t_k),$$

which implies:

$$\begin{aligned} & v_k(f(t), t_k) + \frac{1}{\gamma_k}\beta_{f(t)} + \sum_{k' \neq k} \frac{\gamma_{k'}}{\gamma_k}\, v_{k'}(f(t), t_{k'}) \\ \geq\ & v_k(f(\hat{t}), t_k) + \frac{1}{\gamma_k}\beta_{f(\hat{t})} + \sum_{k' \neq k} \frac{\gamma_{k'}}{\gamma_k}\, v_{k'}(f(\hat{t}), t_{k'}). \end{aligned}$$

If we subtract $h_k(t_{-k})$ on both sides of this inequality, we get on the left hand side the utility of agent k for truth-telling, and on the right hand side its utility when reporting $\hat{t}_k$. □

Note that the generality of the functions h_k gives some flexibility to define payments. In auctions, for example, one uses this flexibility to adjust prices such that agents who do not win any object pay 0. Note further that, for fixed type report t_{-k}, agent k pays a price that depends only on the allocation that is selected by the affine maximizer, and not on its particular type report by which this allocation is achieved. It is easy to see that this so-called *taxation principle* does not only hold for affine maximizers and their VCG payments, but it must hold for any truthful mechanism.

Let us provide an example application of VCG payments in the context of scheduling.

Example 4.19. *Suppose there is a single machine, and there are job-agents whose characteristics – weights w_j and processing times p_j – are private information. The valuation of an agent is the negative of its weighted completion time (in order to*

make the agents utility maximizers). Let f be the exact allocation algorithm, i.e., f chooses a schedule minimizing the weighted sum of completion times. Note that this is an affine maximizer, since it maximizes the sum of agent valuations. Let us use the notation $C_k(t)$ for the completion time of agent k in the optimal solution of the instance with all agents, and $C_k(t_{-j})$ as the completion of agent k in the optimal solution of the instance in which j is not present. Now choose $h_j(t_{-j})$ as the negative of the optimal weighted sum of completion times if agent j is not present. We get the following VCG payments.

$$\begin{aligned}\pi_j(t) &= -\sum_{k\neq j} w_k C_k(t_{-j}) + \sum_{k\neq j} w_k C_k(t) \\ &= \sum_{k\neq j} w_k (C_k(t) - C_k(t_{-j})) = \sum_{k \text{ delayed by } j} w_k p_j.\end{aligned}$$

Here, the last sum is restricted to those jobs that are delayed due to the presence of j. In other words, job j pays for the decrease in utility of other agents. By Theorem 4.18, with these payments, agents maximize their utility by reporting their types truthfully.

Notice that the scheduling problem of Example 4.19 does not only allow for a truthful implementation, but at the same time is the allocation algorithm a polynomial time algorithm: The exact allocation algorithm f just schedules the jobs in the order of non-increasing ratios w_j/p_j (Smith 1956). Given the reports of all agents, also the payments can be computed efficiently.

It is often the case, however, that an exact allocation algorithm is not that easily obtainable, for example because the underlying optimization problem is NP-hard. If instead of an exponential time exact allocation algorithm, we use an allocation algorithm that is a (suboptimal) heuristic, this algorithm is generally not an affine maximizer, and computing payments with the VCG formula on the basis of the solutions computed by the heuristic does not necessarily yield a truthful mechanism (Nisan and Ronen 2000; Ronen 2006).

On the other hand, even if an allocation algorithm is *not* an affine maximizer, it might be truthfully implementable. In order to verify whether a given allocation algorithm is truthfully implementable, and in order to determine the required payments, a characterization of truthfully implementable allocation algorithms is of great importance. Such characterizations have been given by Monderer (2007) and Saks and Yu (2005), generalizing earlier results by Bikhchandani, Chatterjee,

Lavi, Mu'alem, Nisan, and Sen (2006), Gui, Müller, and Vohra (2004), and Roberts (1979). We review the definition of monotonicity, which has been stated in Chapter 3 for more specific valuation functions. Often, monotonicity is also referred to as weak monotonicity.

Definition 4.20. *An allocation algorithm f is said to satisfy* monotonicity *if for all agents k, for all types t_k, $\hat{t}_k$ of agent k, and for all types t_{-k} of other agents:*

$$v(f(t_k, t_{-k}), t_k) - v(f(t_k, t_{-k}), \hat{t}_k) \geq v(f(\hat{t}_k, t_{-k}), t_k) - v(f(\hat{t}_k, t_{-k}), \hat{t}_k).$$

Monotonicity is a necessary condition for a truthful implementation of an allocation algorithm. Indeed, using the types as given in the definition, and assuming that truthful payments exist, we get the following two inequalities, from which monotonicity follows:

$$\begin{aligned} v(f(t_k, t_{-k}), t_k) - \pi(t_k, t_{-k}) &\geq v(f(\hat{t}_k, t_{-k}), t_k) - \pi(\hat{t}_k, t_{-k}) \\ v(f(\hat{t}_k, t_{-k}), \hat{t}_k) - \pi(\hat{t}_k, t_{-k}) &\geq v(f(t_k, t_{-k}), \hat{t}_k) - \pi(t_k, t_{-k}). \end{aligned}$$

For some settings, monotonicity of an allocation rule is even a sufficient condition for truthful implementability, see Theorem 4.24 below. Saks and Yu have shown the more general result that monotonicity is sufficient for convex domains:

Theorem 4.21 (Saks and Yu 2005)**.** *Let the set of outcomes $\mathfrak{S}$ be finite, and let for all agents k the type be represented as a valuation vector with a valuation for every possible outcome $\sigma \in \mathfrak{S}$. That is, $T_k \subseteq \mathbb{R}^{\mathfrak{S}}$ and $v_k(\sigma, t_k) = t_{k\sigma}$, $\sigma \in \mathfrak{S}$. Furthermore, assume that all T_k are convex. Then an allocation algorithm $f : A \to \mathfrak{S}$ is truthfully implementable if and only if f satisfies monotonicity.*

The proof of Theorem 4.21 is based on the link between the implementability of an allocation algorithm and the absence of negative cycles in the allocation graphs obtained for any agent and any fixed report of the other agents, as introduced in Chapter 2. In fact, in many settings, the constructed allocation graphs have no negative cycle if and only if they have no negative 2-cycle. Theorem 4.21 provides an example of such a setting. In Section 4.4.2 we show how allocation graphs can be used to provide an alternative proof for a result given originally by Archer and Tardos (2001).

Monderer (2007) generalizes Theorem 4.21 to domains with convex closure and provides a simpler proof than Saks and Yu (2005).

4.4.2 Performance of Truthful Mechanisms in Machine Scheduling

In this section, we regard specific scheduling models from a mechanism design perspective and show how the techniques described in the previous section can be used to analyze them. We investigate the trade-off between mechanism design goals (truthfulness) and optimization objectives, such as exact optimization, approximation and competitiveness (in the case of online algorithms). The first two models studied in the following refer to off-line situations where the machines are the agents. The model regarded thereafter is an online scheduling model with job-agents. Another online scheduling model is studied in Chapter 5.

By the *performance guarantee* of an (off-line) allocation algorithm for a minimization problem, we refer to an upper bound on the ratio between the worst possible objective value that can be achieved by the allocation algorithm and the optimal objective value. For a mechanism, the performance guarantee is defined with respect to the worst possible objective value that can occur when all agents report their types truthfully. Note that we do not demand any performance guarantee for non-truthful agents.

In an online optimization problem, the instance is not known entirely beforehand, but part of it is only revealed over time. Therefore, any online algorithm has to make decisions on the basis of incomplete information. In our strategic mechanism design setting, the incompleteness of information is due to two sources – the online setting and the fact that agents have private information. The goal is to design online mechanisms that have good properties with respect to truthfulness and performance. In the online model we describe in this chapter, the objective is equivalent to affine maximization. However, the problem does not allow for an exact allocation algorithm due to the online situation. We say that an online algorithm with minimization objective has performance guarantee ϱ for $\varrho \geq 1$ if the schedule resulting from the online algorithm has an objective value no more than ϱ times the optimal off-line objective value. That is, we compare the online algorithm to the best solution that could have been achieved if the entire instance had been known in advance. For an online mechanism, we demand the performance guarantee only with respect to truthful agents.

Unrelated Machine Scheduling with Machine Agents

In this section, we illustrate the conflict between optimizing the objective function of a given problem and obtaining a truthful mechanism. In the setting that we discuss, it turns out to be impossible to design a mechanism that is at the same time truthful and optimizing the objective function. This section is based on the

work of Nisan and Ronen (2001).

Consider the following strategic version of unrelated machine scheduling. The agents are the m machines, on which n jobs have to be scheduled. The type of each machine i is the vector $t_i = (t_{i1}, \ldots, t_{in})$, where t_{ij} denotes the time that machine i needs to process job j. Hence, the type spaces are n-dimensional. The valuation of a machine for a certain schedule is the negative of the total time it needs to process all the jobs assigned to it. The objective of the optimization problem is to minimize the makespan. Obviously, the allocation algorithm that chooses the optimal schedule for every instance does not belong to the class of affine maximizers.

Nisan and Ronen (2001) show that no truthful mechanism for the regarded problem can approximate the optimal solution with an approximation factor better than 2. We give here an alternative proof for their result using monotonicity.

Theorem 4.22 (Nisan and Ronen 2001)**.** *There does not exist a truthful mechanism for the strategic version of the unrelated machine scheduling problem that has a performance guarantee better than 2.*

Proof. Suppose, there is a truthful mechanism $\mu = (f, \pi)$ with performance guarantee $\varrho < 2$. Let there be n jobs $1, \ldots, n$, where $n > 2\varrho/(2-\varrho)$. Let there be two machines 1 and 2 with types $t_1 = t_2 = (1, \ldots, 1)$. Let n_1 be the number of jobs assigned by f to machine 1 and let n_2 be the number of jobs assigned to machine 2. Without loss of generality, we assume that jobs $1, \ldots, n_1$ are assigned to machine 1 and jobs $n_1 + 1, \ldots, n$ are assigned to machine 2. We also assume $n_1 < n_2$ without loss of generality, i.e., the resulting makespan is n_2. The optimal makespan is $\lceil n/2 \rceil$.

We first claim that $n_1 > 0$. In fact, if $n_1 = 0$, then the makespan in the schedule produced by f is n and

$$n > \frac{\varrho}{2-\varrho} \Leftrightarrow n > \varrho \cdot \frac{n+1}{2} \geq \varrho \left\lceil \frac{n}{2} \right\rceil,$$

which contradicts that μ is a ϱ-approximation.

Let us now change type t_1 to $t_1' = (\varepsilon, \ldots, \varepsilon, 1+\varepsilon, \ldots, 1+\varepsilon)$, such that the processing time of machine 1 is ε for the first n_1 jobs and $1+\varepsilon$ for jobs $n_1+1, \ldots, n$. We assume that ε is positive but very close to 0. Type t_2 remains unchanged. We claim that in the new situation, f has to reassign at least one of the jobs $n_1+1, \ldots, n$ to machine 1.

Case 1. Let n_2 be even. Then the new optimal makespan is at most $(n_2/2)(1+$

$\varepsilon) + n_1\varepsilon$. If all the jobs $n_1 + 1, \ldots, n$ remained on machine 2, the new makespan would be at least n_2. But

$$n_2 > \varrho\left(\frac{n_2}{2}(1+\varepsilon) + n_1\varepsilon\right)$$

for sufficiently small ε, contradicting that μ is a ϱ-approximation.

Case 2. Let n_2 be odd. For sufficiently small ε, the optimal makespan is achieved by assigning $\lceil n_2/2 \rceil$ of the jobs $n_1 + 1 \ldots, n$ to machine 2 and all the other jobs to machine 1. The optimal makespan is then $\lceil n_2/2 \rceil = (n_2 + 1)/2$ for ε small enough. If all the jobs $n_1 + 1, \ldots, n$ remained on machine 2, then the new makespan would be at least n_2. But

$$2n_2 \geq n > \frac{2\varrho}{2-\varrho} \Rightarrow n_2 > \frac{\varrho}{2-\varrho} \Rightarrow n_2 > \varrho \cdot \frac{n_2+1}{2}.$$

That contradicts that μ is a ϱ-approximation.

Therefore, one of the jobs $n_1 + 1, \ldots, n$ must move from machine 1 to machine 2. We now show that monotonicity is violated. Let T be the set of jobs initially assigned to machine 1 for type t_1 and let T' be the set of jobs assigned to machine 1 for type t_1'. Monotonicity for machine 1 implies

$$\sum_{j \in T \setminus T'} (t'_{1j} - t_{1j}) + \sum_{j \in T' \setminus T} (t_{1j} - t'_{1j}) \geq 0.$$

The left hand side is equal to $|T \setminus T'|(\varepsilon - 1) + |T' \setminus T|(1 - (1 + \varepsilon))$, where the first term is at most 0 and the second term is strictly negative, since $|T' \setminus T|$ is at least 1. Therefore, the left hand side is strictly smaller than 0 and monotonicity is violated.

The example can easily be modified to show the result for any larger number of jobs and machines. □

In view of this negative result, the question arises which performance guarantee a truthful mechanism can achieve. Nisan and Ronen (2001) suggest the following MinWork mechanism, which can be viewed as auctioning each task separately in a Vickrey auction.

MinWork mechanism.
Allocation algorithm: After each machine has declared its type, assign

each job to the machine that has declared the lowest processing time for that job. Ties are broken arbitrarily. For a vector $t = (t_1, \ldots, t_m) \in T^m$ of machine declarations, the set of jobs allocated to machine i is denoted by $f_i(t)$.
Payment scheme: For a vector $t = (t_1, \ldots, t_m) \in T^m$ of machine declarations, the payment for machine i is defined as $\pi_i(t) = -\sum_{j \in f_i(t)} \min_{i' \neq i} t_{i'j}$. That is, each machine receives for each job that it processes a payment that equals the second lowest declaration of any machine for that job.

Theorem 4.23 (Nisan and Ronen 2001)**.** *MinWork is truthful and an m-approximation.*

Proof. The MinWork mechanism minimizes the total work done and therefore maximizes the sum of the valuations of all machine-agents. Therefore, the allocation algorithm of the mechanism is an affine maximizer. Set $h_i(t_{-i}) := -\sum_{j=1}^{n} \min_{i' \neq i} t_{i'j}$ to see that the payment scheme of MinWork is a VCG payment scheme. Therefore, MinWork is truthful according to Theorem 4.18.

For the performance guarantee with respect to the makespan objective, note that the optimum makespan V_{OPT} is lower bounded by

$$V_{OPT} \geq \frac{1}{m} \sum_{j=1}^{n} \min_{i=1,\ldots,m} t_{ij}.$$

The makespan V_{MW} resulting from the allocation algorithm of the MinWork mechanism is upper bounded by

$$V_{MW} \leq \sum_{j=1}^{n} \min_{i=1,\ldots,m} t_{ij},$$

i.e., $V_{MW} \leq m V_{OPT}$, assuming that all agents report their true types. □

In fact, Nisan and Ronen prove that the mechanism is strongly truthful, i.e., truthtelling is the only dominant strategy for every agent.

As mentioned before, no truthful mechanism for the regarded problem can approximate the optimal solution better than a factor of 2. Therefore, MinWork is best possible for two machines. Moreover, the authors conjecture that also for the general case with m machines, the upper bound of m is tight, yet this remains an open question.

The scheduling problem regarded in this section and variants thereof have become important and well studied problems exemplifying the conflict between approximation goals and mechanism design goals. Since the seminal work of Nisan and Ronen (2001), improvements on the lower bound for the approximation ratio of the strategic unrelated machine scheduling problem have been obtained. To the best of our knowledge, the currently best known lower bound is $1 + \varphi \approx 2.618$ proven by Koutsoupias and Vidali (2007). That means that no truthful mechanism can achieve an approximation ratio better than $1 + \varphi$ for every number m of machines.

Related Machine Scheduling with Machine Agents

Archer and Tardos (2001) consider a similar model for related machine scheduling. Again, the agents are the machines. In contrast to the model from the previous section, the processing times of different jobs on one machine are not independent and the type spaces of the machine-agents are one-dimensional. More precisely, each machine i has a speed s_i and the type is defined to be the inverse of this speed $t_i := 1/s_i$. Each job j has a (unit-speed) processing time p_j. The time that is needed to process job j on machine i is $t_i p_j = p_j/s_i$. The action of each machine is to declare its type. If machine i is assigned the set of jobs $f_i(t) \subseteq J$ for a vector of declarations of all machines $t = (t_1, \ldots, t_m) \in T^m$, then its valuation is $v_i(t, t_i) = -\sum_{j \in f_i(t)} t_i p_j$. The objective is again to minimize the makespan.

For this setting, Archer and Tardos derive a necessary and sufficient condition for an allocation algorithm to be truthfully implementable. For an allocation algorithm, denote by $L_i(t) = L_i(t_i, t_{-i})$ the total workload assigned to machine i. Then an agent's valuation can be written as $v_i(t, t_i) = -t_i \cdot L_i(t_i, t_{-i})$.

Theorem 4.24 (Archer and Tardos 2001)**.** *An allocation algorithm is truthfully implementable if and only if for all agents i and all $t_{-i} \in T^{m-1}$ the function $L_i(t_i, t_{-i})$ is a decreasing function of t_i. If this is the case, then the appropriate payments take the following form*

$$\pi_i(t_i, t_{-i}) = -\left(h_i(t_{-i}) + t_i L_i(t_i, t_{-i}) - \int_0^{t_i} L_i(u, t_{-i}) du \right).$$

Here, h_i are arbitrary functions that depend on the declarations of all agents except i.

Instead of giving the original proof by Archer and Tardos (2001), we show how Theorem 4.24 is implied by the results of Saks and Yu (2005) and Gui, Müller, and Vohra (2004).

Proof. Convexity of the type spaces in the sense of Theorem 4.21 can be easily verified. It is due to the fact that the speed and therefore its inverse can be an arbitrary positive real number and the valuation of an agent for a certain schedule depends linearly on the inverse of the speed.

From Theorem 4.21, we know that monotonicity is a necessary and sufficient condition for the allocation algorithm to be truthfully implementable. To verify monotonicity, let t_i and $\hat{t}_i$ be different types of an agent i with $t_i < \hat{t}_i$, and let the reports of the other agents be fixed as t_{-i}. Then monotonicity is equivalent to

$$\begin{aligned} & v_i((t_i, t_{-i}), t_i) - v_i((t_i, t_{-i}), \hat{t}_i) \geq v_i((\hat{t}_i, t_{-i}), t_i) - v_i((\hat{t}_i, t_{-i}), \hat{t}_i) \\ \Leftrightarrow \quad & -t_i L_i(t_i, t_{-i}) + \hat{t}_i L_i(t_i, t_{-i}) \geq -t_i L_i(\hat{t}_i, t_{-i}) + \hat{t}_i L_i(\hat{t}_i, t_{-i}) \\ \Leftrightarrow \quad & (\hat{t}_i - t_i)(L_i(t_i, t_{-i}) - L_i(\hat{t}_i, t_{-i})) \geq 0. \end{aligned}$$

This condition is satisfied whenever $L_i(t_i, t_{-i}) - L_i(\hat{t}_i, t_{-i}) \geq 0$, i.e., if and only if the function L_i is decreasing in the report of agent i.

For the second part of the theorem, we will use the results of Gui, Müller, and Vohra (2004) to derive the payment scheme given above. Let f be an allocation algorithm that satisfies the decreasing work curves condition, let agent i and the report of the other agents t_{-i} be fixed and let for simplicity of notation $L(t_i) := L_i(t_i, t_{-i})$ denote the workload assigned by f to i when reporting t_i. First, we observe that there are only finitely many possible schedules that can yield only finitely many different values of $L(t_i)$. We denote those values by $L_{max} = \sum_{j \in J} p_j > \cdots > L_1 > L_0 = 0$. Using that f satisfies the decreasing work curve condition, we get the picture in Figure 4.1 for the graph of $L(t_i)$. In order to determine the payments

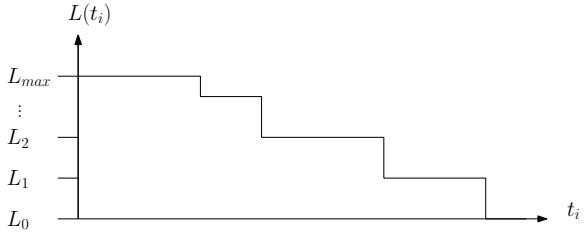

Figure 4.1: decreasing work curve

according to the method of Gui, Müller, and Vohra (2004), we have to determine

shortest paths in the allocation graph as described in Chapter 2. Let π_r denote the payment that machine i has to make if it is assigned workload L_r by the allocation algorithm. We define $\pi_0 = 0$ and determine the shortest path from the node L_0 to node L_r. It can easily be shown that if $L_{r_3} > L_{r_2} > L_{r_1}$ then the arc lengths satisfy $\ell(L_{r_1}, L_{r_3}) \geq \ell(L_{r_1}, L_{r_2}) + \ell(L_{r_2}, L_{r_3})$. Therefore, $[L_0, L_1, \ldots, L_r]$ is a shortest path from L_0 to L_r. Hence, the payments can be written as:

$$\begin{aligned} \pi_r &= \sum_{i=1}^{r} \ell(L_{i-1}, L_i) = \sum_{i=1}^{r} \inf_{t_i : L(t_i) = L_i} (-t_i L_i + t_i L_{i-1}) \\ &= -\sum_{i=1}^{r} \left(\sup_{t_i : L(t_i) = L_i} t_i \right) (L_i - L_{i-1}). \end{aligned}$$

Thus, if machine i is assigned a total workload of L_r, it receives a payment that is equal to the area under the graph of $\min(L(t_i), L_r)$, as depicted in Figure 4.2. Thus,

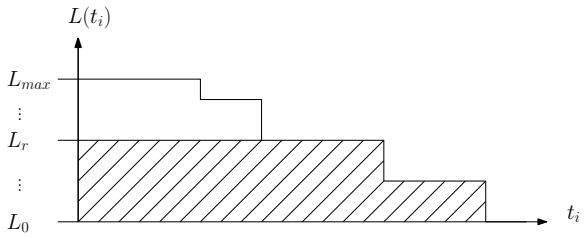

Figure 4.2: payment for workload L_r

the payment to machine i is $t_i L(t_i) + \int_{t_i}^{\infty} L(u)du$. If we now let $h_i(t_{-i}) = \int_0^{\infty} L(u)du$, then the payment that machine i receives equals $t_i L(t_i) + \int_{t_i}^{\infty} L(u)du = h_i(t_{-i}) + t_i L(t_i) - \int_0^{t_i} L(u)du$, which proves the claim.

To see that every payment scheme in a truthful mechanism takes the form obtained above, notice that revenue equivalence holds according to Corollary 2.13. Therefore, the payments are uniquely defined up to $h_i(t_{-i})$, which completes the proof. □

Archer and Tardos (2001) give a randomized polynomial time allocation algorithm, which is based on bin-packing and rounding fractional assignments of jobs to bins in a random fashion. The allocation algorithm fulfills the decreasing work curves condition with respect to the expected utilities of the agents. The following result is obtained.

Theorem 4.25 (Archer and Tardos 2001)**.** *For the strategic version of related machine scheduling with machine agents whose private information is their speed, there exists a mechanism with the following properties:*

- *the mechanism is truthful when agents maximize expected utilities,*
- *it yields a 3-approximation for the makespan of the schedule independent of the random choices,*
- *the payments can be computed in polynomial time.*

There also exists a deterministic allocation algorithm that fulfills the monotonicity condition of Theorem 4.24 due to Kovacs (2005). The allocation algorithm given by Kovacs (2005) runs in polynomial time and yields a 3-approximation as well. Thus, the existence of a truthful payment scheme is guaranteed by Theorem 4.24. However, it is not clear whether the associated payments can be computed in polynomial time. Other papers that have improved some of the results of Archer and Tardos (2001) are by Auletta, De Prisco, Penna, and Persiano (2004), Ambrosio and Auletta (2005), and Andelman, Azar, and Sorani (2005).

Online Scheduling with Job Agents

We now turn to an online scheduling model with job agents. The following single machine model was analyzed by Porter (2004). There is one machine that has to process n jobs, where n is not known beforehand. Preemption of jobs is allowed. Each job j has a release date r_j, a processing time p_j, a deadline d_j and a weight w_j. Those four values are private information. The type of agent j is thus $t_j = (r_j, p_j, d_j, w_j)$. The aim is to design a direct revelation mechanism, that is, jobs have to report their types to the mechanism. We assume that a job can declare a release date $\hat{r}_j \geq r_j$ and a processing time $\hat{p}_j \geq p_j$, while it can declare an arbitrary deadline $\hat{d}_j$ and an arbitrary weight $\hat{w}_j$. The reason that we do not admit declaring a shorter processing time is that this could be easily detected and punished by the mechanism. If job j is completed before its deadline, then its valuation is w_j, otherwise its valuation is zero. Jobs have to pay for being processed. The payments have to be determined online as well, i.e., the payment for a job must be determined at the latest when the job leaves the system. The central objective is to maximize the sum of the weights of all jobs that are completed by their deadline, i.e., the goal is affine maximization. However, an exact allocation algorithm does not exist due to the online nature of the problem. Lower bounds on the performance guarantee of any online algorithm for the problem are given by Baruah, Koren, Mao, Mishra,

Raghunathan, Rosier, Shasha, and Wang (1992). Those bounds imply in particular that there is no exact allocation algorithm. Porter (2004) shows the following.

Theorem 4.26 (Porter 2004)**.** *For the described single machine model, there exists a truthful mechanism with performance guarantee* $(1+\sqrt{k})^2+1$*, where* k *is an upper bound on* $\max_{j,l}(w_l p_j)/(p_l w_j)$ *which is known to the mechanism.*

Notice that the mechanism is assumed to know an upper bound k on the maximum ratio $\max_{j,l}(w_l p_j)/(p_l w_j)$. In addition, the mechanism needs to know a lower bound δ_{min} on the possible ratios w_j/p_j for all jobs j. The single machine processes at any point in time a job that is chosen among the available jobs that still have a chance to be completed before their declared deadline. The choice depends on the declared weights of the jobs, the time already spent processing each job, on k and on δ_{min}. Jobs are only returned at their declared deadlines. The payment that a job has to make is zero if it is not completed before its declared deadline. Otherwise it is equal to the minimum weight that the job could have declared such that it still would have been finished in time, given the declarations of the other jobs and given its own declarations on release date, deadline and processing time.

The proof of the truthfulness of the mechanism given by Porter (2004) is quite technical. Intuitively, the payments can be seen as VCG-payments and are chosen such that the mechanism is truthful with respect to the weights. By returning the job not until the declared deadline it is achieved that a job has no incentive to declare a larger deadline than the true one. Therefore, no job has an incentive to declare a "better" type. It can be shown that also declaring a "worse" type does not pay off.

Interestingly, Porters mechanism is essentially best possible.

Theorem 4.27 (Porter 2004)**.** *Under a number of (weak) conditions and assuming that* $k > 1$*, no deterministic truthful online mechanism can have a performance guarantee better than* $(1+\sqrt{k})^2+1$ *for the described single machine model.*

This is of special interest in view of the existence of an algorithm with performance guarantee $(1+\sqrt{k})^2$ for the non-strategic online setting, due to Koren and Shasha (1995).

In Chapter 5, we analyze another online machine scheduling model on parallel machines. There, unlike in Porter's model, jobs have to be processed without preemption and the central objective is to minimize the total weighted completion time. Furthermore, we introduce the concept of decentralization, which reflects the lack of coordination in distributed settings. Thereby, a link between techniques from this

Section and Section 4.3.2 is established: the mechanism has to account for both, the strategic behavior of agents with respect to reporting their private information as well as their strategic behavior with respect to the selection of a machine.

Another reference, which we do not discuss here is Hajiaghayi, Kleinberg, Mahdian, and Parkes (2005). They derive mechanisms for an online single machine model with job agents, where jobs are available only within certain time windows.

4.5 Discussion

In this chapter, we have given an introduction to the application of game theory and mechanism design to (mostly) multiple machine scheduling models. Thereby we have chosen to limit the scope of models and techniques in order to be able to present the most important techniques in detail. We see several avenues of research departing from here.

Within the narrow scope of this chapter, the most promising research questions seem to us related to mechanism design in the presence of multi-dimensional types, exemplified by the conjecture by Nisan and Ronen mentioned above. Roberts (1979) has shown that in cases where the type space is completely unrestricted, meaning that every agent can have any valuation for each of the outcomes, only affine maximizers are truthfully implementable. For more restrictive type spaces, Bikhchandani, Chatterjee, Lavi, Mu'alem, Nisan, and Sen (2006) could show a similar characterization only for allocation algorithms that satisfy some additional properties. If the conjecture by Nisan and Ronen is true, it would show that in this particular case we cannot do better than using an affine maximizer, if we want to guarantee truthfulness. This would indicate that also in this case affine maximizers are the only truthfully implementable algorithms.

Moreover, we want to emphasize that the area of game theory and mechanism design has established theoretical models that cover many more issues than those treated in this chapter. For example, game theory knows a plentitude of refinements of equilibrium notions, which might be of practical relevance, and mechanism design is also interested in other criteria than only efficiency and truthfulness. For example, various definitions of fairness can be found in the literature. As a point of reference we mention here the literature on matching markets, e.g., Roth, Sotomayor, and Chesher (1990).

Finally, it is likely that also in scheduling, we can benefit from the wealth of results that have recently been derived in the context of combinatorial auctions. We recommend the book edited by Cramton, Shoham, and Steinberg (2006) for reference. While from an optimization point of view, combinatorial auctions deal "only"

with a set packing problem, many of the game theoretic issues in their design are also relevant when the allocation rule is of a different nature, like assigning jobs to machines, and determining their start time on the machines. In combinatorial auctions, those issues are, for example, revenue for the seller, collusion, bidding under multiple identities (false-name bidding), information revelation, and communication complexity. It turns out that a mechanism like the VCG mechanism, when applied to combinatorial auctions, performs badly in terms of such additional criteria, unless rather restrictive assumptions on the bidders valuations are fulfilled (Ausubel and Milgrom 2006). Also in scheduling, such considerations may yield interesting insights and may be of practical importance.

Chapter 5

Mechanism Design for Decentralized Online Machine Scheduling

In Chapter 4, we have highlighted some techniques and phenomena from the recent literature about machine scheduling models that have been analyzed from a game theoretic or mechanism design point of view. This chapter is devoted to the analysis of mechanism design for a parallel machine scheduling model where jobs arrive online over time. As in the settings described in Chapter 4, there is no central decision maker and problem data is distributed among job agents who take autonomous decisions. Instead of centrally assigning jobs to machines, each machine implements a local sequencing policy and jobs decide for machines themselves. In this context, we introduce the concept of a myopic best response equilibrium, a concept weaker than the classical dominant strategy equilibrium, but appropriate for online problems. Our main result is a polynomial time, online mechanism that – assuming rational behavior of jobs – results in an equilibrium schedule that is 3.281-competitive with respect to the maximal social welfare. This is only slightly worse than state-of-the-art algorithms with central coordination.[1]

[1]The results of this chapter were first presented in Heydenreich, Müller, and Uetz (2009).

5.1 Introduction

Scheduling arriving jobs online on a set of parallel machines is a key issue both in business and engineering applications. Examples can be found in service operations management and distributed computing. The problem has been well studied in the traditional setting, where a central decision maker strives to optimize a global system performance measure and is assumed to have access to all relevant data. However, in the environments mentioned above, data is usually not centrally available, but is distributed among selfish job owners or agents. As already observed for the models surveyed in Chapter 4, this gives incentives for strategic behavior of agents, possibly leading to sub-optimal system performance. This challenge calls for mechanism design to align the individual goals of selfish agents with overall system performance. On the other hand, in dynamic environments like distributed computing, machines are locally dispersed and administratively independent and may be dynamically added to or removed from the system. A typical example are web servers, where content and/or computational resources are nowadays distributed over the whole world and service requests need to be allocated online. In such settings, it is indispensable to keep communication complexity low and to design local protocols that machines have to adopt rather than centrally coordinating the distribution of jobs over machines. This has been observed, for example, in the context of analyzing the price of anarchy e.g. by Christodoulou, Koutsoupias, and Nanavati (2004), Immorlica, Li, Mirrokni, and Schulz (2008) and Azar, Jain, and Mirrokni (2008). A description of these models has been given in Chapter 4, as well. In this chapter, we define decentralized online mechanisms that account for all mentioned requirements.

More specifically, we study the online version of the classical parallel machine scheduling problem to minimize the total weighted completion time – $\mathrm{P} \mid r_j \mid \sum w_j C_j$ in the notation of Graham, Lawler, Lenstra, and Rinnooy Kan (1979) – from a game theoretic, or *strategic* perspective. In the online version, jobs j with processing times p_j and weights w_j arrive online over time at release times r_j, and at any given time the scheduler does not know if, or what type of jobs are still to appear in the future. The classical goal in online optimization is to design online algorithms that are *competitive*, that is, even though faced with an online situation, such algorithms compare reasonably well to the optimal offline solution. An online algorithm is called *ϱ-competitive* if it always achieves a solution that is not more than a factor ϱ away from the optimum offline solution. We assume that each job is a selfish agent, and a job's release time r_j, its processing time p_j and its weight w_j is only known to the job itself, but not to the system or any other job. Any job j is interested in being finished as early as possible, and the weight w_j represents j's cost per unit

waiting time. While jobs may strategically report false values $(\tilde{r}_j, \tilde{p}_j, \tilde{w}_j)$ in order to be scheduled earlier, the total *social welfare* is maximized whenever the weighted sum of completion times $\sum w_j C_j$ is minimized.

Next to the game theoretic challenge due to selfishly behaving jobs, distributed systems ask for low communication complexity and local protocols that machines have to commit to rather than centralized coordination. Our goal is to meet the following requirements, which we refer to as *decentralization*: Jobs may communicate with machines, but neither do jobs communicate with each other, nor do machines communicate with each other. In particular, there is no central scheduling unit hosting all the data of the problem. This leads to a setting where the jobs themselves must select the machine to be processed on, and any machine sequences the jobs according to a (known) local sequencing policy. Such models have already been discussed in Chapter 4, Section 4.3.2.

Our goal is to set up an online mechanism that copes with the strategic and decentralized setting while yielding a reasonable overall performance with respect to the total social welfare, that is, minimize $\sum w_j C_j$. The mechanism should motivate the jobs to reveal their private information truthfully. In addition, as we require decentralization, each machine needs to be equipped with a local sequencing policy, and jobs must be induced to select the machines in such a way that the objective $\sum w_j C_j$ does not deteriorate. The online algorithm with the currently best known competitive ratio by Correa and Wagner (2005) crucially requires central coordination to distribute jobs over machines. Instead, we build upon an approach by Megow, Uetz, and Vredeveld (2006), developed for a setting with stochastic job durations, which turns out to be appropriate for the decentralized setting that we aim at.

Related Work. As computational complexity is concerned, the scheduling problem $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$ is well-understood in the non-strategic setting with centralized coordination. First, scheduling to minimize the weighted sum of completion times with release dates is NP-hard, even in the off-line case on a single machine. For more than one machine, the problem is NP-hard even if all release dates are zero (Lenstra, Rinnooy Kan, and Brucker 1977). In the online setting, it is well known that no online algorithm for the single machine problem can be better than 2-competitive (Hoogeveen and Vestjens 1996) regardless of the question whether or not P=NP. On parallel machines, no online algorithm can be better than 1.309-competitive, and this bound can be improved for a specific number of machines (Vestjens 1997). The best possible algorithm for the single machine case is 2-competitive and thus matches the lower bound (Anderson and Potts 2004). For the parallel machine setting, the currently best known online algorithm is 2.62-

competitive (Correa and Wagner 2005), improving upon an earlier algorithm by Megow and Schulz (2004). The algorithm by Megow et al. (2006) is a modification of the latter. Here, jobs are locally sequenced according to an online variant of the well known WSPT rule (Smith 1956), and arriving jobs are assigned to machines in order to minimize an expression that approximates the (expected) increase of the objective value. The algorithms by Megow and Schulz (2004) and Megow et al. (2006) both achieve a competitive ratio of 3.281.

For related literature on mechanism design in combination with the design of approximation algorithms for scheduling problems we refer to Chapter 4, Section 4.4.2.

Decentralization for scheduling models with job agents is regarded in the papers discussed in Chapter 4, Section 4.3.2.

Our Contribution. We present a polynomial time, decentralized online mechanism, called Decentralized Local Greedy Mechanism. Thereby we provide also a new algorithm for the non-strategic, centralized setting, inspired by the Min-Increase Algorithm of Megow et al. (2006), but improving upon the latter in terms of simplicity. The Decentralized Local Greedy Mechanism is easy to implement and we show that it is 3.281-competitive. This coincides with the performance bound achieved by Megow and Schulz (2004) for the non-strategic, centralized setting. The currently best known bound for this setting, however, is 2.62 (Correa and Wagner 2005). Giving up on decentralization, it is possible to design a 2.62-competitive mechanism on the basis of the Correa-Wagner algorithm with a dominant strategy equilibrium in which all agents report truthfully. We discuss the resulting mechanism in Section 5.6.2.

As usual in mechanism design, the Decentralized Local Greedy Mechanism defines *payments* that have to be made by the jobs for being processed. Naturally, we require from an *online* mechanism that also the payments are computed online. Hence, they can be completely settled by the time at which a job leaves the system. We also show that the payments result in a balanced budget. The payments induce rational jobs to truthfully report about their private data. With respect to release dates and processing times, we can show that truthfulness is a dominant strategy equilibrium. With respect to the weights, however, we can only show that truthful reports are myopic best responses (in a sense to be made precise later). Most importantly, the payments induce the jobs to select 'the right' machines, that is, the machines which a centralized mechanism would select in order to achieve a good competitive ratio. Intuitively, the mechanism uses the payments to mimic a corresponding Local Greedy online algorithm in the classical (non-strategic, centralized) parallel machine setting $\mathrm{P} \mid r_j \mid \sum w_j C_j$. In addition, we show that there

does not exist a payment scheme leading to the same selection of machines where truthful reporting of all private information is a dominant strategy equilibrium. This is even true, when only the weight w_j is considered private information and p_j and r_j are publicly known. Hence, for the decentralized online setting that we consider, it is not clear if a constant competitive ratio can be achieved by means of a dominant strategy equilibrium of some mechanism – even if weights are the only private information.

Organization. The chapter is structured as follow. We formalize the model and introduce notation in Section 5.2. Especially, we define the notion of a decentralized online scheduling mechanism and the myopic best response equilibrium in that section. In Section 5.3 the Local Greedy Algorithm is defined. In Section 5.4, this algorithm is adapted to the strategic setting and extended by a payment scheme yielding the Decentralized Local Greedy Mechanism. Moreover, our main results are presented in that section. We analyze the performance of the resulting mechanism in Section 5.5. In Section 5.6, we prove the mentioned negative result and reflect on mechanisms that have dominant strategy equilibria, giving up on decentralization. We conclude with a short discussion in Section 5.7.

5.2 Model and Notation

The considered problem is online parallel machine scheduling with non-trivial release dates, with the objective to minimize the weighted sum of completion times, $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$. We are given a set of jobs $J = \{1, \ldots, n\}$, where each job needs to be processed on any of the parallel, identical machines from the set $M = \{1, \ldots, m\}$. The processing of each job must not be preempted, and each machine can process at most one job at a time. Each job j is viewed as a selfish agent and has the following private information: a release date $r_j \geq 0$, a processing time $p_j > 0$, and an indifference cost, or weight, denoted by $w_j \geq 0$. The release date denotes the time when the job comes into existence, whereas the weight represents the cost to a job for one additional unit of time spent waiting. Without loss of generality, we assume that the jobs are numbered in order of their release dates, i.e., $j < k \Rightarrow r_j \leq r_k$. The triple (r_j, p_j, w_j) is also denoted as the *type* of a job, and we use the shortcut notation $t_j = (r_j, p_j, w_j)$. By $T = \mathbb{R}_0^+ \times \mathbb{R}^+ \times \mathbb{R}_0^+$ we denote the space of possible types of each job. In the model we analyze, a job j can report an arbitrary weight $\tilde{w}_j \neq w_j$, an elongated processing time $\tilde{p}_j \geq p_j$ (e.g. by adding unnecessary work), and it can artificially delay its true release time r_j to $\tilde{r}_j \geq r_j$. We do not allow a job to report a processing time shorter than the true p_j, as this can easily be discovered and punished by the system, for example by preempting the job after the declared

processing time $\tilde{p}_j$ before it is actually finished. Furthermore, we assume that any job j comes into existence only at its release time r_j, thus it does not make sense that a job reports a release time smaller than the true value r_j.

We next introduce the concept of a decentralized online scheduling mechanism. It accounts for the various challenges mentioned in the introduction: It takes into account that necessary information is not centrally available, but has to be communicated from jobs to machines, while keeping the resulting communication complexity down to a minimum. It does not use central coordination, but rather defines a protocol according to which machines process jobs and compute payments that jobs have to make. Our goal will be to find such a mechanism where rational job behavior results in an equilibrium in which the social welfare is not too far from optimum.

Definition 5.1. *A* decentralized online scheduling mechanism *is a procedure that works as follows.*

1. *Each job j has a release time r_j, but may pretend to come into existence at any time $\tilde{r}_j \geq r_j$. At $\tilde{r}_j$, the job communicates to every machine reports $\tilde{w}_j$ and $\tilde{p}_j$.*[2]

2. *Machines communicate on the basis of that information a tentative completion time $\hat{C}_j$ and a tentative payment $\hat{\pi}_j$.*

3. *Based on the responses of all machines at time $\tilde{r}_j$, the job chooses a machine to be processed on.*

4. *There is no communication between machines or between jobs.*

5. *Machines may revise $\hat{C}_j$ and $\hat{\pi}_j$ only if later another job chooses the same machine, leading to an ex-post completion time C_j and an ex-post payment π_j.*

Hereby, we assume that jobs with equal reported release times arrive in some given order and communicate to machines in that order. Next, we define two important properties of the payment scheme.

Definition 5.2. *If for every job j payments to and from j are only made between time $\tilde{r}_j$ and time C_j, then we call the payment scheme an* online payment scheme.

[2] A job could even report different values to different machines. However, we prove existence of equilibria where the jobs do not make use of that option.

Definition 5.3. *A payment scheme satisfies the* balanced budget *condition if the payments made by all jobs sum up to zero, i.e.,* $\sum_{j \in J} \pi_j = 0$.

One of our goals is to design competitive online mechanisms, which are defined as follows.

Definition 5.4. *Let A be an online mechanism that seeks to minimize a certain objective function. Let $V_A(I)$ be the objective value computed by A for a problem instance I and let $V_{OPT}(I)$ be the offline optimal objective value for I. Then A is called* ϱ-competitive *if for all instances I of the problem,*

$$V_A(I) \leq \varrho \cdot V_{OPT}(I) \,.$$

The factor ϱ is also called performance ratio *of the mechanism.*

We assume that the *valuation* equals $v_j(C_j, t_j) = -w_j C_j$, such that smaller completion times are preferred. We furthermore assume *quasi-linear utilities*, that is, the *utility* of job j equals $u_j(C_j, \pi_j, t_j) = v_j(C_j, t_j) - \pi_j$, which is equal to $-w_j C_j - \pi_j$. Unlike in other mechanism design settings, where jobs always have the option not to participate in the mechanism and to obtain zero utility, we assume that the jobs have no such option and they have to be processed on one of the machines.

The communication with machines, and the decision for a particular machine are called *actions* of the jobs; they constitute the strategic actions jobs can take in the non-cooperative game induced by the mechanism. A *strategy* x_j of a job j maps a type t_j to an action for every possible state of the system in which the job is required to take some action. A *strategy profile* is a vector $(x_1, \ldots, x_n)$ of strategies, one for each job. Given a mechanism, a strategy profile, and a realization of types t, we denote by $u_j(x, t)$ the (ex-post) utility that agent j receives. Recall the definition of a dominant strategy equilibrium.

Definition 5.5. *A strategy profile $x = (x_1, \ldots, x_n)$ is called a* dominant strategy equilibrium *if for all jobs $j \in J$, all types t of the jobs, all strategies $\tilde{x}_{-j}$ of the other jobs, and all strategies $\tilde{x}_j$ that j could play instead of x_j,*

$$u_j((x_j, \tilde{x}_{-j}), t) \geq u_j((\tilde{x}_j, \tilde{x}_{-j}), t) \,.$$

The dominant strategy equilibrium is a very sound, yet strong concept, and in many cases dominant strategy equilibria do not exist; see for example the discussion by Nisan (2007). Several alternatives have been studied in the literature

that weaken the definition of rational agent behavior, e.g. ex-post Nash equilibria, Bayes-Nash equilibria or myopic best responses. The latter has for instance been used in auction theory in the context of combinatorial auctions, see e.g. Parkes (1999) and Parkes and Ungar (2000). There, the VCG mechanism (where truthful revelation of private information is a dominant strategy equilibrium) suffers from severe computational difficulties. Instead, an iterative auction with several rounds is proposed that results in a welfare maximizing allocation of goods if bidders are myopic. Myopic bidders aim to maximize their utility with respect to current price and allocation information, rather than taking game theoretic look-ahead to future rounds. Similarly myopic bidders are assumed by Demange, Gale, and Sotomayor (1986) for multi-item auctions, Bein and Wein (2003) and Gallien and Wein (2005) for procurement auctions and by Wellman, Walsh, Wurman, and MacKie-Mason (2001) for the allocation of time slots. We find this concept appropriate and natural also for our setting. We assume that rational agents maximize their *tentative utility*, that is, the utility that a job is communicated upon arrival at the system. Note that the concept shares with the dominant strategy equilibrium the property that it does not require any reasoning about other agents' valuations. In that sense it is prior-free, which is a desirable property.

Definition 5.6. *Given a decentralized, online scheduling mechanism as in Definition 5.1, a strategy profile x, and type profile t. Let $\hat{C}_j$ and $\hat{\pi}_j$ denote the tentative completion time and the tentative payment of job j at time $\tilde{r}_j$. Then $\hat{u}_j(x,t) := -\hat{C}w_j - \hat{\pi}_j$ denotes j's tentative utility at time $\tilde{r}_j$.*

If x and t are clear from the context, we will use $\hat{u}_j$ as short notation.

Definition 5.7. *A strategy profile $(x_1, \ldots, x_n)$ is called a* myopic best response equilibrium*, if for all jobs $j \in J$, all types t of the jobs, all strategies $\tilde{x}_{-j}$ of the other jobs and all strategies $\tilde{x}_j$ that j could play instead of x_j,*

$$\hat{u}_j((x_j, \tilde{x}_{-j}), t) \geq \hat{u}_j((\tilde{x}_j, \tilde{x}_{-j}), t).$$

Notice that the only difference in the definitions of the two equilibria is the utility that agents are concerned with: In the dominant strategy equilibrium, it is the *ex-post* utility that drives an agent, while in the weaker myopic best response equilibrium, it is the *immediate* utility that is observable at the moment in time where the agent chooses an action.

Proposition 5.8. *For any decentralized online scheduling mechanism with online*

payment scheme, every dominant strategy equilibrium is a myopic best response equilibrium.

Proof. In a dominant strategy equilibrium, job j's strategy maximizes j's ex-post utility for all possible strategies of the other jobs. In a decentralized online scheduling mechanism with online payment scheme, there is always a strategy of the other jobs x_{-j} such that j's tentative utility equals j's ex-post utility (e.g., jobs arriving later than j can choose to delay their arrival behind j's completion time). Then none of these jobs can change j's completion time, and if the payment scheme is online, neither can they influence j's payment[3]. Consequently, j's tentative utility must be maximized in any dominant strategy equilibrium, too. □

Hence, the class of myopic best response equilibria is a larger class of equilibria than dominant strategy equilibria, and we will see later that it is indeed a strictly larger class.

Finally, notice that jobs will also have to select a machine according to Definition 5.1. This additional action of jobs has been introduced to distinguish between decentralized and centralized scheduling mechanisms. One might argue that one can nevertheless make use of the *revelation principle*, which asserts that an arbitrary mechanism that has an equilibrium, for example a dominant strategy equilibrium, always induces a direct revelation mechanism with an equilibrium of the same strength. Thus questions with respect to the existence of mechanisms with a particular equilibrium can be answered by restricting to direct revelation mechanisms. However, not all direct revelation mechanisms can be decentralized in the sense of Definition 5.1. For example, we cannot decentralize the algorithm in Correa and Wagner (2005), because it crucially requires a central queue for the jobs. Hence, given that we aim at decentralized mechanisms, we cannot make use of the revelation principle. Equilibria of decentralized online scheduling mechanisms, however, give rise to a particular form of revelation mechanisms, namely mechanisms in which jobs report their types to so-called *proxy agents*, each of them representing exactly one job, and behaving on behalf of the jobs as prescribed by the equilibrium strategy. But introducing proxy agents requires a trustworthy mediator, which can be seen as a hidden form of centralization.

[3]If $m > 1$ it is not necessary to require the payment scheme to be online. The tentative utility equals the ex-post utility, e.g., if later jobs choose a different machine than j.

5.2.1 Critical jobs

For convenience of presentation, we make the following assumption for the main part of the chapter. Fix some constant $0 < \alpha \leq 1$ (α will be discussed later). Let us call job j *critical* if $r_j < \alpha p_j$. Intuitively, a job is critical if it is long and appears comparably early in the system. The assumption we make is that such critical jobs do not exist, that is

$$r_j \geq \alpha\, p_j \quad \text{for all jobs } j \in J\,.$$

This assumption is a tribute to the desired performance guarantee, and in fact, it is well known that critical jobs must not be scheduled early to achieve constant performance ratios (see Anderson and Potts 2004 and Megow and Schulz 2004). However, the assumption is only made due to cosmetic reasons. In the following, we first define an algorithm and a mechanism on the refined type space, where all jobs are non-critical. In Section 5.5.1, we extend the type space and slightly adapt the mechanism. The adapted mechanism can handle critical jobs without sacrificing the performance bound, while all desired properties concerning the strategic behavior of the jobs are preserved.

5.3 The Local Greedy Algorithm

For the time being, we assume that the job characteristics, namely release date r_j, processing time p_j and indifference cost w_j, are given.

The idea of the MinIncrease algorithm of Megow et al. (2006) is that each machine uses (the online version of) the well known WSPT rule (Smith 1956) locally; an idea that we adopt also here. More precisely, each machine implements a priority queue containing the not yet scheduled jobs that have been assigned to the machine. The queue is organized according to WSPT, that is, jobs with higher ratio w_j/p_j have higher priority. In case of ties, jobs with lower index have higher priority. As soon as the machine falls idle, the currently first job from this priority queue is scheduled (if any). Given this local scheduling policy on each of the machines, any arriving job is assigned to that machine where the increase in the objective $\sum w_j\, C_j$ is minimal.

In the formulation of the algorithm, we utilize some shortcut notation. We let $j \to i$ denote the fact that job j is assigned to machine i. Let S_j be the time when job j eventually starts being processed. For any job j, $H(j)$ denotes the set of jobs that have higher priority than j, $H(j) = \{k \in J \mid w_k p_j > w_j p_k\} \cup \{k \leq j \mid w_k p_j = w_j p_k\}$. Note that $H(j)$ includes j, too. Similarly, $L(j) = J \setminus H(j)$ denotes the set of jobs with lower priority. At a given point τ in time, machine i might be busy processing

a job. We let $b_i(\tau)$ denote the remaining processing time of that job at time τ, i.e., at time τ machine i will be blocked during $b_i(\tau)$ units of time for new jobs. If machine i is idle at time τ, we let $b_i(\tau) = 0$. With these definitions, if job j arrives at time r_j and is assigned to machine i, the increase of the objective $\sum w_j C_j$ is

$$z(j,i) := w_j \Big[r_j + b_i(r_j) + \sum_{\substack{k \in H(j) \\ k \to i \\ k < j \\ S_k \geq r_j}} p_k + p_j \Big] + p_j \sum_{\substack{k \in L(j) \\ k \to i \\ k < j \\ S_k > r_j}} w_k \,.$$

Algorithm 1: Local Greedy algorithm

Local Sequencing Policy:
When a machine falls idle, it processes the job with highest (WSPT) priority among all jobs assigned to it.

Assignment:
If job j arrives at time r_j, it is assigned to machine $i_j \in \operatorname{argmin}_{i \in M} z(j,i)$ with minimum index.

Clearly, the Local Greedy algorithm still makes use of central coordination. On the other hand, the WSPT rule can be run locally on every machine and does not require communication between the machines. Therefore, the Local Greedy algorithm qualifies for decentralization, which will be done in the next Section.

5.4 Payments for Myopic Rational Jobs

The payments we introduce can be motivated as follows: A job j pays at the moment of its placement on one of the machines an amount that compensates the decrease in utility of the other jobs. The final payment of each job j resulting from this mechanism will then consist of the immediate payment j has to make when selecting a machine and of the payments j gets when being displaced by other jobs. Furthermore, the WSPT rule is run locally on every machine and jobs select a machine themselves. We will prove that utility maximizing jobs have an incentive to report truthfully and to choose the machine that the Local Greedy Algorithm would have selected, too. We will see in the next section that this yields a constant competitive ratio, given that the jobs behave rationally. The algorithm including the payments is displayed below. Let here the indices of the jobs be defined according to the reported release dates, i.e., $j < k \Rightarrow \tilde{r}_j \leq \tilde{r}_k$. Let $\tilde{H}(j)$ and $\tilde{L}(j)$ be defined analogously to $H(j)$ and $L(j)$ on the basis of the reported weights and

processing times. Then for job j, arriving at time $\tilde{r}_j$, the tentative completion time and payment, respectively, at machine i are

$$\hat{C}_j(i) = \tilde{r}_j + b_i(\tilde{r}_j) + \sum_{\substack{k \in \tilde{H}(j) \\ k \to i \\ k < j \\ S_k \geq \tilde{r}_j}} \tilde{p}_k + \tilde{p}_j \quad \text{and} \quad \hat{\pi}_j(i) = \tilde{p}_j \sum_{\substack{k \in \tilde{L}(j) \\ k \to i \\ k < j \\ S_k > \tilde{r}_j}} \tilde{w}_k \,.$$

Algorithm 2: DecentralizedLocal Greedy Mechanism

Local Sequencing Policy: When a machine falls idle, it processes the job with highest (WSPT) priority among all available jobs queuing at this machine.

Assignment:

1. At time $\tilde{r}_j$ job j arrives and reports weight $\tilde{w}_j$ and processing time $\tilde{p}_j$ to all machines.
2. Every machine i informs j about both $\hat{C}_j(i)$ and $\hat{\pi}_j(i)$.
3. Job j chooses a machine $i_j \in M$. Its tentative utility for being queued at machine i is $\hat{u}_j(i) := -w_j \hat{C}_j(i) - \hat{\pi}_j(i)$.
4. The job is queued at i_j according to WSPT among all currently available jobs on i_j whose processing has not started yet. The payment $\hat{\pi}_j(i_j)$ has to be paid by j.
5. The (tentative) completion time for every job $k \in \tilde{L}(j)$, $k \to i_j$, $k < j$, $S_k > \tilde{r}_j$ increases by $\tilde{p}_j$ due to j's presence. As compensation, k receives a payment of $\tilde{w}_k \tilde{p}_j$.

Notice that the payments result in a balanced budget for the scheduler. That is, the payments paid and received by the jobs sum up to zero, since every arriving job immediately makes its payment to the jobs that are displaced by it. Also notice that the payments are online in the sense of Definition 5.2.

Theorem 5.9. *Regard any type vector t, any strategy profile x and any job j with report $(\tilde{r}_j, \tilde{p}_j, \tilde{w}_j)$, and machine choice $\tilde{i} \in M$. Then changing the report to $(\tilde{r}_j, \tilde{p}_j, w_j)$ and choosing a machine that maximizes tentative utility at time $\tilde{r}_j$ does not decrease j's tentative utility.*

Proof. We first regard the single machine case, i.e., $m = 1$. Suppose, at the arrival time $\tilde{r}_j$ of job j, jobs $k_1, k_2, \ldots, k_r$ with corresponding reported processing times $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_r$ and reported weights $\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_r$ are queueing to be processed on the machine, but none of them has started being processed yet. Without loss of

generality let $\tilde{w}_1/\tilde{p}_1 \geq \tilde{w}_2/\tilde{p}_2 \geq \cdots \geq \tilde{w}_r/\tilde{p}_r$. Given the reported processing time $\tilde{p}_j$, job j could receive any position in front of, between or behind the already present jobs in the priority queue by choosing its weight appropriately. Therefore, it has to decide for every job k_s, $s \in \{1, \ldots, r\}$, whether it wants to be placed in front of k_s or not. Displacing k_s would increase $\hat{\pi}_j(1)$ by $\tilde{w}_s\tilde{p}_j$, whereas $\hat{C}_j(1)$ is decreased by $\tilde{p}_s$. Thus, j's tentative utility changes by $w_j\tilde{p}_s - \tilde{w}_s\tilde{p}_j$ if j displaces k_s compared to not displacing k_s. Therefore, it is rational for j to displace k_s if and only if $w_j\tilde{p}_s - \tilde{w}_s\tilde{p}_j > 0$, which is equivalent to $w_j/\tilde{p}_j > \tilde{w}_s/\tilde{p}_s$. As the machine schedules according to WSPT, j is placed at the position that maximizes its tentative utility when reporting w_j.

For $m > 1$, recall that j can select a machine itself. As reporting the truth maximizes its tentative utility on every single machine, and as j can then choose the machine that maximizes its tentative utility among all machines, truth-telling and choosing a machine maximizing $\hat{u}_j$ will maximize j's tentative utility. □

Lemma 5.10. *Consider any job $j \in J$. Then for all reports of all other agents as well as all choices of machines of the other agents, the following is true:*

(*a*) *If j reports $\tilde{w}_j = w_j$, then the tentative utility when queued at any of the machines will be preserved over time, i.e., it equals j's ex-post utility.*

(*b*) *If j reports $\tilde{w}_j = w_j$, then selecting the machine that the* Local Greedy *Algorithm would have selected maximizes j's ex-post utility.*

Proof. To see (a), note that whenever j's tentative completion time changes, j is immediately compensated for that by a payment. If $\tilde{w}_j = w_j$ then the payment exactly equals the loss in utility. Claim (b) follows from (a) and the fact that the machine chosen by the Local Greedy Algorithm maximizes j's tentative utility. □

Theorem 5.9 implies that a myopic agent should report its true weight. Lemma 5.10 implies that such an agent is guaranteed to receive an ex-post utility as high as its tentative utility. The alternative is gambling: Recall that we have defined a restricted communication paradigm where jobs, upon arrival, are only informed about (tentative) completion times and (tentative) payments. In particular, jobs do not get to know which jobs are already queuing at the machines and what reports the

already present jobs have made. One can construct simple examples that demonstrate that overstating or understating the weight bears the risk of arbitrarily high utility losses in comparison to truthful reporting. More specifically, if a job j overstates its weight, this can result in a position in front of a job already present on the chosen machine whose weight over processing time ratio is smaller than j's true ratio. The payment j has to make in this case can be arbitrarily higher than the valuation j gains. Understating the weight can lead to a later job displacing j, but paying to j arbitrarily less than j's actual loss in valuation. On the other hand, one can similarly show that agents also have the chance of arbitrary high utility gains by overstating their weight. In this light, reporting truthfully becomes particularly attractive for risk-averse agents.

Theorem 5.11. *Consider the restricted strategy space where all $j \in J$ report $\tilde{w}_j = w_j$. Then the strategy profile where all jobs j truthfully report $\tilde{r}_j = r_j$, $\tilde{p}_j = p_j$ and choose a machine that maximizes $\hat{u}_j$ is a dominant strategy equilibrium.*

Proof. Let us start with $m = 1$. Suppose $\tilde{w}_j = w_j$, fix any $\tilde{r}_j$ and regard any $\tilde{p}_j > p_j$. Let u_j denote j's (ex-post) utility when reporting p_j truthfully and let $\tilde{u}_j$ be its (ex-post) utility for reporting $\tilde{p}_j$. As $\tilde{w}_j = w_j$, the ex-post utility equals in both cases the tentative utility at decision point $\tilde{r}_j$ according to Lemma 5.10(a). Let us therefore regard the latter utilities. Clearly, according to the WSPT-priorities, j's position in the queue at the machine for report p_j will not be behind its position for report $\tilde{p}_j$. Let us divide the jobs already queuing at the machine upon j's arrival into three sets: Let $J_1 = \{k \in J \,|\, k < j,\, S_k > \tilde{r}_j,\, \tilde{w}_k/\tilde{p}_k \geq w_j/p_j\}$, $J_2 = \{k \in J \,|\, k < j,\, S_k > \tilde{r}_j,\, w_j/p_j > \tilde{w}_k/\tilde{p}_k \geq w_j/\tilde{p}_j\}$ and $J_3 = \{k \in J \,|\, k < j,\, S_k > \tilde{r}_j,\, w_j/\tilde{p}_j > \tilde{w}_k/\tilde{p}_k\}$. That is, J_1 comprises the jobs that are in front of j in the queue for both reports, J_2 consists of the jobs that are only in front of j when reporting $\tilde{p}_j$ and J_3 includes only jobs that queue behind j for both reports. Therefore, $\tilde{u}_j - u_j$ equals

$$\begin{aligned} & - \sum_{k \in J_1 \cup J_2} w_j \tilde{p}_k - \sum_{k \in J_3} \tilde{p}_j \tilde{w}_k - w_j \tilde{p}_j - \left(- \sum_{k \in J_1} w_j \tilde{p}_k - \sum_{k \in J_2 \cup J_3} p_j \tilde{w}_k - w_j p_j \right) \\ = & \sum_{k \in J_2} (p_j \tilde{w}_k - w_j \tilde{p}_k) - \sum_{k \in J_3} (\tilde{p}_j - p_j) \tilde{w}_k - w_j (\tilde{p}_j - p_j). \end{aligned}$$

According to the definition of J_2, the first term is smaller than or equal to zero. As $\tilde{p}_j > p_j$, the whole right hand side becomes non-positive. Therefore $\tilde{u}_j \leq u_j$, i.e.,

truthfully reporting p_j maximizes j's ex-post utility on a single machine.

Let us now fix $\tilde{w}_j = w_j$ and any $\tilde{p}_j \geq p_j$ and regard any $\tilde{r}_j > r_j$. There are two effects that can occur when arriving later than r_j. Firstly, jobs queued at the machine already at time r_j may have been processed or may have started receiving service by time $\tilde{r}_j$. But either j would have had to wait for those jobs anyway or it would have increased its utility at decision point r_j by displacing a job and paying the compensation. So, j cannot gain from this effect by lying. The second effect is that new jobs have arrived at the machine between r_j and $\tilde{r}_j$. Those jobs either delay j's completion time and j looses the payment it could have received by arriving earlier. Or the jobs do not delay j's completion time, but j has to pay the jobs for displacing them when arriving at $\tilde{r}_j$. If j arrived at time r_j, it would not have to pay for displacing such a job. Hence, j cannot gain from this effect either. Thus the tentative utility at time r_j will be at least as large as the one at time $\tilde{r}_j$. Therefore, j maximizes its tentative utility by choosing $\tilde{r}_j = r_j$. As $\tilde{w}_j = w_j$, it follows from Lemma 5.10(a) that choosing $\tilde{r}_j = r_j$ also maximizes the job's ex-post utility on a single machine.

For $m > 1$, note that on every machine, the tentative utility of job j at decision point $\tilde{r}_j$ is equal to its ex-post utility and that j can select a machine itself that maximizes its tentative utility and therefore its ex-post utility. Therefore, given that $\tilde{w}_j = w_j$, a job's ex-post utility is maximized by choosing $\tilde{r}_j = r_j$, $\tilde{p}_j = p_j$ and, according to Lemma 5.10(b), by choosing the machine that the Local Greedy Algorithm would have chosen. □

Theorem 5.12. *Given the types of all jobs, the strategy profile where each job j reports $(\tilde{r}_j, \tilde{p}_j, \tilde{w}_j) = (r_j, p_j, w_j)$ and chooses a machine maximizing its tentative utility $\hat{u}_j$ is a myopic best response equilibrium.*

Proof. Regard job j. According to the proof of Theorem 5.9, $\hat{u}_j$ on any machine is maximized by reporting $\tilde{w}_j = w_j$ for any $\tilde{r}_j$ and $\tilde{p}_j$. According to Theorem 5.11 and Lemma 5.10(b), $\tilde{p}_j = p_j$, $\tilde{r}_j = r_j$ and choosing a machine that maximizes j's tentative utility at time $\tilde{r}_j$ maximize j's ex-post utility if j truthfully reports $\tilde{w}_j = w_j$. According to Lemma 5.10(a) this ex-post utility is equal to $\hat{u}_j$ if j reports $\tilde{w}_j = w_j$. Therefore, any job j maximizes $\hat{u}_j$ by truthful reports and choosing the machine as claimed. □

In order to obtain the myopic best response equilibrium, payments paid by an arriving job j need not necessarily be given to the jobs delayed by j. We formulate this fact as a Corollary.[4]

Corollary 5.13. *If the* Decentralized Local Greedy *Mechanism is modified such that payments are collected from jobs, but not given to the other jobs, then truth-telling and choosing a machine that maximizes the tentative utility $\hat{u}_j$ is a myopic best response equilibrium.*

Proof. According to Theorem 5.9, truthfully reporting w_j maximizes j's tentative utility for any $\tilde{p}_j$ and $\tilde{r}_j$. Furthermore, similar to the proof of Theorem 5.11 it can be shown that truthfully reporting the processing time maximizes j's tentative utility for any $\tilde{r}_j$ and truthful $\tilde{w}_j = w_j$. Concerning the release date, arriving late at time $\tilde{r}_j$ instead of r_j does not increase the tentative utility for the following reasons. Jobs that were present at r_j are already finished or have started receiving service or are still waiting at time $\tilde{r}_j$. For those jobs, j either would have had to wait anyway or j could have increased its utility by displacing such a job and paying the compensation. Jobs that have arrived between time r_j and $\tilde{r}_j$ can only delay j or increase the amount that j has to pay. In any case, j cannot benefit from arriving late. Therefore, arriving at r_j maximizes j's tentative utility. This proves the claim. □

Although paying jobs when being displaced is not necessary to obtain the equilibrium, it is desirable for other reasons. First of all, the resulting ex-post payments yield a balanced budget and in equilibrium, the tentative utility at arrival is preserved and equals the ex-post utility of every job (Lemma 5.9). Furthermore, paying jobs for their delay results in a dominant strategy equilibrium in a restricted type space (Theorem 5.11).

5.5 Performance of the Mechanism

As shown in Section 5.4, jobs have a motivation to report truthfully about their data. Therefore we will call a job *rational* if it truthfully reports w_j, p_j and r_j and chooses a machine maximizing its tentative utility $\hat{u}_j$. In this section, we will show

[4]We add an extra proof, since the proof of Theorem 5.12 uses the detour via ex-post utilities, which is not possible if jobs are not compensated for delays.

that if all jobs are rational, then the Decentralized Local Greedy Mechanism is 3.281-competitive.

5.5.1 Handling Critical Jobs

Recall that from Section 5.2.1 on, we assumed that no critical jobs exist, as we defined the Decentralized Local Greedy Mechanism only for jobs j with $r_j \geq \alpha\, p_j$. We will now relax this assumption and allow jobs to have types from the more general type space $\{(r_j, p_j, w_j) | r_j \geq 0,\, p_j \geq 0,\, w_j \in \mathbb{R}\}$. Without the assumption, the DecentralizedLocal Greedy Mechanism as stated above does not yet yield a constant performance ratio; simple examples can be constructed in the same flavor as by Megow and Schulz (2004). In fact, it is well known that early arriving jobs with large processing times have to be delayed (Anderson and Potts 2004; Megow and Schulz 2004; Megow et al. 2006). In order to achieve a constant performance ratio, we also adopt this idea and use modified release dates as Megow and Schulz (2004) and Megow et al. (2006). To this end, we define the modified release date of every job $j \in J$ as $r'_j = \max\{r_j, \alpha p_j\}$, where $\alpha \in (0, 1]$ will later be chosen appropriately. For our decentralized setting, this means that a machine will not admit any job j to its priority queue before time $\max\{\tilde{r}_j, \alpha\tilde{p}_j\}$ if j arrives at time $\tilde{r}_j$ and reports processing time $\tilde{p}_j$. Moreover, machines refuse to provide information about the tentative completion time and payment to a job before its modified release date (with respect to the job's reported data). Note that this modification is part of the local scheduling policy of every machine and therefore does not restrict the required decentralization. Note further that any myopic rational job j still reports $\tilde{w}_j = w_j$ according to Theorem 5.9 and that a rational job reports $\tilde{p}_j = p_j$ as well as communicates to machines at the earliest opportunity, i.e., at time $\max\{r_j, \alpha p_j\}$, according to the arguments in the proof of Theorem 5.11. Moreover, the aforementioned properties concerning the balanced budget, the conservation of utility in the case of a truthfully reported weight, and the online property of the payments still apply to the algorithm with modified release dates.

5.5.2 Proof of the Competitive Ratio

It is not a goal in itself to have a truthful mechanism, but to use the truthfulness in order to achieve a reasonable overall performance in terms of the social welfare $\sum w_j\, C_j$.

Theorem 5.14. *Suppose every job is rational in the sense that it reports r_j, p_j, w_j and selects a machine that maximizes its tentative utility at arrival. Then the*

Decentralized Local Greedy *Mechanism is ϱ-competitive, with $\varrho = 3.281$.*

The proof of the theorem partly follows the lines of the corresponding proof by Megow et al. (2006). But the distribution of jobs over machines in their algorithm differs. Therefore, our result is not implied by the result by Megow et al. (2006) and it is necessary to give a proof here.

Proof. A rational job communicates to the machines at time $\max\{r_j, \alpha p_j\}$ and chooses a machine i_j that maximizes its utility upon arrival $\hat{u}_j(i_j)$. That is, it selects a machine i that minimizes

$$-\hat{u}_j(i) = w_j \hat{C}_j(i) + \hat{\pi}_j(i) = w_j \big[r'_j + b_i(r'_j) + \sum_{\substack{k \in H(j) \\ k \to i \\ k < j \\ S_k \geq r'_j}} p_k + p_j \big] + p_j \sum_{\substack{k \in L(j) \\ k \to i \\ k < j \\ S_k > r'_j}} w_k .$$

This, however, exactly equals the immediate increase of the objective value $\sum w_j C_j$ that is due to the addition of job j to the schedule. We now claim that we can express the objective value Z of the resulting schedule as

$$Z = \sum_{j \in J} -\hat{u}_j(i_j) \,,$$

where i_j is the machine selected by job j. Here, it is important to note that $-\hat{u}_j(i_j)$ does not express the total (ex-post) contribution of job j to $\sum w_j C_j$, but only the increase *upon arrival* of j on machine i_j. However, further contributions of job j to $\sum w_j C_j$ only appear when job j is displaced by some later arriving job with higher priority, say k. This contribution by job j to $\sum w_j C_j$, however, will be accounted for when adding $-\hat{u}_k(i_k)$.

Next, since we assume that any job maximizes its utility upon arrival, or equivalently minimizes $-\hat{u}_j(i)$ when selecting a machine i, we can apply an averaging argument over the number of machines, like in Megow et al. (2006), to obtain:

$$Z \leq \sum_{j \in J} \frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i) \,.$$

Next, recall that upon arrival of job j on any of the machines i (at time r'_j), machine

i is blocked for time $b_i(r'_j) \leq r'_j/\alpha$. Therefore, we get for any j,

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} -\hat{u}_j(i) &= w_j r'_j + w_j \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j \sum_{i=1}^{m} \sum_{\substack{k\in H(j)\\ k\to i\\ k<j\\ S_k\geq r'_j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{i=1}^{m} \sum_{\substack{k\in L(j)\\ k\to i\\ k<j\\ S_k> r'_j}} \frac{w_k}{m} \\
&= w_j r'_j + w_j \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j \sum_{\substack{k\in H(j)\\ k<j\\ S_k\geq r'_j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{\substack{k\in L(j)\\ k<j\\ S_k> r'_j}} \frac{w_k}{m} \\
&\leq w_j r'_j + w_j \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m} \\
&\leq w_j r'_j + w_j \frac{r'_j}{\alpha} + w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m}\,.
\end{aligned}$$

Thus,

$$Z \leq \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + \sum_{j\in J} w_j p_j + \sum_{j\in J} p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m}.$$

The last term can be rewritten as follows:

$$\sum_{j\in J} p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m} = \sum_{\substack{(j,k):\\ j\in H(k)\\ k<j}} p_j \frac{w_k}{m} = \sum_{\substack{(j,k):\\ k\in H(j)\\ j<k}} p_k \frac{w_j}{m} = \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k>j}} \frac{p_k}{m}\,.$$

Therefore,

$$\begin{aligned}
Z &\leq \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + \sum_{j\in J} w_j p_j + \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k>j}} \frac{p_k}{m} \\
&= \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \sum_{j\in J} w_j \sum_{k\in H(j)} \frac{p_k}{m} + \frac{m-1}{m}\sum_{j\in J} w_j p_j\,.
\end{aligned}$$

Now, we apply a lower bound on the optimal off-line schedule by Eastman, Even,

and Isaacs (1964, Thm. 1), namely

$$Z^{OPT} \geq \sum_{j \in J} w_j \sum_{k \in H(j)} \frac{p_k}{m} + \frac{m-1}{2m} \sum_{j \in J} w_j p_j \,,$$

yielding:

$$\begin{aligned} Z &\leq Z^{OPT} + \sum_{j \in J} w_j (1 + \frac{1}{\alpha}) r'_j + \frac{m-1}{2m} \sum_{j \in J} w_j p_j \\ &\leq Z^{OPT} + \sum_{j \in J} w_j (1 + \frac{1}{\alpha})(r_j + \alpha p_j) + \frac{m-1}{2m} \sum_{j \in J} w_j p_j \\ &= Z^{OPT} + \sum_{j \in J} w_j \left[(1 + \frac{1}{\alpha}) r_j + (1 + \alpha + \frac{m-1}{2m}) p_j \right], \end{aligned}$$

where in the second inequality $r_j + \alpha p_j$ is used as an upper bound on r'_j. Applying the trivial lower bound $\sum_{j \in J} w_j (r_j + p_j) \leq Z^{OPT}$, we get:

$$\begin{aligned} Z &\leq Z^{OPT} + \max \left\{ 1 + \frac{1}{\alpha}, 1 + \alpha + \frac{m-1}{2m} \right\} Z^{OPT} \\ &= 2Z^{OPT} + \max \left\{ \frac{1}{\alpha}, \alpha + \frac{m-1}{2m} \right\} Z^{OPT}. \end{aligned}$$

Therefore, we get the performance bound

$$\varrho = 2 + \max \left\{ \frac{1}{\alpha}, \alpha + \frac{m-1}{2m} \right\}.$$

This can now be optimized for α, which was already done in Megow and Schulz (2004). There it was shown that $\varrho < 3.281$ for $\alpha = (\sqrt{17m^2 - 2m + 1} - m + 1)/(4m)$. □

5.6 On Dominant Strategy Equilibria

We show that any mechanism that has a dominant strategy equilibrium with truthful reports must necessarily differ from the Decentralized Local Greedy Mechanism in the allocation of jobs to machines and time slots; so it remains unclear what the performance of such a mechanism might be. Giving up on decentralization, however, we show that it is possible to define a constant competitive mechanism on

the basis of the algorithm by Correa and Wagner (2005) such that truthtelling is a dominant strategy equilibrium. Finally, notice that for the single machine case, decentralization is not an issue. We comment on the Correa-Wagner Algorithm for the single machine case and show that also the Local Greedy Algorithm can be made truthful with respect to the stronger dominant strategy equilibrium. The payment scheme required to achieve that, however, is different from the payment scheme of the Decentralized Local Greedy Mechanism.

5.6.1 A Negative Result

Recall that in the Local Greedy algorithm, jobs are centrally assigned to machines such as to minimize the increase in the global objective function $\sum w_j C_j$. We can see this algorithm as the allocation algorithm of a mechanism where the only action of any job is to report its type. Recall that such mechanisms are known as direct revelation mechanisms. In that context, a *truthful* (*direct revelation*) *mechanism* denotes a mechanism where truthtelling is a dominant strategy equilibrium. The question arises if the Local Greedy algorithm can be augmented by some payment scheme to a truthful mechanism. For the case of parallel machines, however, we have the following negative result.

Theorem 5.15. *There does not exist a payment scheme that extends the* Local Greedy *algorithm to a truthful mechanism.*

Proof. We first derive a necessary condition for making truthful reports a dominant strategy equilibrium, and then show that it is violated by the Local Greedy Algorithm. Suppose there is a payment scheme π such that truthful reporting is a dominant strategy for each job. Fix job j and the reports of all other jobs. Let also j's report about processing time and release date be fixed. Consider two weights of j, $w^{(1)}$ and $w^{(2)}$, and denote by $C^{(1)}$ and $C^{(2)}$ the resulting completion times and by $\pi^{(1)}$ and $\pi^{(2)}$ the resulting payments when j reports $w^{(1)}$ and $w^{(2)}$, respectively. As truthful reporting is a dominant strategy, reporting $w^{(1)}$ must maximize j's ex-post utility when j has true weight $w^{(1)}$, especially, $-w^{(1)}C^{(1)} - \pi^{(1)} \geq -w^{(1)}C^{(2)} - \pi^{(2)}$. Similarly, $-w^{(2)}C^{(2)} - \pi^{(2)} \geq -w^{(2)}C^{(1)} - \pi^{(1)}$. Adding up these two inequalities

yields[5]

$$(w^{(2)} - w^{(1)})(C^{(1)} - C^{(2)}) \geq 0. \tag{5.1}$$

Especially, if $w^{(1)} < w^{(2)}$ we must have $C^{(1)} \geq C^{(2)}$.

Consider now the following example. Let $[\tilde{w}/\tilde{p}]$ denote a job with (reported) weight $\tilde{w}$ and (reported) processing time $\tilde{p}$. Suppose that we have to schedule the following four jobs on two machines: $[6/3], [5/4], j = [\tilde{w}/\frac{1}{7}], [20/4]$, where $\tilde{w}$ is a parameter. Let all jobs have a common release date large enough such that no job has to be delayed according to the modified release dates (say $r > 4\alpha$, with α as in Section 5.5.1), but assume that they arrive in the given order.

The first job [6/3] increases the objective value on both machines by the same amount and is therefore scheduled on the first machine. The second job [5/4] is then assigned to the second machine. We consider two values for the weight of j, namely $w^{(1)} = \frac{1}{14}$ and $w^{(2)} = \frac{1}{2}$. In the first case the weight over processing time ratio is $\frac{1}{2}$ and therefore smaller than the respective ratios of the two jobs already assigned to machines. Thus, j would be scheduled last on each of the machines according to the WSPT rule. It would cause the following increases:

$$z(j,1) = (r + \frac{1}{7} + 3)w^{(1)}$$

$$z(j,2) = (r + \frac{1}{7} + 4)w^{(1)}.$$

Therefore, j is assigned to the end of machine 1.

The second case for $w^{(2)} = \frac{1}{2}$ yields a ratio of $\frac{7}{2}$, which would place j first on both machines. The respective increases are:

$$z(j,1) = (r + \frac{1}{7})w^{(2)} + 6 \cdot \frac{1}{7}$$

$$z(j,2) = (r + \frac{1}{7})w^{(2)} + 5 \cdot \frac{1}{7}.$$

Job j would be scheduled on machine 2.

[5]Condition (5.1) corresponds to the notion of monotonicity as introduced by Bikhchandani, Chatterjee, Lavi, Mu'alem, Nisan, and Sen (2006). Furthermore, in the single parameter setting where only the weights are private information, it is equivalent to the decreasing work curves condition by Archer and Tardos (2001). Cf. Chapters 3 and 4.

The last job [20/4] has a ratio larger than all the ratios of the present jobs. Therefore it would be scheduled first on both machines. In both cases, the total weight of jobs on the first machine is larger than the total weight of jobs on the second machine. Therefore, the increase in the objective value caused by the last job is in both cases smaller on the second machine. Thus the job is scheduled on the second machine, which increases j's completion time only in the second case. Thus, j is completed at time $r+3+\frac{1}{7}$ when reporting $\frac{1}{14}$ and at time $r+4+\frac{1}{7}$ when reporting $\frac{1}{2}$. Hence, condition (5.1) is violated. □

Remark 5.16. *Theorem 5.15 does not depend on the fact that p_j and r_j are private information. Hence, even if only the weights are private and the other job characteristics are public, the* LocalGreedy *Algorithm cannot be augmented to a truthful mechanism.*

5.6.2 On the Correa–Wagner Algorithm

As mentioned in the introduction, the currently best known performance guarantee for the regarded online scheduling problem is 2.62 due to Correa and Wagner (2005). In this section, we show how this algorithm can be used to obtain a centralized mechanism that admits a dominant strategy equilibrium where all jobs report truthfully.

The Correa-Wagner algorithm maintains a virtual single machine that can process jobs m times faster than the original machines. The virtual machine preemptively processes at any point in time the available job with the highest w_j/p_j ratio. For $\alpha \in [0,1]$, the α-point of a job is defined as the point in time, when for the first time an α-fraction of the job is processed on the virtual machine. Jobs enter a FIFO-queue in the order of their α-points. Whenever one of the m 'real' machines becomes idle, it starts processing the next job from the FIFO-queue. The mentioned performance bound is achieved by choosing $\alpha = (\sqrt{5}-1)/2$.

Theorem 5.17. *For the parallel machine problem, consider a fixed job j and let the reports of the other jobs be fixed as well. For given reports $\tilde{p}_j$ and $\tilde{r}_j$, let $C^0 \leq \cdots \leq C^k \leq \cdots \leq C^{MAX}$ be the (finitely many) possible values for j's completion time when j varies its report $\tilde{w}_j$. Let furthermore $\varrho^\xi = \inf\{w \mid \text{report } w \text{ leads to } C^{\xi-1}\}$ and define*

$$\theta_j^k = \sum_{\xi=k+1}^{MAX} \left[\varrho^\xi \left(C^\xi - C^{\xi-1}\right)\right]$$

to be the payment that j has to pay if j's completion time is C^k. Then the Correa–Wagner Algorithm together with the payment scheme θ has a dominant strategy equilibrium in which all jobs are truthful.

The payment scheme θ is under certain conditions related to the VCG-payment scheme. But especially in an online setting, this relationship does not hold; see Section 5.6.3.

For the induced single parameter problem, where only weights are private information, the result in Archer and Tardos (2001) implies that a sufficient condition for the existence of a truthful payment scheme is that the completion time of each job depends non-increasingly on the job's reported weight. For the Correa–Wagner Algorithm, indeed a higher report for the weight can only improve a job's priority on the virtual machine and therefore can only decrease the completion time of the job. Therefore, the existence of a payment scheme that makes truth-telling w_j a dominant strategy (for fixed processing time and release date) is guaranteed by the result of Archer and Tardos (2001). The above defined payments can be obtained using the methods by Gui, Müller, and Vohra (2004) or by Archer and Tardos (2001). Here, we only show that the payments θ_j^k indeed make truth-telling a dominant strategy with respect to all three job characteristics.

Proof of Theorem 5.17. Fix a job j and the reports of the other jobs. For given reports $\tilde{p}_j$ and $\tilde{r}_j$ of job j, let $C(w_j)$ denote j's completion time as a function of the reported weight w_j. As mentioned above, $C(w_j)$ is a non-increasing function; in fact, it is a non-increasing step function. With that insight, it can be easily verified that if j achieves C^k under report w_j, then the payment satisfies

$$\theta_j^k = \int_0^{w_j} (C(x) - C(w_j))\ dx.$$

Suppose that j has true weight w_j and let $\tilde{w}_j > w_j$ be some untruthful report. Then the corresponding incentive constraint reads as

$$-w_j C(w_j) - \int_0^{w_j} (C(x) - C(w_j))\ dx \geq -w_j C(\tilde{w}_j) - \int_0^{\tilde{w}_j} (C(x) - C(\tilde{w}_j))\ dx,$$

which is equivalent to

$$\int_{w_j}^{\tilde{w}_j} (C(x) - C(\tilde{w}_j))\ dx \geq 0.$$

The latter is true, since $C(\cdot)$ is non-increasing and thus the integrand is non-negative.

For $\tilde{w}_j < w_j$, the corresponding incentive constraint can be verified similarly.[6]

From the above analysis follows that truthfully reporting the weight is a dominant strategy for job j, no matter what reports j makes about its processing time and release date. Let us now turn to j's processing time. For the true processing time p_j and fixed release date, let $C(w_j)$, $w_j \geq 0$, be the completion time for weight report w_j. Define $\tilde{C}(w_j)$ analogously for some untruthful processing time $\tilde{p}_j > p_j$. Clearly, $\tilde{C}(w_j) \geq C(w_j)$ for all w_j. The corresponding incentive constraint with respect to processing times reads as

$$\begin{aligned} -w_j C(w_j) - \int_0^{w_j} (C(x) - C(w_j))\, dx &\geq -w_j \tilde{C}(w_j) - \int_0^{w_j} \left(\tilde{C}(x) - \tilde{C}(w_j)\right) dx \\ \Leftrightarrow \int_0^{w_j} \left(\tilde{C}(x) - C(x)\right) dx &\geq 0. \end{aligned}$$

The latter is implied by $\tilde{C}(x) \geq C(x)$ for all $x > 0$.

It remains to show that arriving no later than the true release date is a weakly dominant strategy, too. Assume j's report is truthful in w_j and p_j. For fixed processing time p_j and fixed release date r_j, let $C(w_j)$, $w_j \geq 0$, be the completion time for weight report w_j. Define $\tilde{C}(w_j)$ analogously for release date $\tilde{r}_j$. Clearly, $\tilde{C}(w_j) \geq C(w_j)$ for all w_j. This implies the incentive constraints with respect to the release date similar to the above.

□

The payments θ_j^k can be computed in polynomial time, which can be seen as follows. For every job j, given the actual reports of the other jobs and j's report about processing time and release date, it is sufficient to know j's completion time as a function of the reported weight in order to determine the payments. It can be easily verified that this function is a non-increasing step function whose points of discontinuity are a subset of the set of reported ratios w_ℓ/p_ℓ for the $n-1$ jobs $\ell \neq j$. Therefore, it is sufficient to determine j's completion time for n values of j's weight. Determining the completion time for one of these values requires running the Correa-Wagner algorithm once again and thus takes polynomial time. Hence, determining the payments for all n jobs can be done in polynomial time, too.

However, the payment scheme θ is not an online payment scheme and it does not satisfy the balanced budget condition. The described mechanism is heavily

[6] For a similar, but more intuitive proof for the induced single parameter problem we refer to Lavi (2007).

dependent on centralization in maintaining the fast virtual machine and the FIFO-queue. Therefore, we hardly see a chance to turn the mechanism into a decentralized one.

5.6.3 The Single Machine Case

For a single machine, the decentralization requirement is redundant. In this case, the Correa-Wagner algorithm is equivalent to the algorithm by Goemans, Queyranne, Schulz, Skutella, and Wang (2002), which yields a performance guarantee of 2.42. This way, we get a truthful mechanism with performance guarantee 2.42.

Although this is better than the performance guarantee of 3 for the Local Greedy Algorithm on a single machine (see Megow and Schulz 2004), we briefly comment on the latter, too. Even for a single machine, the Decentralized Local Greedy Mechanism does in general not have a dominant strategy equilibrium where all jobs report truthfully: Consider a job j with $w_j = p_j = 1$ arriving first and a job k with $w_k = 2$ and $p_k = 1$ arriving second. Suppose that both jobs report truthfully. Job j is displaced by k and receives a payment of $w_j p_k = 1$. But with any report $1 < \tilde{w}_j < 2$, job j would still be displaced by k, receiving a higher payment $\tilde{w}_j p_k > 1$. Even when we give up on a balanced budget and j does not receive the payment when being displaced by k, j would be better off lying $\tilde{w}_j = 3$ and not being displaced at all.

However, the Local Greedy Algorithm together with payment scheme θ from Section 5.6.2 yields a mechanism in which truth-telling becomes a dominant strategy equilibrium. This can be proven analogously to Theorem 5.17. Again, in contrast to the payment scheme of the Decentralized Local Greedy mechanism, the payment scheme θ does not satisfy the balanced budget condition and is not an online payment scheme.

In the offline case with trivial release dates, i.e., when $r_j = 0$ for all $j \in J$, the WSPT rule minimizes $\sum_{j \in J} w_j C_j$ and therefore maximizes the total social welfare. In this case, the WSPT rule together with payments given by θ coincide with the VCG mechanism, see also Example 4.19. It can be verified that the payment that job j has to make according to θ equals the product of j's processing time and the total weight of the jobs that are processed behind j. This is exactly the externality that j imposes on the other jobs and therefore equals the VCG-payment. For the online case with nontrivial release dates, however, an exact algorithm does not exist (Hoogeveen and Vestjens 1996). Hence, it is not possible to set up the VCG mechanism. As a consequence, neither of the previously discussed mechanisms is the VCG mechanism.

5.7 Discussion

We leave it as an open problem to find a decentralized, constant competitive online mechanism where it is a *dominant strategy equilibrium* to report truthfully. A decentralized algorithm that mimics the Local Greedy Algorithm is not a candidate, as the latter cannot be augmented by any payment scheme to a mechanism with a dominant strategy equilibrium.

We have shown that the algorithm with the currently best performance bound for the non-strategic, centralized setting can be turned into a truthful mechanism with competitive ratio 2.62. But, the resulting mechanism is not decentralized and the given payment scheme is not online. Thus, an intriguing open question remains: Is the decentralized setting actually harder than the setting with central coordination?

Finally, we believe that it would be interesting to identify general rules that allow for a transformation of centralized algorithms to decentralized mechanisms – our work can be seen as one instance of such a result.

References

Ambrosio, P. and V. Auletta (2005). Deterministic monotone algorithms for scheduling on related machines. In G. Persiano and R. Solis-Oba (Eds.), *Approximation and Online Algorithms*, Volume 3351 of *Lecture Notes in Computer Science*, pp. 267–280. Springer.

Andelman, N., Y. Azar, and M. Sorani (2005). Truthful approximation mechanisms for scheduling selfish related machines. In V. Diekert and B. Durand (Eds.), *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science*, Volume 3404 of *Lecture Notes in Computer Science*, pp. 69–82. Springer.

Anderson, E. J. and C. N. Potts (2004). Online scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research 29*(3), 686–697.

Angel, E., E. Bampis, and F. Pascual (2005). Truthful algorithms for scheduling selfish tasks on parallel machines. In X. Deng and Y. Ye (Eds.), *Internet and Network Economics*, Volume 3828 of *Lecture Notes in Computer Science*, pp. 698–707. Springer.

Archer, A. and E. Tardos (2001). Truthful mechanisms for one-parameter agents. In *Proc. 42nd Annual Symposium on Foundations of Computer Science*, pp. 482–491. IEEE Computer Society.

Armstrong, M. (2000). Optimal multi-object auctions. *Review of Economic Studies 67*, 455–481.

Aspnes, J., Y. Azar, A. Fiat, S. Plotkin, and O. Waarts (1997). On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM 44*(3), 486–504.

Auletta, V., R. De Prisco, P. Penna, and G. Persiano (2004). Deterministic truthful approximation mechanisms for scheduling related machines. In V. Diekert and M. Habib (Eds.), *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science*, Volume 2996 of *Lecture Notes in Computer Science*, pp. 608–619. Springer.

Ausubel, L. and P. Milgrom (2006). The lovely but lonely vickrey auction. In P. Cramton, Y. Shoham, and R. Steinberg (Eds.), *Combinatorial Auctions*, pp. 17–40. MIT Press.

Azar, Y., K. Jain, and V. Mirrokni (2008). Optimal coordination mechanisms for unrelated machine scheduling. In *Proc. Symposium on Discrete Algorithms (SODA'08)*, pp. 323–332.

Baruah, S., G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang (1992). On the competitiveness of on-line real-time task scheduling. *Journal of Real-Time Systems 4*(2), 125–144.

Bein, D. and L. Wein (2003). An inverse-optimization based auction mechanism to support a multiattribute RFQ process. *Management Science 49*(11), 1529–1545.

Bikhchandani, S., S. Chatterjee, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen (2006). Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica 74*(4), 1109–1132.

Briest, P., P. Krysta, and B. Vöcking (2005). Approximation techniques for utilitarian mechanism design. In *Proc. 37th Annual ACM Symposium on Theory of Computing*, New York, pp. 39–48. ACM.

Cachon, G. and M. Lariviere (1999). Capacity choice and allocation: Strategic behavior and supply chain performance. *Management Science 45*(8), 1091–1108.

Cho, Y. and S. Sahni (1980). Bounds for list schedules on uniform processors. *SIAM Journal on Computing 9*(1), 91–103.

Christodoulou, G., E. Koutsoupias, and A. Nanavati (2004). Coordination mechanisms. In *Automata, Languages and Programming*, Volume 3142 of *Lecture Notes in Computer Science*, pp. 345–357. Springer.

Chung, K.-S. and W. Olszewski (2007). A non-differentiable approach to revenue equivalence. *Theoretical Economics 2*, 469–487.

Clarke, E. H. (1971). Multipart pricing of public goods. *Public Choice 11*(1), 17–33.

Cole, R., Y. Dodis, and T. Roughgarden (2006). How much can taxes help selfish routing? *Journal of Computer and System Sciences 72*(3), 444–467.

Correa, J. R. and M. R. Wagner (2005). LP-based online scheduling: from single to parallel machines. In M. Jünger and V. Kaibel (Eds.), *Integer Programming and Combinatorial Optimization*, Volume 3509 of *Lecture Notes in Computer Science*, pp. 196–209. Springer.

Cramton, P., Y. Shoham, and R. Steinberg (Eds.) (2006). *Combinatorial Auctions*. MIT Press.

Czumaj, A. (2004). Selfish routing on the internet. In J. Leung (Ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapter 42. CRC Press.

Czumaj, A. and B. Vöcking (2007). Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms 3*(1).

Demange, G., D. Gale, and M. Sotomayor (1986). Multi-item auctions. *The Journal of Political Economy 94*(4), 863–872.

Dobson, G. (1984). Scheduling independent tasks on uniform processors. *SIAM Journal on Computing 13*(4), 705–716.

Eastman, W. L., S. Even, and I. M. Isaacs (1964). Bounds for the optimal scheduling of n jobs on m processors. *Management Science 11*, 268–279.

Feldmann, R., M. Gairing, T. Lücking, B. Monien, and M. Rode (2003). Nashification and the coordination ratio for a selfish routing game. In *Proc. of the 30th International Colloquium on Automata, Languages and Programming (ICALP 2003)*, pp. 514–526.

Finn, G. and E. Horowitz (1979). A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics 19*(3), 312–320.

Friesen, D. (1987). Tighter bounds for LPT scheduling on uniform processors. *SIAM Journal on Computing 16*(3), 554–560.

Gallien, J. and L. Wein (2005). A smart market for industrial procurement with capacity constraints. *Management Science 51*(1), 76–91.

Garey, M. and D. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. Freeman.

Goemans, M., M. Queyranne, A. Schulz, M. Skutella, and Y. Wang (2002). Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics 15*(2), 165–192.

Graham, R. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal 45*(9), 1563–1581. published also as Graham (1969).

Graham, R. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics 17*, 416–429.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287–326.

Green, J. and J.-J. Laffont (1977). Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica 45*(2), 427–438.

Groves, T. (1973). Incentives in teams. *Econometrica 41*, 617–631.

Gui, H., R. Müller, and R. Vohra (2004). Dominant strategy mechanisms with multidimensional types. Discussion Paper 1392, The Center for Mathematical Studies in Economics & Management Sciences, Northwestern University, Evanston, IL.

Hajiaghayi, M., R. Kleinberg, M. Mahdian, and D. Parkes (2005). Online auctions with re-usable goods. In *Proc. 6th ACM Conf. on Electronic Commerce (EC'05)*, pp. 165–174.

Hartline, J. and A. Karlin (2007). Profit maximization in mechanism design. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge University Press.

Heydenreich, B., D. Mishra, R. Müller, and M. Uetz (2008). Optimal mechanisms for single machine scheduling. In C. Papadimitriou and S. Zhang (Eds.), *Internet and Network Economics - WINE 2008*, Volume 5385 of *Lecture Notes in Computer Science*, pp. 414–425. Springer.

Heydenreich, B., R. Müller, and M. Uetz (2007). Games and mechanism design in machine scheduling - an introduction. *Production and Operations Management 16*(4), 437–454.

Heydenreich, B., R. Müller, and M. Uetz (2009). Mechanism design for decentralized online machine scheduling. *Operations Research*. to appear.

Heydenreich, B., R. Müller, M. Uetz, and R. Vohra (2009). Characterization of revenue equivalence. *Econometrica 77*(1), 307–316.

Holmström, B. (1979). Groves' scheme on restricted domains. *Econometrica 47*(5), 1137–1144.

Hoogeveen, J. A. and A. P. A. Vestjens (1996). Optimal on-line algorithms for single machine scheduling. In W. H. Cunningham, S. T. McCormick, and

M. Queyranne (Eds.), *Integer Programming and Combinatorial Optimization*, Volume 1084 of *Lecture Notes in Computer Science*, pp. 404–414. Springer.

Ibarra, O. and C. Kim (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. Assoc. Comput. Mach. 24*(2), 280–289.

Immorlica, N., L. Li, V. S. Mirrokni, and A. Schulz (2008). Coordination mechanisms for selfish scheduling. *Theoretical Computer Science*. to appear.

Jehiel, P. and B. Moldovanu (2001). Efficient design with interdependent valuations. *Econometrica 69*(5), 12371259.

Jehiel, P., B. Moldovanu, and E. Stacchetti (1996). How (not) to sell nuclear weapons. *American Economic Review 86*(4), 814–829.

Jehiel, P., B. Moldovanu, and E. Stacchetti (1999). Multidimensional mechanism design for auctions with externalities. *Journal of Economic Theory 85*(2), 258–293.

Kawaguchi, T. and S. Kyan (1986). Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing 15*(4), 1119–1129.

Kittsteiner, T. and B. Moldovanu (2005). Priority auctions and queue disciplines that depend on processing time. *Management Science 51*, 236–248.

Klemperer, P. (1999). Auction theory: A guide to the literature. *Journal of Economic Surveys 13*(3), 227–286.

Koren, G. and D. Shasha (1995). D-over: An optimal on-line scheduling algorithm for overloaded real-time systems. *SIAM Journal on Computing 24*(2), 318–339.

Koutsoupias, E. and C. Papadimitriou (1999). Worst-case equilibria. In C. Meinel and S. Tison (Eds.), *STACS 1999, 16th Annual Symposium on Theoretical Aspects of Computer Science*, Volume 1563 of *Lecture Notes in Computer Science*, pp. 404–413. Springer.

Koutsoupias, E. and A. Vidali (2007). A lower bound of 1+phi for truthful scheduling mechanisms. pp. 454–464.

Kovacs, A. (2005). Fast monotone 3-approximation algorithm for scheduling related machines. In G. S. Brodal and S. Leonardi (Eds.), *Algorithms - ESA 2005*, Volume 3669 of *Lecture Notes in Computer Science*, pp. 616–627. Springer.

Kreipl, S. and M. Pinedo (2004). Planning and scheduling supply chains: An overview of issues in practice. *Production and Operations Management 13*(1),

77–92.

Krishna, V. (2002). *Auction Theory*. Academic Press.

Krishna, V. and E. Maenner (2001). Convex potentials with an application to mechanism design. *Econometrica 69*(4), 1113–1119.

Lavi, R. (2007). Computationally efficient approximation mechanisms. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge University Press.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (1993). Sequencing and scheduling: Algorithms and complexity. In *Logistics of Production and Inventory*, Volume 4 of *Handbooks in Operations Research and Management Science*, pp. 445–522. Amsterdam: North-Holland.

Lenstra, J. K., A. H. G. Rinnooy Kan, and P. Brucker (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343–362.

Leung, J. (Ed.) (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press.

Malakhov, A. and R. Vohra (2007). An optimal auction for capacity constrained bidders: a network perspective. *Economic Theory*.

Mas-Colell, A., M. D. Whinston, and J. R. Green (1995). *Microeconomic Theory*. Oxford University Press.

Megow, N. and A. S. Schulz (2004). On-line scheduling to minimize average completion time revisited. *Operations Research Letters 32*, 485–490.

Megow, N., M. Uetz, and T. Vredeveld (2006). Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research 31*(3), 513–525.

Milgrom, P. (2004). *Putting Auction Theory to Work*. New York: Cambridge University Press.

Milgrom, P. and I. Segal (2002). Envelope theorems for arbitrary choice sets. *Econometrica 70*(2), 583–601.

Mitra, M. (2001). Mechanism design in queueing problems. *Economic Theory 17*(2), 277–305.

Mitra, M. (2005). Incomplete information and multiple machine queueing problems. *European Journal of Operational Research 165*(1), 251–266.

Monderer, D. (2007). Monotonicity and implementability. Discussion paper, Technion, Haifa.

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

Moulin, H. (2007). On scheduling fees to prevent merging, splitting, and transferring of jobs. *Mathematics of Operations Research 32*, 266–283.

Müller, R., A. Perea, and S. Wolf (2007). Weak monotonicity and Bayes-Nash incentive compatibility. *Games and Economic Behavior 61*(2), 344–358.

Munkres, J. R. (2000). *Topology* (2nd ed.). Prentice Hall.

Myerson, R. (1981). Optimal auction design. *Mathematics of Operations Research 6*(1), 58–73.

Myerson, R. (1991). *Game Theory: Analysis of Conflict*. Harvard University Press.

Nisan, N. (2007). Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge University Press.

Nisan, N. and A. Ronen (2000). Computationally feasible VCG mechanisms. In *Proc. 2nd ACM Conference on Electronic Commerce*, pp. 242–252.

Nisan, N. and A. Ronen (2001). Algorithmic mechanism design. *Games and Economic Behavior 35*, 166–196.

Owen, G. (1995). *Game theory* (Third ed.). San Diego, CA: Academic Press Inc.

Papadimitriou, C. (1994). *Computational Complexity*. Reading(MA): Addison-Wesley.

Parkes, D. C. (1999). *i*Bundle: An efficient ascending price bundle auction. In *Proc. 1st ACM Conf. on Electronic Commerce (EC-99)*, pp. 148–157.

Parkes, D. C. and L. H. Ungar (2000). Iterative combinatorial auctions: Theory and practice. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pp. 74–81.

Porter, R. (2004). Mechanism design for online real-time scheduling. In J. S. Breese, J. Feigenbaum, and M. I. Seltzer (Eds.), *Proc. 5th ACM Conference on Electronic Commerce*, pp. 61–70. ACM.

Pruhs, K., J. Sgall, and E. Torng (2004). Online scheduling. In J. Y.-T. Leung (Ed.), *Handbook of Scheduling*, Chapter 15. CRC Press LLC.

Roberts, K. (1979). The characterization of implementable choice rules. In J.-J. Laffont (Ed.), *Aggregation and Revelation of Preferences*. North Holland Publishing Company.

Rochet, J.-C. (1987). A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics 16*(2), 191–200.

Ronen, A. (2006). Incentive compatibility in computationally feasible combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg (Eds.), *Combinatorial Auctions*, pp. 369–394. MIT Press.

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

Roth, A., M. A. O. Sotomayor, and A. Chesher (Eds.) (1990). *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Cambridge University Press.

Roughgarden, T. (2004). Stackelberg scheduling strategies. *SIAM Journal on Computing 33*(2), 332–350.

Saks, M. and L. Yu (2005). Weak monotonicity suffices for truthfulness on convex domains. In *Proc. 6th ACM conference on Electronic Commerce*, pp. 286 – 293.

Sandholm, T. (2003). Automated mechanism design: A new application area for search algorithms. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 19–36.

Schrijver, A. (2003). *Combinatorial Optimization*, Volume A. Springer.

Schuurman, P. and T. Vredeveld (2007). Performance guarantees of local search for multiprocessor scheduling. *Informs Journal on Computing 19*(1), 52–63.

Smith, W. (1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly 3*, 59–66.

Suijs, J. (1996). On incentive compatibility and budget balancedness in public decision making. *Economic Design 2*, 193–209.

Vestjens, A. P. A. (1997). *On-line Machine Scheduling*. Ph. D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Vickrey, W. (1961). Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance 16*, 8–37.

Wellman, M. P., W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason (2001). Auction protocols for decentralized scheduling. *Games and Economic Behavior 35*(1-2), 271–303.

Subject Index

Nederlandse samenvatting

Grafen, Mechanismen en Roostering. Traditionele optimalisatie houdt zich bezig met de efficiënte selectie van een optimale oplossing voor een minimalisatie- of maximalisatieprobleem. Normaliter wordt er aangenomen dat er een centrale planner is, die volledige toegang heeft tot het model en de daarvoor relevante data. Bijvoorbeeld, deze planner moet in een productieproces een aantal taken aan verschillende machines toewijzen, op een dusdanige manier dat de in totaal benodigde tijd om alle opgaven af te ronden geminimaliseerd wordt. Of hij moet de beste manier bepalen om een aantal goederen van verschillende locaties naar verschillende eindbestemmingen in een stratennetwerk te vervoeren. In tegenstelling tot de centrale planning zijn er in de realiteit meestal verschillende agenten, die mogelijk over privé informatie beschikken, en deze agenten moeten samenwerken om een oplossing voor het probleem te bereiken. Dat gebeurt bijvoorbeeld op het internet, waar verschillende onafhankelijke en egoïstische gebruikers hun datapakketten over de communicatieverbindingen van het netwerk verzenden. Of in het dagelijkse verkeer, als een bestuurder probeert om de snelste route voor zichzelf te vinden en geen rekening houdt met het effect op de andere bestuurders. In zulke situaties zijn agenten gestimuleerd om zich strategisch te gedragen. Dit strategische gedrag kan tot niet-optimale oplossingen leiden. De analyse van dit soort situaties vraagt niet alleen om technieken uit de klassieke optimalisatie maar ook om technieken uit de speltheorie en de economische theorie.

Dit proefschrift levert verscheidene bijdragen aan het onderzoeksgebied in de overlapping van optimalisatie en speltheorie / economische theorie. In deel I bestuderen we klassieke vraagstukken in mechanism design door gebruik te maken van grafen. Hoofdstuk 2 bevat een nieuwe karakterisering van revenue equivalence. Om duidelijk te maken wat revenue equivalence is, bekijken we een veiling-situatie, waar één enkel goed moet worden verkocht aan één persoon van een groep van bieders.

Elke bieder heeft persoonlijke informatie over hoeveel waarde hijzelf aan het goed hecht, maar is onzeker over de waardering van de andere bieders voor het goed. Laten we aannemen dat het doel is om het goed te verkopen aan de bieder die het het meeste waardeert, maar tegelijkertijd een prijs te eisen, die voor geen bieder motivatie geeft om iets anders dan zijn ware waardering voor het goed te bieden. Dan is de verwachte opbrengst voor de veilingmeester altijd hetzelfde, onafhankelijk van de precieze opzet van de veiling (Myerson 1981). Dit feit wordt revenue equivalence genoemd. Resultaten over revenue equivalence kunnen in veel algemenere situaties worden bekeken. We bekijken zo'n algemene situatie en bewijzen een karakterisering van revenue equivalence met behulp van elementaire grafentheorie. Onze stelling kan worden toegepast in situaties waar alle bekende stellingen niet van toepassing zijn, en heeft het voordeel dat deze eenvoudig en elementair is.

In het volgende hoofdstuk, Hoofdstuk 3, bestuderen we optimale mechanismen voor een roosteringsprobleem. Laten we weer naar de veiling-situatie kijken om een idee te krijgen wat een optimaal mechanisme zou kunnen zijn. In de al eerder beschreven veiling van een enkel goed is een optimaal mechanisme één, dat de opbrengst voor de veilingmeester maximaliseert. Volgens Myerson (1981) geeft het optimale mechanisme het goed niet altijd aan de bieder met de hoogste waarde ervoor, maar de veilingmeester kan het bijvoorbeeld zelf houden als alle biedingen minder zijn dan een bepaalde gedefinieerde prijs. Ook optimale mechanismen zijn niet alleen interessant in veiling-situaties, maar in elk spel met prijzen. We bekijken een eenvoudig roosteringsprobleem en bepalen optimale mechanismen. We zien ook, dat zelfs het eenvoudigste roosteringsprobleem wezenlijk ingewikkelder is met betrekking tot het bepalen van optimale mechanismen dan het veiling-probleem.

Deel II gaat over de toepassing van mechanism design en speltheoretische concepten op roosteringsproblemen met meerdere machines. Hoofdstuk 4 bevat een overzicht van de meest interessante en meest recente onderwerpen op dit gebied. Het omvat de zogenoemde prijs van anarchie dan wel approximatie algoritmen voor optimalisatieproblemen in situaties met agenten.

In het laatste hoofdstuk, Hoofdstuk 5, bekijken we een online roosteringsprobleem vanuit het mechanism design oogpunt. De aanvullende complicatie in een online situatie is dat de taken, die op de machines moeten worden toegewezen, alleen te zijner tijd bekend worden. Beslissingen moeten al worden genomen voordat alle taken bekend zijn. Bovendien eisen we een gedecentraliseerde opzet. Dat betekent dat de taken hun machine zelf kunnen kiezen. We stellen een nieuw evenwichtsconcept voor, dat geschikt is voor deze situatie. Daarnaast presenteren we een mechanisme voor het gedecentraliseerde online roosteringsprobleem. Ons mechanisme levert een goede approximatie op voor een bepaalde doelfunctie.

Curriculum Vitae

Birgit Heydenreich was born on October 7, 1978 in Berlin, Germany. She obtained a university qualifying degree (*Abitur*) from *Johann-Gottfried-Herder-Oberschule Berlin* in June 1998. Subsequently, she enrolled in the program 'Mathematics with Computer Science and Business Administration' at the Department of Mathematics of *Technische Universität Berlin*. There, she obtained a master's degree (*Dipl.-Math.oec.*) in June 2004, marked with distinction. Next to her studies, she had part-time employments at Siemens, Ubis, Deutsche Bahn and as a teaching assistant at the Department of Mathematics at TU Berlin. The academic year 2000/01, she spent at the *University of Durham*, UK.

Birgit joined the OR group of *Maastricht University* as a Ph.D. candidate in October 2004. Under the supervision of Prof. Dr. Rudolf Müller and Prof. Dr. Marc Uetz, she studied mechanism design and its applications to scheduling models. The results of this research are presented at various conferences, in a number of journal articles, and in this thesis.

Acknowledgements

This thesis benefitted from the effort of a number of people, some of whom I would like to mention at this point. Most importantly, I would like to thank my supervisors Marc Uetz and Rudolf Müller for their continuous teaching, guidance and advice, and for the always open doors. Parts of the thesis are joint work with Rakesh Vohra and Debasis Mishra - I am grateful for their collaboration and what I have learned from it. I thank Hans Peters, Benny Moldovanu and Tjark Vredeveld for refereeing the thesis as members of the Degree Committee. Thanks to Joyce van Loon for the "language support" (Dutch, C++ and more).

Special thanks to all my colleagues from the Operations Research group, the Mathematical Economics group and others for the nice working atmosphere and for the professional as well as non-professional discussions and advice.

The research for this thesis was carried out from 2004 to 2008 at the Department of Quantitative Economics, Maastricht University. It was financed by the Netherlands Organisation for Scientific Research (NWO) under the project title "Local Decisions in Decentralised Planning Environments". The support of both Maastricht University and NWO is gratefully acknowledged.

Birgit Heydenreich
Maastricht, January 2009