Decentralization and Mechanism Design for Online Machine Scheduling[1]

Birgit Heydenreich[2] and Rudolf Müller and Marc Uetz

Abstract

We study the online version of the classical parallel machine scheduling problem to minimize the total weighted completion time from a new perspective: We assume that the data of each job, namely its release date r_j, its processing time p_j and its weight w_j is only known to the job itself, but not to the system. Furthermore, we assume a decentralized setting where jobs choose the machine on which they want to be processed themselves. We study this problem from the perspective of algorithmic mechanism design. We introduce the concept of a myopic best response equilibrium, a concept weaker than the dominant strategy equilibrium, but appropriate for online problems. We present a polynomial time, online scheduling mechanism that, assuming rational behavior of jobs, results in an equilibrium schedule that is 3.281-competitive. The mechanism deploys an online payment scheme that induces rational jobs to truthfully report their private data. We also show that the underlying local scheduling policy cannot be extended to a mechanism where truthful reports constitute a dominant strategy equilibrium.

1 Introduction

We study the online version of the classical parallel machine scheduling problem to minimize the total weighted completion time[3] from a new perspective: We assume a strategic setting, where the data of each job, namely its release date r_j, its processing time p_j and its weight w_j is only known to the job itself, but not to the system. Any job j is interested in being finished as early as possible, and the weight w_j represents its indifference cost for spending one additional unit of time waiting. The time when job j is finished is called its completion time C_j. 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. Furthermore, we assume a restricted communication paradigm, referred 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 coordination authority hosting all the data of the problem. This

[1]A version of this paper has been published in the Proceedings of the Scandinavian Workshop on Algorithm Theory (SWAT) 2006, LNCS 4059, pp. 136-147, Springer

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

[3]The problem is $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$ in the notation of Graham et al. [1].

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.

The problem $P \mid r_j \mid \sum w_j C_j$ 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 [2]. Second, no online algorithm for the single machine problem can be better than 2-competitive [3] regardless of the question whether or not P=NP, and lower bounds exist for parallel machines, too [4]. The best possible algorithm for the single machine case is 2-competitive [5]. For the parallel machine setting, the currently best known online algorithm is 2.61-competitive [6].

In the strategic setting, selfish agents trying to maximize their own benefit can do so by reporting strategically about their private information, thus manipulating the resulting schedule. In the model we propose, a job can report an arbitrary weight, an elongated processing time (e.g. by adding unnecessary work), and it can artificially delay its true release date r_j. We do not allow a job to report a processing time shorter than 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, as we assume that any job j comes into existence only at its release date r_j, it obviously does not make sense that a job reports a release date smaller than the true value r_j.

Our goal is to set up a mechanism that yields a reasonable overall performance with respect to the objective function $\sum w_j C_j$. To that end, the mechanism needs to motivate the jobs to reveal their private information truthfully. In addition, as we require decentralization, each machine must be equipped with a local sequencing policy that is publicly known, and jobs must be induced to select the machines in such a way that $\sum w_j C_j$ is not too large. Known algorithms with the best performance ratio, e.g. [6, 7], crucially require central coordination to distribute jobs over machines. An approach by Megow et al. [8], developed for an online setting with release dates and stochastic job durations, however, turns out to be appropriate for being adopted to the decentralized, strategic setting.

Related Work and Contribution. Mechanism design in combination with the design of approximation algorithms for scheduling problems has been studied, e.g., by Nisan and Ronen [10], Archer and Tardos [11], and Kovacs [12]. In those papers, not the jobs but the machines are the selfishly behaving parts of the system, and their private information is the time they need to process the jobs. A scheduling model where the jobs are the selfish agents of the system has been studied by Porter [13]. He addresses a single machine scheduling problem, where the private data of each job consists of a release date, its processing time, its weight, and a deadline. In all mentioned papers, the only action of an agent (machine or job, respectively) is to reveal its private data; the resulting mechanisms are also called direct revelation mechanisms. The mechanism suggested in this paper is not a direct revelation mechanism, since in addition to the revelation of private data, jobs must select the machine

to be processed on.

In the algorithm of Megow et al. [8], jobs are locally sequenced according to an online variant of the well known WSPT rule [9], and arriving jobs are assigned to machines in order to minimize an expression that approximates the (expected) increase of the objective value. This algorithm achieves a performance ratio of 3.281. The mechanism we propose develops their idea further. We present a polynomial time, decentralized online mechanism, called Decentralized LocalGreedy Mechanism. Thereby we provide also a new algorithm for the non-strategic, centralized setting, inspired by the MinIncrease Algorithm of [8], but improving upon the latter in terms of simplicity. We show that the Decentralized LocalGreedy Mechanism is 3.281-competitive as well. The currently best known bound for the non-strategic setting is 2.61 [6].

As usual in mechanism design, the Decentralized LocalGreedy 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 the jobs to select 'the right' machines. Intuitively, the mechanism uses the payments to mimic a corresponding LocalGreedy online algorithm in the classical (non-strategic, centralized) parallel machine setting $\mathrm{P} \mid r_j \mid \sum w_j C_j$. Moreover, 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). In addition, we show that there does not exist a payment scheme extending the allocation rule of the Decentralized LocalGreedy Mechanism to a mechanism where truthful reporting of all private information is a dominant strategy equilibrium.

This extended abstract is organized as follows. We formalize the model and introduce the required notation in Section 2. In Section 3 the LocalGreedy algorithm is defined. In Section 4, this algorithm is adapted to the strategic setting and extended by a payment scheme, yielding the Decentralized LocalGreedy Mechanism. Moreover, our main results are presented in that section. We analyze the performance of the mechanism in Section 5, mention a negative result in Section 6, and 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. 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 pro-

cess 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.

Definition 1. *A* decentralized online scheduling mechanism *is a procedure that works as follows.*

1. *Each job j has a release date r_j, but may pretend to come into existence at any time $\tilde{r}_j \geq r_j$. At that chosen release date, the job communicates to every machine reports $\tilde{w}_j$ and $\tilde{p}_j$ (which may differ from the true w_j and p_j)*[4].
2. *Machines communicate on the basis of that information a (tentative) completion time $\hat{C}_j$ and a (tentative) payment $\hat{\pi}_j$ to the job. This information is tentative due to the online situation. The values $\hat{C}_j$ and $\hat{\pi}_j$ can only change if later another job chooses the same machine.*
3. *Based on this response, the job chooses a machine. This choice is binding. The entire communication takes place at one point in time, namely $\tilde{r}_j$.*
4. *There is no communication between machines or between jobs.*
5. *Depending on later arrivals of jobs, machines may revise $\hat{C}_j$ and $\hat{\pi}_j$. Eventually, the mechanism leads to an (ex-post) completion time C_j and an (ex-post) payment π_j of each job.*

Hereby, we assume that jobs with equal reported release date arrive in some given order and communicate to machines in that order. Next, we define an online property of the payment scheme and the performance ratio of an online mechanism.

Definition 2. *If in a decentralized online scheduling mechanism 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 of the mechanism an* online payment scheme.

Definition 3. *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 an 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*

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

[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.

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

We assume that each job j prefers a lower completion time to a higher one and model this by the valuation $v_j(C_j \,|\, t_j) = -w_j\, C_j$. We 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 $-w_j\, C_j - \pi_j$. In this model, the utility u_j is always negative. Therefore, we assume that a job has a constant and sufficiently large utility for 'being processed at all'. Note that the total social welfare is maximized whenever the weighted sum of completion times $\sum_{j \in J} w_j\, C_j$ is minimum, which is independent of whether we do or do not carry these constants with us.

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 utility that agent j receives.

Definition 4. *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,*

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

We could simplify notation if we restricted ourselves to *direct revelation mechanisms*, that is mechanisms in which the only action of a job is to report its type. However, a decentralized online scheduling mechanism requires that jobs decide themselves on which machine they are scheduled. Since these decisions are likely to influence the utility of the jobs, they have to be modelled as actions in the game. Therefore, it is not sufficient to restrict oneself to direct revelation mechanisms.

We will see that the mechanism proposed in this paper does not have a dominant strategy equilibrium, whatever modification we might apply to the payment scheme. However, a weaker equilibrium concept applies, which we define next. That definition uses the concept of the tentative utility, i.e., the utility a job would have if it was the last to be accepted on its machine.

Definition 5. *Given a decentralized, online scheduling mechanism as in Definition 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 6. *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,*

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

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 [5, 7]. 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 such that also critical jobs can be dealt with. This slight adaption leads to a constant performance bound while preserving all desired properties concerning the strategic behaviour of the jobs.

3 The LocalGreedy Algorithm

We next formulate an online scheduling algorithm that is inspired by the Min-Increase Algorithm from Megow et al. [8]. 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. In the next section, we discuss how to turn this algorithm into a mechanism for the strategic, decentralized setting that we aim at.

The idea of the algorithm is that each machine uses (an online version of) the well known WSPT rule [9] locally. 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 t in time, machine i might be busy processing a job. We let $b_i(t)$ denote the remaining processing time of that job at time t, i.e., at time t machine i will be blocked during $b_i(t)$ units of time for new jobs. If machine i is idle at time t, we let $b_i(t) = 0$.

Algorithm 1: LocalGreedy algorithm

Local Sequencing Policy:
Whenever a machine becomes idle, it starts processing the job with highest (WSPT) priority among all jobs assigned to it.
Assignment:
(1) At time r_j job j arrives; the immediate increase of the objective $\sum w_j C_j$, given that j is assigned to machine i, is

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

(2) Job j is assigned to machine $i_j \in \operatorname{argmin}_{i \in M} z(j,i)$ with minimum index.

Clearly, the LocalGreedy algorithm still makes use of central coordination in Step (2). In the sequel we will introduce payments that allow to transform the algorithm into a decentralized online scheduling mechanism.

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 receives when being displaced by other jobs. We will prove that utility maximizing jobs have an incentive to report truthfully and to choose the machine that the LocalGreedy Algorithm would have selected, too. Furthermore, the WSPT rule can be run locally on every machine and does not require communication between the machines. We will see in the next section that this yields a constant-factor approximation of the off-line optimum, given that the jobs behave rationally. The algorithm including the payments is displayed below as the Decentralized LocalGreedy Mechanism. Let 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.

Algorithm 2: DecentralizedLocalGreedy Mechanism

Local Sequencing Policy:
Whenever a machine becomes idle, it starts processing 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 a weight $\tilde{w}_j$ and a processing time $\tilde{p}_j$ to all machines.

(2) Every machine i computes

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

and 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 with $k \in \tilde{L}(j)$, $k \to i_j$, $k < j$, $S_k > \tilde{r}_j$ increases by $\tilde{p}_j$ due to j's presence. As compensation, k receives a payment of $\tilde{w}_k\tilde{p}_j$.

The DecentralizedLocalGreedy Mechanism together with the stated 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. Notice that the payments are made online in the sense of Definition 2.

Theorem 7. *Regard any type vector t, any strategy profile s and any job j such that j reports $(\tilde{r}_j, \tilde{p}_j, \tilde{w}_j)$ and chooses machine $\tilde{m} \in M$. Then changing the report to $(\tilde{r}_j, \tilde{p}_j, w_j)$ and choosing a machine that maximizes its tentative utility at time $\tilde{r}_j$ does not decrease j's tentative utility under the* Decentralized LocalGreedy *Mechanism.*

Proof. We only give the idea here. For the single machine case, an arriving job j gains tentative utility $\tilde{p}_k w_j - \tilde{p}_j\tilde{w}_k$ from displacing an already present job k. WSPT assigns j in front of k if and only if $\tilde{p}_k\tilde{w}_j - \tilde{p}_j\tilde{w}_k > 0$. Thus, $\tilde{w}_j = w_j$ maximizes j's tentative utility. For $m > 1$, the theorem follows from the fact that j can select a machine itself. □

Lemma 8. *Consider any job $j \in J$. Then, under the* Decentralized LocalGreedy *Mechanism, 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 LocalGreedy *Algorithm would have selected maximizes j's ex-post utility.*

Proof. See full version of the paper. □

Theorem 9. *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 under the* Decentralized LocalGreedy *Mechanism.*

Proof. Let us start with $m = 1$. Suppose $\tilde{w}_j = w_j$, fix any pretended release date $\tilde{r}_j$ 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 8(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 false release date $\tilde{r}_j > r_j$. 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 immediate 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 from those jobs 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 immediate utility at time r_j will be at least as large as its immediate utility at time $\tilde{r}_j$. Therefore, j maximizes its immediate utility at time $\tilde{r}_j$ by choosing $\tilde{r}_j = r_j$. As $\tilde{w}_j = w_j$, it follows from Lemma 8(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 immediate 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 immediate 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 8(b), by choosing a machine that minimizes the immediate increase in the objective function. □

Theorem 10. *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 under the* DECENTRALIZED LOCALGREEDY *Mechanism.*

Proof. Regard job j. According to the proof of Theorem 7, $\hat{u}_j$ on any machine is maximized by reporting $\tilde{w}_j = w_j$ for any $\tilde{r}_j$ and $\tilde{p}_j$. According to Theorem 9 and Lemma 8(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 8(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. □

Given the restricted communication paradigm, jobs do not know at their arrival which jobs are already queuing at the machines and what reports the already present jobs have made. Therefore it is easy to see that for any non-truthful report of an arriving job about its weight, instances can be constructed in which this report yields a strictly lower utility for the job than a truthful report would have given. With arguments similar to those in the proof of Theorem 9, the same holds for false reports about the processing time and the release date.

Note that in order to obtain the myopic best response equilibrium (Theorem 10), payments paid by an arriving job j need not necessarily be given to the jobs delayed by j. But by doing so, the resulting ex-post payments result in a balanced budget and the tentative utility at arrival is preserved and equals the ex-post utility of every job (Lemma 7). Furthermore, paying jobs for their delay results in a dominant strategy equilibrium in a restricted type space (Theorem 9).

5 Performance of the Mechanism

As shown in Section 4, jobs have a motivation to report truthfully about their data: According to Theorem 7, it is a myopic best response for a job j to report the true weight w_j, no matter what the other jobs do and no matter which $\tilde{p}_j$ and $\tilde{r}_j$ are reported by j itself. Given a true report of w_j, it was proven in Theorem 9 that reporting the true processing time and release date as well as choosing a machine maximizing the tentative utility at arrival maximizes the job's ex-post utility. 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 LOCALGREEDY Mechanism is 3.281-competitive.

5.1 Handling Critical Jobs

Recall that from Section 2.1 on, we assumed that no critical jobs exist, i.e. we defined the DECENTRALIZED LOCALGREEDY 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 DecentralizedLocalGreedy Mechanism as stated above does not yet yield a constant performance ratio; simple examples can be constructed in the same flavor as in [7]. In fact, it is well known that early arriving jobs with large processing times have to be delayed [5, 7, 8]. In order to achieve a constant performance ratio, we also adopt this idea and use modified release dates as [7, 8]. 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 7 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 9. 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 Performance 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 $\sum w_j\, C_j$. We derive a constant performance ratio for the Decentralized LocalGreedy Mechanism by the following theorem:

Theorem 11. *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 LocalGreedy *Mechanism is ϱ-competitive, with $\varrho = 3.281$.*

The proof of the theorem partly follows the lines of the corresponding proof of Megow et al. [8]. But the distribution of jobs over machines in their algorithm differs from the decentralized distribution in the Decentralized LocalGreedy Mechanism when rational jobs are assumed. Therefore, our result is not implied by the result of Megow et al. [8] and it is necessary to give a proof here.

Proof. A rational job communicates to the machines at time $r'_j = \max\{r_j, \alpha p_j\}$ 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 [8], to obtain:

$$Z \leq \sum_{i \in J} \frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i) \,.$$

The remainder of the proof utilizes the definitions of $\hat{u}_j(i)$ and particulary the fact that, upon arrival of job j on any of the machines i (at time r'_j), machine i is blocked for time $b_i(r'_j)$, which is upper bounded by r'_j/α. This upper bound is machine-independent, and follows from the definition of r'_j, since any job k in process at time r'_j fulfills $\alpha p_k \leq r'_k \leq r'_j$. Furthermore, the proof utilizes a lower bound on any (off-line) optimum schedule from Eastman et al. [14, Thm. 1]. For details, we refer to the full version of the paper. The resulting performance bound 3.281 is identical to the one of [8] (for deterministic processing times), when α is $(\sqrt{17m^2 - 2m + 1} - m + 1)/(4m)$. □

6 Negative Result

Theorem 12. *There does not exist a payment scheme that extends the* LocalGreedy *algorithm to a truthful mechanism. Therefore, it is not possible to turn the* Decentralized LocalGreedy *Mechanism into a mechanism with a dominant strategy equilibrium in which all jobs report truthfully by only modifying the payment scheme.*

Proof. If the Decentralized LocalGreedy Mechanism can be turned into a truthful mechanism by only modifying the payment scheme, then the LocalGreedy algorithm can be completed by a payment scheme to a truthful mechanism. Furthermore, we can show that a necessary condition for truthfulness, called weak monotonicity, is not satisfied by the LocalGreedy algorithm. Weak monotonicity has been introduced in [15]. □

7 Discussion

It would be interesting to find a constant competitive decentralized online scheduling mechanism such that there is a *dominant strategy equilibrium* in which the jobs report all data truthfully. As we have seen in Section 6, the LocalGreedy Algorithm cannot be extended by a payment scheme such that the resulting mechanism has the described properties. Furthermore, recall that the currently best known performance bound for the non-strategic, centralized setting is 2.61 [6]. This algorithm crucially requires a centralized distribution of jobs over machines, and therefore does not seem to be suited for decentralization. Nevertheless, it remains an interesting question to identify general rules for the transformation of centralized algorithms to decentralized mechanisms.

References

[1] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: A survey. Ann. Discr. Math. **5** (1979), 287–326

[2] Lenstra, J.K., Rinnoy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. Ann. of Discr. Math. **1** (1977), 343–362

[3] Hoogeveen, J.A., Vestjens, A.P.A.: Optimal on-line algorithms for single machine scheduling. In: Cunningham, W.H., McCormick, S.T., Queyranne, M., eds.: IPCO 1996, LNCS 1084 (1996), 404-414

[4] Vestjens, A.P.A.: On-line Machine Scheduling. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (1997)

[5] Anderson, E.J., Potts, C.N.: Online scheduling of a single machine to minimize total weighted completion time. Math. Oper. Res. **29** (2004), 686-697

[6] Correa, J.R., Wagner, M.R.: LP-based online scheduling: from single to parallel machines. In: Jünger, M., Kaibel, V., eds.: IPCO 2005. LNCS 3509 (2005), 196-209

[7] Megow, N., Schulz, A.S.: On-line scheduling to minimize average completion time revisited. Oper. Res. Letters **32** (2004), 485-490

[8] Megow, N., Uetz, M., Vredeveld, T.: Models and algorithms for stochastic online scheduling. Math. Oper. Res., to appear.

[9] Smith, W.: Various optimizers for single stage production. Nav. Res. Log. Quarterly **3** (1956), 59-66

[10] Nisan, N., Ronen, A.: Algorithmic mechanism design. Games and Economic Behavior **35** (2001), 166-196

[11] Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: Proc. 42nd FOCS. IEEE Computer Society (2001), 482-491

[12] Kovacs, A.: Fast monotone 3-approximation algorithm for scheduling related machines. In: Brodal, G.S., Leonardi, S., eds.: ESA 2005. LNCS 3669 (2005), 616-627

[13] Porter, R.: Mechanism design for online real-time scheduling. Proc. 5th ACM Conf. Electronic Commerce, ACM Press (2004), 61-70

[14] Eastman, W.L., Even, S., Isaacs, I.M.: Bounds for the optimal scheduling of n jobs on m processors. Management Science **11** (1964), 268–279

[15] S. Bikhchandani, S. Chatterjee, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant strategy implementation,. *Econometrica*, 74(4):1109–1132, 2006.

Birgit Heydenreich and Rudolf Müller and Marc Uetz
Maastricht University,
Quantitative Economics,
P.O.Box 616,
6200 MD Maastricht,
The Netherlands.
Email: {b.heydenreich,r.muller,m.uetz}@ke.unimaas.nl