Mathematics of Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Greed Works—Online Algorithms for Unrelated Machine Stochastic Scheduling

Varun Gupta, Benjamin Moseley, Marc Uetz, Qiaomin Xie

To cite this article:
Varun Gupta, Benjamin Moseley, Marc Uetz, Qiaomin Xie (2020) Greed Works—Online Algorithms for Unrelated Machine Stochastic Scheduling. Mathematics of Operations Research 45(2):497-516. https://doi.org/10.1287/moor.2019.0999

http://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 45, No. 2, May 2020, pp. 497–516
ISSN 0364-765X (print), ISSN 1526-5471 (online)

Greed Works—Online Algorithms for Unrelated Machine Stochastic Scheduling

Varun Gupta,[a] Benjamin Moseley,[b] Marc Uetz,[c] Qiaomin Xie[d]

[a] Booth School of Business, University of Chicago, Chicago, Illinois 60637; [b] Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213; [c] Department of Applied Mathematics, University of Twente, 7522 NB Enschede, Netherlands; [d] Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Contact: varun.gupta@chicagobooth.edu, http://orcid.org/0000-0001-7373-1734 (VG); moseleyb@andrew.cmu.edu, https://orcid.org/0000-0001-8162-017X (BM); m.uetz@utwente.nl, http://orcid.org/0000-0003-4223-2435 (MU); qxie@mit.edu, http://orcid.org/0000-0003-2834-6866 (QX)

Received: November 29, 2017
Revised: July 9, 2018; October 29, 2018
Accepted: December 23, 2018
Published Online in Articles in Advance: January 29, 2020

MSC 2000 Subject Classification: Primary: 90B36, 68M20; secondary: 90C27, 90C59
OR/MS Subject Classification: Primary: analysis of algorithms: suboptimal algorithms; production/scheduling: stochastic, multiple machine, sequencing: stochastic, approximations/heuristic

https://doi.org/10.1287/moor.2019.0999

Abstract. This paper establishes performance guarantees for online algorithms that schedule stochastic, nonpreemptive jobs on unrelated machines to minimize the expected total weighted completion time. Prior work on unrelated machine scheduling with stochastic jobs was restricted to the offline case and required linear or convex programming relaxations for the assignment of jobs to machines. The algorithms introduced in this paper are purely combinatorial. The performance bounds are of the same order of magnitude as those of earlier work and depend linearly on an upper bound on the squared coefficient of variation of the jobs' processing times. Specifically for deterministic processing times, without and with release times, the competitive ratios are 4 and 6, respectively. As to the technical contribution, this paper shows how dual fitting techniques can be used for stochastic and nonpreemptive scheduling problems.

Funding: This work was funded by the Simons Institute for the Theory of Computing at the University of California, Berkeley; the National Science Foundation (NSF) [Grants CCF-1617724, CCF-1830711, CCF-1824303, and CCF-1733873]; a Yahoo Research Award; and a Google Research Award.

Keywords: unrelated machine scheduling • stochastic scheduling • online scheduling

> The point is, ladies and gentleman, that greed, for lack of a better word, is good. Greed is right, greed works. (Gordon Gekko, *Wall Street* [43])

1. Introduction

Scheduling jobs on multiple parallel machines is a fundamental problem both in combinatorial optimization and systems theory. There is a vast number of different model variants as well as applications, to which the existence of the handbook by Leung [25] testifies. A well-studied class of problems is scheduling a set of n nonpreemptive jobs that arrive over time on m unrelated machines with the objective of minimizing the total weighted completion time. In the unrelated machines model, the matrix that describes the processing times of all jobs on all machines can have any rank larger than 1. The offline version of the problem is denoted by $R|r_j|\sum w_jC_j$ in the three-field notation of Graham et al. [11], and the problem has been a cornerstone problem for the development of new techniques in the design of (approximation) algorithms; see, for example, Bansal et al. [4], Horowitz and Sahni [16], Lenstra et al. [24], and Skutella [38].

This paper addresses the online version of the problem where jobs sizes are stochastic. In the online model, jobs arrive over time, and the set of jobs is unknown a priori. For pointers to relevant work on online models in scheduling, see Im et al. [18] and Pruhs et al. [34]. In many systems, the scheduler may not know the exact processing times of jobs when the jobs arrive to the system. Different approaches have been introduced to cope with this uncertainty. If jobs can be preempted, then nonclairvoyant schedulers have been studied that do not know the processing time of a job until the job is completed (Becchetti and Leonardi [5], Gupta et al. [13], Im et al. [20], Kalyanasundaram and Pruhs [22], Motwani et al. [33]). Unfortunately, if preemption is not allowed, then any algorithm has poor performance in the nonclairvoyant model, because the lower bound for the competitive ratio against the offline optimal schedule is $\Omega(n)$. This is even true if we consider the special case where all jobs have the same unit weight w_j.

This lower bound suggests that the nonclairvoyant model is too pessimistic for nonpreemptive problems. Even if exact processing times are unknown to the scheduler, it can be realistic to assume that at least an

estimate of the true processing times is available. For such systems, a model that is used is *stochastic scheduling*. In the stochastic scheduling model, the jobs' processing times are given by random variables. A *nonanticipatory* scheduler only knows the random variable that encodes the possible realizations of a job's processing time. If the scheduler starts a job on a machine, then that job must be run to completion *nonpreemptively*, and it is only when the job completes that the scheduler learns the actual processing time of the job. With respect to the random processing times, both the scheduler and the optimal solution are required to be nonanticipatory, which means that only the (conditional) distribution of a job's processing time may be used at any point in time. Stochastic scheduling has been well studied in fundamental work such as that by Möhring et al. [30, 31] and approximation algorithms by, for example, Möhring et al. [32], Skutella and Uetz [39], Megow et al. [29], Skutella et al. [40], and Schulz [36].

This paper considers online scheduling of nonpreemptive, stochastic jobs in the unrelated machine model to minimize the total weighted completion time. This is the same problem as considered by Megow et al. [29], but here we address the more general *unrelated* machine model. In the stochastic unrelated machine model, the scheduler is given machine-dependent probability distributions that describe a job's potential processing time for each of the machines. For a given job, the processing times across different machines need not be independent, but the processing times of different jobs are assumed to be independent.

1.1. Identical Machines, Special Processing Time Distributions

Restricting attention to nonpreemptive policies, when all machines are identical, perhaps the most natural algorithm is weighted shortest expected processing time (WSEPT) first. When a machine is free, WSEPT always assigns the job to be processed that has the maximum ratio of weight over expected size. When all jobs have unit weight, this algorithm boils down to the shortest expected processing time (SEPT) algorithm that greedily schedules jobs with the smallest expected size. When there is a single machine and all jobs arrive at the same time, WSEPT is optimal (Rothkopf [35]). For multiple machines with equal weights for all jobs, if the job sizes are deterministic and arrive at the same time, SEPT is optimal (Horn [15]). For multiple identical machines with equal weights for all jobs, SEPT is optimal if job sizes are exponentially distributed (Bruno et al. [6], Weiss and Pinedo [45]) or, more generally, are stochastically comparable in pairs (Weber et al. [44]). Some extensions of these optimality results to the problem with weights exist as well (Kämpke [23]). For more general distributions, simple solutions fail (Uetz [41]), and our knowledge of optimal scheduling policies is limited.

1.2. Identical Machines, Arbitrary Processing Times

To cope with these challenges, approximation algorithms have been studied. With the notable exception of Im et al. [19], all approximation algorithms have performance guarantees that depend on an upper bound Δ on the squared coefficient of variation of the underlying random variables. Möhring et al. [32] established the first approximation algorithms for stochastic scheduling on identical machines via a linear programming relaxation. Their work gave a $(3+\Delta)$-approximation for jobs released over time (yet known offline), and they additionally showed that WSEPT is a $(3+\Delta)/2$-approximation when jobs arrive together.[1] These results have been built on and generalized in several settings (Jäger and Skutella [21], Megow and Vredeveld [28], Megow et al. [29], Schulz [36], Skutella and Uetz [39], Skutella et al. [40]), notably in Megow et al. [29] and Schulz [36] for the online setting. The currently best known result when jobs are released over time (yet known offline) is a $(2+\Delta)$-approximation by Schulz [36]. In the online setting, Schulz [36] gives an algorithm with performance guarantee of $(2.309+1.309\Delta)$. These results build on an idea from Correa and Wagner [10] to use a preemptive, fast single machine relaxation, next to the relaxation of Möhring et al. [32]. The work of Im et al. [19] gave the first results independent of Δ showing that there exist polylogarithmic approximation algorithms under some assumptions. All these papers address problems with identical machines.

1.3. Unrelated Machines, Arbitrary Processing Times

For some 15 years after the results of Möhring et al. [32] for the case of identical machines, no nontrivial results were known for the case of unrelated machines despite it being a target in the area. Recently, Skutella et al. [40] gave a $(3+\Delta)/2$-approximation algorithm for the unrelated machines model when jobs arrive at the same time and a $(2+\Delta)$-approximation when jobs are released over time (yet known offline). Central to unlocking an efficient approximation algorithm for the unrelated machines case was the introduction of a time-indexed linear program (LP) that lower bounds the objective value of the optimal nonanticipatory scheduling policy. It is this LP that allows the authors to overcome the complexities of the stochastic unrelated machine setting.

The work introduced in this paper targets the more realistic *online* setting for scheduling stochastic jobs on unrelated machines. A priori, it is not clear that there should exist an algorithm with a small competitive ratio for this problem. Prior work for the offline problem requires sophisticated linear (Skutella et al. [40]) or convex (Balseiro et al. [3]) programming relaxations. Good candidates for online algorithms that might also have practical impact should be simple and combinatorial, but even discovering an offline approximation algorithm that is simple and combinatorial has remained an open problem for (stochastic) scheduling on unrelated machines.

1.4. Related Work for Deterministic Processing Times

For special cases and deterministic processing times, approximation algorithms have been known to exist. For example for the online unrelated machine case with deterministic processing times, Hall et al. [14] obtained an 8-competitive algorithm. Their algorithm is based on the idea to partition time into geometrically increasing intervals and then to maximize the total weight of (available) jobs that can be scheduled in these intervals. Algorithms with better competitive ratios were obtained by Chakrabarti et al. [7] by using randomization in the definition of these intervals, resulting in a randomized 5.78-competitive algorithm. As far as we know, this is the state of the art when it comes to competitive analysis for the online problem with release times and on unrelated machines. The deterministic greedy algorithm proposed in this paper is 6-competitive.

For the offline problem with deterministic processing times, the following is known: when there are no release times and processing times are deterministic, the currently best known approximation algorithms have performance bounds slightly below 3/2, based on semidefinite relaxations (Bansal [4]) and, more recently, also on linear relaxations (Li [26]). For the offline case with release times, the $(2+\varepsilon)$-approximation of Schulz and Skutella [37] was the best known until recently, when Im and Li [17] gave a 1.878-approximation algorithm. The problem has also been looked at through the lens of game theory, and for the offline problem without release times, Cole et al. [8] showed that when machines follow the weighted shortest processing time (WSPT) rule, Nash equilibria of selfish jobs that minimize their own completion time yield schedules with cost at most four times the optimal. Interestingly, our paper shows that the same approximation guarantee can be obtained online by a simple greedy algorithm. We note that the work of Cole et al. [8] is offline, and moreover, their algorithm and analysis differ from those of this paper.

With respect to lower bounds on performance guarantees of online algorithms, we are aware of only one lower bound on the competitive ratio of any online algorithm, which is the 1.309 lower bound of Vestjens [42]. This lower bound holds for the problem on identical machines with deterministic processing times.

1.5. Results

This paper suggests two combinatorial online algorithms for stochastic scheduling on unrelated machines that have a performance guarantee of order $O(\Delta)$, where Δ is an upper bound on the squared coefficient of variation of the processing time distributions P_{ij}. More specifically, in the *online-list* model, where jobs arrive online (at time 0) and must be assigned to a machine immediately upon arrival, this paper establishes a performance guarantee of $(4+2\Delta)$. For this problem, the algorithm assigns jobs to machines to minimize the expected contribution to the objective function, whereas per machine, the jobs are sequenced by largest ratio $w_i/\mathbb{E}[P_{ij}]$ first. For deterministic processing times, the proposed greedy algorithm has a competitive ratio of 4, and this paper also gives a lower bound instance showing that the analysis is tight. Arguably more relevant is the *online-time* model, where jobs arrive over time at individual release times. Here this paper establishes a performance guarantee of $(6+3\Delta)h(\Delta)$, where $h(\Delta) = 1+\sqrt{\Delta}/2$ for $\Delta \le 1$ and $h(\Delta) = 1+\Delta/(\Delta+1)$ for $\Delta \ge 1$. Observe that $h(\cdot)$ is a concave, increasing function of Δ that is bounded from above by 2. Here the greedy assignment of jobs to machines is the same, but the sequencing per machine is augmented by possibly introducing forced idle time. The idea here is to work with a "nominal" schedule based on expected processing times and never start processing a job before its nominal starting time. For deterministic processing times, $\Delta = 0$, and $h(\Delta) = 1$; hence, the competitive ratio equals 6. Because the algorithm is a deterministic algorithm, this improves upon the competitive ratio 8 from Hall et al. [14] and falls only slightly behind the randomized 5.78-competitive algorithm that was proposed by Chakrabarti et al. [7].

Even though the performance bounds for the case with nontrivial release times are most probably not tight, we believe that our results are interesting for the following reasons: (1) this is the first analysis of a combinatorial algorithm for stochastic scheduling on unrelated machines and the first results for stochastic online scheduling in the unrelated machine model, (2) even for the deterministic setting, this is the first analysis of an (arguably) intuitive combinatorial algorithm that simply assigns jobs to machines where their expected contribution is minimal, (3) the analysis uses the idea of dual fitting; hence, we demonstrate that this technique can be used for bounding the performance of scheduling policies in nonpreemptive and stochastic

scheduling, and (4) the performance bounds, even where not tight, have the same order of magnitude as those of earlier results in the literature, while considering a more general problem.

We now briefly discuss the proposed algorithms in relation to prior work. The algorithms rest on the following ingredients to solve the machine assignment and scheduling problem. Generally speaking, at any point in time, a job with the highest ratio $w_j/\mathbb{E}[P_{ij}]$ is scheduled from the set of jobs that are assigned to and available on machine i. This is the well-known WSEPT rule. For the case with release dates, however, jobs possibly have to wait for "artificial" release times before they are declared available for processing. The necessity of such forced idle time is well known whenever jobs are released over time, even for single-machine problems (Megow and Schulz [27]). Next, to this standard manipulation of release times, we work with a nominal schedule that is based on expected processing times and never allow jobs to be started before their nominal starting times. The assignment of jobs to machines is solved by greedily assigning jobs to the machines where (a proxy for) the expected increase of the objective is minimal. Comparable greedy-type algorithms have also been used before, for example, in Avrahami and Azar [2], Megow and Schulz [27], and Megow et al. [29], but not for an unrelated machine setting. Note that the $\Omega(\Delta)$ lower bound for fixed assignment policies in Skutella et al. [40] yields that our results are asymptotically tight in Δ among policies that must irrevocably assign jobs to machines at the time of their release. As mentioned earlier, the analysis proposed in this paper uses dual fitting techniques. The technique has been used, for example, in Anand et al. [1] for deterministic and preemptive scheduling problems.

2. Notation and Preliminaries

The input to the problem consists of a set of unrelated parallel machines M of cardinality m. The set of jobs J, of cardinality n, is unknown and only disclosed gradually over time. Each job needs to be executed on exactly one (and any one) of the machines in M, and each machine can process at most one job at a time. The jobs are nonpreemptive. This means that a job, once started, must not be interrupted until its completion.

This paper considers two online models. In the first model, known as the *online-list model*, the scheduler is presented the jobs $j \in J$ one after the other. Whenever a job is presented, the algorithm has to assign it to one of the machines before the next job is presented. The decision when the job begins being processed can be deferred until all jobs have arrived. It is unknown how many jobs will arrive, but once all jobs in J have arrived, the jobs assigned to any one of the machines must be sequenced on that machine in some order. In the second model, known as the *online-time model*, time progresses and jobs appear over time at their individual release times. Let r_j denote the release time of job j. At the moment of arrival r_j, or possibly at a later point in time, a job must be assigned to a machine. Once assigned to a machine, the job may possibly wait until an even later point in time to be processed.

The jobs are stochastic, meaning that each job j's processing time is revealed to the scheduler at the point of arrival in the form of a random variable P_{ij} for every machine $i \in M$. If job j is assigned to machine i, its processing time will be random according to P_{ij}. It is allowed that certain jobs $j \in J$ cannot be processed on certain machines $i \in M$, in which case $\mathbb{E}[P_{ij}] = \infty$.

In the stochastic scheduling model, the realization of the processing time of a job j becomes known at the moment that the job completes. This paper considers designing a nonanticipatory scheduling policy Π that minimizes the expected total weighted completion time $\mathbb{E}[\sum_j w_j C_j]$, where C_j denotes the random variable for the completion time of job j under policy Π.

We assume that the random variables P_{ij} are discrete and integer valued. This can be assumed at the cost of a multiplicative factor of $(1+\varepsilon)$ in the final approximation ratio for any $\varepsilon > 0$ (Skutella et al. [40]). Our analysis will make use of the following facts about first and second moments of discrete random variables; these facts also appear in Skutella et al. [40].

Lemma 1. *Let X be an integer-valued nonnegative random variable. Then*

$$\sum_{r \in \mathbb{Z}_{\geq 0}} \mathbb{P}[X > r] = \mathbb{E}[X] \quad \text{and} \quad \sum_{r \in \mathbb{Z}_{\geq 0}} \left(r + \frac{1}{2}\right) \mathbb{P}[X > r] = \frac{1}{2}\mathbb{E}\left[X^2\right].$$

Definition 1. Let X be a nonnegative random variable. The *squared coefficient of variation* is defined as the scaled variance of X; that is,

$$\mathbb{CV}[X]^2 := \mathbb{V}\mathrm{ar}[X]/\mathbb{E}[X]^2,$$

where $\mathbb{V}\mathrm{ar}[X] = \mathbb{E}[X^2] - E[X]^2$.

2.1. Stochastic Online Scheduling and Policies

The setting considered in this paper is that of stochastic online scheduling as defined in Megow et al. [29]. This means that (the existence of) a job j is unknown before it arrives, and upon arrival at time r_j, only the distribution of the random variables P_{ij} for the possible processing times on machines $i = 1, \dots, m$ are known to the scheduler. At any given time t, a nonanticipatory online scheduling policy is allowed to use only the information that is available at time t. In particular, it may anticipate the (so far) realized processing times of jobs up to time t. For example, a job that has possible size 1, 3, or 4 with probability 1/3 each, and has been running for two time units, will have a processing time 3 or 4, each with probability 1/2. It is well known that adaptivity over time is needed to minimize the expectation of the total weighted completion time (e.g., Uetz [41]). We refer the reader to Megow et al. [29] for a more thorough discussion of the stochastic online model.

For simplicity of notation, denote by OPT the expected total weighted completion time of an optimal nonanticipatory scheduling policy for the problem where the set of jobs, their release times r_j, and their processing time distributions P_{ij} are known in advance; that is to say, the benchmark that we compare our algorithms to knows the set of jobs and their parameters, but not the actual realizations of processing time distributions P_{ij}.

We seek to find a nonanticipatory online scheduling policy (an algorithm) ALG with expected performance close to OPT. For convenience, and in a slight abuse of notation, we use the same notation for both the algorithm and its expected performance; that is to say, both ALG and OPT denote the expected performance of nonanticipatory scheduling policies, and by linearity of expectation, we have $\mathsf{ALG} = \sum_j w_j \mathbb{E}[C_j^{\mathsf{ALG}}]$ and $\mathsf{OPT} = \sum_j w_j \mathbb{E}[C_j^{\mathsf{OPT}}]$.

Definition 2. A scheduling policy is said to have a (multiplicative) *performance guarantee* $\alpha \geq 1$ if, for every possible input instance,

$$\mathsf{ALG} \leq \alpha \mathsf{OPT}.$$

We remark that OPT is not restricted to assigning jobs to machines at the time of their arrival. The only restrictions on OPT are that it must schedule jobs nonpreemptively and be nonanticipatory. Note that our approximation guarantees hold against an adversary who knows all the jobs and their release times r_j, as well as the processing time distributions P_{ij} in advance, but not the actual realizations of P_{ij}. This implies that the model generalizes the classic offline stochastic scheduling model (assuming all parameters are disclosed to the scheduler, too), as well as traditional competitive analysis (assuming deterministic processing times).

Finally, we may assume without loss of generality (w.l.o.g.) that no pair of job and machine exists with $\mathbb{E}[P_{ij}] = 0$. That said, we may further assume that $\mathbb{E}[P_{ij}] \geq 1$ for all machines i and jobs j by scaling.

3. Linear Programming Relaxations

This section introduces a linear programming relaxation for the problem. This relaxation was discussed previously in Skutella et al. [40, section 8]. The LP uses variables y_{ijs} to denote the probability that job j is being processed on machine i within the time interval $[s, s+1]$ under some given and fixed scheduling policy. It is known that y_{ijs} can be linearly expressed in terms of the variables x_{ijt}, which denote the probability that job j is started at time t on machine i, as follows:

$$y_{ijs} = \sum_{t=0}^{s} x_{ijt} \, \mathbb{P}[P_{ij} > s - t]. \tag{1}$$

The fact that any machine can process at most one job at a time can be written as

$$\sum_{j \in J} y_{ijs} \leq 1 \qquad \text{for all } i \in M,\ s \in \mathbb{Z}_{\geq 0}. \tag{2}$$

Moreover, by the fact that scheduling policies are nonanticipatory, we know that whenever a job j is started on a machine i at time t, it will in expectation be processed for time $\mathbb{E}[P_{ij}]$, so its expected completion time is $t + \mathbb{E}[P_{ij}]$. Now, conditioning on a job being processed on machine i and making use of (1) and the first part of Lemma 1, together with the fact that each job must be completely processed, gives the constraint that $\sum_{s \in \mathbb{Z}_{\geq 0}} \frac{y_{ijs}}{\mathbb{E}[P_{ij}]} = 1$. Unconditioning on the machine assignment yields the following constraints:

$$\sum_{i \in M} \sum_{s \in \mathbb{Z}_{\geq 0}} \frac{y_{ijs}}{\mathbb{E}[P_{ij}]} = 1 \qquad \text{for all } j \in J. \tag{3}$$

Finally, with the help of (1) and the second part of Lemma 1, the expected completion time of a job j can be expressed in y_{ijs} variables as

$$C_j^S := \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\geq 0}} \left(\frac{y_{ijs}}{\mathbb{E}[P_{ij}]}\left(s+\frac{1}{2}\right)+\frac{1-\mathbb{CV}[P_{ij}]^2}{2}\,y_{ijs}\right) \qquad \text{for all } j\in J, \tag{4}$$

where we labeled the expected completion time variable with a superscript S for "stochastic" for reasons that will become clear shortly. For completeness, Equation (4) is proved in Lemma A.1 (in the appendix).

For the analysis to follow, we also need to express the fact that the expected completion time of a job cannot be smaller than its expected processing time, which is generally not implied by (4):

$$C_j^S \geq \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\geq 0}} y_{ijs} \qquad \text{for all } j\in J. \tag{5}$$

The following linear programming relaxation for the unrelated machine scheduling problem can be derived with these observations. This LP extends the LP given in Skutella et al. [40] by adding the constraints (5):

$$\begin{aligned} \min \quad & z^S = \sum_{j\in J} w_j C_j^S \\ \text{s.t.} \quad & (2),\ (3),\ (4),\ (5), \\ & y_{ijs}\geq 0 \qquad \text{for all } j\in J, i\in M, s\in\mathbb{Z}_{\geq 0}. \end{aligned} \tag{S}$$

The analysis in this paper will work with the dual of this relaxation. However, the term $-\mathbb{CV}[P_{ij}]^2$ in the primal objective would appear in the dual constrains. Because we do not know how to deal with this negative term in the analysis that is to follow, we are going to factor it out.

To that end, define a simpler (i.e., deterministic) version for the expected completion times (4), labeled with P to distinguish it from the previous formulation, by letting

$$C_j^P = \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\geq 0}} \left(\frac{y_{ijs}}{\mathbb{E}[P_{ij}]}\left(s+\frac{1}{2}\right)+\frac{y_{ijs}}{2}\right) \qquad \text{for all } j\in J. \tag{6}$$

Consider the following linear programming problem:

$$\begin{aligned} \min \quad & z^P = \sum_{j\in J} w_j C_j^P \\ \text{s.t.} \quad & (2),\ (3),\ (6), \\ & y_{ijs}\geq 0 \qquad \text{for all } j\in J, i\in M, s\in\mathbb{Z}_{\geq 0}. \end{aligned} \tag{P}$$

This corresponds to a time-indexed linear programming relaxation for a purely deterministic, unrelated machine scheduling problem where the random processing times are fixed at their expected values $\mathbb{E}[P_{ij}]$. Also note that we have dropped constraints (5).

In the following, a relationship between these two relaxations is established. To begin, define an upper bound on the squared coefficient of variation by the following.

Definition 3. Define Δ as a universal upper bound on the squared coefficient of variation of the processing time of any job on any machine, that is,

$$\Delta := \max_{i,j} \mathbb{CV}[P_{ij}]^2.$$

Observe that $\Delta = 0$ for deterministic processing times and $\Delta = 1$ for processing times that are new better than used in expectation (NBUE); that is, the expected remaining processing time of a job never exceeds its total expected processing time. Specifically, $\Delta = 1$ for exponential distributions. Next, for any given solution $\boldsymbol{y}$ of (S) or (P), define

$$H(\boldsymbol{y}) := \sum_{j\in J} w_j \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\geq 0}} y_{ijs}.$$

Let $\boldsymbol{y}^S$ denote an optimal solution to (S), and recall that OPT is the expected total weighted completion time of an optimal nonanticipatory scheduling policy. By constraints (5),

$$H(\boldsymbol{y}^S) = \sum_{j\in J} w_j \sum_{i\in M} \sum_{s\in\mathbb{Z}_{\geq 0}} y^S_{ijs} \leq \sum_{j\in J} w_j C^S_j = z^S(\boldsymbol{y}^S) \leq \mathsf{OPT}.$$

The following lemma establishes the relation between the two relaxations and is crucial for our analysis.

Lemma 2. *The optimal solution values z^P and z^S of the linear programming relaxations (P) and (S) fulfill*

$$z^P \leq \left(1+\frac{\Delta}{2}\right) z^S.$$

Proof. Let $\boldsymbol{y}^P$ be an optimal solution to (P), and let $\boldsymbol{y}^S$ be an optimal solution to (S). Clearly, $\boldsymbol{y}^S$ is a feasible solution also for (P), which is less constrained. Hence, we get the following, where $z^P(\boldsymbol{y}^P)$ is the value of $\boldsymbol{y}^P$ on LP (P):

$$\begin{aligned} z^P = z^P(\boldsymbol{y}^P) &\leq z^P(\boldsymbol{y}^S) \\ &= z^S(\boldsymbol{y}^S) + \sum_{j\in J} w_j \sum_{i\in M} \sum_{s\in\mathbb{Z}_{\geq 0}} \frac{\mathbb{CV}[P_{ij}]^2}{2} y^S_{ijs} \\ &\leq z^S(\boldsymbol{y}^S) + \frac{\Delta}{2} H(\boldsymbol{y}^S) \leq \left(1+\frac{\Delta}{2}\right) z^S(\boldsymbol{y}^S). \end{aligned} \tag{7}$$

Note that the second-to-last inequality uses only the definitions of Δ and $H(\cdot)$. The last inequality holds because $H(\boldsymbol{y}^S) \leq z^S(\boldsymbol{y}^S)$. □

Recalling that (S) is a relaxation for the stochastic scheduling problem, we conclude the following.

Corollary 1. *The optimal solution value z^P of the linear programming relaxation (P) is bounded by the expected performance of an optimal scheduling policy by*

$$z^P \leq \left(1+\frac{\Delta}{2}\right) \mathsf{OPT}.$$

The dual program of (P) will have unconstrained variables α_j for all $j \in J$ and nonnegative variables β_{is} for all $i \in M$ and $s \in \mathbb{Z}_{\geq 0}$:

$$\begin{aligned} \max \quad & z^D = \sum_{j\in J} \alpha_j - \sum_{i\in M} \sum_{s\in\mathbb{Z}_{\geq 0}} \beta_{is} \\ \text{s.t.} \quad & \frac{\alpha_j}{\mathbb{E}[P_{ij}]} \leq \beta_{is} + w_j \left(\frac{s+\frac{1}{2}}{\mathbb{E}[P_{ij}]} + \frac{1}{2}\right) && \text{for all } i \in M, j \in J, s \in \mathbb{Z}_{\geq 0}, \\ & \beta_{is} \geq 0 && \text{for all } i \in M, s \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{D}$$

Like the analysis in Anand et al. [1], we will define a feasible solution for the dual (D) such that this solution corresponds to the schedule created by an online greedy algorithm for the original stochastic scheduling problem. Similar greedy algorithms have been used before, in both deterministic and stochastic scheduling on parallel machines, for example, in Avrahami and Azar [2], Megow and Schulz [27], and Megow et al. [29].

4. Greedy Algorithm and Analysis for the Online-List Model

In this section, the online-list model is considered. Assume w.l.o.g. that the jobs are presented in the order $1, 2, \ldots, |J|$. On any machine i, let $H(j,i)$ denote the jobs that have priority no less than that of job j according to the ratios $w_k/\mathbb{E}[P_{ik}]$, breaking ties by index; that is,

$$H(j,i) := \{k \in J \mid w_k/\mathbb{E}[P_{ik}] > w_j/\mathbb{E}[P_{ij}]\} \cup \{k \in J \mid k \leq j, w_k/\mathbb{E}[P_{ik}] = w_j/\mathbb{E}[P_{ij}]\}.$$

Note that $j \in H(j,i)$. Also let $L(j,i) := J \setminus H(j,i)$. Furthermore, let $k \to i$ denote that a job k has been assigned to machine i by the algorithm.

4.1. Greedy Algorithm (Online-List Model)

Whenever a new job $j \in J$ is presented to the algorithm, compute for each of the machines $i \in M$ the *instantaneous expected increase* in the cost if job j is assigned to machine i and all jobs already present on i are scheduled in nonincreasing order of the ratio's weight over expected processing time. Because the expected completion time of the new job j will be determined by the sum of expected processing times of all jobs in $H(j,i)$, and all the jobs in $L(j,i)$ will be delayed in expectation by an additional time $\mathbb{E}[P_{ij}]$, this cost increase equals

$$\operatorname{cost}(j \to i) := w_j \left(\sum_{k \to i, k \le j, k \in H(j,i)} \mathbb{E}[P_{ik}] \right) + \mathbb{E}[P_{ij}] \sum_{k \to i, k < j, k \in L(j,i)} w_k.$$

The greedy algorithm assigns the job to one of the machines where this quantity is minimal; that is, a job is assigned to machine $m(j) := \arg\min_{i \in M}\{\operatorname{cost}(j \to i)\}$; ties are broken arbitrarily. Once all jobs have arrived and are assigned, the jobs assigned to a fixed machine are sequenced in nonincreasing order of their ratio of weight over expected processing time. This WSEPT ordering is optimal conditioned on the given assignment (Rothkopf [35]).

The analysis of this greedy algorithm will proceed by defining a dual solution (α, β) in a way similar to that done in Anand et al. [1]. Let

$$\alpha_j := \operatorname{cost}(j \to m(j)) \quad \text{for all } j \in J.$$

That is, α_j is defined as the instantaneous expected increase in the total weighted completion time on the machine job j is assigned to by the greedy algorithm. Let

$$\beta_{is} := \sum_{j \in A_i(s)} w_j,$$

where $A_i(s)$ is defined as the total set of jobs assigned to machine i by the greedy algorithm but restricted to those that have not yet been completed by time s if the jobs' processing times were their expected values $\mathbb{E}[P_{ij}]$. In other words, β_{is} is exactly the expected total weight of yet unfinished jobs on machine i at time s, given the assignment (and sequencing) of the greedy algorithm.

It is now shown that these dual variables are feasible for the dual LP. Later this fact will allow us to relate the variables to the optimal solution's objective.

Lemma 3. *The solution* $(\boldsymbol{\alpha}/2, \boldsymbol{\beta}/2)$ *is feasible for* (D).

Proof. This proof shows that

$$\frac{\alpha_j}{\mathbb{E}[P_{ij}]} \le \beta_{is} + w_j \left(\frac{s}{\mathbb{E}[P_{ij}]} + 1 \right) \tag{8}$$

holds for all $i \in M$, $j \in J$, and $s \in \mathbb{Z}_{\ge 0}$. This implies the feasibility of $(\boldsymbol{\alpha}/2, \boldsymbol{\beta}/2)$ for (D). Fix a job j and machine i, and recall that $k \to i$ denotes a job k being assigned to machine i by the greedy algorithm. By definition of α_j and by choice of $m(j)$ as the minimizer of $\operatorname{cost}(j \to i)$, for all i, it is the case that

$$\frac{\alpha_j}{\mathbb{E}[P_{ij}]} \le \frac{\operatorname{cost}(j \to i)}{\mathbb{E}[P_{ij}]} = w_j + w_j \sum_{k \to i, k < j, k \in H(j,i)} \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} + \sum_{k \to i, k < j, k \in L(j,i)} w_k. \tag{9}$$

Next, we are going to argue that the right-hand side of (9) is upper bounded by the right-hand side of (8), from which the claim follows. Observe that the term w_j cancels. Observe that any job $k \to i$, $k \ne j$, can appear on the right-hand side of (9) at most once, either with value w_k, namely, when $k \in L(j,i)$, or with value $w_j \mathbb{E}[P_{ik}]/\mathbb{E}[P_{ij}] \le w_k$, when $k \in H(j,i)$. We show that each of these values on the right-hand side of (9) is accounted for on the right-hand side of (8), for any $s \ge 0$.

Fix any such job $k \to i$. First consider the case where the time s is small enough that our job $k \to i$ is still alive at time s, so $s < \sum_{\ell \to i, \ell \in H(k,i)} \mathbb{E}[P_{i\ell}]$. Then w_k is accounted for in the definition of β_{is}.

Now consider the case where $s \ge \sum_{\ell \to i, \ell \in H(k,i)} \mathbb{E}[P_{i\ell}]$, which means that job k is already finished at time s. In this case, we distinguish two cases. □

Case 1. $k \in L(j,i)$. In this case, job k contributes to the right-hand side of (9) a value of w_k, but as $s \geq \sum_{\ell \to i, \ell \in H(k,i)} \mathbb{E}[P_{i\ell}]$, the term $w_j(s/\mathbb{E}[P_{ij}])$ on the right-hand side of (8) contains the term $w_j(\mathbb{E}[P_{ik}]/\mathbb{E}[P_{ij}]) \geq w_k$.

Case 2. $k \in H(j,i)$. In this case, job k contributes to the right-hand side of (9) a value of $w_j(\mathbb{E}[P_{ik}]/\mathbb{E}[P_{ij}])$, which is exactly what is also contained in the term $w_j(s/\mathbb{E}[P_{ij}])$, because $s \geq \sum_{\ell \to i, \ell \in H(k,i)} \mathbb{E}[P_{i\ell}]$. □

In the following lemma, the online algorithm's objective is expressed in terms of the dual variables, which follows more or less directly from the definition of the dual variables $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Let us denote by ALG the total expected value achieved by the greedy algorithm.

Lemma 4. *The total expected value of the greedy algorithm is*

$$\mathsf{ALG} = \sum_{j \in J} \alpha_j = \sum_{i \in M} \sum_{s \in \mathbb{Z}_{\geq 0}} \beta_{is}.$$

Proof. For the first equality, recall that α_j is the instantaneous increase in ALG's expected total weighted completion time. Summing this over all jobs gives exactly the total expected value of ALG's objective. For a formal proof of this, see, for example, Megow et al. [29, lemma 4.1] for the case of parallel identical machines. That lemma and its proof can directly be applied to the case of unrelated machines.

The second equality follows from the fact that the (expected) total weighted completion time of any schedule can be alternatively expressed by weighting each period of time by the total weight of yet unfinished jobs. The equality is true here because $\boldsymbol{\beta}$ was defined on the basis of the same distribution of jobs over machines as given by ALG and because each job k's weight w_k, given $k \to i$, appears in β_{is} for all times s up to a job k's expected completion time, given jobs' processing times are fixed to their expected values. This is exactly what happens in computing the expected completion times under the greedy algorithm because it is a "fixed assignment" algorithm that assigns all jobs to machines at time 0 and sequences the jobs per machine thereafter. □

5. Speed Augmentation and Analysis

The preceding analysis of the dual feasible solution $(\boldsymbol{\alpha}/2, \boldsymbol{\beta}/2)$ yields a dual objective value equal to 0 by Lemma 4. This is of little help to bound the algorithm's performance. However, following Anand et al. [1], define another dual solution that has an interpretation in the model where all machines run at faster speed $f \geq 1$, meaning in particular that all (expected) processing times get scaled (down) by a factor f.

Define ALG^f as the expected solution value obtained by the same greedy algorithm, except that all the machines run at a speed increased by a factor of f, where $f \geq 1$ is an integer. Note that $\mathsf{ALG} = f\mathsf{ALG}^f$ by definition. We denote by $(\boldsymbol{\alpha}^f, \boldsymbol{\beta}^f)$ the exact same dual solution that was defined before, only for the new instance with faster machines. The following establishes feasibility of a slightly modified dual solution.

Lemma 5. *Whenever $f \geq 2$, the solution $(\boldsymbol{\alpha}^f, \frac{1}{f}\boldsymbol{\beta}^f)$ is a feasible solution for the dual (D) in the original (unscaled) problem instance.*

Proof. By definition of $(\boldsymbol{\alpha}^f, \frac{1}{f}\boldsymbol{\beta}^f)$, to show feasibility for (D), it suffices to show the slightly stronger constraint that

$$\frac{\alpha_j^f}{\mathbb{E}[P_{ij}]} \leq \frac{1}{f}\beta_{is}^f + w_j\left(\frac{s}{\mathbb{E}[P_{ij}]} + \frac{1}{2}\right)$$

for all i, j, s. Indeed, in this inequality, we have only dropped the nonnegative term $w_j/(2\mathbb{E}[P_{ij}])$ from the right-hand side of (D); hence, the above implies the feasibility of $(\boldsymbol{\alpha}^f, \frac{1}{f}\boldsymbol{\beta}^f)$ for (D). By definition of $\boldsymbol{\alpha}$, we have $\alpha_j = f\alpha_j^f$. So the above is equivalent to

$$\frac{\alpha_j}{\mathbb{E}[P_{ij}]} \leq \beta_{is}^f + w_j\left(\frac{f \cdot s}{\mathbb{E}[P_{ij}]} + \frac{f}{2}\right). \tag{10}$$

Because the assumption was that $f \geq 2$, (10) is implied by

$$\frac{\alpha_j}{\mathbb{E}[P_{ij}]} \leq \beta_{is}^f + w_j\left(\frac{f \cdot s}{\mathbb{E}[P_{ij}]} + 1\right). \tag{11}$$

Now observe that $\beta_{is}^f = \beta_{i(f\cdot s)}$ (and recall that f is integer), so (11) is nothing but inequality (8) with variable s replaced by $f \cdot s$. The validity of (11) therefore follows directly from (8) in our earlier proof of Lemma 3 to demonstrate the feasibility of $(\boldsymbol{\alpha}/2, \boldsymbol{\beta}/2)$ for (D). □

The first main theorem of the paper is now established.

Theorem 1. *The greedy algorithm has a performance guarantee of* $(4 + 2\Delta)$ *for online scheduling of stochastic jobs on unrelated machines to minimize the expectation of the total weighted completion times* $\mathbb{E}[\sum_j w_j C_j]$; *that is,* $\mathsf{ALG} \le (4 + 2\Delta)\mathsf{OPT}$.

Proof. We know from Corollary 1 that $z^D(\boldsymbol{\alpha}^f, \frac{1}{f}\boldsymbol{\beta}^f) \le z^D = z^P \le \left(1 + \frac{\Delta}{2}\right)\mathsf{OPT}$, given that $f \ge 2$. Next, recall that $\mathsf{ALG}^f = \sum_{j\in J} \alpha_j^f = \sum_{i\in M} \sum_{s\in\mathbb{Z}_{\ge 0}} \beta_{is}^f$ by Lemma 4, and $\mathsf{ALG} = f\mathsf{ALG}^f$. The theorem now follows from evaluating the objective value of the specifically chosen dual solution $(\boldsymbol{\alpha}^f, \frac{1}{f}\boldsymbol{\beta}^f)$ for (D), as

$$z^D\left(\boldsymbol{\alpha}^f, \frac{1}{f}\boldsymbol{\beta}^f\right) = \sum_{j\in J} \alpha_j^f - \frac{1}{f}\sum_{i\in M}\sum_{s\in\mathbb{Z}_{\ge 0}} \beta_{is}^f = \frac{f-1}{f}\mathsf{ALG}^f = \frac{f-1}{f^2}\mathsf{ALG}.$$

Putting together this equality with the previous inequality yields a performance bound equal to $\frac{f^2}{f-1}(1 + \frac{\Delta}{2})$, where we have the constraint that $f \ge 2$. This term is minimal and equal to $(4 + 2\Delta)$, exactly when we choose $f = 2$. □

We end this section with the following theorem, which we believe was unknown before.

Theorem 2. *The greedy algorithm for the deterministic online scheduling problem has competitive ratio 4 for minimizing the total weighted completion times* $\sum_j w_j C_j$ *on unrelated machines, and there is a tight lower bound of 4 for the performance of the greedy algorithm.*

Proof. The upper bound follows as a special case of Theorem 1 as $\Delta = 0$. As to the lower bound, we use a parametric instance from Correa and Queyranne [9], which we briefly reproduce here for convenience. The instances are denoted by I^k, where $k \in \mathbb{N}$. There are m machines, with m defined large enough that $m/h^2 \in \mathbb{N}$ for all $h = 1, \ldots, k$. There are jobs $j = (h, \ell)$ for all $h = 1, \ldots, k$ and all $\ell = 1, \ldots, m/h^2$. The processing times of a job $j = (h, \ell)$ on a machine i is defined as

$$p_{ij} = \begin{cases} 1 & \text{if } i \le \ell, \\ \infty & \text{otherwise.} \end{cases}$$

In other words, job $j = (h, \ell)$ can be processed only on machines $1, \ldots, \ell$. All jobs have weight $w_j = 1$. Because jobs have unit length on the machines on which they can be processed, we assume that the greedy algorithm breaks ties on each machine so that jobs with larger second index ℓ go first.

The optimal schedule is to assign all jobs $j = (h, \ell)$ to machine ℓ, resulting in m/h^2 jobs finishing at time h, for $h = 1, \ldots, k$, and hence a total cost $m \sum_{h=1}^k 1/h$. Now assume that the online sequence of jobs is by decreasing order of their second index. Then, because this is the same priority order as on each of the machines, the greedy algorithm assigns each job at the end of all previously assigned jobs. This means that the greedy algorithm assigns each job j to one of the machines that minimizes its own completion time C_j. Here we assume that ties are broken in favor of lower machine index. It is shown in Correa and Queyranne [9] that the resulting schedule, which is in fact a Nash equilibrium in the game where jobs select machine to minimize their own completion times, has a total cost of at least $4m \sum_{i=1}^k 1/i - \mathrm{O}(m)$. The lower bound of 4 follows by letting $k \to \infty$. □

6. The Online-Time Model

This section addresses the online-time model, where jobs arrive over time; that is, a job j arrives at release time $r_j \ge 0$. In particular, the presence of job j is unknown before time r_j. Upon time r_j, the job becomes available, and the processing time distributions P_{ij} become known for all machines $i = 1, \ldots, m$. We may assume w.l.o.g. that jobs are indexed such that $r_j \le r_k$ for $j < k$.

The difficulty in analyzing the problem where jobs arrive over time lies in jobs that block a machine for a long time, while shortly after, other jobs might be released that cannot be scheduled. This is a well-known problem for the total weighted completion time objective in general, even for a single machine (Megow and Schulz [27]). To counter this effect, a job j is started only after an additional, forced delay that depends on its own expected processing time. For example, for identical machine problems, Megow and Schulz [27] and Megow et al. [29] work with modified release times of the form $r'_j := \max\{r_j, c\mathbb{E}[P_j]\}$, for some parameter $c \ge 1$.

Another idea to counter the same effect was used in Schulz [36], namely, to start a job no earlier than its (expected) starting time in a preemptive relaxation on a single machine that works m times faster. For the unrelated machine problem that we consider here, we use a combination of these two ideas. The assignment of jobs to machines will follow the same idea as for the case without release dates, namely, to assign each job to a machine where (an approximation of) the expected increase of the objective value is minimal. Once jobs are assigned to machines, for the stochastic case, the modified release times will be defined on the basis of a nominal schedule, where processing times P_{ij} are fixed at their expected values $\mathbb{E}[P_{ij}]$. For this reason, this section first considers the deterministic problem where the processing times are defined by $p_{ij} := \mathbb{E}[P_{ij}]$ for all jobs j and machines i.

6.1. Nominal Schedule: Online-Time Model with Deterministic Processing Times

Let us first describe the greedy algorithm that is used to assign jobs to machines and schedule jobs on machines. Per machine, it is actually the same greedy WSPT rule that prefers to schedule jobs with the highest ratios of weight over processing time w_j/p_{ij}, with the only difference that we also take into account modified release times. The assignment of jobs to machines is done greedily, too.

6.1.1. Greedy Algorithm (Online-Time Model for Deterministic Processing Times). Consider any fixed job j that is released at time $t = r_j$ with processing times p_{ij} on machines $i = 1, \dots, m$. Then we proceed as follows:

1. Define modified release times. On machine i, the release time of job j is modified to $r_{ij} := \max\{r_j, c \cdot p_{ij}\}$. We will optimize parameter $c \geq 1$ later.
2. Let $U_i(t)$ denote the jobs that have been assigned to machine i at time t and that have not been started yet (excluding the fixed job j). Let $X_i(t)$ denote the remaining processing time of the job that is potentially being processed at time t on machine i. If there is no such job, $X_i(t) = 0$.
3. Consider a hypothetical single machine schedule of the job set $U_i(t) \cup \{j\}$ on machine i, namely, according to the greedy WSPT rule, with job release times r_{ik}, where we include the job of processing time $X_i(t)$ as being scheduled at time t. We also consider the same hypothetical single machine schedule *without* job j. Again, let $\text{cost}(j \to i)$ be the instantaneous increase in the total weighted completion times on machine i as a result of including job j, that is, the cost difference between these two schedules.
4. Assign job j to a machine i that minimizes $\text{cost}(i \to j)$, among all machines $i \in \{1, \dots, m\}$; ties are broken arbitrarily.
5. On each machine i, schedule jobs following the greedy WSPT rule with modified release times r_{ij}; that is, as soon as a machine falls idle at time t, schedule, among all unscheduled jobs k assigned to machine i with $r_{ik} \leq t$, any job j with maximal ratio w_k/p_{ik}.

6.1.2. Analysis. We now show that this greedy online algorithm is 6-competitive. This is interesting in its own right because it improves on the best prior algorithm that was known to be 8-competitive (Hall et al. [14]). As before, let us denote by ALG the total value achieved by the greedy algorithm and by OPT the optimal solution value.

Our first observation is to bound the increase in the total weighted completion times on machine i resulting from including job j.

Lemma 6.

$$\begin{aligned}\text{cost}(i \to j) &\leq w_j \left(r_{ij} + p_{ij} + X_i(r_j) + \sum_{k \in U_i(r_j),\, \frac{w_k}{p_{ik}} \geq \frac{w_j}{p_{ij}}} p_{ik} \right) + \sum_{k \in U_i(r_j),\, \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij} \\ &\leq w_j \left(\left(1 + \frac{1}{c}\right) r_j + (1 + c) p_{ij} + \sum_{k \in U_i(r_j),\, \frac{w_k}{p_{ik}} \geq \frac{w_j}{p_{ij}}} p_{ik} \right) + \sum_{k \in U_i(r_j),\, \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij}.\end{aligned}$$

Proof. The first inequality follows by definition of the expected increase that is caused by including job j simply by considering the latest possible completion time of job j caused by jobs with higher priority and the additional delay that job j imposes on jobs with lower priority. The second inequality is true as a result of the definition of the modified release times because with h being the job in process at time r_j, it must hold that

$$X_i(r_j) \leq p_{ih} \leq \frac{r_{ih}}{c} \leq \frac{r_j}{c}.$$

Here the last inequality must be true because job h was available before time r_j. □

Note the following: it may, of course, happen that some jobs of lower priority can get delayed by even more than p_{ij} because the presence of job j on machine i can cause other higher-priority jobs to become available later on, thereby causing low-priority jobs to be delayed even further. But such an increase will be accounted for at the time that such jobs appear; the total increase in total weighted completion time caused by job j *at time* r_j is bounded as above.

Theorem 3. *The greedy algorithm for the deterministic online scheduling problem with release times has competitive ratio 6 for minimizing the total weighted completion times* $\sum_j w_j C_j$ *on unrelated machines; that is,* $\mathsf{ALG} \le 6\mathsf{OPT}$.

Proof. Let $m(k)$ be the machine to which job k got assigned. As before, define α_j as the actual increase in total weighted completion time resulting from the presence of job j and $\beta_{i,s}$ as the sum of weights of unfinished jobs on machine i at time s; that is,

$$\alpha_j := \text{cost}(j \to m(j)),$$
$$\beta_{i,s} := \sum_{k:m(k)=i;\ r_k \le s;\ C_k \ge s} w_k.$$

As before, we clearly have

$$\mathsf{ALG} = \sum_j \alpha_j = \sum_{i,s} \beta_{i,s}.$$

For this analysis, we again consider a speed-scaled problem instance, but now we need to modify both the release times and the processing times by a factor f as

$$r_j^f := \frac{r_j}{f} \text{ and } p_{ij}^f := \frac{p_j}{f},$$

so that we have

$$r_{ij}^f = \frac{r_{ij}}{f}.$$

Observe that the machine assignment in the speed-scaled instance is exactly the same as in the original instance. In fact, the speed-scaled instance just scales time by a factor of f. Define, α_j^f and $\beta_{i,s}^f$ analogously as the increase in total weighted completion time resulting from the presence of job j in the speed-scaled instance and $\beta_{i,s}^f$ as the weight of the unfinished jobs on machine i at time s in the speed-scaled instance. Then

$$\begin{aligned} \alpha_j^f &= \frac{\alpha_j}{f}, \\ \beta_{is}^f &= \beta_{i(f\cdot s)}. \end{aligned} \tag{12}$$

(Here we assume w.l.o.g. that all job sizes are integer multiples of f, which can be achieved by scaling.) Also, let us denote by ALG^f the value achieved by the greedy algorithm for the speed-scaled instance, and note that $\mathsf{ALG}^f = \sum_j \alpha_j^f = \sum_{i,s} \beta_{i,s}^f = \mathsf{ALG}/f$.

In the next section, we are going to prove Lemma 7, which gives a lower bound on the optimal solution value OPT, again via some feasible solution for the dual of a linear programming relaxation of the form $\left(\frac{\alpha^f}{a}, \frac{\beta^f}{b}\right)$ for some constants (a, b), which will yield that

$$\mathsf{OPT} \ge \sum_j \frac{\alpha_j^f}{a} - \sum_{i,s} \frac{\beta_{is}^f}{b} = \frac{\mathsf{ALG}}{f}\left(\frac{1}{a} - \frac{1}{b}\right),$$

or

$$\mathsf{ALG} \le \frac{f \cdot \mathsf{OPT}}{\frac{1}{a} - \frac{1}{b}}.$$

Now, setting parameters $c = 1$, $a = 4/3, b = 4$, and speed $f = 3$ is a feasible choice for using Lemma 7, which gives $\mathsf{ALG} \le 6 \cdot \mathsf{OPT}$. □

6.2. Linear Programming Relaxation and Dual Lower Bound

Analogous to the earlier linear programming relaxation (S), we can define the same linear programming relaxation for the instance with release times r_j. We omit repeating this linear programming relaxation here because it is exactly the same as (S), except that the variables y_{ijs} are defined only for times $s \geq r_j$. Let us refer to this modified linear programming relaxation for the problem with release dates (S_r) and its optimal solution value z^{S_r}. Similarly, analogous to (P), we can define a linear programming relaxation for the deterministic version of the same problem with deterministic processing times $\mathbb{E}[P_{ij}]$ by dropping all terms $-\mathbb{C}\mathbb{V}[P_{ij}]^2$ from the relaxation (S_r) and eliminating constraints (5). Let us refer to this deterministic linear programming relaxation (P_r) with optimal solution value z^{P_r}. Lemma 2 and Corollary 1 apply to this linear programming relaxation in exactly the same way as before; that is, when OPT denotes the expected value of an optimal stochastic scheduling policy for the unrelated machine scheduling problem with release dates, we have that

$$z^{P_r} \leq \left(1 + \frac{\Delta}{2}\right) z^{S_r} \leq \left(1 + \frac{\Delta}{2}\right) \mathsf{OPT}. \tag{13}$$

Specifically, for the purpose of the proof of Theorem 3, observe that for the case of deterministic processing times where $\Delta = 0$, the optimal linear programming solution value z^{P_r} is simply a lower bound for OPT.

6.2.1. Dual Lower Bound. By duality, we can lower bound the optimal solution value z^{P_r} for linear programming relaxation (P_r) by any feasible solution to its dual LP, which is

$$\begin{aligned} \max \quad & z^{D_r} = \sum_{j \in J} \alpha_j - \sum_{i \in M} \sum_{s \in \mathbb{Z}_{\geq 0}} \beta_{is} \\ \text{s.t.} \quad & \frac{\alpha_j}{\mathbb{E}[P_{ij}]} \leq \beta_{is} + w_j \left(\frac{s + \frac{1}{2}}{\mathbb{E}[P_{ij}]} + \frac{1}{2} \right) && \text{for all } i \in M, j \in J, s \in \mathbb{Z}_{\geq r_j}, \\ & \beta_{is} \geq 0 && \text{for all } i \in M, s \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{D_r}$$

Lemma 7. *With $\boldsymbol{\alpha}^f$ and $\boldsymbol{\beta}^f$ as defined in (12), the values $(\frac{\boldsymbol{\alpha}^f}{a}, \frac{\boldsymbol{\beta}^f}{b})$ are a feasible solution for the dual LP (D_r) given that $af \geq 2(1+c)$, $c(a-1)f \geq 1$, and $af \geq b$. Specifically for $c = 1$, $a = 4/3$, $b = 4$, and speed $f = 3$, the objective value of the dual solution equals $z^{D_r}(\frac{\boldsymbol{\alpha}^f}{a}, \frac{\boldsymbol{\beta}^f}{b}) = \frac{\mathsf{ALG}}{f}(\frac{1}{a} - \frac{1}{b})$.*

Proof. We are only left to show the feasibility of the solution $(\frac{\boldsymbol{\alpha}^f}{a}, \frac{\boldsymbol{\beta}^f}{b})$. For convenience, let us write p_{ij} for $\mathbb{E}[P_{ij}]$. Then the dual constraints require that, for all jobs j and machines i, and for all times $s \geq r_j$,

$$\frac{\alpha_j}{p_{ij}} \leq \beta_{is} + w_j \frac{s + \frac{1}{2}}{p_{ij}} + w_j \cdot \frac{1}{2}. \tag{14}$$

Let us fix job j and machine i. Plugging in the values α_j^f/a and β_{is}^f/b, we need to show that

$$\frac{\alpha_j^f}{a \cdot p_{ij}} \leq \frac{\beta_{is}^f}{b} + w_j \frac{s + \frac{1}{2}}{p_{ij}} + w_j \cdot \frac{1}{2} \tag{15}$$

for all $s \geq r_j$. Equivalently, noting that $\boldsymbol{\alpha}^f = \boldsymbol{\alpha}/f$, we have to show that

$$\frac{\alpha_j}{p_{ij}} \leq af \cdot \frac{\beta_{is}^f}{b} + w_j \frac{s + \frac{1}{2}}{p_{ij}} \cdot af + w_j \cdot \frac{af}{2}. \tag{16}$$

Because $\beta_{is}^f = \beta_{i,fs}$ (the version with machines' speeds scaled by f is just scaling down time by a factor of f), and replacing $s + 1/2$ by s, it suffices to show that

$$\frac{\alpha_j}{p_{ij}} \leq af \cdot \frac{\beta_{i,fs}}{b} + w_j \frac{s}{p_{ij}} \cdot af + w_j \cdot \frac{af}{2} \tag{17}$$

for all $s \geq r_j$. Because of Lemma 6 and our choice of α_j as minimizer of cost$(j \to i)$, we have that

$$\frac{\alpha_j}{p_{ij}} \leq \frac{w_j\left(\left(1+\frac{1}{c}\right)r_j + (1+c)p_{ij} + \sum_{k\in U_i(r_j),\, \frac{w_k}{p_{ik}} \geq \frac{w_j}{p_{ij}}} p_{ik}\right) + \sum_{k\in U_i(r_j),\, \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij}}{p_{ij}}. \tag{18}$$

Hence, it suffices to show that the right-hand side of (18) is upper bounded by the right-hand side of (17). To that end, we even show that a slightly stronger inequality is true; recall that $\beta_{i,fs}$ is the total weight of jobs k assigned to machine i and unfinished at time fs but with $r_k \leq fs$. Because $f \geq 1$ and $r_j \leq s$, we have $r_j \leq fs$. Hence, $\beta_{i,fs} \geq \sum_{k:m(k)=i,\, r_k \leq r_j,\, C_k \geq fs} w_k \geq \sum_{k\in U_i(r_j),\, C_k \geq fs} w_k$. Therefore, it suffices to show that the right-hand side of (18) is bounded from above by

$$\begin{aligned}
&\frac{af}{b}\cdot \sum_{k\in U_i(r_j), C_k \geq fs} w_k + w_j \frac{s}{p_{ij}}\cdot af + w_j \cdot \frac{af}{2} \\
&= \left(\frac{af}{b}\cdot \sum_{k\in U_i(r_j), C_k \geq fs} w_k + \frac{w_j}{p_{ij}}\cdot (fs - r_j)\right) + \frac{w_j}{p_{ij}}\cdot (fs(a-1) + r_j) + w_j \cdot \frac{af}{2}.
\end{aligned}$$

This means that we need to argue that the following inequality is true:

$$\begin{aligned}
&w_j\left(\left(1+\frac{1}{c}\right)r_j + (1+c)p_{ij} + \sum_{k\in U_i(r_j),\, \frac{w_k}{p_{ik}} \geq \frac{w_j}{p_{ij}}} p_{ik}\right) + \sum_{k\in U_i(r_j),\, \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij} \\
&\qquad \leq \left(\frac{af}{b}\cdot \sum_{k\in U_i(r_j): C_k \geq fs} w_k p_{ij} + w_j \cdot (fs - r_j)\right) + w_j \cdot (fs(a-1) + r_j) + w_j \cdot \frac{af}{2}\cdot p_{ij}.
\end{aligned}$$

Let us rewrite this more conveniently as

$$\begin{aligned}
&\underbrace{w_j r_j}_{I} + \underbrace{\frac{w_j r_j}{c}}_{II} + \underbrace{w_j(1+c)p_{ij}}_{III} + \underbrace{w_j \sum_{k\in U_i(r_j),\, \frac{w_k}{p_{ik}} \geq \frac{w_j}{p_{ij}}} p_{ik}}_{IV} + \underbrace{\sum_{k\in U_i(r_j),\, \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij}}_{V} \\
&\qquad \leq \underbrace{\frac{af}{b}\cdot \sum_{k\in U_i(r_j): C_k \geq fs} w_k p_{ij}}_{V} + \underbrace{w_j \cdot (fs - r_j)}_{IV} + \underbrace{w_j \cdot (fs(a-1))}_{II} + \underbrace{w_j r_j}_{I} + \underbrace{w_j \cdot \frac{af}{2}\cdot p_{ij}}_{III}.
\end{aligned} \tag{19}$$

The following observations and conditions are sufficient for this inequality to be true:

1. Terms I: These terms are identical.
2. Terms II: Noting that $r_j \leq s$, it suffices that $1/c \leq f(a-1)$.
3. Terms III: It suffices that $2(1+c) \leq af$.
4. Terms IV + V: We have by definition of $U_i(r_j)$ that

$$w_j(fs - r_j) \geq w_j \sum_{k\in U_i(r_j), C_k < fs} p_{ik}.$$

Therefore, under the assumption that $af/b \geq 1$, the terms IV + V on the right-hand side of (19) are indeed larger because jobs with lower priority than j that complete before time fs have a contribution $w_k p_{ij}$ to the left-hand side but a larger contribution $w_j p_{ik}$ to the right-hand side because for these jobs $w_k p_{ij} < w_j p_{ik}$. □

6.3. Online-Time Model with Stochastic Processing Times

Let us first describe how we modify the greedy algorithm from the preceding section for the case with stochastic processing times.

6.3.1. Greedy Algorithm (Online-Time Model with Stochastic Processing Times). When job sizes are stochastic, we use exactly the same greedy assignment of jobs to machines as we used in the preceding section for the deterministic case using processing times $p_{ij} := \mathbb{E}[P_{ij}]$. The only difference lies in the scheduling of jobs per machine, which works by restricting jobs to start no earlier than in the nominal schedule with deterministic processing times $p_{ij} = \mathbb{E}[P_{ij}]$. Specifically, the jobs assigned to any machine i are scheduled exactly in the same order as in the nominal schedule, with the ℓth job to start on machine i at time

$$S_{i,\ell} = \max\{s_{i,\ell}, S_{i,\ell-1} + P_{i,\ell-1}\}.$$

Here $s_{i,\ell}$ denotes the deterministic starting time of the ℓth job in the nominal schedule where $p_{ij} = \mathbb{E}[P_{ij}]$ for all jobs j and machines i. Here note that the identity of the ℓth job to be scheduled on machine i is the same in both cases. Also note that for the greedy algorithm for the stochastic case, the assignment of jobs to machines is deterministic and not dependent on the realized processing times of jobs. The following two remarks are probably helpful.

Remark 1. One may wonder if and how the algorithm can actually be executed online. This simply works by concurrently building the greedy WSPT schedule with deterministic processing times $p_{ij} := \mathbb{E}[P_{ij}]$. Consider any job j that was released at time r_j. For the assignment of job j to its correct machine $i = m(j)$, indeed only information that is available at time r_j is needed. Also observe that it may be the case that neither the value $s_{i,j}$ is necessarily known at time r_j nor which of the jobs are the predecessors of job j on machine i. But this is not necessary because job j is simply blocked for processing as long as the same job has not started being processed in the corresponding deterministic schedule.

Remark 2. Observe that we may introduce forced idleness before the processing of any job j; that is, even if the machine $i = m(j)$ is idle, we might not process any of the available jobs, and this delay depends on the nominal schedule for the underlying deterministic instance with $p_{ij} = \mathbb{E}[P_{ij}]$. One may wonder why this forced idleness is actually necessary. Apart from the analysis that is to come, the reason to do this can most easily be seen by considering the following example. There are n^2 "bad" jobs of weight $\epsilon \ll 1$ released at time 0 with independent and identically distributed processing requirements $P_{\text{bad}} = 0$ with probability $1 - 1/n^2$ and $P_{\text{bad}} = n$ with probability $1/n^2$ and one "good" job released at time 1 with weight 1 and a deterministic processing time of 1. With the proposed algorithm that never starts a job before its starting time in the nominal schedule, we can schedule at most n bad jobs before the good job is released because $\mathbb{E}[P_{\text{bad}}] = 1/n$. This yields $\mathbb{E}[C_{\text{good}}] = \mathrm{O}(1)$. However, without this forced idle time, a greedy algorithm would keep scheduling bad jobs until there were none (if all were of size 0) or a rare long bad job was encountered. This would yield $\mathbb{E}[C_{\text{good}}] = \Omega(n)$, which is problematic.

The analysis of the greedy algorithm for the stochastic setting is based on a comparison with the nominal schedule, as expressed in the following lemma.

Lemma 8. *The expected starting time of a job j on machine i in the stochastic case is bounded in terms of its starting time in the underlying nominal schedule*[2] *by $\mathbb{E}[S_{i,j}] \le h(\Delta)s_{i,j}$, where*

$$h(\Delta) = \begin{cases} 1 + \dfrac{\sqrt{\Delta}}{2}, & \Delta \le 1, \\ 1 + \dfrac{\Delta}{\Delta + 1}, & \Delta \ge 1. \end{cases}$$

Observe that $h(\cdot)$ is a concave increasing function of Δ, that $h(\Delta) \le 2$ for all $\Delta \ge 0$, and that $h(0) = 1$. Specifically, Lemma 8 implies the weaker bound $\mathbb{E}[S_{i,j}] \le 2s_{i,j}$.

Proof. For simplicity of notation, let us say that the jobs $k = 1, \dots, j$ are the jobs that have been assigned to machine i in this order. By definition of the algorithm for the stochastic setting, and by the fact that both the assignment to machines and the sequencing per machine are identical to those in the nominal schedule, the following equality holds per realization of the processing times:

$$\begin{aligned} S_{i,j} &= \max\{s_{i,j}, S_{i,j-1} + P_{i,j-1}\} \\ &= \max\{s_{i,j}, s_{i,j-1} + P_{i,j-1}, s_{i,j-2} + P_{i,j-2} + P_{i,j-1}, \dots, P_{i,1} + \cdots + P_{i,j-1}\} \\ &=: F_{i,j}(P_{i,1}, P_{i,2}, \cdots, P_{i,j-1}). \end{aligned}$$

Noting that the function $F_{i,j}$ is nondecreasing and Lipschitz continuous with coefficient 1 in each of its coordinates, and $F_{i,j}(\mathbb{E}[P_{i,1}],\dots,\mathbb{E}[P_{i,j-1}]) = s_{i,j}$, we have

$$\begin{aligned} F_{i,j}(P_{i,1},\dots,P_{i,j-1}) &= F_{i,j}\big(\mathbb{E}[P_{i,1}],\dots,\mathbb{E}[P_{i,j-1}]\big) + \big(F_{i,j}(P_{i,1},\dots,P_{i,j-1}) - F_{i,j}(\mathbb{E}[P_{i,1}],\dots,\mathbb{E}[P_{i,j-1}])\big) \\ &\le s_{i,j} + \sum_{k=1}^{j-1}\big(P_{i,k} - \mathbb{E}[P_{i,k}]\big)^+. \end{aligned}$$

Lemma A.2., which is proved in the appendix, yields $\mathbb{E}[(P_{i,k} - \mathbb{E}[P_{i,k}])^+] \le (h(\Delta) - 1)\mathbb{E}[P_{i,k}]$. Hence, taking expectations, we get

$$\begin{aligned} \mathbb{E}[S_{i,j}] &\le s_{i,j} + \sum_{k=1}^{j-1}\mathbb{E}[P_{i,k}](h(\Delta) - 1) \\ &\le h(\Delta)s_{i,j}. \end{aligned}$$

The last inequality holds because for the nominal schedule, the jth job cannot begin before time $\sum_{k=1}^{j-1}\mathbb{E}[P_{ik}]$, which means that $s_{i,j} \ge \sum_{k=1}^{j-1}\mathbb{E}[P_{ik}]$. □

We conclude with the main theorem of this section.

Theorem 4. *The greedy algorithm has a performance guarantee of* $(6 + 3\Delta)h(\Delta)$ *for online scheduling of stochastic jobs with release times on unrelated machines to minimize the expectation of the total weighted completion times* $\mathbb{E}[\sum_j w_j C_j]$; *that is,* $\mathsf{ALG} \le (6 + 3\Delta)h(\Delta)\mathsf{OPT}$.

Proof. Let us denote by C_j^P the completion time of job j in the nominal schedule as computed by the greedy algorithm as described in Section 6.1, where $p_{ij} = \mathbb{E}[P_{ij}]$. Let us denote by $\mathsf{ALG}_P = \sum_j w_j C_j^P$ the objective value achieved by that nominal schedule. Also, let us denote by $\mathsf{ALG} = \sum_j w_j \mathbb{E}[C_j^S]$ the expected performance of the greedy algorithm for the stochastic case as described in this section.

It follows from Lemma 8 that the expected completion time of any job j under the greedy algorithm for the stochastic case fulfills $\mathbb{E}[C_j] \le h(\Delta)C_j^P$, and therefore,

$$\mathsf{ALG} \le h(\Delta)\mathsf{ALG}_P.$$

What we have shown in Lemma 7 is that there exists a solution to the dual linear programming relaxation (D_r) with value equal to $\mathsf{ALG}_P/6$. Therefore, by linear programming duality, we get that $\mathsf{ALG}_P \le 6z^{P_r}$, with z^{P_r} being the optimal solution value for the linear programming relaxation (P_r). This yields

$$\mathsf{ALG} \le h(\Delta)\mathsf{ALG}_P \le h(\Delta)6z^{P_r} \le h(\Delta)6\left(1 + \frac{\Delta}{2}\right)z^{S_r} \le (6 + 3\Delta)h(\Delta)\mathsf{OPT}.$$

Here the third inequality follows by (13). □

7. Conclusions

The performance guarantees for the greedy algorithm obtained in this paper are in the order $\mathrm{O}(\Delta)$, which is the same order of magnitude as earlier results that have been obtained for offline problems on unrelated machines (Skutella et al. [40]) and of the same order of magnitude as earlier bounds for the online identical machines setting (Megow et al. [29]). Getting results independent of Δ is an interesting open problem.

We also believe that the presented competitive analyses for the online deterministic problems are interesting in their own right, even if better competitive ratios can be obtained. We think so because the proposed greedy algorithm is (arguably) simple and intuitive and hence practical. Finding a (matching) lower bound for the case with release times would be interesting.

Another direction for future work is the derivation of genuine lower bounds on the approximability of the optimal expected performance of efficiently computable policies for stochastic scheduling problems, even in the offline setting. This would allow us to separate the computational complexity of stochastic problems from the corresponding deterministic special cases.

Acknowledgments

This work was started while all four authors were with the Simons Institute for the Theory of Computing at the University of California Berkeley. The authors thank the institute for the financial support and the organizers of the semester on algorithms

and uncertainty for providing a very stimulating atmosphere. B. Moseley was employed at Washington University in St. Louis while some of this research was conducted. V. Gupta acknowledges support from the Booth School of Business, University of Chicago. All the authors express their gratitude to the referees for their helpful comments and for challenging them to improve their results. A conference publication with preliminary results appeared in the 2017 Proceedings of the Conference on Integer Programming and Combinatorial Optimization (Gupta et al. [12]).

Appendix. Auxiliary Lemmas

Lemma A.1. *We focus on a single machine and job. Let P denote the random variable for the processing time with support $\mathbb{Z}_{>0}$. Let x_t denote the probability that a job starts processing on the machine at time t ($t = 0, 1, \ldots$). For a given set of $\{x_t\}$ variables, let y_s denote the probability that the job is being processed on the machine during time slot s. Then the expected completion time of the job is given by*

$$C = \sum_{s \in \mathbb{Z}_{\geq 0}} \left(\frac{y_s}{\mathbb{E}[P]} \left(s + \tfrac{1}{2}\right) + \frac{1 - \mathbb{CV}[P]^2}{2} y_s \right).$$

Proof. It follows from the fact that policies are nonanticipatory that in terms of $\{x_t\}$ variables, the expected completion time is

$$C = \sum_{t=0}^{\infty} x_t (t + \mathbb{E}[P]).$$

Furthermore, from (1),

$$y_s = \sum_{t=0}^{s} x_t \cdot \mathbb{P}[P > s - t],$$

which also gives

$$\sum_{s=0}^{\infty} y_s = \mathbb{E}[P] \sum_{t=0}^{\infty} x_t.$$

Consider the summation

$$\begin{aligned}
\sum_{s=0}^{\infty} y_s \left(s + \frac{1}{2}\right) &= \sum_{s=0}^{\infty} \left(s + \frac{1}{2}\right) \sum_{t=0}^{s} x_t \cdot \mathbb{P}[P > s - t] \\
&= \sum_{t=0}^{\infty} x_t \sum_{s=t}^{\infty} \left(s + \frac{1}{2}\right) \mathbb{P}[P > s - t] \\
&= \sum_{t=0}^{\infty} x_t \left(t \sum_{r=0}^{\infty} \mathbb{P}[P > r] + \sum_{r=0}^{\infty} \left(r + \frac{1}{2}\right) \mathbb{P}[P > r] \right) \\
&= \sum_{t=0}^{\infty} x_t \left(t \cdot \mathbb{E}[P] + \frac{1}{2} \mathbb{E}[P^2] \right) \\
&= \mathbb{E}[P] \sum_{t=0}^{\infty} x_t \cdot t + \frac{1}{2} \mathbb{E}[P^2] \sum_{t=0}^{\infty} x_t \\
&= \mathbb{E}[P] \left(\sum_{t=0}^{\infty} x_t \cdot t + \frac{1 + \mathbb{CV}[P]^2}{2} \sum_{s=0}^{\infty} y_s \right)
\end{aligned}$$

or

$$\sum_{t=0}^{\infty} x_t \cdot t = \sum_{s=0}^{\infty} \left(\frac{y_s}{\mathbb{E}[P]} \left(s + \frac{1}{2}\right) - \frac{1 + \mathbb{CV}[P]^2}{2} y_s \right).$$

Adding $\sum_{t=0}^{\infty} x_t \mathbb{E}[P] = \sum_{s=0}^{\infty} y_s$ to the above gives

$$C = \sum_{t=0}^{\infty} x_t (t + \mathbb{E}[P]) = \sum_{s=0}^{\infty} \left(\frac{y_s}{\mathbb{E}[P]} \left(s + \frac{1}{2}\right) + \frac{1 - \mathbb{CV}[P]^2}{2} y_s \right). \quad \square$$

Lemma A.2. *Let X be a nonnegative random variable with mean μ and squared coefficient of variation Δ. Then*

$$\mathbb{E}[(X - \mu)^+] \leq \begin{cases} \mu \dfrac{\sqrt{\Delta}}{2} & \Delta \leq 1, \\ \mu \dfrac{\Delta}{\Delta + 1} & \Delta \geq 1. \end{cases} \tag{A.1}$$

Proof. We consider the problem of finding the measure on $\mathbb{R}_+$ for X that maximizes $\mathbb{E}[(X-\mu)^+]$. Assuming that X has a density, this can be written as an infinite-dimensional linear program. Using duality, we provide these bounds by exhibiting feasible solutions to its dual. It is instructive to follow this path, and later we argue that the bound holds for any distribution of X. We note that the bounds are in fact tight, which can be easily proved by analyzing the distributions in the duality-based proof, but we omit this step here.

We begin with the primal problem of finding the extremal measure on $\mathbb{R}_+$ as an infinite-dimensional LP, assuming the extremal measure has density $f(\cdot)$:

$$\begin{aligned} \max_{f(\cdot)} \quad & \int_0^\infty (x-\mu)^+ f(x)dx \\ \text{s.t.} \quad & \int_0^\infty f(x) = 1, \\ & \int_0^\infty x \cdot f(x)dx = \mu, \\ & \int_0^\infty x^2 \cdot f(x)dx = \mu^2(1+\Delta). \end{aligned}$$

The dual problem to this is

$$\begin{aligned} \min_{\alpha,\beta,\gamma} \quad & \alpha + \mu\beta + \mu^2(1+\Delta)\gamma \\ \text{s.t.} \quad & \alpha + \beta x + \gamma x^2 \geq (x-\mu)^+ \qquad \text{for all } x \geq 0. \end{aligned}$$

Case $\Delta \geq 1$. We begin by making a guess about the extremal distribution $f(\cdot)$, namely, that it is a parametric distribution with as limiting case two atoms, one of which is at 0. Under this assumption, one observes that $\mathbb{P}[X=0] = \frac{\Delta}{\Delta+1}$, and $\mathbb{P}[X=\mu(\Delta+1)] = \frac{1}{\Delta+1}$ yields the desired conditions that $\mathbb{E}[X] = \mu$, $\mathbb{E}[X^2] = \mu^2(\Delta+1)$, and $\mathbb{E}[(X-\mu)^+] = \mu\frac{\Delta}{\Delta+1}$. For the given guess, complementary slackness implies that

$$\alpha + \beta \cdot 0 + \gamma \cdot 0 = 0, \tag{A.2}$$

$$\alpha + \beta \cdot \mu(\Delta+1) + \gamma \cdot \mu^2(\Delta+1)^2 = \mu\Delta. \tag{A.3}$$

The first of the above gives $\alpha = 0$, and the second gives the dual objective value of $\mu\frac{\Delta}{\Delta+1}$. It now remains to verify dual feasibility. Dual feasibility is met if the gradient of the quadratic function $\alpha + \beta x + \gamma x^2$ at $x = 0$ is nonnegative, and it is tangent to $(x-\mu)^+$ at $x = \mu(\Delta+1)$. The latter implies that

$$\beta + 2\gamma\mu(\Delta+1) = 1, \tag{A.4}$$

And this, together with (A.3), gives, $\beta = \frac{\Delta-1}{\Delta+1}$ and $\gamma = \frac{1}{\mu(\Delta+1)^2}$. The nonnegativity of the gradient at $x = 0$ is true if and only if $\beta \geq 0$. Therefore, if $\Delta \geq 1$, then $\alpha = 0, \beta = \frac{\Delta-1}{\Delta+1}, \gamma = \frac{1}{\mu(\Delta+1)^2}$ is a feasible dual solution with objective value $\mu\frac{\Delta}{\Delta+1}$.

To turn this idea into a formal proof, we claim that for any $x \geq 0$ and $\Delta \geq 1$, we have

$$(x-\mu)^+ \leq \frac{\Delta-1}{\Delta+1}x + \frac{1}{\mu(\Delta+1)^2}x^2.$$

This follows from $(x-\mu(\Delta+1))^2 \geq 0$ by adding $2x = (\Delta+1)x + (\Delta-1)x$ to both sides. Then basic algebra yields $(x-\mu) \leq (\Delta-1)x/(\Delta+1) + x^2/(\mu(\Delta+1)^2)$, where we assume w.l.o.g. that $\mu > 0$. Therefore, for any random variable $X \geq 0$ with $\mathbb{E}[X] > 0$, we have

$$\mathbb{E}\big[(X-\mu)^+\big] \leq \frac{\Delta-1}{\Delta+1}\mathbb{E}[X] + \frac{1}{\mu(\Delta+1)^2}\mathbb{E}\big[X^2\big].$$

Substituting $\mathbb{E}[X] = \mu, \mathbb{E}[X^2] = \mu^2(1+\Delta)$, we get

$$\mathbb{E}\big[(X-\mu)^+\big] \leq \mu\frac{\Delta}{\Delta+1},$$

which proves the second case of (A.1).

Case $\Delta \leq 1$. The constraints of the dual suggest that we should look for a quadratic function $\alpha + \beta x + \gamma x^2$ that is tangent to $(x-\mu)^+$ at two points, one of which must be in the interval $[0,\mu]$. Therefore, $\alpha + \beta x + \gamma x^2 = \gamma(x-\nu_1)^2$ for some $0 \leq \nu_1 \leq \mu$. Let this quadratic be tangent to $(x-\mu)$ at $\nu_2 \geq \mu$. The tangency conditions give

$$\gamma(\nu_2-\nu_1)^2 = \nu_2 - \mu, \tag{A.5}$$

$$2\gamma(\nu_2-\nu_1) = 1, \tag{A.6}$$

which together imply that $\gamma = \frac{1}{4(\mu-\nu_1)}$. Thus, the dual minimization problem becomes the following single-parameter optimization problem over ν_1:

$$\min_{\nu_1} \underbrace{\frac{\nu_1^2}{4(\mu-\nu_1)}}_{=:\alpha(\nu_1)} + \mu \underbrace{\left(-\frac{\nu_1}{2(\mu-\nu_1)}\right)}_{=:\beta(\nu_1)} + \mu^2(\Delta+1)\underbrace{\frac{1}{4(\mu-\nu_1)}}_{=:\gamma(\nu_1)}$$
$$= \min_{\nu_1} \frac{\mu^2\Delta}{4(\mu-\nu_1)} + \frac{\mu-\nu_1}{4} = \frac{\mu\sqrt{\Delta}}{2}.$$

The minimizer is $\nu_1^* = \mu(1-\sqrt{\Delta})$, which is indeed in the interval $[0,\mu]$ for $\Delta \le 1$, as desired for dual feasibility.

As before, we can turn this into a formal proof by showing that for any $x \ge 0$ and $0 \le \Delta \le 1$,

$$(x-\mu)^+ \le \frac{1}{4(\mu-\nu)}(x-\nu)^2,$$

where we have defined $\nu = \mu(1-\sqrt{\Delta})$. Then, for any random variable $X \ge 0$ with $\mathbb{E}[X] > 0$,

$$\mathbb{E}[(X-\mu)^+] \le \mathbb{E}\left[\frac{1}{4(\mu-\nu)}(X-\nu)^2\right].$$

After substituting $\mathbb{E}[X] = \mu, \mathbb{E}[X^2] = \mu^2(1+\Delta)$, this gives

$$\mathbb{E}[(X-\mu)^+] \le \mu\frac{\sqrt{\Delta}}{2}.$$

This completes the proof for the first case of (A.1).

Remark A.1. It was pointed out to us by one of the referees that the bound for the case $\Delta \le 1$ follows even more simply by observing $\mathbb{E}[(X-\mu)^+] = \frac{1}{2}\mathbb{E}[|X-\mu|_2] \le \frac{1}{2}\sqrt{\mathbb{E}[(X-\mu)^2]} = \frac{\mu\sqrt{\Delta}}{2}$. Here the first equality follows because $X-\mu$ has mean zero, and the inequality is by Jensen's inequality. The linear programming argument, however, is constructive in the sense that it also gives tight examples. □

Endnotes

[1] The ratio is slightly better, but for simplicity, we ignore the additive $\Theta(1/m)$ term.

[2] We write $S_{i,j}$ to indicate that job j was assigned to machine i, only for notational convenience. Because the assignment of jobs to machines is deterministic, observe that $S_j = S_{i,j}$.

References

[1] Anand S, Garg N, Kumar A (2012) Resource augmentation for weighted flow-time explained by dual fitting. Rabani Y, ed. *Proc. 23rd Annual ACM-SIAM Sympos. Discrete Algorithms* (Association for Computing Machinery, New York), 1228–1241.

[2] Avrahami N, Azar Y (2007) Minimizing total flow time and total completion time with immediate dispatching. *Algorithmica* 47(3):253–268.

[3] Balseiro S, Brown D, Chen C (2018) Static routing in stochastic scheduling: Performance guarantees and asymptotic optimality. *Oper. Res.* 66(6):1641–1660.

[4] Bansal N, Srinivasan A, Svensson O (2016) Lift-and-round to improve weighted completion time on unrelated machines. Wichs D, Mansour Y, eds. *Proc. 48th Ann. ACM Sympos. Theory Computing (STOC)* (Association for Computing Machinery, New York), 156–167.

[5] Becchetti L, Leonardi S (2001) Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines. Vitter J, Spirakis P, Yannakiakis M, eds. *Proc. 33rd Annual ACM Sympos. Theory Comput.* (Association for Computing Machinery, New York), 94–103.

[6] Bruno J, Downey PJ, Frederickson G (1981) Sequencing tasks with exponential service times to minimize the expected flowtime or makespan. *J. ACM* 28(1):100–113.

[7] Chakrabarti S, Phillips CA, Schulz AS, Shmoys DB, Stein C, Wein J (1996) Improved scheduling algorithms for minsum criteria. Meyer auf der Heide F, Monien B, eds. *Automata, Languages and Programming—ICALP 1996*, Lecture Notes in Computer Science, vol. 1099 (Springer, Berlin), 646–657.

[8] Cole R, Correa JR, Gkatzelis V, Mirrokni VS, Olver N (2011) Inner product spaces for minsum coordination mechanisms. Fortnow L, Vadhan SP, eds. *Proc. 43rd ACM Sympos. Theory Comput.* (Association for Computing Machinery, New York), 539–548.

[9] Correa J, Queyranne M (2012) Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. *Naval Res. Logist.* 59(5):384–395.

[10] Correa J, Wagner M (2008) LP-based online scheduling: From single to parallel machines. *Math. Programming* 119(1):109–136.

[11] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.* 5:287–326.

[12] Gupta V, Moseley B, Xie Q, Uetz M (2017) Stochastic online scheduling on unrelated machines. Eisenbrand F, Koennemann J, eds. *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, vol. 10328 (Springer, Berlin), 228–240.

[13] Gupta A, Im S, Krishnaswamy R, Moseley B, Pruhs K (2012) Scheduling heterogeneous processors isn't as easy as you think. *Proc. 23rd Annual ACM-SIAM Sympos. Discrete Algorithms* (Association for Computing Machinery, New York), 1242–1253.

[14] Hall LA, Schulz AS, Shmoys DB, Wein J (1997) Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Math. Oper. Res.* 22(3):513–544.
[15] Horn W (1973) Minimizing average flowtime with parallel machines. *Oper. Res.* 21(3):846–847.
[16] Horowitz E, Sahni S (1976) Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM* 23(2):317–327.
[17] Im S, Li S (2016) Better unrelated machine scheduling for weighted completion time via random offsets from non-uniform distributions. Dinur I, ed. *Proc. IEEE 57th Annual Sympos. Foundations Comput. Sci.* (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 138–147.
[18] Im S, Moseley B, Pruhs K (2011) A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*. 42(2):83–97.
[19] Im S, Moseley B, Pruhs K (2015) Stochastic scheduling of heavy-tailed jobs. Mayr EW, Ollinger N, eds. *Proc. 32nd Sympos. Theoret. Aspects Comput. Sci.*, vol. 30 (Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany), 474–486.
[20] Im S, Kulkarni J, Munagala K, Pruhs K (2014) Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. *Proc. 55th IEEE Annual Sympos. Foundations Comput. Sci.* (Institute of Electrical and Electronics Engineers, Piscataway, NJ) 531–540.
[21] Jäger S, Skutella M (2018) Generalizing the Kawaguchi-Kyan bound to stochastic parallel machine scheduling. Niedermeier R, Vallée B, eds. *35th Sympos. Theoret. Aspects Comput. Sci.*, Leibniz International Proceedings in Informatics, vol. 96 (Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), 43:1–43:14.
[22] Kalyanasundaram B, Pruhs K (2000) Speed is as powerful as clairvoyance. *J. ACM* 47(4):617–643.
[23] Kämpke T (1987) On the optimality of static priority policies in stochastic scheduling on parallel machines. *J. Appl. Probab.* 24(2):430–448.
[24] Lenstra J, Shmoys DB, Tardos É (1990) Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming* 46: 259–271.
[25] Leung JYT, ed. (2004) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (Chapman & Hall/CRC, Boca Raton, FL).
[26] Li S (2017) Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. Umans C, ed. *Proc. 58th IEEE Annual Sympos. Foundations Comput. Sci*. (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 283–294.
[27] Megow N, Schulz A (2004) On-line scheduling to minimize average completion time revisited. *Oper. Res. Lett.* 32(5):485–490.
[28] Megow N, Vredeveld T (2011) A tight 2-approximation for preemptive stochastic scheduling. *Math. Oper. Res.* 39(4):1297–1310.
[29] Megow N, Uetz M, Vredeveld T (2006) Models and algorithms for stochastic online scheduling. *Math. Oper. Res.* 31(3):513–525.
[30] Möhring RH, Radermacher FJ, Weiss G (1984) Stochastic scheduling problems I: General strategies. *Zeitschrift für Oper. Res.* 28:193–260.
[31] Möhring RH, Radermacher FJ, Weiss G (1985) Stochastic scheduling problems II: Set strategies. *Zeitschrift für Oper. Res.* 29:65–104.
[32] Möhring RH, Schulz AS, Uetz M (1999) Approximation in stochastic scheduling: The power of LP-based priority policies. *J. ACM* 46(6):924–942.
[33] Motwani R, Phillips S, Torng E (1994) Non-clairvoyant scheduling. *Theoret. Comput. Sci.* 130(1):17–47.
[34] Pruhs K, Sgall J, Torng E (2004) Online scheduling. Leung J Y-T, ed. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (CRC Press Inc., Boca Raton, FL), 115–124.
[35] Rothkopf MH (1966) Scheduling with random service times. *Management Sci.* 12(9):703–713.
[36] Schulz AS (2008) Stochastic online scheduling revisited. Yang B, Du DZ, Wang C, eds. *Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, vol. 5165 (Springer, Berlin), 448–457.
[37] Schulz AS, Skutella M (2002) Scheduling unrelated machines by randomized rounding. *SIAM J. Discrete Math.* 15(4):450–469.
[38] Skutella M (2001) Convex quadratic and semidefinite programming relaxations in scheduling. *J. ACM* 48(2):206–242.
[39] Skutella M, Uetz M (2005) Stochastic machine scheduling with precedence constraints. *SIAM J. Comput.* 34(4):788–802.
[40] Skutella M, Sviridenko M, Uetz M (2016) Unrelated machine scheduling with stochastic processing times. *Math. Oper. Res.* 41(3):851–864.
[41] Uetz M (2003) When greediness fails: Examples from stochastic scheduling. *Oper. Res. Lett.* 31(6):413–419.
[42] Vestjens A (1997) Online machine scheduling. PhD thesis, TU Eindhoven, Eindhoven, Netherlands.
[43] *Wall Street* (1987) Stone O, dir. Twentieth Century Fox, Los Angeles.
[44] Weber R, Varaiya P, Walrand J (1986) Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected owtime. *J. Appl. Probab.* 23(3):841–847.
[45] Weiss G, Pinedo M (1980) Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *J. Appl. Probab.* 17(1):187–202.