Mechanism Design for Decentralized Online Machine Scheduling[1]

Birgit Heydenreich[2,4] Rudolf Müller[2] Marc Uetz[3]

April 16, 2008

Abstract

Traditional optimization models assume a central decision maker who optimizes a global system performance measure. However, problem data is often distributed among several agents, and agents take autonomous decisions. This gives incentives for strategic behavior of agents, possibly leading to sub-optimal system performance. Furthermore, in dynamic environments, machines are locally dispersed and administratively independent. Examples are found both in business and engineering applications. We investigate such issues for a parallel machine scheduling model where jobs arrive online over time. Instead of centrally assigning jobs to machines, each machine implements a local sequencing rule 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 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

[1]A preliminary version of this paper with parts of the results appeared in the conference proceedings SWAT 2006 (Heydenreich, Müller, and Uetz 2006).

[2]Maastricht University, Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands. Email: {b.heydenreich,r.muller}@ke.unimaas.nl

[3]University of Twente, Applied Mathematics, P.O. Box 217, 7500 AE Enschede, The Netherlands. Email: m.uetz@utwente.nl

[4]Supported by NWO grant 2004/03545/MaGW 'Local Decisions in Decentralised Planning Environments'.

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, called *agents*. 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 (2005) and Azar, Jain, and Mirrokni (2008). In this paper, 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 a model was lately also discussed by Azar, Jain, and Mirrokni (2008).

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 $\mathrm{P} \mid r_j \mid \sum w_j C_j$ 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.

Mechanism design in combination with the design of approximation algorithms for scheduling problems has been studied, e.g., by Nisan and Ronen (2001) and Archer and Tardos (2001). The models in these papers are such that the processing time of a job depends on the machine that processes the job (unrelated and related machine scheduling, respectively). Furthermore, the machines are the selfishly behaving parts of the system. Their private information is the time they need to process the jobs. Among other things, these papers present mechanisms where rational (selfish) behavior of agents yields equilibrium solutions that are only a constant factor away from the optimum. A scheduling model that comes closer to the model that we address has been studied by Porter (2004). He analyzes a single machine online scheduling problem where the jobs are the selfish agents of the system, and the private data of each job consists of a release date, its processing time, its weight, and a deadline. He, too, obtains a mechanism where selfish behavior of agents yields competitive equilibrium solutions. Also Hajiaghayi, Kleinberg, Mahdian, and Parkes (2005) derive mechanisms for an online single machine model with job agents, where jobs are available only within certain time windows.

Decentralization for scheduling models with job agents is regarded in the papers by Christodoulou, Koutsoupias, and Nanavati (2004), Immorlica, Li, Mirrokni, and Schulz (2005) and Azar, Jain, and Mirrokni (2008). Those papers analyze the price of anarchy for different local scheduling policies.

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 MinIncrease 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 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 $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$. 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 of the Paper. We formalize the model and introduce notation in Section 2. Especially, we define the notion of a decentralized online scheduling mechanism and the myopic best response equilibrium in that section. In Section 3 the Local Greedy Algorithm is defined. In Section 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. In Section 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 7.

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, e.g. 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 jobs' behavior results in an equilibrium where the social welfare is not too far from optimum.

Definition 2.1. *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 $\tilde{p}_j$.*[4]
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 2.2. *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.

[4] 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 2.3. *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 2.4. *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* s_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 $(s_1, \ldots, s_n)$ of strategies, one for each job. Given a mechanism, a strategy profile, and a realization of types t, we denote by $u_j(s,t)$ the (ex-post) utility that agent j receives.

Definition 2.5. *A strategy profile $s = (s_1, \ldots, s_n)$ is called a* dominant strategy equilibrium *if for all jobs $j \in J$, all types t of the jobs, all strategies $\tilde{s}_{-j}$ of the other jobs, and all strategies $\tilde{s}_j$ that j could play instead of s_j,*

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

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 2.6. *Given a decentralized, online scheduling mechanism as in Definition 2.1, a strategy profile s, 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 $\hat{u}_j(s,t) := -\hat{C}w_j - \hat{\pi}_j$ denotes j's tentative utility at time $\tilde{r}_j$.*

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

Definition 2.7. *A strategy profile $(s_1, \ldots, s_n)$ is called a* myopic best response equilibrium*, if for all jobs $j \in J$, all types t of the jobs, all strategies $\tilde{s}_{-j}$ of the other jobs and all strategies $\tilde{s}_j$ that j could play instead of s_j,*

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

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 2.1. *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 s_{-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[5]. 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 2.1. 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

[5]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.

sense of Definition 2.1. 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.

2.1 Critical jobs

For convenience of presentation, we make the following assumption for the main part of the paper. 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 5.1, 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.

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 are 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.

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 complection 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 results 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 2.2.

Theorem 4.1. *Regard any type vector t, any strategy profile s 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 4.1. *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 4.1 implies that a myopic agent should report its true weight. Lemma 4.1 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 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 4.2. *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 4.1(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 4.1(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 4.1(b), by choosing the machine that the Local Greedy Algorithm would have chosen. □

Theorem 4.3. *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 4.1, $\hat{u}_j$ on any machine is maximized by reporting $\tilde{w}_j = w_j$ for any $\tilde{r}_j$ and $\tilde{p}_j$. According to Theorem 4.2 and Lemma 4.1(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 4.1(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.[6]

Corollary 4.1. *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 4.1, 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 4.2 it can be shown that truthfully reporting the processing time maximizes j's tentative utility for any $\tilde{r}_j$ and truthful

[6] We add an extra proof, since the proof of Theorem 4.3 uses the detour via ex-post utilities, which is not possible if jobs are not compensated for delays.

$\tilde{w}_j = w_j$. 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 4.1). Furthermore, paying jobs for their delay results in a dominant strategy equilibrium in a restricted type space (Theorem 4.2).

5 Performance of the Mechanism

As shown in Section 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 that if all jobs are rational, then the Decentralized Local Greedy Mechanism is 3.281-competitive.

5.1 Handling Critical Jobs

Recall that from Section 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 4.1 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 4.2. 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.

5.2 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 5.1. *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; it is presented in the appendix.

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.

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 6.1. *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

$$(w^{(2)} - w^{(1)})(C^{(1)} - C^{(2)}) \geq 0. \tag{1}$$

Especially, if $w^{(1)} < w^{(2)}$ we must have $C^{(1)} \geq C^{(2)}$.[7]

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 5.1), 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.

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

[7] Condition (1) corresponds to the notion of weak 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).

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 (1) is violated. □

Remark 6.1. *Theorem 6.1 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.*

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 a $\alpha \in [0, 1]$, 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 6.2. *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*

$$\sigma_j^k = \sum_{\xi=k+1}^{MAX} \left[\varrho^\xi \left(C^\xi - C^{\xi-1}\right)\right]$$

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 σ has a dominant strategy equilibrium in which all jobs are truthful.

The payment scheme σ is under certain conditions related to the VCG-payment scheme. But especially in an online setting, this relationship does not hold; see Section 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 σ_j^k indeed make truth-telling a dominant strategy with respect to all three job characteristics.

Proof of Theorem 6.2. 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

$$\sigma_j^k = \int_0^{w_j} (C(x) - C(w_j))\ dx.$$

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.[8]

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.

□

[8] For a similar, but more intuitive proof for the induced single parameter problem we refer to Lavi (2007).

The payments σ_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 σ is not an online payment scheme and it does not satisfy the balanced budget condition. The described mechanism is heavily 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.

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 σ from Section 6.2 yields a mechanism in which truth-telling becomes a dominant strategy equilibrium. This can be proven analogously to Theorem 6.2. Again, in contrast to the payment scheme of the Decentralized Local Greedy mechanism, the payment scheme σ 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 σ coincide with the VCG mechanism: It can be verified that the payment that job j has to make according to σ 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.

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.

Acknowledgements

We thank the anonymous referees for very useful comments and suggestions.

References

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.

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.

Azar, Y., K. Jain, and V. Mirrokni (2008). Optimal coordination mechanisms for unrelated machine scheduling. In *Proc. Symposium on Discrete Algorithms (SODA'08)*. To appear.

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.

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.

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.

Demange, G., D. Gale, and M. Sotomayor (1986). Multi-item auctions. *The Journal of Political Economy 94*(4), 863–872.

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.

Gallien, J. and L. Wein (2005). A smart market for industrial procurement with capacity constraints. *Management Science 51*(1), 76–91.

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. 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.

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.

Heydenreich, B., R. Müller, and M. Uetz (2006). Decentralization and mechanism design for online machine scheduling. In L. Arge and R. Freivalds (Eds.), *Algorithm Theory - SWAT 2006*, Volume 4059 of *Lecture Notes in Computer Science*, pp. 136–147. Springer.

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.

Immorlica, N., L. Li, V. S. Mirrokni, and A. Schulz (2005). Coordination mechanisms for selfish scheduling. In X. Deng and Y. Ye (Eds.), *Internet and Network Economics*, Volume 3828 of *Lecture Notes in Computer Science*, pp. 55–69. Springer.

Lavi, R. (2007). Computationally efficient approximation mechanisms. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge University Press.

Lenstra, J. K., A. H. G. Rinnooy Kan, and P. Brucker (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343–362.

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.

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 (2001). Algorithmic mechanism design. *Games and Economic Behavior 35*, 166–196.

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.

Smith, W. (1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly 3*, 59–66.

Vestjens, A. P. A. (1997). *On-line Machine Scheduling*. Ph. D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

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.

Appendix: Proof of Theorem 5.1

Theorem 5.1. *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 $\varrho = 3.281$.*

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 $b_i(r'_j) \le r'_j/\alpha$. Therefore we get, for any j,

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} -\hat{u}_j(i) &= w_j r'_j + w_j \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j \sum_{i=1}^{m} \sum_{\substack{k\in H(j)\\ k\to i\\ k<j\\ S_k\ge r'_j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{i=1}^{m} \sum_{\substack{k\in L(j)\\ k\to i\\ k<j\\ S_k> r'_j}} \frac{w_k}{m} \\
&= w_j r'_j + w_j \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j \sum_{\substack{k\in H(j)\\ k<j\\ S_k\ge r'_j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{\substack{k\in L(j)\\ k<j\\ S_k> r'_j}} \frac{w_k}{m} \\
&\le w_j r'_j + w_j \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m} \\
&\le w_j r'_j + w_j \frac{r'_j}{\alpha} + w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + w_j p_j + p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m} .
\end{aligned}$$

Thus,

$$Z \le \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + \sum_{j\in J} w_j p_j + \sum_{j\in J} p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m}$$

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,

$$\begin{aligned}
Z &\le \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + \sum_{j\in J} w_j p_j + \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k>j}} \frac{p_k}{m} \\
&= \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \sum_{j\in J} w_j \sum_{k\in H(j)} \frac{p_k}{m} + \frac{m-1}{m}\sum_{j\in J} w_j p_j .
\end{aligned}$$

Now, we apply a lower bound on the optimal off-line schedule by Eastman, Even, and Isaacs (1964, Thm. 1), namely

$$Z^{OPT} \ge \sum_{j\in J} w_j \sum_{k\in H(j)} \frac{p_k}{m} + \frac{m-1}{2m}\sum_{j\in J} w_j p_j ,$$

yielding:

$$\begin{aligned}
Z &\le Z^{OPT} + \sum_{j\in J} w_j(1+\frac{1}{\alpha})r'_j + \frac{m-1}{2m}\sum_{j\in J} w_j p_j \\
&\le Z^{OPT} + \sum_{j\in J} w_j(1+\frac{1}{\alpha})(r_j+\alpha p_j) + \frac{m-1}{2m}\sum_{j\in J} w_j p_j \\
&= Z^{OPT} + \sum_{j\in J} w_j \left[(1+\frac{1}{\alpha})r_j + (1+\alpha+\frac{m-1}{2m})p_j\right],
\end{aligned}$$

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 $\alpha = (\sqrt{17m^2 - 2m + 1} - m + 1)/(4m)$. □