Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0000-0000 | EISSN 0000-0000 | 00 | 0000 | 0001
DOI 10.1287/xxxx.0000.0000

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 purely 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 Δ on the squared coefficient of variation of the jobs' processing times. They are $4+2\Delta$ when there are no release dates, and $72+36\Delta$ when there are release dates. Bounds on the performance of combinatorial algorithms for problems with unrelated machines were previously unknown, even when processing times are not stochastic. For the special case of deterministic processing times and without release times, the performance bound equals 4, which is tight. 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 [19]. 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

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

rank larger than 1. The offline version of the problem is denoted $\mathrm{R}\,|\,r_j\,|\sum w_jC_j$ in the three-field notation of Graham et al. [9], and the problem has been a cornerstone problem for the development of new techniques in the design of (approximation) algorithms, e.g. [4, 12, 18, 30].

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 [14, 27]. 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 [26, 5, 16, 10, 13]. 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 this random variable P_j that encodes the possible realizations of job j'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 [23, 24] and approximation algorithms, e.g. [25, 32, 21, 31, 29].

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 [21] by Megow et al., but here we address the more general *unrelated* machines model. In the stochastic unrelated machine model, the scheduler is given a machine-dependent probability distribution of a job's processing time. For a given job the processing times across different machines need not be independent, but the processing time vectors 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, WSEPT is optimal [28]. In the case where job sizes are deterministic and arrive at the same time, SEPT is optimal [11]. In the identical machines setting, SEPT is optimal if job sizes are exponentially distributed [6, 35], or more generally, are stochastically comparable in pairs [34]. Some extensions of these optimality results to the problem with weights exist as well [17]. For more general distributions, simple solutions fail [33], 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 [15], 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, Schulz and Uetz [25] 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 [32, 22, 21, 33, 31, 29], notably in [21] 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 [29]. In the online setting Schulz gives a $(2.309+1.309\Delta)$-competitive algorithm [29]. These results build on an idea from [8] to use a preemptive, fast single machine relaxation, next to the relaxation of [25]. The work of Im, Moseley and Pruhs [15] gave the first results independent of Δ 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 [25] 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 et al. [31] 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 offline). 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

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

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 [31] or convex [3] programming relaxations. Good candidates for online algorithms are simple and combinatorial, but even discovering an offline approximation algorithm that is simple and combinatorial remains an open problem.

Results: This paper shows that there exists an online, $O(\Delta)$-competitive, *combinatorial* algorithm for stochastic scheduling on unrelated machines. 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 competitive ratio of $(4+2\Delta)$. Specifically, for deterministic processing times this bound equals 4, which we show is tight. In the online-time model, where jobs arrive over time, this paper derives an algorithm with competitive ratio $(72+36\Delta)$. Even though the last competitive ratio is far from tight, we believe our results are interesting for at least four 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) It is the first competitive analysis for a combinatorial algorithm for scheduling on unrelated machines even for the deterministic setting. (3) The analysis uses the idea of dual fitting, hence we demonstrate for the first time that this technique can be used for bounding the performance of scheduling policies in non-preemptive and stochastic scheduling. (4) The performance bounds, even if not tight, have the same order of magnitude as those of earlier results in the literature, namely $O(\Delta)$.

The combinatorial algorithm rests on the straightforward idea to greedily assign jobs to the machines where the expected increase of the objective is minimal. The same idea that was used also before, e.g. in [2, 20, 21], but never it was analyzed for unrelated machine settings. Note that the $\Omega(\Delta)$ lower bound for fixed assignment policies in [31] yields that these results are asymptotically tight in Δ among policies that must irrevocably assign jobs to machines at the time of their release. As mentioned already 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 establishes the new insight 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 $r)_j$, but also at a later point in time a job must be assigned to a machine. Once assigned to a machine, the job may wait until a later 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 at the moment that the job completes. This paper considers designing a non-anticipatory scheduling policy Π 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 Π.

This paper assumes 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 $\varepsilon > 0$ [31]. This analysis will make use of the following facts about first and second moments of discrete random variables; these facts also appear in [31].

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

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

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 [21]. This means that (the existence of) a job j is unknown before it arrives, and upon arrival at time r_j, only the distributions of the random variables P_{ij} for the possible processing times on machine $i = 1, \ldots, m$ 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. [33]. We refer the reader to [21] 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 online scheduling policy for the problem. We seek to find a non-anticipatory online scheduling policy (an algorithm) with expected performance ALG close to OPT. For convenience we use the same notation for both the algorithm and its expected performance.

We remark that OPT is not restricted to assigning jobs to machine at the time of their arrival. The only restriction on OPT is that it must schedule jobs nonpreemptively, and that it is non-anticipatory. In fact, our approximation guarantees hold against an even stronger OPT benchmark which 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}.

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 [31, §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, making use of (1) and the first part of Lemma 1, the fact that each job needs to be completely processed translates into the 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

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

The following LP relaxation for the unrelated machine scheduling problem can be derived with these observations. This LP extends the LP given in [31] 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. 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

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

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 $\mathbb{E}[P_{ij}]$.

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 2. Define Δ as a universal upper bound on the 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 follow 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 OPT is the expected total weighted completion time of an optimal non-anticipatory algorithm. By constraints (5),

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

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).

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

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

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*

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

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 [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, 20, 21].

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 $1, 2 \ldots, |J|$. On any machine i, let $H(j,i)$ denote the set of all jobs that have higher priority than j according to their order in non-increasing ratios $w_j/\mathbb{E}[P_{ij}]$, breaking ties by index. That is,

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

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

Greedy Algorithm: 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 total weighted completion time if j is assigned to i and the jobs already present on each machine were to be scheduled in non-increasing order of the ratios weight over expected processing time. This is,

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

The greedy algorithm assigns the job to one of the machines where this quantity is minimal. That is, a job is assigned to machine $i(j) := \operatorname{argmin}_{i\in M}\{\mathrm{EI}(j\to i)\}$; 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 [28].

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

$$\alpha_j := \mathrm{EI}(j \to i(j)) \quad \text{for all } j \in J\,.$$

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 linear program. Later this fact will allow us to relate the variables to the optimal solution's objective.

Fact 1. The solution $(\alpha/2, \beta/2)$ 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 $(\alpha/2, \beta/2)$ 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 α_j and by choice of $i(j)$ as the minimizer of $\mathrm{EI}(j \to i)$, for all i it is the case that

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

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 $w_j \mathbb{E}[P_{ik}]/\mathbb{E}[P_{ij}] \le w_k$ 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 $s \ge 0$.

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 β_{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 (α, β). Let us denote by ALG the total expected value achieved by the greedy algorithm.

LEMMA 3. *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 [21, 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 β 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 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 3. 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^{-1}.

Define ALG^f as the expected solution value obtained by the same greedy algorithm, except that all the machine run at a speed increased by a factor of f. Note that $\mathsf{ALG} = f\mathsf{ALG}^f$, by definition. We denote by (α^f, β^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 4. *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 α 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

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

But now observe that $\beta_{is}^f = \beta_{i(f \cdot s)}$, 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 Fact 1 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 is a $(4+2\Delta)$-competitive algorithm for online scheduling of stochastic jobs to minimize the expectation of the total weighted completion times $\mathbb{E}[\sum_j w_j C_j]$.*

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 $\mathsf{ALG}^f = \sum_{j\in J}\alpha_j^f = \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\ge 0}}\beta_{is}^f$ by Lemma 3, 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

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

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 is a 4-competitive algorithm for deterministic online scheduling to minimize the total weighted completion times $\sum_j w_j C_j$ on unrelated machines, and this bound is tight.*

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 [7], 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 $h = 1, \ldots, k$. 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 only be processed on machines $1, \ldots, \ell$. 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 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, 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 [7] 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 $k \to \infty$. □

6. The Online Time Model

This section adresses the online-time model where jobs arrive over time. A job j arrives at release date $r_j \geq 0$. Assume w.l.o.g. jobs are indexed such that $r_j \leq r_k$ for $j < k$.

This section introduces a new algorithm, which is an adaptation of an analogous algorithm considered in [21] for the parallel identical machines setting. In the greedy algorithm described below, each job j will be irrevocably assigned to a machine upon time r_j, but once assigned to a machine, we work with modified release dates for the scheduling of jobs per machine. Indeed, in order to get reasonable performance bounds even for single machine problems, it is well known that long jobs with small release dates must be delayed to avoid blocking short jobs with larger release dates; see, e.g. [20] for corresponding examples.

Because we will need to consider several variants of a given problem instance, and correspondingly different modified release dates, for what follows it will be convenient to refer to a stochastic scheduling instance by the tuple $(\{r_j\}_{j\in J}, \{P_{ij}\}_{i\in M, j\in J})$, or $(\{r_j\}, \{P_{ij}\})$ for short. Moreover, for any such instance, let us use the notation $(\{r_j\}, \{P_{ij}\}) - \{r_{ij}\}$ to mean that the greedy algorithm uses modified release times $r_{ij} \geq r_j$ for the scheduling of jobs per machine.

Greedy Algorithm:

1. Assignment of jobs to machines: At time r_j, the algorithm computes for each of the machines the quantity $\mathrm{EI}(j \to i)$ in a slightly different way as for the case without release times. Specifically, for problem $(\{r_j\}, \{P_{ij}\})$ and modified release dates r_{ij} that will be defined subsequently, let

$$\mathrm{EI}(j \to i) := w_j \left(2r_{ij} + \sum_{k \to i, k \leq j, k \in H(j,i)} \mathbb{E}[P_{ik}] \right) + \mathbb{E}[p_{ij}] \sum_{k \to i, k < j, k \in L(j,i)} w_k \,,$$

and job j is assigned to any machine i that minimizes $\mathrm{EI}(j \to i)$.

2. Job Processing: For job j assigned to machine i at time r_j, the algorithm modifies its release date to

$$r_{ij} := \max\{f r_j, \mathbb{E}[P_{ij}]\}$$

for a speed-up parameter $f \geq 1$. Later we choose $f = 2$. If a machine i falls idle at a time t, among all unfinished jobs j assigned to machine i where $r_{ij} \leq t$, the algorithm finds the job k with the highest ratio $w_k/\mathbb{E}[P_{ik}]$. Now the algorithm inserts even more forced idleness than done in prior work, as it forces the machine to remain idle for another $\mathbb{E}[P_{ik}]$ units of time, and only then begins the actual processing of job k.

The reason to introduce (more) forced idle time than earlier work is twofold. First, in the analysis to follow we again work with a speed scaling argument, where all machines work at speed f. To

get the necessary scaling argument work out, we define the modified release time of a job j by $\max\{fr_j, \mathbb{E}[P_{ij}]\}$ (and not $\max\{r_j, \mathbb{E}[P_{ij}]\}$ which was used in earlier work [21]). Next, note that the additional, forced idle time of $\mathbb{E}[P_{ik}]$ right before the actual processing of a job k on machine i means that the job will be started after that forced idle time, even if other jobs with higher priority get released in the meantime. Effectively this extends the processing time of any job by its expected processing time, and increases the performance bound by no more than a factor two. This additional, forced idle time however, allows us to bound the expected remaining processing time of a job that potentially could block a machine at a modified release time r_{ij} (see Lemma 5 and its proof). In comparison to earlier work, specifically [29] and [31], this is necessary because we use a greedy algorithm and have otherwise neither control on the amount of jobs that precede a given job j, nor on the expected remaining processing time of a potential blocking job[2]. That problem does not arise in the papers [29] and [31], because those algorithms are steered by corresponding (linear programming) relaxations. Finally in [21], which also analyzes a greedy algorithm, the expected remaining processing time of potential blocking jobs explicitly appears in the performance bounds. By means of the additional forced idle time, we here avoid that. That comes at at the expense of increasing the constant in the performance bound. In fact note that our analysis also works if instead of the forced idleness $\mathbb{E}[P_{ik}]$ we modify the processing time of any job k on machine i to $\max\{P_{ik}, \mathbb{E}[P_{ik}]\}$. This is strictly better e.g. when $\Delta = 0$, but does not improve our analysis in general. The main result of this section is:

THEOREM 3. *For the stochastic online scheduling problem on unrelated parallel machines with release dates, the greedy algorithm is* $(72 + 36\Delta)$*-competitive.*

The complete section that follows is devoted to proving this theorem.

7. Proof of Theorem 3.

The paper first gives a proof outline to provide an overview; the details are given in Section 7.2.

7.1. Proof Outline.

Define the expected total cost of the greedy algorithm as ALG_S and the expected total cost of the optimal non-anticipatory policy as OPT. The goal is to prove $\mathsf{ALG}_S \leq (72 + 36\Delta)\mathsf{OPT}$.

[2] Consider the following single machine example to illustrate this point: 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 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 forced idleness, the 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 turns out to be problematic for our subsequent analysis.

Step 1: As in the online-list model, the core of the argument proceeds via an instance where the machines speeds are augmented by factor $f \geq 1$. Given instance $(\{r_j\}, \{P_{ij}\})$, define instance $(\{r_j\}, \{P_{ij}^f\})$ parameterized by speedup parameter $f \geq 1$, which has the same release times r_j but processing times

$$P_{ij}^f := P_{ij}/f\,.$$

Denote by ALG_S^f the expected cost of the greedy algorithm for $(\{r_j\}, \{P_{ij}^f\}) - \{\max\{r_j, \mathbb{E}[P_{ij}^f]\}\}$, that is, using the modified release dates

$$r_{ij}^f := \max\{r_j, \mathbb{E}[P_{ij}^f]\}\,.$$

Then a simple time scaling argument shows

$$\mathsf{ALG}_S = f \cdot \mathsf{ALG}_S^f.$$

In fact, the equality holds even in distribution and not just for expectations.

Step 2: For the stochastic instance $(\{r_j\}, \{P_{ij}^f\})$, define a corresponding deterministic instance $(\{r_j\}, \{\mathbb{E}[P_{ij}^f]\})$ where the processing time of job j on machine i is non-stochastic and equals $\mathbb{E}[P_{ij}^f]$. For this instance, define the greedy online algorithm as before, except that the algorithm begins processing a job as soon as a machine is idle. That is, there is *no additional forced idleness* before scheduling a job, yet modified release dates are still equal to $\max\{r_j, \mathbb{E}[P_{ij}^f]\}$. Let ALG_D^f denote the total cost of this greedy algorithm on this instance. The proof will establish that

$$\mathsf{ALG}_S^f \leq 6 \cdot \mathsf{ALG}_D^f.$$

Step 3: As in Section 3, define an LP relaxation of the online stochastic machine scheduling problem. Let z^{S_o} denote the optimal solution to this stochastic LP. The only difference compared to the LP given earlier in Section 3 is that the decision variables y_{ijs} are forced to be 0 for all times $s \leq r_j$. Also, let z^{P_o} be the optimal solution value of a modification of that LP as described earlier in Section 3, by simply dropping the term $-\mathbb{CV}[P_{ij}]^2$. As before, it can be established that,

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

The proof will then proceed with a dual fitting argument to show that for speedup parameter $f \geq 2$

$$\mathsf{ALG}_D^f \leq \frac{3f}{f-1} z^{P_o}.$$

Combining these three steps now shows that $\mathsf{ALG}_S = f\mathsf{ALG}_S^f \leq 6f \cdot \mathsf{ALG}_D^f \leq 18f^2/(f-1) \cdot z^{P_o} \leq 18f^2/(f-1)\,(1+\Delta/2) \cdot \mathsf{OPT}$. Choosing $f = 2$ completes the proof of Theorem 3.

7.2. Proof details.

Now we provide the complete proof of Theorem 3 by following the outline described above.

Step 1. Linking ALG_S to a stochastic instance with machine speedup f. Recall that the instance with speedup f was defined to be $(\{r_j\},\{P_{ij}^f\})$, and ALG_S^f denotes the expected objective value of the greedy algorithm for $(\{r_j\},\{P_{ij}^f\}) - \{\max\{r_j,\mathbb{E}[P_{ij}^f]\}\}$, that is, each job j is made available to be processed at time $r_{ij}^f = \max\{r_j,\mathbb{E}[P_{ij}^f]\}$ if the job is assigned to machine i. Note that for the original stochastic instance $(\{r_j\},\{P_{ij}\})$, the time at which job j is made available to be scheduled on machine i by the greedy algorithm is $r_{ij} = \max\{fr_j,\mathbb{E}[P_{ij}]\}$. Hence, $r_{ij}^f = r_{ij}/f$. Therefore, all parameters that are used to compute $\mathrm{EI}(j \to i)$ for the assignment of jobs to machines in the variant with speedup f are scaled consistently, hence the assignments of jobs to machines are identical for both algorithms. That results in the fact that the schedules are identical, only the time axis is compressed by a factor f for ALG_S^f on instance $(\{r_j\},\{P_{ij}^f\})$. This shows that $\mathsf{ALG}_S = f\mathsf{ALG}_S^f$, even in distribution.

Step 2. Upper bound on the performance of ALG_S^f by a deterministic counterpart. For the moment consider an arbitrary stochastic instance with release times r_j, that is $(\{r_j\},\{P_{ij}\})$. Consider the greedy algorithm (including the additional, forced idleness before actually processing any job), however using as modified release dates any value $R_{ij} \geq \max\{r_j,\mathbb{E}[P_{ij}]\}$. In the following we show how to upper bound the expected completion time of any job j. This upper bound actually rests on the use of the additional, forced idle time before the actual processing of any job, and without it, we are not aware how to derive such upper bound.

LEMMA 5. *For the stochastic instance $(\{r_j\},\{P_{ij}\})$, if a job j is assigned to machine i under the greedy algorithm, and $R_{ij} \geq \max\{r_j,\mathbb{E}[P_{ij}]\}$ is used as the modified release date, the expected completion time of job j is bounded as follows.*

$$\mathbb{E}[C_j \,|\, j \to i] \leq 4R_{ij} + 2 \sum_{k \to i, k \in H(j,i)} \mathbb{E}[P_{ik}]\,.$$

Proof. Job j becomes available on machine i at time R_{ij}. Let the random variable X denote the remaining processing time of a job $j-1$ being processed at time R_{ij}, if any such job exists. Note that X also includes the forced idle time before $(j-1)'s$ actual processing. Job j will not be started until job $j-1$ and any available job with higher priority is completed. In any case we can bound the expectation of the completion time C_j of job j by

$$\mathbb{E}[C_j \,|\, j \to i] \leq R_{ij} + \mathbb{E}[X] + 2 \sum_{k \to i, k \in H(j,i)} \mathbb{E}[P_{ik}]\,. \tag{12}$$

Note that the factor 2 comes from the idleness that is inserted before the actual processing of jobs.

Let us first assume that $X = 0$, so machine i is idle at time R_{ij}. Then we immediately see that the even stronger inequality

$$\mathbb{E}[C_j \,|\, j \to i] \le R_{ij} + 2 \sum_{k \to i, k \in H(j,i)} \mathbb{E}[P_{ik}]$$

holds and we are done. So let us assume that $X > 0$. In fact, the only difficulty in bounding $\mathbb{E}[X]$ lies in bounding the expected remaining processing time of job $j-1$ when $j-1$ has started its *actual* processing before R_{ij}, as otherwise $\mathbb{E}[X] \le 2\mathbb{E}[P_{ij-1}] \le 2R_{ij-1} \le 2R_{ij}$, and then

$$\mathbb{E}[C_j \,|\, j \to i] \le 3R_{ij} + 2 \sum_{k \to i, k \in H(j,i)} \mathbb{E}[P_{ik}]\,.$$

To bound the term $\mathbb{E}[X]$ in case there is a job $j-1$ that has started its actual processing before time R_{ij} and is still in process at R_{ij}, we define a sequence of random variables $Y_1, Y_2, \cdots$ where Y_k measures the time interval between the completion time of the $(k-1)$st job and that of the kth job completed on machine i. Let I_k denote the "true" idle time of machine i within this interval, and define A_k to be the sum of I_k and the expected processing time of the kth job completed. Figure 1 illustrates these definitions for job j. Under the greedy algorithm, for all k we have

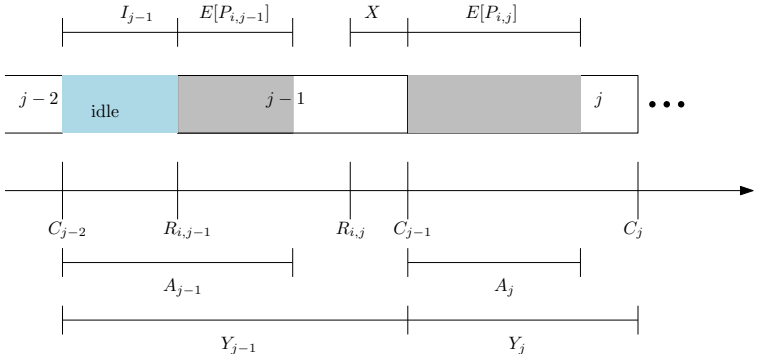

Figure 1 **Illustration for the definitions used in the proof of Lemma 5. Note that $I_j = 0$.**

$$Y_k \ge A_k \;\; w.p.\ 1, \text{ and } \mathbb{E}[Y_k \,|\, Y_1, \ldots, Y_{k-1}] \le 2A_k\,,$$

as $Y_k = A_k + P_{ik}$ and $\mathbb{E}[Y_k \,|\, Y_1, \ldots, Y_{k-1}] = 2\mathbb{E}[P_{ik}] + I_k \le 2A_k$. Here, note that I_k is in fact deterministic given the history $Y_1, \ldots, Y_{k-1}$. Define the $R_{ij}-$stopping time τ with respect to the sequence $Y_1, Y_2, \ldots$ as

$$\tau := \min\{k : Y_1 + Y_2 + \cdots + Y_k \ge R_{ij}\}\,.$$

In the case we consider, as also illustrated in Figure 1, note that we have $\tau = j-1$, and hence it holds that $A_k \le R_{ij}$ for all $k = 1, \dots, \tau$. Therefore we can use Lemma 10 from Appendix A (with stopping time $T = R_{ij}$), which yields

$$R_{ij} + \mathbb{E}[X] = \mathbb{E}[Y_1 + Y_2 + \cdots + Y_\tau] \le 4R_{ij}\,.$$

Hence, plugging that into (12) yields

$$\mathbb{E}[C_j | j \to i] \le 4R_{ij} + 2 \sum_{k \to i, k \in H(j,i)} \mathbb{E}[P_{ik}]\,.$$

□

Recalling our definition of modified release dates $r_{ij}^f = \max\{r_j, \mathbb{E}[P_{ij}^f]\}$, we derive an upper bound on the cost ALG_S^f of the greedy algorithm for the stochastic instance $(\{r_j\}, \{P_{ij}^f\}) - \{r_{ij}^f\}$ in terms of the cost ALG_D^f of the greedy algorithm for the corresponding deterministic instance $(\{r_j\}, \{\mathbb{E}[P_{ij}^f]\}) - \{r_{ij}^f\}$.

LEMMA 6. *Given the definitions of* ALG_S^f *and* ALG_D^f*, we have*

$$\mathsf{ALG}_S^f \le 6\mathsf{ALG}_D^f\,.$$

Proof. Let C_j^f denote the completion time of job j of the greedy algorithm for the stochastic instance $(\{r_j\}, \{P_{ij}^f\}) - \{r_{ij}^f\}$. Using Lemma 5 with $R_{ij} = r_{ij}^f$ we see that

$$\begin{aligned}
\mathsf{ALG}_S^f &= \mathbb{E}[\sum_j w_j C_j^f] = \sum_i \sum_{j:j\to i} w_j \mathbb{E}[C_j^f \,|\, j \to i] \\
&\le \sum_i \sum_{j:j\to i} \left[4w_j r_{ij}^f + 2w_j \sum_{k\to i, k\in H(j,i)} \mathbb{E}[P_{ik}^f] \right] \\
&\le 4 \left[\sum_i \sum_{j:j\to i} w_j (r_j + \mathbb{E}[P_{ij}^f]) \right] \;+\; 2 \left[\sum_i \sum_{j:j\to i} w_j \sum_{k\to i, k\in H(j,i)} \mathbb{E}[P_{ik}^f] \right],
\end{aligned}$$

where the last inequality follows because $r_{ij}^f \le r_j + \mathbb{E}[P_{ij}^f]$. Next we want to argue that each of the six terms on the right-hand side is a lower bound for ALG_D^f. Since $w_j(r_j + \mathbb{E}[P_{ij}^f])$ is a trivial lower bound for the completion time of a job assigned to machine i, and since $\sum_{j:j\to i} w_j \sum_{k\to i, k\in H(j,i)} \mathbb{E}[P_{ik}^f]$ is the value of the optimal solution for the jobs assigned to machine i without considering release times [28], indeed each of the six terms is a lower bound for ALG_D^f as long as the jobs are assigned to the same machines for both ALG_S^f and ALG_D^f. But this is true because the assignment of jobs to machines only depends on the ordering of the original release dates, expected processing times of jobs, and the modified release dates. These are indeed the same, as the instances are $(\{r_j\}, \{P_{ij}^f\}) - \{r_{ij}^f\}$, respectively $(\{r_j\}, \{\mathbb{E}[P_{ij}^f]\}) - \{r_{ij}^f\}$. □

Step 3.1 LP Relaxation Analogous to the earlier linear programming relaxation (S), we will define the LP relaxation for the instance $(\{r_j\},\{P_{ij}\})$. We omit writing out this LP relaxation in detail as it is exactly the same as (S), except that the variables y_{ijs} are defined only for $s \geq r_j$. Let us call this LP "S_o" and its optimal solution value z^{S_o}. Similarly, analogous to (P) we can write the LP relaxation for the deterministic version $(\{r_j\},\{\mathbb{E}[P_{ij}]\})$ by dropping all terms $-\mathbb{CV}[P_{ij}]^2$. Let us call this deterministic LP relaxation "P_o" with optimal solution value z^{P_o}. Lemma 2 and Corollary 1 apply to z^{P_o} and z^{S_o} in exactly the same way. That is,

$$z^{P_o} \leq \big(1+\frac{\Delta}{2}\big) z^{S_o} \leq \big(1+\frac{\Delta}{2}\big)\mathsf{OPT}\,.$$

Step 3.2. Lower bound on z^{P_o}. To lower bound the cost of the deterministic Primal LP relaxation P_o, we will define a feasible solution to its dual LP, which is:

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

To define a feasible solution for (D_o), we consider the greedy algorithm ALG_D^f for the instance $(\{r_j\},\{\mathbb{E}[P_{ij}^f]\})-\{r_{ij}^f\}$. Recall that this variant of the greedy algorithm does not insert additional, forced idle time before actually processing a job.

For job $j\in J$, we define α_j^f as an upper bound on the expected increase of the expected objective value when job j is assigned to machine i. That is,

$$\alpha_j^f := \min_i \left[2w_j r_{ij}^f + w_j \sum_{k\to i, k\leq j, k\in H(j,i)} \mathbb{E}[P_{ik}^f] \;+\; \mathbb{E}[P_{ij}^f] \sum_{k\to i, k<j, k\in L(j,i)} w_k \right].$$

For a machine i and time s, we denote by $A_i^f(s)$ the total set of jobs assigned to machine i that have not been completed by time s. Define β_{is}^f as the total weight of jobs in $A_i^f(s)$, i.e.,

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

Lemma 7. *The values* $(\{\alpha_j\},\{\beta_{is}\}) := \left(\{\frac{\alpha_j^f}{3}\},\{\frac{\beta_{is}^f}{3f}\}\right)$ *are feasible for* (D_o).

Proof. Fix job j and machine i. We need to show that

$$\frac{f\alpha_j^f}{\mathbb{E}[P_{ij}]} \leq \beta_{is}^f + 3fw_j\left(\frac{s+\frac{1}{2}}{\mathbb{E}[P_{ij}]}+\frac{1}{2}\right) \qquad (13)$$

holds for all $i \in M$, $j \in J$, and $s \geq r_j$. By the definition of α_j^f, for any machine i, we have

$$\alpha_j^f \;\leq 2w_j r_{ij}^f + w_j \sum_{k \to i, k \leq j, k \in H(j,i)} \mathbb{E}[P_{ik}^f] \;+\; \mathbb{E}[P_{ij}^f] \sum_{k \to i, k < j, k \in L(j,i)} w_k \,.$$

As $r_{ij}^f \leq r_j + \mathbb{E}[P_{ij}^f]$, we get after multiplication with $f/\mathbb{E}[P_{ij}]$

$$\begin{aligned}\frac{f\alpha_j^f}{\mathbb{E}[P_{ij}]} &\leq \frac{2w_j f(r_j + \mathbb{E}[P_{ij}^f])}{\mathbb{E}[P_{ij}]} + w_j \sum_{k \to i, k \leq j, k \in H(j,i)} \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \;+ \sum_{k \to i, k < j, k \in L(j,i)} w_k \\ &\leq w_j \left(2\frac{fs}{\mathbb{E}[P_{ij}]} + 3\right) + w_j \sum_{k \to i, k < j, k \in H(j,i)} \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \;+ \sum_{k \to i, k < j, k \in L(j,i)} w_k \,,\end{aligned}$$

where the last inequality follows by the fact that $r_j \leq s$. Therefore to prove that (13) holds, assuming $f \geq 2$ it suffices to show that

$$w_j \sum_{k \to i, k < j, k \in H(j,i)} \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \;+ \sum_{k \to i, k < j, k \in L(j,i)} w_k \;\leq \beta_{is}^f + w_j \frac{fs}{\mathbb{E}[P_{ij}]}. \tag{14}$$

Let $D_{ij}^f(s)$ denote the set of jobs $k < j$ assigned to machine i and completed by time s, and $U_{ij}^f(s)$ be the set of jobs $k < j$ assigned to machine i and still unfinished (alive) at time s (including those are assigned but not available according to modified release times r_{ij}^f). Observe that $U_{ij}^f(s) \subset A_i^f(s)$. Hence by the definition of β_{is}^f,

$$\sum_{k \in U_{ij}^f(s)} w_k \leq \sum_{k \in A_i^f(s)} w_k = \beta_{is}^f. \tag{15}$$

Here

$$\sum_{k \in D_{ij}^f(s)} \frac{\mathbb{E}[P_{ik}]}{f} \leq s. \tag{16}$$

Note that if $k \in H(j,i)$, $w_j \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \leq w_k$, and if $k \in L(j,i)$, $w_j \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} > w_k$. Then we can upper bound the left-hand side (LHS) of (14) as follows:

$$\begin{aligned}\text{LHS of (14)} &= \sum_{k \in H(j,i) \cap D_{ij}^f(s)} w_j \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \;+ \sum_{k \in L(j,i) \cap D_{ij}^f(s)} w_k \\ &\quad + \sum_{k \in H(j,i) \cap U_{ij}^f(s)} w_j \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \;+ \sum_{k \in L(j,i) \cap U_{ij}^f(s)} w_k \\ &\leq \sum_{k \in H(j,i) \cap D_{ij}^f(s)} w_j \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \;+ \sum_{k \in L(j,i) \cap D_{ij}^f(s)} w_j \frac{\mathbb{E}[P_{ik}]}{\mathbb{E}[P_{ij}]} \\ &\quad + \sum_{k \in H(j,i) \cap U_{ij}^f(s)} w_k \;+ \sum_{k \in L(j,i) \cap U_{ij}^f(s)} w_k \\ &= \frac{w_j}{\mathbb{E}[P_{ij}]} \sum_{k \in D_{ij}^f(s)} \mathbb{E}[P_{ik}] + \sum_{k \in U_{ij}^f(s)} w_k\end{aligned}$$

$$\leq \frac{w_j}{\mathbb{E}[P_{ij}]} fs + \beta_{is}^f,$$

where the last inequality follows from (15)-(16). □

As the so-defined variables $(\{\frac{\alpha_j^f}{3}\}, \{\frac{\beta_{is}^f}{3f}\})$ are feasible for the dual (D_o), their objective value provides a lower bound for the optimal solution of the primal (P_o) by duality, and we note the following.

COROLLARY 2. *The optimal solution of the deterministic LP relaxation (P_o) is bounded by:*

$$z^{P_o} \geq \frac{1}{3}\left(\sum_{j\in J} \alpha_j^f - \frac{1}{f}\sum_{i\in M}\sum_{s} \beta_{is}^f\right).$$

Step 3.3 Upper bound on ALG_D^f . Finally, to complete the proof, we show that the dual variables (α^f, β^f) can be linked to the cost ALG_D^f of the greedy algorithm for the deterministic instance $(\{r_j\}, \{\mathbb{E}[P_{ij}^f]\}) - \{r_{ij}^f\}$. This in turn allows us to upper bound ALG_D^f in terms of the optimal value of the deterministic LP relaxation z^{P_o}.

LEMMA 8. *The total weighted completion time of the greedy algorithm ALG_D^f on instance $(\{r_j\}, \{\mathbb{E}[P_{ij}^f]\}) - \{r_{ij}^f\}$ satisfies*

$$\mathsf{ALG}_D^f \leq \sum_{j\in J} \alpha_j^f, \qquad \mathsf{ALG}_D^f = \sum_{i\in M}\sum_{s\in\mathbb{Z}_{\geq 0}} \beta_{is}^f.$$

Combined with Corollary 2, the above yields

$$\mathsf{ALG}_D^f \leq \frac{3f}{f-1} z^{P_o}.$$

Proof. For each job j, let i_j denote the machine to which it is assigned by ALG_D^f. By the same argument as that for (12), we can obtain the following upper bound on the completion time of job j

$$C_j \leq r_{i_j j}^f + X + \sum_{k\to i_j, k\in H(j,i_j)} \mathbb{E}[P_{i_j k}^f],$$

where X is the (deterministic) remaining processing time of a job $j-1$ in process at time $r_{i_j j}^f$, if any such job exists; otherwise X has value 0. Next observe that $X \leq r_{i_j j}^f$: This clearly holds if machine i is idle at $r_{i_j j}^f$. Assume that job $j-1$ is being processed at time $r_{i_j j}^f$. Then it must be true that $\max\{r_j, \mathbb{E}[P_{i_j j-1}^f]\} = r_{i_j j-1}^f \leq r_{i_j j}^f$. Hence $X \leq \mathbb{E}[P_{i_j j-1}^f] \leq r_{i_j j}^f$. Therefore, we have

$$\mathsf{ALG}_D^f = \sum_j w_j C_j^f \leq 2\sum_j w_j r_{i_j j}^f + \sum_j w_j \sum_{k\to i_j, k\in H(j,i_j)} \mathbb{E}[P_{i_j k}^f]. \tag{17}$$

By applying the following standard index rearrangement,

$$\sum_j w_j \sum_{\substack{k\to i_j \\ k\in H(j,i_j) \\ k>j}} \mathbb{E}[P_{i_j k}^f] = \sum_j \mathbb{E}[P_{i_j j}^f] \sum_{\substack{k\to i_j \\ k\in L(j,i_j) \\ k<j}} w_k,$$

we can rewrite the second part of the right hand side of (17) as

$$\sum_j w_j \sum_{\substack{k\to i_j \\ k\in H(j,i_j)}} \mathbb{E}[P^f_{i_j k}] = \sum_j w_j \sum_{\substack{k\to i_j \\ k\in H(j,i_j) \\ k\le j}} \mathbb{E}[P^f_{i_j k}] + \sum_j \mathbb{E}[P^f_{i_j j}] \sum_{\substack{k\to i_j \\ k\in L(j,i_j) \\ k<j}} w_k.$$

We thus obtain

$$\begin{aligned}
\mathsf{ALG}^f_D &\le \sum_j \left(2w_j r^f_{i_j j} + w_j \sum_{k\to i_j, k\in H(j,i_j), k\le j} \mathbb{E}[P^f_{i_j k}] + \mathbb{E}[P^f_{i_j j}] \sum_{k\to i_j, k\in L(j,i_j), k<j} w_k \right) \\
&= \sum_j \left(\frac{1}{f} \mathrm{EI}(j \to i_j) \right) \\
&= \sum_j \alpha^f_j,
\end{aligned}$$

where the first inequality follows from the definition of $\mathrm{EI}(j \to i)$ and the second equality follows by the fact that i_j also minimizes $\mathrm{EI}(j \to i)$. This is true because the assignment of jobs to machines is the same for both, the greedy algorithm ALG_S for the original, stochastic instance $(\{r_j\}, \{P_{ij}\}) - \{fr^f_{ij}\}$, and the greedy algorithm ALG^f_D for the deterministic instance $(\{r_j\}, \{\mathbb{E}[P^f_{ij}]\}) - \{r^f_{ij}\}$. Here, note that we have $fr^f_{ij} = \max\{fr_j, \mathbb{E}[P_{ij}]\}$.

Finally, the proof for the identity $\mathsf{ALG}^f_D = \sum_i \sum_s \beta^f_{is}$ is exactly the same as that of Lemma 3, and therefore omitted. □

8. Conclusions

The main result of this paper is to show that simple, combinatorial online algorithms *can* be worst-case analyzed even for the most general of all machine scheduling models and uncertain job sizes. Even if the competitive ratio for the case where jobs are release over time is obviously far from being tight, all performance bounds that we derive for the greedy algorithm are $\mathrm{O}(\Delta)$, which is the same order of magnitude as earlier bounds for offline problems on unrelated machines [31], and the same order of magnitude as earlier bounds for the online identical machines setting [21]. We believe that the derivation of genuine lower bounds for stochastic problems that would allow to separate from the corresponding deterministic special cases is the most interesting yet probably challenging direction for future work.

Acknowledgements. This work was done 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. 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-1617724.

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] 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.

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

[9] 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.

[10] 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.

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

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

[13] 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.

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

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

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

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

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

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

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

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

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

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

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

[25] 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.

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

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

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

[29] 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.

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

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

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

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

[34] 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.

[35] 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 slot t ($t=0,1,\ldots$). For a given $\{x_t\}$ 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 is easy to see 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 following sum:

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

or,

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

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

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

□

LEMMA 10. *Let $Y_1, Y_2, \ldots,$ be a sequence of random variables, with Y_k adapted to the filtration $\mathcal{F}_{k-1} = \sigma(Y_1, \ldots, Y_{k-1})$ for all $k \geq 1$. Further, let $A_1, A_2, \ldots$ be another sequence, with A_k adapted to $\mathcal{F}_{k-1}$ satisfying*

1. *$0 \leq A_k \leq T$ almost surely,*
2. *$X_k \geq \alpha A_k$ almost surely, and $\mathbb{E}[Y_k] \leq (1+\alpha)A_k$ for some $\alpha > 0$.*

Define the T-stopping time of the sequence $Y_1, Y_2, \ldots$ as:

$$\tau := \min\{k : Y_1 + \cdots + Y_k \geq T\} \tag{18}$$

Under the assumption that the stopping time τ satisfies $\mathbb{E}[\tau] < \infty$, we have

$$\mathbb{E}\left[\sum_{k=1}^{\tau} Y_k\right] \leq \frac{(1+\alpha)^2}{\alpha} T. \tag{19}$$

In particular, choosing $\alpha = 1$, so that $\mathbb{E}[Y_k|\mathcal{F}_{k-1}] \leq 2A_k$ and $Y_k \geq A_k$, we have

$$\mathbb{E}\left[\sum_{k=1}^{\tau} Y_k\right] \leq 4T.$$

Proof. The lemma is a straightforward consequence of the Optional Stopping Theorem. We first note that since $\mathbb{E}[Y_k|\mathcal{F}_{k-1}] \leq (1+\alpha)A_k$, the sequence:

$$M_\ell = \sum_{k=0}^{\ell} (Y_k - (1+\alpha)A_k)$$

defines a supermartingale with $M_0 = 0$. Under the assumption that $\mathbb{E}[\tau] < \infty$, the Optional Stopping Theorem gives

$$\mathbb{E}[M_\tau] \leq M_0 = 0\,.$$

Therefore,

$$\begin{aligned}
\mathbb{E}\left[\sum_{k=1}^{\tau} Y_k\right] &\leq \mathbb{E}\left[(1+\alpha)\sum_{k=1}^{\tau} A_k\right] \\
&= (1+\alpha)\mathbb{E}[A_\tau] + (1+\alpha)\mathbb{E}\left[\sum_{k=1}^{\tau-1} A_k\right] \\
&\leq (1+\alpha)T + (1+\alpha)\mathbb{E}\left[\sum_{k=1}^{\tau-1} A_k\right] \\
&\leq (1+\alpha)T + (1+\alpha)\frac{T}{\alpha} \\
&= \frac{(1+\alpha)^2}{\alpha} T\,.
\end{aligned}$$

Where the second inequality follows from $A_\tau \leq T$, and the last inequality follows from the observations: $A_k \leq \frac{1}{\alpha} Y_k$ almost surely, and $\sum_{k=1}^{\tau-1} Y_k \leq T$. □