Decentralization and Mechanism Design for Online Machine Scheduling †

Birgit Heydenreich* Rudolf Müller* Marc Uetz*

February 9, 2006

Abstract

We study the online version of the classical parallel machine scheduling problem to minimize the total weighted completion time – $\mathrm{P}\,|\,r_j\,|\,\sum w_j C_j$ in the classical notation of [5] – 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. Furthermore, we assume a decentralized setting, where jobs choose the machine on which they want to be processed themselves. We study this setting from the perspective of algorithmic mechanism design and present a polynomial time decentralized online scheduling mechanism that induces rational jobs to select their machine in such a way that the resulting schedule is 3.281-competitive. The mechanism deploys an online payment scheme that induces rational jobs to truthfully report about their private data: with respect to release dates and processing times, truthfully reporting is a dominant strategy equilibrium, whereas truthfully reporting the weights is a myopic best response equilibrium. We show that the local scheduling policy used in the mechanism cannot be extended to a mechanism where truthful reports with respect to weights 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 – $\mathrm{P}\,|\,r_j\,|\,\sum w_j\, C_j$ in the notation of Graham et al. [5] – 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. 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 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.

*Maastricht University, Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: {b.heydenreich,r.muller,m.uetz}@ke.unimaas.nl

†This research was supported by NWO grant 2004/03545/MaGW 'Local Decisions in Decentralised Planning Environments'.

The problem $P \,|\, r_j \,|\, \sum w_j\, C_j$ is well-understood in the non-strategic setting with centralized coordination. First, scheduling to minimize the weighted sum of completion times with release dates, is NP-hard, even in the off-line case [9]. Second, no online algorithm for the single machine problem can be better than 2-competitive [6] regardless of the question whether or not P=NP, and lower bounds exist for parallel machines, too [15]. The best possible algorithm for the single machine case is 2-competitive [1]. For the parallel machine setting, the currently best known online algorithm is 2.61-competitive [3].

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 outcome. 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 competitive ratio, e.g. [3, 10], crucially require central coordination to distribute jobs over machines. An approach by Megow et al. [11], developed for an online setting with release dates and stochastic job durations, however, turns out to be quite appropriate for being adopted to the decentralized, strategic setting. In their paper, jobs are locally sequenced according to an online variant of the well known WSPT rule [14], 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 competitive ratio of 3.281. The mechanism we propose develops their idea further: by introducing appropriate payments, jobs are induced to not only truthfully report their private data, but also to select those machines that minimize the immediate increase of the objective value. This way, we obtain the same competitive ratio of 3.281, but in the strategic, decentralized setting.

Related Work. Mechanism design in combination with the design of approximation algorithms for scheduling problems has been studied, e.g., by Nisan and Ronen [12], Archer and Tardos [2], and Kovacs [7]. In those papers, not the jobs but the machines are the selfishly behaving parts of the system, and the private information is the processing speed of the machine. A scheduling model where the jobs are the selfish agents of the system has been studied by Porter [13]. The paper addresses preemptive scheduling on a single machine, where the private data of each job consists of a release date, its processing time, its weight, and a deadline. The objective function is the sum of the weights of all jobs that are completed by their deadline. Porter proposes a $((1+\sqrt{k})^2+1)$-competitive incentive compatible mechanism and proves that this is best possible for deterministic mechanisms that satisfy certain conditions. Here, $k = \max_{i,j}(w_j p_i)/(p_j w_i)$ is the maximum ratio of the value densities of two jobs, where the value density of job j is defined as weight divided by processing time w_j/p_j.

Contribution. We present a polynomial time, decentralized online mechanism, called Decentralized LocalGreedy Mechanism, for the strategic and decentralized setting of the online scheduling problem $P \,|\, r_j \,|\, \sum w_j\, C_j$. Thereby, we provide also a new algorithm for the non-strategic, centralized setting, that is inspired by the MinIncrease Algorithm of [11], but improves on the latter in terms of simplicity. We show that the Decentralized LocalGreedy Mechanism is 3.281-competitive which coincides with the bound that is known for the non-strategic, centralized setting in [10, 11].

As usual in mechanism design, our 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. The payments in our mechanism 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 $\mathrm{P} \,|\, r_j \,|\, \sum w_j \, C_j$. We show that the payments result in a balanced budget. Moreover, the payments induce rational jobs to truthfully report about their private data. With respect to release dates and processing times, truthfulness is a dominant strategy equilibrium. With respect to the weights, truthful reports are myopic best responses (in a sense to be made precise later). 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.

The paper 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 resulting mechanism in Section 5, prove the mentioned negative result in Section 6 and conclude with a short discussion in Section 7.

2 Model and Notation

The considered problem is online scheduling with non-trivial release dates on parallel machines, 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 $M = \{1, \ldots, m\}$. 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, which we denote 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 its true type).*
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. $\hat{C}_j$ 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 at all between machines or between jobs.*

5. *Depending on later arrivals of jobs, machines may revise $\hat{C}_j$ and $\hat{\pi}_j$. Altogether, the interaction between jobs and machines as well as their local decision making leads to an (ex-post) completion time C_j and an (ex-post) payment π_j of each job.*

Next, we define a property of the payments in a decentralized online scheduling 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*.*

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*, $u_j(C_j, \pi_j \,|\, t_j) = v_j(C_j \,|\, t_j) - \pi_j$, which is in our case equal to $-w_j\, C_j - \pi_j$. In this model, u_j is always negative. Therefore, we assume that a job has a constant and sufficiently large utility for 'being processed at all'. Carrying this over to the notation would add a constant to u_j. Since this does not change the jobs' behavior when maximizing their utility, we will omit the constant and continue working with u_j. For the sake of simplicity, we drop the dependence of the mechanism in our notation. 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 again independent of whether we do or do not carry constants with us.

The messages jobs send to machines are called *actions*, since they constitute the strategic actions jobs can take in the non-cooperative game induced by the mechanism. A *strategy* of a job j maps each type 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 3. *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 our notation if we would restrict ourselves to *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 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 a tentative utility, i.e. the utility a job would have if it was the last to be accepted to its chosen machine.

Definition 4. *Given a decentralized, online scheduling mechanism, 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 at time $\tilde{r}_j$ from Definition 1 if the jobs have types t and play strategies according to s. Then we let $\hat{u}_j(s,t) := \hat{C} w_j - \hat{\pi}_j$ denote the 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 5. *A strategy profile* $(s_1, \dots, s_n)$ *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 additionally make the following assumption for the main part of the paper. Fix some constant $0 < \alpha \leq 1$ that will be discussed later. Let us call jobs *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

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

That means that long jobs cannot appear very early in the system. This assumption can be seen as a tribute to the desired performance guarantee, and in fact, it is well known that critical jobs must not be scheduled early in order to achieve constant competitive ratios [1, 10]. We point out however, that this assumption is only made due to cosmetic reasons. In Section 5.1, we show how to relax this assumption, and we discuss how critical jobs can be dealt with without leaving the framework of decentralized online scheduling mechanisms or losing any of our results.

3 The LocalGreedy Algorithm

We next formulate an online scheduling algorithm that is inspired by the MinIncrease Algorithm from Megow et al. [11]. For the time being, we assume that the job characteristics such as 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 [14] 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 were the increase in the objective $\sum w_j\, C_j$ is minimal.

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.

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 $b_i(t) = 0$.

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 gets 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 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.

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 6. *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 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.

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

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 7. *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. (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.

(b) Follows from (a) and the fact that the machine chosen by the LocalGreedy Algorithm maximizes j's tentative utility. □

Theorem 8. *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 7(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 at 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,

$$\begin{aligned}
\tilde{u}_j - u_j &= -\sum_{k \in J_1 \cup J_2} w_j \tilde{p}_k - \sum_{k \in J_3} \tilde{p}_j \tilde{w}_k - w_j \tilde{p}_j - \left(-\sum_{k \in J_1} w_j \tilde{p}_k - \sum_{k \in J_2 \cup J_3} p_j \tilde{w}_k - w_j p_j \right) \\
&= \sum_{k \in J_2} (p_j \tilde{w}_k - w_j \tilde{p}_k) - \sum_{k \in J_3} (\tilde{p}_j - p_j)\tilde{w}_k - w_j(\tilde{p}_j - p_j).
\end{aligned}$$

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 at the machine. 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 and the immediate utility at decision point r_j will be at least as large as its immediate utility at decision point $\tilde{r}_j$. Therefore, for a single machine, j maximizes its immediate utility at decision point $\tilde{r}_j$ by choosing $\tilde{r}_j = r_j$. As $\tilde{w}_j = w_j$, it follows from Lemma 7(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 7(b), choosing a machine that minimizes the immediate increase in the objective function. □

Theorem 9. *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 6, $\hat{u}_j$ on any machine is maximized by reporting $\tilde{w}_j = w_j$ for any $\tilde{r}_j$ and $\tilde{p}_j$. According to Theorem 8 and Lemma 7(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 7(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 our 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 8, the same holds for false reports about the processing time and the release date.

5 Performance of the Mechanism

As shown in Section 4, jobs have a motivation to report truthfully about their data: According to Theorem 6, 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 8 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. that $r_j \geq \alpha\, p_j$ for all jobs $j \in J$. We will now relax this assumption. Without the assumption, the Decentralized-LocalGreedy Mechanism as stated above does not yet yield a constant approximation factor; simple examples can be constructed in the same flavor as in [10]. In fact, it is well known that early arriving jobs with large processing times have to be delayed [1, 10, 11]. In order to achieve a constant competitive ratio, we also adopt this idea and use modified release dates as [10, 11]. 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 6 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 8. 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 $\sum w_j\, C_j$. We derive a constant competitive ratio for the Decentralized LocalGreedy Mechanism by the following theorem:

Theorem 10. *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 $\varrho = 3.281$.*

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 [11], 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. [4, Thm. 1]. The details are moved to Appendix C. The resulting performance bound 3.281 is identical to the one of [11] (for deterministic processing times), when α is $(\sqrt{17m^2 - 2m + 1} - m + 1)/(4m)$. $\square$

6 Negative Result

In this section we will show that by modifying the payment scheme it is not possible to turn the Decentralized LocalGreedy Mechanism into a mechanism that has a dominant strategy equilibrium, in which all jobs truthfully report about their data.

First note that in any dominant strategy equilibrium, each job must select a machine that maximizes its tentative utility at arrival. That is due to the fact that the choice of the machine must also be optimal if all jobs arriving later than j choose their release date so large such that j's tentative utility is not changed any more.

Let us for a moment integrate the choice of the machine into the Decentralized LocalGreedy Mechanism, that is an arriving job reports its data and is centrally assigned to one of the machines maximizing its tentative utility according to the reports made. The resulting mechanism is a direct revelation mechanism that differs from the LocalGreedy Algorithm only in the fact that jobs have to report their types. Therefore, it is enough to prove that the LocalGreedy Algorithm cannot be completed by a payment scheme to a *truthful* mechanism, i.e. a mechanism in which truth-telling is a dominant strategy equilibrium. To illustrate the latter, we use the following necessary condition formulated by Lavi et al. [8].

Definition 11. (Weak Monotonicity) *Let $v_j(A(\tilde{t})|t_j)$ denote the valuation of job j for a schedule that results from allocation algorithm A when type vector $\tilde{t}$ is reported and j has the true type t_j. Allocation algorithm A satisfies* weak monotonicity *if for any agent $j \in J$, every fixed report vector of the other agents $\tilde{t}_{-j}$ and every pair of possible types $t_j^{(1)}$ and $t_j^{(2)}$*

$$v_j(A(\tilde{t}_{-j}, t_j^{(1)})|t_j^{(1)}) - v_j(A(\tilde{t}_{-j}, t_j^{(1)})|t_j^{(2)}) \geq v_j(A(\tilde{t}_{-j}, t_j^{(2)})|t_j^{(1)}) - v_j(A(\tilde{t}_{-j}, t_j^{(2)})|t_j^{(2)}). \qquad (1)$$

Lemma 12. (Lavi, Mu'alem, and Nisan [8]) *Let A be an allocation algorithm. If there is a payment scheme π such that A together with π is a truthful mechanism, then A satisfies weak monotonicity.*

This result is now applied to our model. Lemma 13 formulates a necessary condition for truthfulness in our setting.

Lemma 13. *For a job $j \in J$ and fixed reports $\tilde{t}_{-j}$ by the other jobs, let $A(\tilde{t}_{-j}, \tilde{t}_j)$ denote the resulting schedule if j reports $\tilde{t}_j$. Let $C_j(A(\tilde{t}_{-j}, \tilde{t}_j))$ be the corresponding (ex-post) completion time of j in that schedule. Then A satisfies weak monotonicity in the described model only if it satisfies*

$$w_j^{(1)} < w_j^{(2)} \Rightarrow C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))) \geq C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)})))$$
$$\forall\, j \in J, \quad \forall\, \tilde{t}_{-j} \in T^{n-1}, \quad \forall\, w_j^{(1)}, w_j^{(2)} \geq 0, \quad \forall p_j > 0, \quad \forall r_j \geq 0.$$

Proof. See Appendix A. □

The above condition is in fact equivalent to the notion of decreasing work curves as formulated by Archer and Tardos [2] for the case with one-dimensional types. An example in Appendix B shows that the LocalGreedy Algorithm does not satisfy weak monotonicity, and therefore does not allow a payment scheme that extends the algorithm to a mechanism where truth-telling is a dominant strategy equilibrium. Let us summarize this.

Theorem 14. *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. Use Lemma 13 and the example in Appendix B. □

7 Discussion

The Decentralized LocalGreedy Mechanism induces rational jobs to report truthfully about their data, where this motivation is 'myopic' with respect to the weight and a dominant strategy equilibrium in the restricted strategy space where all jobs report their true weight. However, it would be desirable to define a decentralized online scheduling mechanism such that there is a dominant strategy equilibrium in which all jobs report all their data truthfully. Furthermore, the desired mechanism should yield a constant competitive ratio for all such equilibria. It remains an open question, whether such a mechanism exists. 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.

References

[1] E. J. Anderson and C. N. Potts. Online scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research*, 29(3):686–697, August 2004.

[2] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd Annual Symposium on Foundations of Computer Science*, pages 482–491. IEEE Computer Society, 2001.

[3] J. R. Correa and M. R. Wagner. LP-based online scheduling: from single to parallel machines. In M. Jünger and V. Kaibel, editors, *Proc. 11th Conference on Integer Programming and Combinatorial Optimization*, volume 3509 of *Lecture Notes in Computer Science*, pages 196–209. 2005.

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

[5] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[6] J. A. Hoogeveen and A. P. A. Vestjens. Optimal on-line algorithms for single machine scheduling. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Proc. 5th Conference on Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, pages 404–414. Springer, 1996.

[7] A. Kovacs. Fast monotone 3-approximation algorithm for scheduling related machines. In G. S. Brodal and S. Leonardi, editors, *Proc. 13th Annual European Symposium on Algorithms*, Lecture Notes in Computer Science. Springer, 2005, to appear.

[8] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. 44th Annual Symposium on Foundations of Computer Science*, pages 574–583. IEEE Computer Society, 2003.

[9] E. L. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:243–362, 1977.

[10] N. Megow and A. S. Schulz. On-line scheduling to minimize average completion time revisited. *Operations Research Letters*, 32:485–490, 2004.

[11] N. Megow, M. Uetz, and T. Vredeveld. Stochastic online scheduling on parallel machines. In G. Persiano and R. Solis-Oba, editors, *Proc. Second International Workshop on Approximation and Online Algorithms*, volume 3351 of *Lecture Notes in Computer Science*, pages 167–180. 2005.

[12] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[13] R. Porter. Mechanism design for online real-time scheduling. *Proc. of the ACM Conference on Electronic Commerce (EC'04)*, 2004.

[14] W. Smith. Various optimizers for single stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[15] A. P. A. Vestjens. *On-line Machine Scheduling.* PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1997.

Appendices

A Proof of Lemma 13

Lemma 13. *For a job $j \in J$ and fixed reports $\tilde{t}_{-j}$ by the other jobs, let $A(\tilde{t}_{-j}, \tilde{t}_j)$ denote the resulting schedule if j reports $\tilde{t}_j$. Let $C_j(A(\tilde{t}_{-j}, \tilde{t}_j))$ be the corresponding (ex-post) completion time of j in that* *schedule. Then A satisfies weak monotonicity in the described model only if it satisfies*

$$\begin{aligned} w_j^{(1)} < w_j^{(2)} \Rightarrow C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))) \geq C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))) \\ \forall\, j \in J, \quad \forall\, \tilde{t}_{-j} \in T^{n-1}, \quad \forall\, w_j^{(1)}, w_j^{(2)} \geq 0, \quad \forall p_j > 0, \quad \forall r_j \geq 0. \end{aligned}$$

Proof. Weak monotonicity is satisfied only if Equation 1 is satisfied for all possible pairs $t_j^{(1)}$ and $t_j^{(2)}$. Let $w_j^{(1)} < w_j^{(2)}$ and define $t_j^{(1)} := (r_j, p_j, w_j^{(1)})$ and $t_j^{(2)} := (r_j, p_j, w_j^{(2)})$. Then Equation 1 implies

$$\begin{aligned} & v_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))|(r_j, p_j, w_j^{(1)})) - v_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))|(r_j, p_j, w_j^{(2)})) \\ \geq\; & v_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))|(r_j, p_j, w_j^{(1)})) - v_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))|(r_j, p_j, w_j^{(2)})) \end{aligned}$$

Using the special structure of the valuation function in our model the above condition becomes:

$$\begin{aligned} & -w_j^{(1)} C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))) + w_j^{(2)} C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))) \\ & +w_j^{(1)} C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))) - w_j^{(2)} C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))) \geq \quad 0 \\ \Leftrightarrow \quad & (w_j^{(1)} - w_j^{(2)})[C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))) - C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)})))] \geq \quad 0 \\ \Leftrightarrow \quad & C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(2)}))) - C_j(A(\tilde{t}_{-j}, (r_j, p_j, w_j^{(1)}))) \geq \quad 0\,, \end{aligned}$$

where the last equivalence follows from $w_j^{(2)} < w_j^{(1)}$. □

B The LocalGreedy algorithm is not weakly monotone

Example 15. *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 the common release date $r > 4\alpha$, but let us assume that they nevertheless arrive in the given order. (We could alternatively enforce this order by adding small but positive constants to some of the release dates without changing the effect demonstrated below.)*

Let us consider the LocalGreedy algorithm. Note that no job has to be delayed according to the modified release dates. 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:

$$incr(j, 1) = (r + \frac{1}{7} + 3)w^{(1)}$$

$$incr(j, 2) = (r + \frac{1}{7} + 4)w^{(1)}.$$

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:

$$incr(j, 1) = (r + \frac{1}{7})w^{(2)} + 6 \cdot \frac{1}{7}$$

$$incr(j, 2) = (r + \frac{1}{7})w^{(2)} + 5 \cdot \frac{1}{7}.$$

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 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 $\frac{1}{2}$. Therefore, the MinIncrease Algorithm does not satisfy weak monotonicity.

C Proof of Theorem 10

Theorem 10. *Suppose every job is rational in reporting $\tilde{w}_j$, $\tilde{p}_j$, choosing its release date $\tilde{r}_j$ and selecting a machine. Then the* Decentralized LocalGreedy *Mechanism is ϱ-competitive, with $\varrho = 3.281$.*

Proof. Recall that Z denotes the objective value of the final schedule produced by the Decentralized LocalGreedy Mechanism. Let Z^{OPT} denote the value of the optimum off-line solution. We have already argued that

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

Next, recall that upon arrival of job j on any of the machines i (at time r'_j), machine i is blocked for time $b_i(r'_j) \leq r'_j/\alpha$. Therefore we get, for any j,

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

Thus,

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

The last term can be rewritten as follows:

$$\sum_{j\in J} p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k}{m} = \sum_{\substack{(j,k):\\ j\in H(k)\\ k<j}} p_j \frac{w_k}{m} = \sum_{\substack{(j,k):\\ k\in H(j)\\ j<k}} p_k \frac{w_j}{m} = \sum_{j\in J} w_j \sum_{\substack{k\in H(j)\\ k>j}} \frac{p_k}{m}\,.$$

Therefore,

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

Now, we apply a lower bound on the optimal off-line schedule from [4, Thm. 1], namely

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

yielding:

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

where in the second inequality $r_j + \alpha p_j$ is used as an upper bound on r'_j. Applying the trivial lower bound $\sum_{j \in J} w_j (r_j + p_j) \leq Z^{OPT}$, we get:

$$\begin{aligned} Z &\leq Z^{OPT} + \max\left\{ 1 + \frac{1}{\alpha},\, 1 + \alpha + \frac{m-1}{2m} \right\} Z^{OPT} \\ &= 2Z^{OPT} + \max\left\{ \frac{1}{\alpha},\, \alpha + \frac{m-1}{2m} \right\} Z^{OPT}. \end{aligned}$$

Therefore, we get the performance bound

$$\varrho = 2 + \max\left\{ \frac{1}{\alpha},\, \alpha + \frac{m-1}{2m} \right\}.$$

This can now be optimized for α, which was already done in [10]. There it was shown that $\varrho < 3.281$ for $\alpha = (\sqrt{17m^2 - 2m + 1} - m + 1)/(4m)$. $\square$