MATHEMATICS OF OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0364-765X | EISSN 1526-5471 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000
© 0000 INFORMS

# Greed Works* – Online Algorithms For Unrelated Machine Stochastic Scheduling

Varun Gupta
University of Chicago, varun.gupta@chicagobooth.edu,

Benjamin Moseley
Carnegie Mellon University, moseleyb@andrew.cmu.edu

Marc Uetz
University of Twente, m.uetz@utwente.nl

Qiaomin Xie
Massachusetts Institute of Technology, qxie@mit.edu

This paper establishes the first performance guarantees for a combinatorial online algorithm that schedules 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 sophisticated linear or convex programming relaxations for the assignment of jobs to machines. The algorithm introduced in this paper is based on a combinatorial assignment of jobs to machines, hence it also works online. The performance bounds are of the same order of magnitude as those of earlier work, and depend linearly on an upper bound $\Delta$ on the squared coefficient of variation of the jobs' processing times. They are $4+2\Delta$ when there are no release dates, and $(6+3\Delta)h(\Delta)$ when jobs are released over time. Here, $1 \le h(\cdot) \le 2$ is a continuous function of $\Delta$ with $h(0)=1$. Specifically for deterministic processing times, without and with release times, the competitive ratios are therefore 4 and 6, respectively. As to the technical contribution, the paper shows for the first time how dual fitting techniques can be used for stochastic and nonpreemptive scheduling problems.

## 1. Introduction.

Scheduling jobs on multiple, parallel machines is a fundamental problem both in combinatorial optimization and systems theory. There is a vast amount of different model variants as well as applications, which is testified by the existence of the handbook [25]. A well studied class of problems is scheduling a set of $n$ nonpreemptive jobs that arrive over time on $m$

* Gordon Gekko (Michael Douglas) in Oliver Stone's "Wall Street" (Twentieth Century Fox, 1987).

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 $\mathrm{R} \,|\, r_j \,|\, \sum w_j C_j$ 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, e.g. [4, 16, 24, 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, refer to [19, 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 non-clairvoyant schedulers have been studied that do not know the processing time of a job until the job is completed [33, 5, 22, 12, 17]. Unfortunately, if preemption is not allowed then any algorithm has poor performance in the non-clairvoyant model, as 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 non-clairvoyant model is too pessimistic for non-preemptive 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 *non-anticipatory* 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 *non-preemptively*, and it is only when the job completes that the scheduler learns the actual processing time of the job. Both the scheduler and the optimal solution are non-anticipatory, which roughly means that the future is uncertain for both, the scheduler and the adversary. Stochastic scheduling has been well-studied, including fundamental work such as [30, 31] and approximation algorithms, e.g. [32, 40, 28, 39, 36].

This paper considers online scheduling of non-preemptive, stochastic jobs in the unrelated machine model to minimize the total weighted completion time. This is the same problem as considered in the paper [28] by Megow et al., but here we address the more general *unrelated* machines 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.

**Identical machines, special processing time distributions.** Restricting attention to non-preemptive 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 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 [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 [15]. For multiple identical machines with equal weights for all jobs, SEPT is optimal if job sizes are exponentially distributed [6, 44], or more generally, are stochastically comparable in pairs [43]. Some extensions of these optimality results to the problem with weights exist as well [23]. For more general distributions, simple solutions fail [41], and our knowledge of optimal scheduling policies is limited.

**Identical machines, arbitrary processing times.** To cope with these challenges, approximation algorithms have been studied. With the notable exception of [20], all approximation algorithms have performance guarantees that depend on an upper bound $\Delta$ on the squared coefficient of variation of the underlying random variables. Möhring, Schulz and Uetz [32] established the first approximation algorithms for stochastic scheduling on identical machines via a linear programming relaxation. Their work gave a $(3+\Delta)$-approximation when jobs are released over time (yet known offline), and they additionally showed that WSEPT is a $\frac{(3+\Delta)}{2}$-approximation when jobs arrive together[1]. These results have been built on and generalized in several settings [40, 29, 28, 36, 39, 21], notably in [28, 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 gives an algorithm with performance guarantee of $(2.309+1.309\Delta)$ [36]. These results build on an idea from Correa and Wagner [10] to use a preemptive, fast single machine relaxation, next to the relaxation of [32]. The work of Im, Moseley and Pruhs [20] gave the first results independent of $\Delta$ showing that there exist poly-logarithmic approximation algorithms under some assumptions. All these papers address problems with identical machines.

**Unrelated machines, arbitrary processing times.** For some 15 years after the results of Möhring et al. [32] for the identical machines case, no non-trivial results were known for the *unrelated* machines case despite being a target in the area. Recently Skutella, Sviridenko and Uetz [39] gave a $\frac{3+\Delta}{2}$-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).

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

Central to unlocking an efficient approximation algorithm for the unrelated machines case was the introduction of a time-indexed linear program that lower bounds the objective value of the optimal non-anticipatory scheduling policy. It is this LP that allows the authors to overcome the complexities of the stochastic unrelated machines setting.

The work introduced in this present 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 small competitive ratio for this problem. Prior work for the offline problem requires sophisticated linear [39] or convex [3] programming relaxations. Good candidates for online algorithms that might also have practical impact are desired to 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.

**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, Schulz, Shmoys and Wein [14] obtain an 8-competitive algorithm. Their algorithm is based on the idea to partition time into geometrically increasing intervals, and then maximizing the total weight of (available) jobs that can be scheduled in these intervals. Algorithms with better competitive ratios have been obtained by Chakrabarti et al. [7] by using randomization in the definition of these intervals, and 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 [4] and more recently also on linear relaxations [26]. For the offline case with release times, the $(2+\varepsilon)$-approximation of [37] was the best known until recently Im and Li [18] 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] show that when machines follows the WSPT rule, Nash equilibria of selfish jobs that minimize their own completion time yield schedules which are at most a factor 4 above optimum. Interestingly, our paper shows that the same approximation guarantee can be obtained online by a simple greedy algorithm. We note that the work of [8] is offline, and moreover their algorithm and analysis differs from that 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.

**Results.** This paper suggests two combinatorial online algorithm for stochastic scheduling on unrelated machines that have a performance guarantee of order $\mathrm{O}(\Delta)$, where $\Delta$ 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 is a simple greedy assignment that assigns jobs to machines so as to minimize the expected contribution to the objective function, while 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 the 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, the 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 $\Delta$ which 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. For deterministic processing times $\Delta=0$ and $h(\Delta)=1$, hence the competitive ratio equals 6. As the algorithm is a deterministic algorithm, this improves upon the competitive ratio 8 from [14], and falls only slightly behind the randomized 5.78-competitive algorithm that was proposed in [7].

Even though the performance bounds for the case with nontrivial release times are most probably not tight, we believe our results are interesting for the following reasons: (1) It is the first analysis of a combinatorial algorithm for stochastic scheduling on unrelated machines, and the first result for stochastic online scheduling in the unrelated machine model. (2) Even for the deterministic setting, it is the first time to analyze 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 non-preemptive and stochastic scheduling. (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 algorithm rests on two ingredients. The first is that per machine $i$, at any point in time a job with highest ratio $w_j/\mathbb{E}[P_{ij}]$ is scheduled from the set of jobs available on that machine. This is the well known WSEPT rule. For the case with release dates, it needs to be slightly modified in that jobs will have to wait for "artificial" release times before they become available for processing. The necessity of such forced idle time is well known to be necessary for problems where jobs are released over time, even for single machine problems [27]. The second ingredient is the assignment to machines, which is here solved by greedily assigning jobs to the machines where the (approximate) expected increase of the objective is minimal. Comparable greedy-type algorithms were used also before, e.g. in [2, 27, 28], however not for an unrelated machine setting. Note that the $\Omega(\Delta)$ lower bound for fixed assignment policies in [39] yields that our results are asymptotically tight in $\Delta$ among policies that must irrevocably assign jobs to machines at the time of their release. As mentioned above, the analysis proposed in this paper uses dual fitting techniques. The technique has been used e.g. in [1] for deterministic and preemptive scheduling problems. This paper therefore also shows that dual fitting can be used for bounding the performance of algorithms even for non-preemptive and stochastic settings.

## 2. Notation & Preliminaries.

The input to the problem consists of a set of unrelated parallel machines $M$ of cardinality $m$. This paper considers two online models. In the first model, known as *online-list*, the scheduler is presented a sequence of 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 machine assignment is decided when a job arrives and the decision on the time the job begins being processed can be deferred. 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 scheduled on that machine. In the second model, known as *online-time*, 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 a later point in time to be processed. 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. Moreover, the jobs are stochastic, meaning that each job $j$'s processing time is revealed to the scheduler 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 only at the moment that the job completes. This paper considers designing a non-anticipatory scheduling policy $\Pi$ that minimizes the expected total weighted completion time $\mathbb{E}\left[\sum_j w_j C_j\right]$, where $C_j$ denotes the completion time of job $j$ in the schedule $\Pi$.

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 $\varepsilon > 0$ [39]. Our analysis will make use of the following facts about first and second moments of discrete random variables; these facts also appear in [39].

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

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

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 & Policies** The setting considered in this paper is that of stochastic online scheduling as defined in [28]. 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 $i = 1, \ldots, m$ are known to the scheduler. At any given time $t$, a non-anticipatory 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 sizes 1, 3 or 4 with probabilities 1/3 each, and has been running for 2 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 in order to minimize the expectation of the total weighted completion time, e.g. [41]. We refer the reader to [28] for a more thorough discussion of the stochastic online model.

For simplicity of notation, denote OPT as the expected total weighted completion time of an optimal, non-anticipatory 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 non-anticipatory online scheduling policy (an algorithm) $\mathsf{ALG}$ with expected performance $\mathsf{ALG}$ close to $\mathsf{OPT}$. 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 $\mathsf{ALG}$ and $\mathsf{OPT}$ denote the expected performance of non-anticipatory 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 $\mathsf{OPT}$ is not restricted to assigning jobs to machine at the time of their arrival. The only restriction on $\mathsf{OPT}$ is that it must schedule jobs nonpreemptively, and that it is non-anticipatory. 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 $P_{ij}$. That implies that the model generalizes the classic offline stochastic scheduling model (assuming all paramaters are disclosed to the scheduler, too), as well as traditional competitive analysis (assuming deterministic processing times).

Finally, we may assume 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 previously discussed in [39, §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

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

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 non-anticipatory 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 $i$, 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

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

where we labeled the expected completion time variables with a superscript S for "stochastic", for reasons that will become clear shortly. For completeness, equation (4) is proved in Lemma 9 (Appendix A).

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 LP relaxation for the unrelated machine scheduling problem can be derived with these observations. This LP extends the LP given in [39] by adding the constraints (5).

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

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. As 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

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

Consider the following linear programming problem

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

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

Definition 3. Define $\Delta$ 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 NBUE (new better than used in expectation), that is, the expected remaing processing time of a job never exceeds its total expected processing time. Specifically, $\Delta = 1$ for exponential distributions. Next, for any given solution $y$ of (S) or (P), define

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

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

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

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

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

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

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*

$$z^P \le \big(1+\frac{\Delta}{2}\big)\mathsf{OPT}\,.$$

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

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

Like the analysis in [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, both in deterministic and stochastic scheduling on parallel machines, e. g. in [2, 27, 28].

## 4. Greedy Algorithm & Analysis for the Online-List Model.

In this section the online-list model is considered. Assume without loss of generality that the jobs are presented in the order $1,2\dots,|J|$. 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,

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

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

**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 non-increasing order of the ratios weight over expected processing time. Since 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

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

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

The analysis of this greedy algorithm will proceed by defining a dual solution $(\alpha, \beta)$ in a way similar to that done in [1]. Let

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

That is, $\alpha_j$ 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, $\beta_{is}$ 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 linear program. Later this fact will allow us to relate the variables to the optimal solution's objective.

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

*Proof.* This proof shows that

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

holds for all $i \in M$, $j \in J$, and $s \in \mathbb{Z}_{\geq 0}$. This implies the feasibility of $(\alpha/2, \beta/2)$ 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 $\alpha_j$ and by choice of $m(j)$ as the minimizer of $\operatorname{cost}(j \to i)$, for all $i$ it is the case that

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

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 $k \to i$, $k \neq j$, can appear in the right-hand side of (9) at most once, either with value $w_k$, namely when $k \in L(j,i)$, or with value $w_j \mathbb{E}[P_{ik}]/\mathbb{E}[P_{ij}] \leq w_k$ when $k \in H(j,i)$. We show that each of these values in the right-hand-side of (9) is accounted for in the right-hand side of (8), for any $s \geq 0$.

Fix any such job $k \to i$. First consider the case that the time $s$ is small enough so 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 $\beta_{is}$.

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

*Case 1* is $k \in L(j,i)$: 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 $w_j(s/\mathbb{E}[P_{ij}])$ in the right-hand side of (8) contains the term $w_j(\mathbb{E}[P_{ik}]/\mathbb{E}[P_{ij}]) \geq w_k$.

*Case 2* is $k \in H(j,i)$: 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 directly follows more or less directly from the definition of the dual variables $(\alpha, \beta)$. Let us denote by $\mathsf{ALG}$ 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 $\alpha_j$ is the instantaneous increase in $\mathsf{ALG}$'s expected total weighted completion time. Summing this over all jobs gives exactly the total expected value of $\mathsf{ALG}$'s objective. For a formal proof of this, see for example [28, 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 $\beta$ 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 $\beta_{is}$ 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 also 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 & Analysis** The previous analysis of the dual feasible solution $(\alpha/2, \beta/2)$ yields a dual objective value equal to 0 by Lemma 4. This is of little help to bound the algorithm's performance. However following [1], define another dual solution which 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 $\mathsf{ALG}^f$ 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 $(\alpha^f, \beta^f)$ 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 $(\alpha^f, \frac{1}{f}\beta^f)$ is a feasible solution for the dual (D) in the* original *(unscaled) problem instance.*

*Proof.* By definition of $(\alpha^f, \frac{1}{f}\beta^f)$, 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 the above 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 $(\alpha^f, \frac{1}{f}\beta^f)$ for (D). By definition of $\alpha$ we have $\alpha_j = f\alpha_j^f$. So the above is equivalent to

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

As 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 $\beta^f_{is} = \beta_{i(f\cdot s)}$ (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 directly follows from (8) in our earlier proof of Lemma 3 to demonstrate the feasibility of $(\alpha/2, \beta/2)$ 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* $\mathbb{E}[\sum_j w_jC_j]$. *That is,* $\mathsf{ALG} \le (4+2\Delta)\mathsf{OPT}$.

*Proof.* We know from Corollary 1 that $z^D(\alpha^f, \frac{1}{f}\beta^f) \le z^D = z^P \le \big(1+\frac{\Delta}{2}\big)\mathsf{OPT}$, given that $f\ge 2$. Next, recall that $\mathsf{ALG}^f = \sum_{j\in J}\alpha^f_j = \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\ge 0}}\beta^f_{is}$ 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 $(\alpha^f, \frac{1}{f}\beta^f)$ for (D), as

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

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* $\sum_j w_jC_j$ *on unrelated machines, and there is a tight lower bound of* 4 *for the performance of the greedy algorithm.*

*Proof.* The upper bounds follows as a special case of Theorem 1 as $\Delta = 0$. As to the lower bound, we use a parametric instance from [9], which we briefly reproduce here for convenience. The instances are denoted $I^k$, where $k\in\mathbb{N}$. There are $m$ machines, with $m$ defined large enough so that $m/h^2\in\mathbb{N}$ for all $h = 1,\dots,k$. There are jobs $j=(h,\ell)$ for all $h=1,\dots,k$ and all $\ell = 1,\dots,m/h^2$. 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 only be processed on machines $1,\dots,\ell$. All jobs have weight $w_j = 1$. As 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 $\ell$ go first.

The optimal schedule is to assign all jobs $j=(h,\ell)$ to machine $\ell$, resulting in $m/h^2$ jobs finishing at time $h$, for $h=1,\dots,k$, 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, as 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. That 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 favour of lower machine index. It is shown in [9] that the resulting schedule, which is in fact a Nash equilibrium in the game where jobs select a machine to minimize their own completion time, has a total cost at least $4m\sum_{i=1}^{k} 1/i - \mathrm{O}(m)$. The lower bound of 4 follows by letting $k \to \infty$. $\square$

## 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 $r_j \geq 0$. In particular, the presence of job $j$ is unknown before time $r_j$. Upon time $r_j$, the job becomes available and the processing times distributions $P_{ij}$ become known, for all machines $i = 1, \ldots, m$. We may assume w.l.o.g. jobs are indexed such that $r_j \leq r_k$ 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 [27]. In order to counter that effect, a job $j$ is only started after an additional, forced delay that (typically) depends on its own expected processing time. For example for identical machine problems, [27] and [28] work with modified release times of the form $r'_j := \max\{r_j, c\mathbb{E}[P_j]\}$, for some parameter $c \geq 1$. Another idea to counter the same effect has been used in [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 a job to a machine where (an approximation of) the expected increase of the objective value is minimal. Once assigned to a machine, for the stochastic case the modified release times will be defined on the basis of a greedy schedule where processing times $P_{ij}$ are fixed at their expected values $\mathbb{E}[P_{ij}]$. For that 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. The 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 highest ratios 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.

**Greedy Algorithm (Online Time Model for Deterministic Processing Times).** Consider any fixed job $j$ that is 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 which 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 weighted shortest processing time rule (WSPT), 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 $j$. Let again $\mathrm{cost}(j \to i)$ be the instantaneous increase in the total weighted completion times on machine $i$ due to including job $j$, that is, the cost difference between these two schedules.

4. Assign job $j$ to a machine $i$ that minimizes $\mathrm{cost}(i \to j)$, among all machines $i \in \{1, \dots, m\}$; ties broken arbitrarily.

5. On each machine $i$, we schedule jobs following the greedy weighted shortest processing time rule (WSPT) with modified release times $r_{ij}$. That is, as soon as a machine falls idle at time $t$, we schedule among all unscheduled jobs $k$ assigned to machine $i$ with $r_{ik} \leq t$, any job $j$ with maximal ratio $w_k/p_{ik}$.

**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 [14]. As before, let us denote by $\mathsf{ALG}$ the total value achieved by the greedy algorithm, and $\mathsf{OPT}$ to be the optimal solution value.

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

Lemma 6.

$$\mathrm{cost}(i \to j) \leq w_j \left( r_{ij} + p_{ij} + X_i(r_j) + \sum_{k \in U_i(r_j), \frac{w_k}{p_{ik}} \geq \frac{w_j}{p_{ij}}} p_{ik} \right) + \sum_{k \in U_i(r_j), \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij}$$

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

*Proof.* The first inequality follows by definition of the expected increase that is caused by including job $j$, by simply 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 because of the definition of the modified release times, since with $h$ being the job in process at time $r_j$, it must hold that

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

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 that 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 $\sum_j w_jC_j$ 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 $\alpha_j$ as the actual increase in total weighted completion time due to the presence of job $j$, and $\beta_{i,s}$ is the sum of weights of unfinished jobs on machine $i$ at time $s$. That is,

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

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 follows.

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

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 the original instance. In fact, the speed scaled instance just scales time by a factor of $f$. Define, $\alpha_j^f$ and $\beta_{i,s}^f$ analogously as the increase in total weighted completion time due to 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 $\mathsf{ALG}^f$ 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 $\mathsf{OPT}$, again via some feasible solution for the dual of a linear programming relaxation of the form $\left(\frac{\alpha_j^f}{a}, \frac{\beta_{is}^f}{b}\right)$ for some constants $(a, b)$, which will yield that

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

or

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

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

**6.2. Linear Programming Relaxation and Dual Lower Bound.** Analogous to the earlier linear programming relaxation (S), we can define the same LP relaxation for the instance with release times $r_j$. We omit repeating this LP relaxation here as 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 LP relaxation for the problem with release dates (S$_r$), and its optimal solution value $z^{S_r}$. Similarly, analogous to (P) we can define an LP relaxation for the deterministic version of the same problem with deterministic processing times $\mathbb{E}[P_{ij}]$, by dropping all terms $-\mathbb{CV}[P_{ij}]^2$ from the relaxation (S$_r$), and eliminating constraints (5). Let us refer to this deterministic LP 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 $\mathsf{OPT}$ denotes the expected value of an optimal stochastic scheduling policy for the unrelated machine scheduling problem with release dates, we have that

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

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

**Dual Lower Bound.** By duality, we can lower bound the optimal solution value $z^{P_r}$ for LP relaxation ($\mathrm{P}_r$) by any feasible solution to its dual linear program, which is:

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

Lemma 7. *With $\alpha^f$ and $\beta^f$ as defined in (12), the values $\left(\frac{\alpha_j^f}{a}, \frac{\beta_{is}^f}{b}\right)$ are a feasible solution for the dual ($\mathrm{D}_r$), given that $af \geq 2(1+c)$, $(a-1)f \geq \frac{1}{c}$, and $af \geq b$.*

*Proof.* 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 $\alpha_j^f/a$ and $\beta_{is}^f/b$, we need to show

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

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

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

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

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

for all $s \geq r_j$. Due to Lemma 6, and our choice of $\alpha_j$ as minimizer of $\mathrm{cost}(j \to i)$ 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 in (18) is upper bounded by the right hand side in (17). To that end, we even show 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 $r_k \le fs$. Since $f \ge 1$ and $r_j \le s$, we have $r_j \le fs$. Hence, $\beta_{i,fs} \ge \sum_{k:m(k)=i, r_k \le r_j, C_k \ge fs} w_k \ge \sum_{k \in U_i(r_j), C_k \ge fs} w_k$. Therefore it suffices to show that the right hand side of (18) is bounded from above by

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

That means that we need to argue that the following inequality is true

$$w_j \left( \left(1 + \frac{1}{c}\right) r_j + (1+c)p_{ij} + \sum_{k \in U_i(r_j), \frac{w_k}{p_{ik}} \ge \frac{w_j}{p_{ij}}} p_{ik} \right) + \sum_{k \in U_i(r_j), \frac{w_k}{p_{ik}} < \frac{w_j}{p_{ij}}} w_k p_{ij}$$
$$\le \left( \frac{af}{b} \cdot \sum_{k \in U_i(r_j): C_k \ge fs} w_k p_{ij} + w_j \cdot (fs - r_j) \right) + w_j \cdot (fs(a-1) + r_j) + w_j \cdot \frac{af}{2} \cdot p_{ij} \,.$$

Let us rewrite this more conveniently as

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

The following observations and conditions are sufficient for the above inequality to be true.

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

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

Therefore, under the assumption that $\frac{af}{b} \ge 1$, the right hand side of the terms IV + V in (19) is indeed larger, because jobs with lower priority than $j$ which 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, as for these jobs $w_k p_{ij} < w_j p_{ik}$. □

**6.3. The 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.

**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 is the scheduling of jobs per machine, which works as follows. The jobs assigned to any machine $i$ are scheduled exactly in the same order as in the corresponding deterministic counterpart, with the $\ell$th job to start on machine $i$ restricted to starting 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 $\ell$th job in the corresponding deterministic problem where $p_{ij} = \mathbb{E}[P_{ij}]$ for all jobs $j$ and machines $i$. Here, note that the identity of the $\ell$th 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 is needed that is available at time $r_j$. 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, as 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 greedy WSPT 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 that 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 i.i.d. processing requirements $P_{bad} = 0$ with probability $1 - 1/n^2$ and $P_{bad} = n$ with probability $1/n^2$, and one "good" job released at time 1 with weight 1 and deterministic processing time of 1. With the proposed greedy algorithm that includes forced idleness we can schedule at most $n$ bad jobs before the good job is released, as $\mathbb{E}[P_{bad}] = 1/n$. That yields $\mathbb{E}[C_{good}] = \mathrm{O}(1)$. However without any forced idleness, a greedy algorithm keeps scheduling bad jobs until there are none (if all are of size 0), or a rare long bad job is encountered. That yields $\mathbb{E}[C_{good}] = \Omega(n)$, which is problematic.

The analysis of the greedy algorithm for the stochastic setting is based on a comparison with the underlying deterministic 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 deterministic schedule*[2] *by* $\mathbb{E}[S_{i,j}] \le s_{i,j} h(\Delta)$, *where*

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

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

*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 is identical to the deterministic 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 non-decreasing, Lipschitz continuous with coefficient 1 in each of its coordinates, and $F_{i,j}(\mathbb{E}[P_{i,1}], \cdots, \mathbb{E}[P_{i,j-1}]) = s_{i,j}$, we have:

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

Lemma 10, 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 since for the deterministic case, the $j$th job can not 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.

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

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* $\mathbb{E}[\sum_j w_jC_j]$*. 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 underlying deterministic schedule computed by the greedy algorithm as described in Section 6.1, where $p_{ij} = \mathbb{E}[P_{ij}]$. Let us denote by $\mathsf{ALG}_P = \sum_j w_jC_j^P$ the objective value achieved by that 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 fulfils $\mathbb{E}[C_j] \le h(\Delta)C_j^P$, 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 LP relaxation ($\mathrm{D}_r$) with value equal to $\mathsf{ALG}_P/6$. Therefore by LP duality we get that $\mathsf{ALG}_P \le 6z^{P_r}$, with $z^{P_r}$ being the optimal solution value for the LP relaxation ($\mathrm{P}_r$). That yields

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

Here, the third inequality follows by (13). □

## 7. Conclusions

The performance guarantees for the greedy algorithm obtained in this paper are in the order O( Δ ), which is the same order of magnitude as earlier results that have been obtained for offline problems on unrelated machines [39], and of the same order of magnitude as earlier bounds for the online identical machines setting [28]. 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 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 to separate the computational complexity of stochastic problems from the corresponding deterministic special cases.

**Acknowledgements.** This work was started while all four authors were with the Simons Institute for the Theory of Computing at UC Berkeley. The authors wish to thank the institute for the financial support and the organizers of the semester on "Algorithms & Uncertainty" for providing a very stimulating atmosphere. A conference publication with preliminary results appeared in the proceedings of IPCO 2017 [13]. B. Moseley was employed at Washington University in St. Louis while some of this research was conducted. B. Moseley was supported in part by a Google Research Award, a Yahoo Research Award and NSF Grant CCF-1830711, 1824303, and 1733873. All authors wish to express their gratitude to the referees for their helpful comments and for challenging us to improve the results.

## References

[1] S. Anand, N. Garg, and A. Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 1228–1241, 2012.

[2] N. Avrahami and Y. Azar. Minimizing total flow time and total completion time with immediate dispatching. In *Proc. 15th Symp. on Parallelism in Algorithms and Architectures (SPAA 2003)*, pages 11–18. ACM, 2003.

[3] S. Balseiro, D. Brown, and C. Chen. Static routing in stochastic scheduling: Performance guarantees and asymptotic optimality. Tech. Rep., 2016.

[4] N. Bansal, A. Srinivasan, and O. Svensson. Lift-and-round to improve weighted completion time on unrelated machines. In *Proc. 48th Ann. ACM Symp. Theory Computing (STOC)*, pages 156–167. ACM, 2016.

[5] L. Becchetti and S. Leonardi. Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines. In *STOC*, pages 94–103, 2001.

[6] J. Bruno, P. J. Downey, and G. Frederickson. Sequencing tasks with exponential service times to minimize the expected flowtime or makespan. *Journal of the ACM*, 28:100–113, 1981.

[7] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In F. M. auf der Heide and B. Monien, editors, *Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 646–657. Springer, 1996.

[8] R. Cole, J. R. Correa, V. Gkatzelis, V. S. Mirrokni, and N. Olver. Inner product spaces for minsum coordination mechanisms. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 539–548, 2011.

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

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

[11] 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.

[12] A. Gupta, S. Im, R. Krishnaswamy, B. Moseley, and K. Pruhs. Scheduling heterogeneous processors isn't as easy as you think. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 1242–1253, 2012.

[13] V. Gupta, B. Moseley, Q. Xie, and M. Uetz. Stochastic online scheduling on unrelated machines. In F. Eisenbrand and J. Koennemann, editors, *Integer Programming and Combinatorial Optimization*, volume 10328 of *Lecture Notes in Computer Science*, pages 228–240. Springer, 2017.

[14] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[15] W. Horn. Minimizing average flowtime with parallel machines. *Operations Research*, 21:846– 847, 1973.

[16] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.

[17] S. Im, J. Kulkarni, K. Munagala, and K. Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 531–540, 2014.

[18] S. Im and S. Li. Better unrelated machine scheduling for weighted completion time via random offsets from non-uniform distributions. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 138–147, 2016.

[19] S. Im, B. Moseley, and K. Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011.

[20] S. Im, B. Moseley, and K. Pruhs. Stochastic scheduling of heavy-tailed jobs. In *STACS*, 2015.

[21] S. Jäger and M. Skutella. Generalizing the Kawaguchi-Kyan bound to stochastic parallel machine scheduling. In R. Niedermeier and B. Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, Leibniz International Proceedings in Informatics, pages 43:1–43:14. Schloss Dagstuhl, 2018.

[22] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

[23] T. Kämpke. On the optimality of static priority policies in stochastic scheduling on parallel machines. *Journal of Applied Probability*, 24:430–448, 1987.

[24] J. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[25] J. Y.-T. Leung, editor. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC, 2004.

[26] S. Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 283–294, 2017.

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

[28] N. Megow, M. Uetz, and T. Vredeveld. Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research*, 31(3):513–525, 2006.

[29] N. Megow and T. Vredeveld. A tight 2-approximation for preemptive stochastic scheduling. *Mathematics of Operations Research*, 39:1297 – 1310, 2011.

[30] R. H. Möhring, F. J. Radermacher, and G. Weiss. Stochastic scheduling problems I: General strategies. *ZOR - Zeitschrift für Operations Research*, 28:193–260, 1984.

[31] R. H. Möhring, F. J. Radermacher, and G. Weiss. Stochastic scheduling problems II: Set strategies. *ZOR - Zeitschrift für Operations Research*, 29:65–104, 1985.

[32] R. H. Möhring, A. S. Schulz, and M. Uetz. Approximation in stochastic scheduling: The power of LP-based priority policies. *Journal of the ACM*, 46:924–942, 1999.

[33] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.

[34] K. Pruhs, J. Sgall, and E. Torng. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter Online Scheduling. 2004.

[35] M. H. Rothkopf. Scheduling with random service times. *Management Science*, 12:703–713, 1966.

[36] A. S. Schulz. Stochastic online scheduling revisited. In B. Yang, D.-Z. Du, and C. Wang, editors, *Combinatorial Optimization and Applications*, volume 5165 of *Lecture Notes in Computer Science*, pages 448–457. Springer, 2008.

[37] A. S. Schulz and M. Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15:450–469, 2002.

[38] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48:206–242, 2001.

[39] M. Skutella, M. Sviridenko, and M. Uetz. Unrelated machine scheduling with stochastic processing times. *Math. Oper. Res.*, 41(3):851–864, 2016.

[40] M. Skutella and M. Uetz. Stochastic machine scheduling with precedence constraints. *SIAM Journal on Computing*, 34:788–802, 2005.

[41] M. Uetz. When greediness fails: Examples from stochastic scheduling. *Operations Research Letters*, 31:413–419, 2003.

[42] A. Vestjens. *Online Machine Scheduling.* PhD thesis, TU Eindhoven, 1997.

[43] R. Weber, P. Varaiya, and J. Walrand. Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected owtime. *Journal of Applied Probability*, 23:841–847, 1986.

[44] G. Weiss and M. Pinedo. Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *Journal of Applied Probability*, 17:187–202, 1980.

## Appendix A: Auxiliary Lemmas

LEMMA 9. *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 the 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 non-anticipatory that in terms of $\{x_t\}$ variables, the expected completion time is

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

Further, 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

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

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

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

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

□

Lemma 10. *Let $X$ be a non-negative random variable with mean $\mu$ and squared coefficient of variation $\Delta$. Then,*

$$\mathbb{E}[(X-\mu)^+] \le \begin{cases} \mu\frac{\sqrt{\Delta}}{2} & \Delta \le 1, \\ \mu\frac{\Delta}{\Delta+1} & \Delta \ge 1. \end{cases} \tag{20}$$

*Proof.* The problem of finding the measure on $\mathbb{R}_+$ for $X$ that maximizes $\mathbb{E}[(X-\mu)^+]$ can be written as the following infinite dimensional linear rogram.

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

The dual problem to the above is:

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

We will exhibit the bound in the Lemma by using weak duality and exhibiting a feasible solution to the dual program.

**Case :** $\Delta \ge 1$. We begin by making an informed guess about the extremal distribution $f(\cdot)$. Namely, that it is a distribution with two atoms, one of which is at 0. That is, $\mathbb{P}[X=0] = \frac{\Delta}{\Delta+1}$ and $\mathbb{P}[X=\mu(\Delta+1)] = \frac{1}{\Delta+1}$. It is easy to verify that for this distribution $\mathbb{E}[X]=\mu$, $\mathbb{E}[X^2]=\mu^2(\Delta+1)$ and $\mathbb{E}[(X-\mu)^+] = \mu\frac{\Delta}{\Delta+1}$. For the given guess, complementary slackness implies:

$$\alpha + \beta\cdot 0 + \gamma\cdot 0 = 0 \tag{21}$$

$$\alpha+\beta\cdot\mu(\Delta+1)+\gamma\cdot\mu^2(\Delta+1)^2=\mu\Delta. \tag{22}$$

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 non-negative, and it is tangent to $(x-\mu)^+$ at $x=\mu(\Delta+1)$. The latter implies,

$$\beta+2\gamma\mu(\Delta+1)=1 \tag{23}$$

which together with (22) gives, $\beta=\frac{\Delta-1}{\Delta+1}$ and $\gamma=\frac{1}{\mu(\Delta+1)^2}$. The non-negativity of 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 $\mu\frac{\Delta}{\Delta+1}$ proving the second case of (20).

**Case: $\Delta\leq 1$.** The constraints of the dual suggest that we should look for a quadratic function $\alpha+\beta x+\gamma x^2$ which 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:

$$\gamma(\nu_2-\nu_1)^2=\nu_2-\mu \tag{24}$$

$$2\gamma(\nu_2-\nu_1)=1 \tag{25}$$

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

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

The minimizer is $\nu_1^*=\mu(1-\sqrt{\Delta})$ which is indeed in the interval $[0,\mu]$ for $\Delta\leq 1$ as desired for dual feasibility. This completes the first case of (20). □