Models and Algorithms for Stochastic Online Scheduling [1]

Nicole Megow
Technische Universität Berlin, Institut für Mathematik, Strasse des 17. Juni 136, 10623 Berlin, Germany.
email: nmegow@math.tu-berlin.de

Marc Uetz
Maastricht University, Department of Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands.
email: m.uetz@ke.unimaas.nl

Tjark Vredeveld[2]
Maastricht University, Department of Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands.
email: t.vredeveld@ke.unimaas.nl

We introduce a model for non-preemptive scheduling under uncertainty. In this model, we combine the main characteristics of online and stochastic scheduling in a simple and natural way. Job processing times are assumed to be stochastic, but in contrast to traditional stochastic scheduling models, we assume that jobs arrive online, and there is no knowledge about the jobs that will arrive in the future. The model incorporates both, stochastic scheduling and online scheduling as a special case. The particular setting we analyze is parallel machine scheduling, with the objective to minimize the total weighted completion times of jobs. We propose simple, combinatorial online scheduling policies for that model, and derive performance guarantees that match the currently best known performance guarantees for stochastic and online parallel machine scheduling. For processing times that follow NBUE distributions, we improve upon previously best known performance bounds from stochastic scheduling, even though we consider a more general setting.

Key words: scheduling; stochastic; online; total weighted completion time; approximation

MSC2000 Subject Classification: Primary: 90B36 ; Secondary: 68W40

OR/MS subject classification: Primary: Production/scheduling ; Secondary: approximation/heuristic

---

1. Introduction. Non-preemptive scheduling on identical parallel machines to minimize the total weighted completion time of jobs, $\mathrm{P}\,|\,|\sum w_j\,C_j$ in the three-field notation of Graham et al. [10], is one of the classical problems in combinatorial optimization. The problem plays a role whenever many jobs must be processed on a limited number of machines, with typical applications in manufacturing, parallel computing [5] or compiler optimization [6]. The literature has observed many papers on this problem, as well as its variant where the jobs have individual release dates before which they must not be processed, $\mathrm{P}\,|\,r_j\,|\sum w_j\,C_j$. In the deterministic offline setting, where the set of jobs and their processing times, release dates and weights are known in advance, the complexity status of both problems is solved; both problems are known to be strongly NP-hard [16], and both admit a polynomial time approximation scheme [1, 27].

In order to cope with uncertainty about the future, there are two major frameworks in the theory of scheduling, one is stochastic scheduling, the other online scheduling. The main characteristic of *stochastic scheduling*, in contrast to deterministic models, is the fact that the processing times of jobs are subject to random fluctuations, and the actual processing times become known only upon completion of the jobs. It is generally assumed, though, that the respective random variables, or at least their first moments, are known beforehand. In *online scheduling*, the assumption is that the instance is presented to the scheduler only piecewise. Depending on the precise model, jobs are arriving either one-by-one (online-list), or over time (online-time) [22]. The job characteristics such as weight and processing time are usually disclosed upon arrival of the job, and decisions must be made without any knowledge of the jobs to come.

In this paper, we suggest a model that generalizes both, the stochastic scheduling model as well as the online scheduling model. Like in online scheduling, we assume that the instance is presented to the scheduler piecewise, and nothing is known about jobs that might arrive in the future. Once a job arrives,

[1]Research partially supported by the DFG Research Center MATHEON "Mathematics for key technologies". An extended abstract with parts of this work appeared in the Proceedings of the 2nd Workshop on Approximation and Online Algorithms, 2004.

[2]This research was done while the author was with the Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany

like in stochastic scheduling, we assume that its weight and expected processing time are disclosed, but the actual processing time remains unknown until the job completes. Before we describe the model in more detail, we next discuss related work from both, stochastic scheduling and online scheduling literature.

Stochastic scheduling. Stochastic machine scheduling models have been addressed mainly since the 1980s [7]. Let us briefly recall the concept of a *scheduling policy* as introduced by Möhring et al. [19]. Roughly spoken, at any time t, such a policy specifies which action to perform, in particular which jobs to start at t. In order to decide, it may utilize the complete information contained in the partial schedule up to time t, and it may also use the information about the given input instance, such as the number of jobs and the given job characteristics. However, a policy is *non-anticipatory*, meaning that it must not utilize any information about actual processing times of jobs that will be finished in the future. An *optimal* scheduling policy is defined as one that minimizes the objective function value in expectation. By definition, an optimal stochastic scheduling policy may in general fail to yield an (offline) optimal solution for all realizations of the processing times; this because it is non-anticipatory. Probably the best known scheduling policy for the problem at hand is list scheduling according to the WSEPT rule, which greedily schedules the jobs according to highest relative weight.

Definition 1.1 (WSEPT rule, weighted shortest expected processing time first) *At any point in time when a machine is idle, among all jobs that are available, schedule the job with the highest ratio of weight over expected processing time,* $w_j/\mathbb{E}[P_j]$.

Here, a job j is *available* at a given time t if it has not yet been scheduled, and if its release date has passed, that is, $r_j \leq t$. For unit weights, this reduces to the SEPT rule, list scheduling in the order of shortest expected processing time first. For deterministic processing times, it is also known as the WSPT rule or Smith's rule [29]. The WSEPT rule is indeed a scheduling policy, because the decisions to schedule a job only depend on weights and expected processing times of jobs, but not on the actual processing time realizations.

With the exception of the papers by Weiss [33, 34], who analyzes the WSEPT rule asymptotically, the first approximation algorithms for stochastic machine scheduling have been derived only recently [20, 26]. In these papers, the expected performance of either the WSEPT rule or a linear programming based list scheduling policy is compared against the expected performance of an optimal scheduling policy. A policy is called a ρ–approximation if its expected performance is not worse than ρ times the expected performance of an optimal scheduling policy. The results of [20, 26] are constant-factor approximations for problems with or without release dates [20], and also with precedence constraints [26]. The approach is based upon the solution of linear programming relaxations, and for the models with release dates or precedence constraints, these LP solutions are used in order to define corresponding list scheduling policies.

Recently, another type of analysis has been pursued by Steger et al. in the papers [24, 30]. They analyze the *expected ratio* of the performance of the (W)SEPT rule over the offline optimum solution. This approach has certain advantages over the previously described approach, namely in terms of the effective averaging over different realizations of processing times. We refer to [24, 30] for a discussion of this issue. One of the main differences, however, is the fact that it is based on a comparison against the offline optimal solution, whereas the approach in [20, 26] compares against the optimal stochastic scheduling policy. Nevertheless, restricted to models without release dates or precedence constraints, constant-factor approximation results for the expected ratio have been obtained for the (W)SEPT rule on parallel machines [24, 30].

Online scheduling. Online machine scheduling is an other extensively researched model for scheduling under uncertainty. It is assumed that information about the instance is revealed only piecewise. Thus, an online algorithm has to make decisions based only on partial knowledge of the input instance. In comparison to stochastic scheduling, this lack of knowledge of the actual input instance is even larger, since it is assumed that nothing is known about the jobs that are to come. Even the number of jobs is not given in advance. The quality of online algorithms is typically assessed by their worst-case performance, expressed as the competitive ratio [13, 28]. An algorithm is called ρ-competitive if for any instance a solution is achieved that has an objective function value of at most ρ times the value of an optimal offline solution.

In online optimization, two models exist for piecewise revealing information to an online algorithm, the *online-time* and the *online-list* model. In scheduling terminology, in the online-time model *jobs arrive over time*, that is, information about jobs are revealed over time and the algorithm has to make decisions about which jobs to schedule at each time step. In contrast to that, in the online-list model we must schedule *jobs one by one*, which means that jobs are presented as a list of which the algorithm sees only the first request. A job has to be scheduled immediately, i.e., the decision about the machines and time slots must be made immediately and irrevocably, before the next job can be seen. We omit more details and refer to Borodin and El-Yaniv [3] or Pruhs et al. [22].

Online scheduling to minimize the sum of completion times has been investigated mainly in the online-time model where jobs have release dates and arrive over time. The reason lies in the intractability of the problem in the online-list model: Fiat and Woeginger [9] show that the single machine problem $1 \,|\, r_j \,|\, \sum C_j$ does not allow any deterministic or random online algorithm with a competitive ratio of $\log n$. On the positive side they prove the existence of an online algorithm that is $(\log n)^{1+\epsilon}$-competitive, for any $\epsilon > 0$.

Therefore, we mainly focus on the online-time model. The status of the single machine problem $1 \,|\, r_j \,|\, \sum w_j C_j$ is solved. Several groups proposed 2-competitive algorithms for the setting where all job weights are equal [21, 12, 17]. Anderson and Potts [2] finally extend an algorithm from [12], and provide a 2-competitive online algorithm for the general setting $1 \,|\, r_j \,|\, \sum w_j C_j$. These results are best possible, since Hoogeveen and Vestjens [12] proved a lower bound of 2 on the competitive ratio of any deterministic online algorithm.

Online scheduling on parallel machines is less well understood. Vestjens [32] proved a lower bound of 1.309 for the competitive ratio of any deterministic online algorithm for $\mathrm{P} \,|\, r_j \,|\, \sum C_j$. The currently best known competitive ratio is 3.28. This result has been obtained by Megow and Schulz [18], even in the weighted setting $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$, by modifying release dates and scheduling jobs according to the WSPT rule. The currently best known randomized online algorithms for the problem have an expected competitive ratio of 2 [25].

Stochastic online scheduling (SoS). Results in stochastic scheduling either rely on the (W)SEPT rule [20, 24, 30, 33, 34] for models without release dates, or they use linear programming relaxations to define list scheduling policies other than WSEPT [20, 26]. As soon as we assume that the total population of jobs is not given at the outset, but that jobs arrive online, both approaches fail. Simple examples show that WSEPT may be arbitrarily bad, and LP-based approaches to define list scheduling algorithms do not seem to make sense at all.

In this paper, we overcome this difficulty and suggest simple, combinatorial algorithms that have constant performance bounds also in an online setting with stochastic processing times. Analysiswise, we follow the approach taken by Möhring et al. [20]. In other words, we compare the expected outcome of a certain online scheduling policy against the expected outcome of an optimal scheduling policy. In contrast to the previously mentioned work on stochastic scheduling, however, we consider a model where jobs arrive online over time. At the moment that a job j arrives, its weight w_j and expected processing time $\mathbb{E}[P_j]$ are disclosed to the scheduler. The actual processing time, however, remains unknown until the job completes.

Since the feature of jobs arriving online over time is a new feature in comparison to the traditional stochastic scheduling model, we must clarify how an *optimal scheduling policy* is to be understood in this new model. For convenience, and in line with the nomenclature in online optimization, let us think of the optimal scheduling policy as an *adversary*, as opposed to ourselves, the *scheduler*. Like in online scheduling, we assume that the adversary may freely choose the sequence of jobs arriving over time, together with their characteristics such as weights and processing time distributions. However, the uncertainty of processing times is assumed to be exogenous and not under control of the adversary; neither does the adversary know in advance how the actual processing times will materialize. Thus, the adversary knows in advance all jobs with their respective weights w_j, release dates r_j, and processing time distributions P_j, but with respect to the uncertainty of actual processing times, the adversary is just as powerful as the scheduler. In this sense, the adversary effectively creates an instance of a traditional stochastic scheduling problem $\mathrm{P} \,|\, r_j \,|\, \mathbb{E}[\sum w_j C_j]$. An optimal scheduling policy for the stochastic online scheduling model can therefore be defined as an optimal scheduling policy in the traditional sense, on this particular instance. The *scheduler*, on the other hand, does not know any job before its actual release

date has passed, and is therefore much less knowledgable than the adversary.

Notice that our definition of an optimal scheduling policy is chosen in such a way that it generalizes both, online scheduling and stochastic scheduling. On the one hand, when the processing times are deterministic, we arrive at the well known online-time model for $\mathrm{P}\,|\,r_j\,|\sum w_j\,C_j$. On the other hand, traditional stochastic scheduling corresponds to the case where all jobs are disclosed to the scheduler at the outset, while we assume that jobs are disclosed only at their respective release dates. When all release dates are equal, we thus arrive at the traditional stochastic scheduling model $\mathrm{P}\,|\,|\,\mathbb{E}\,[\sum w_j\,C_j]$. For this case, however, we also consider an online model that resembles the online-list model: The jobs are assumed to be presented sequentially, and the scheduler must assign jobs to machines immediately and irrevocably, without knowing the jobs to come. So instead of the traditional stochastic scheduling model $\mathrm{P}\,|\,|\,\mathbb{E}\,[\sum w_j\,C_j]$, we also consider an online generalization for the case without release dates.

For convenience, let us denote both models by SOS, for stochastic online scheduling.

Discussion of the model. As a matter of fact, the solution of LP relaxations is crucial for the work of Möhring et al. [20] or Skutella and Uetz [26]. For the models with release dates, optimal LP solutions are not only required for the purpose of analysis as lower bounds, but also to define the corresponding list scheduling policies. In order to set up these LP relaxations, it is required to know beforehand the set of jobs, their expected processing times $\mathbb{E}\,[P_j]$, and also a uniform upper bound Δ on the squared coefficient of variation of all processing times distributions

$$\mathrm{CV}\,[P_j]^2 = \mathrm{Var}\,[P_j]/\mathbb{E}\,[P_j]^2 \le \Delta \quad \text{for all jobs } j\,.$$

While an upper bound on the (squared) coefficient of variation is probably reasonable in many cases, one obvious critique of this approach is the fact that (parts of) this data might not be available. The effect is that the algorithm itself cannot be properly defined. This becomes even more apparent in an online setting, where such an LP-based approach appears quite useless. We mention that we still use the very same linear programming relaxation as Möhring et al. [20] within our analysis, but the main difference lies in the fact we do not require the solution of linear programs in advance.

In comparison to traditional online models, we make another remark. Like in traditional online optimization, the adversary in the SOS model may choose an arbitrary sequence of jobs. These jobs are stochastic with corresponding processing time distributions; deterministic processing times being a special case. The actual processing times are realized according to exogenous probability distributions. Thus, the best the adversary can do is indeed to use an optimal stochastic scheduling policy in the traditional sense. In this view, our model somewhat compares to the idea of a *diffuse adversary* as defined by Koutsoupias and Papadimitriou in [15]. Since deterministic processing times are still contained as a special case, however, all lower bounds on the approximability known from (deterministic) online scheduling immediately transfer to the SOS model of the present paper. Hence, we know that no SOS policy can exist with a performance bound better than 1.309 for the case of unit weights [32], and thus also in general.

Results and methodology. We derive guarantees on the expected performance of simple, combinatorial online scheduling policies for models with and without release dates.

For the model without release dates, $\mathrm{P}|\,|\mathbb{E}\,[\sum w_j\,C_j]$, this is a performance guarantee of

$$\rho = 1 + \frac{(m-1)(\Delta+1)}{2m}\,,$$

where Δ is an upper bound for the squared coefficients of variation of the processing time distributions. This matches the previously best known performance guarantee of [20] for the performance of the WSEPT rule. As mentioned already above, also for the case without release dates, we derive this performance bound for a model other than traditional stochastic scheduling. More precisely, we generalize the traditional stochastic scheduling problem to a stochastic online model where the jobs are presented to the scheduler sequentially, and the scheduler must immediately assign jobs to machines, without knowledge of the jobs to come. This assignment of jobs to machines must not be revised later, and therefore we also speak of *fixed-assignment* policies. We show that in this model, there still exists a fixed-assignment policy that indeed achieves the same performance guarantee as the above mentioned bound for the WSEPT rule. In this context, it is probably worthy to mention that the WSEPT rule is clearly not a fixed-assignment policy, since the assignment of jobs to machines depends on the actual realizations of the processing times.

For the model with release dates, $P|r_j|\mathbb{E}[\sum w_jC_j]$ we prove a slightly more complicated performance guarantee that is valid for a class of processing time distributions that we call δ-NBUE, generalizing the well known class of NBUE distributions (new better than used in expectation).

DEFINITION 1.2 (δ-NBUE) *A non-negative random variable X is δ-NBUE if, for $\delta \geq 1$, $\mathbb{E}[X - t \,|\, X > t\,] \leq \delta\,\mathbb{E}[X]$ for all $t \geq 0$.*

Ordinary NBUE distributions are therefore 1-NBUE by definition. For a NBUE random variable X, Hall and Wellner [11] showed that the (squared) coefficient of variation is bounded by 1, that is, $\mathrm{Var}[X]/\mathbb{E}[X]^2 \leq 1$. With their techniques, it also follows that, if X is δ-NBUE, then $\mathrm{Var}[X]/\mathbb{E}[X]^2 \leq 2\delta - 1$. Examples of ordinary NBUE (or 1-NBUE) distributions are exponential, Erlang, uniform, or Weibull distributions (the latter with shape parameter at least 1). For the stochastic online model $P|r_j|\mathbb{E}[\sum w_jC_j]$, we obtain a performance bound of

$$\rho = 1 + \max\left\{1 + \frac{\delta}{\alpha}\ ,\ \alpha + \delta + \frac{(m-1)(\Delta+1)}{2m}\right\},$$

for processing times that follow δ-NBUE distributions. Here, $\alpha > 0$ is an arbitrary parameter, and again, Δ is an upper bound on the squared coefficients of variation $\mathrm{Var}[P_j]/\mathbb{E}[P_j]^2$ of the processing time distributions. In particular, since $\Delta \leq 2\delta - 1$ for δ-NBUE distributions, $\rho \leq 1 + \max\{1 + \delta/\alpha\ ,\ \alpha + \delta(2m-1)/m\}$. Optimizing for α yields that $\rho < 3/2 + (1 - 1/2m)\delta + \sqrt{1 + 4\delta^2}/2$. For for ordinary NBUE (or 1-NBUE) distributions, where $\delta = 1$, we thus obtain a performance bound strictly less than $(\sqrt{5}+5)/2 - 1/(2m) \approx 3.62 - 1/(2m)$. Thereby, we improve upon the previously best known performance guarantee of $4 - 1/m$ for NBUE distributions, which was derived for an LP based list scheduling policy [20]. Notice that this improved bound holds even though we consider an online model in which the jobs arrive over time, and the scheduler does not know anything about the jobs that are about to arrive in the future. For deterministic processing times, where $\Delta = 0$ and $\delta = 1$, we get $\rho = 2 + \max\{1/\alpha\ ,\ \alpha + (m-1)/(2m)\}$, and optimizing for α yields $\rho \approx 3.28$, matching the currently best known bound from [18].

Our results are achieved by the following, quite simple SOS policy, which is in fact a fixed-assignment policy. On every machine, the jobs are processed in the WSEPT order; in the case with release dates, this is in fact a modified version of the WSEPT order that will be explained later. To make the online decisions on machine assignments, at any time when a job is presented, it is assigned to that machine where it causes the minimal expected increase in the objective value, given the jobs that have been assigned so far, and assuming that the jobs on each machine will be scheduled in WSEPT order. In fact, this can be interpreted as the derandomized version of a policy that assigns jobs uniformly at random to the machines. Intuitively, the reason why we can recover, respectively improve the previous best known results in stochastic machine scheduling is the following: On the one hand, we restrict the full power of scheduling policies by fixing machine assignments immediately upon arrival of a job. On the other hand, it is precisely this fixed machine assignment, together with an averaging argument over the number of machines, that allows an improvement in the analysis in comparison to general scheduling policies. We again mention that, to obtain our results, we utilize one of the linear programming based lower bounds of [20]; but we do not require the solution of linear programs for the definition of the online scheduling policies.

2. Model definition, notation, and preliminaries. Let n be the number of jobs, index $j \in J = \{1, \ldots, n\}$ denote a job, with associated weight w_j and processing time distribution P_j. By $\mathbb{E}[P_j]$ we denote its expected processing time, and p_j denotes a particular realization of P_j. The processing time distributions P_j are assumed to be independent. In the model with release dates, r_j denotes the earliest point in time when job j can be started. Given a schedule of start times $S_1, \ldots, S_n$ for a particular realization $p = (p_1, \ldots, p_n)$ of processing times, $C_j = S_j + p_j$ is the completion time of job j, $j = 1, \ldots, n$. Each job must be processed non-preemptively, on any of the m machines, and each machine can process at most one job at a time. The goal is to find a scheduling policy that minimizes the expected value of the weighted completion times of jobs, $\mathbb{E}[\sum w_j\, C_j]$.

A scheduling policy eventually yields a feasible m-machine schedule for each realization p of the processing times. For a given policy, denoted by Π, let $S_j^\Pi(p)$ and $C_j^\Pi(p)$ denote the start and completion

times of job j for a given realization p, and let $S_j^\Pi(P)$ and $C_j^\Pi(P)$ denote the associated random variables. Unless there is danger of ambiguity, we also write S_j and C_j, for short. We let

$$\mathbb{E}\left[Z^\Pi\right] = \mathbb{E}\left[\sum\nolimits_j w_j\, C_j^\Pi(P)\right]$$

denote the expected performance of a scheduling policy Π. Then, if OPT is an optimal scheduling policy according to the most general definition of stochastic scheduling policies in [19], we say that a policy Π is a *ρ–approximation* if, for some $\rho \geq 1$,

$$\mathbb{E}\left[Z^\Pi\right] \leq \rho\, \mathbb{E}\left[Z^{\mathrm{OPT}}\right].$$

We assume that the jobs are arriving over time in the order $1, \ldots, n$. Therefore, we can assume w.l.o.g. that $r_j \leq r_k$ for $j < k$. However, the number of jobs n is not known in advance. When a job arrives, at time r_j, the scheduler is informed about its weight w_j and its expected processing time $\mathbb{E}[P_j]$. For a given job $j \in J$, let us define by $H(j)$ the jobs that have a higher priority in the WSEPT ordering, that is

$$H(j) = \left\{k \in J \mid \frac{w_k}{\mathbb{E}[P_k]} > \frac{w_j}{\mathbb{E}[P_j]}\right\} \cup \left\{k \leq j \mid \frac{w_k}{\mathbb{E}[P_k]} = \frac{w_j}{\mathbb{E}[P_j]}\right\}.$$

Accordingly, define

$$L(j) = J \backslash H(j)$$

as those jobs that have lower priority in the WSEPT order. As a tie-breaking rule for jobs k with equal ratio $w_k/\mathbb{E}[P_k] = w_j/\mathbb{E}[P_j]$ we decide depending on the position in the online sequence relative to j. That is, if $k \leq j$ then k belongs to set $H(j)$, otherwise it is included in set $L(j)$. Notice that, by convention, we assume that $H(j)$ contains job j, too.

It is clear that the online scheduling policies for the SoS model are nothing but a certain subclass of stochastic scheduling policies in general. This because, given a traditional stochastic scheduling problem, we could just define a policy for the SoS model by requiring that the policy must not use any information about jobs that are released in the future, at any time. Therefore, the expected performance of an optimal SoS policy is by definition no less than the expected performance of an optimal policy for a corresponding traditional stochastic problem. Hence, lower bounds on the expected objective value of an optimal policy carry over to the stochastic online setting that we consider in this paper. We crucially exploit that fact, and will utilize the following lower bound on the expected performance $\mathbb{E}\left[Z^{\mathrm{OPT}}\right]$ of an optimal stochastic scheduling policy. This lower bound is a generalization of a lower bound by Eastman et al. [8] for the deterministic setting $\mathrm{P}\,|\,|\sum w_j\, C_j$.

LEMMA 2.1 (Möhring et al. [20]) *For any instance of* $\mathrm{P}\,|\,r_j\,|\,\mathbb{E}[\sum w_j\, C_j]$*, we have that*

$$\mathbb{E}\left[Z^{\mathrm{OPT}}\right] \geq \sum_j w_j \sum_{k \in H(j)} \frac{\mathbb{E}[P_k]}{m} \;-\; \frac{(m-1)(\Delta-1)}{2m} \sum_j w_j \mathbb{E}[P_j]\,,$$

where Δ bounds the squared coefficient of variation of the processing times, that is, $\mathrm{Var}[P_j]/\mathbb{E}[P_j]^2 \leq \Delta$ *for all jobs $j = 1, \ldots, n$ and some $\Delta \geq 0$.*

3. Stochastic online scheduling on a single machine. In this section we consider the problem of stochastic online scheduling on a single machine. When all release dates are identical, it is well known that the WSEPT rule is optimal [23, 29].

Consider now the problem of scheduling jobs with nontrivial release dates on a single machine. The currently best known result from stochastic scheduling is an LP-based list scheduling algorithm with a performance bound of 3 [20]. Inspired by a corresponding algorithm for the deterministic (online) setting on parallel machines from [18], we propose the following scheduling policy for the stochastic online scheduling model on a single machine.

Algorithm 1: α-SHIFT-WSEPT

Modify the release date r_j of each job j to $r'_j = \max\{r_j, \alpha\,\mathbb{E}[P_j]\}$, for some fixed $\alpha > 0$. At any time t, when the machine is idle, start the job with the highest priority in the WSEPT order among all available jobs (respecting the modified release dates).

In order to analyze this policy, we first derive an upper bound on the expected completion time of a job, $\mathbb{E}\left[C_j^\alpha\right]$, when scheduling jobs on a single machine according to the α-SHIFT-WSEPT policy.

LEMMA 3.1 *Let all processing times be δ-NBUE. Then the expected completion time of job j for α-SHIFT-WSEPT on a single machine can be bounded by*

$$\mathbb{E}\left[C_j^\alpha\right] \le (1+\delta/\alpha)\, r'_j + \sum_{k\in H(j)} \mathbb{E}\left[P_k\right] .$$

PROOF. We consider some job j. Let us denote by $\mathcal{B}$ the event that the machine is busy processing some job at time r'_j, and let us denote by $\mathcal{I}$ the complement of $\mathcal{B}$, namely that the machine is idle (or just finished processing some job) at time r'_j. Under the condition $\mathcal{I}$ it could still be that there are higher priority jobs $k \in H(j)\setminus\{j\}$ available at time r'_j, but in any case the expected start time of job j can be postponed by at most

$$\sum_{k\in H(j)\setminus\{j\}} \mathbb{E}\left[P_k \mid \mathcal{I}\,\right] .$$

However, due to independence of processing times, we have that $\mathbb{E}\left[P_k \mid \mathcal{I}\,\right] = \mathbb{E}\left[P_k\right]$, and therefore

$$\mathbb{E}\left[S_j^\alpha \mid \mathcal{I}\,\right] \;\le\; r'_j \;+ \sum_{k\in H(j)\setminus\{j\}} \mathbb{E}\left[P_k\right] .$$

Consider condition $\mathcal{B}$ and let us denote by $\mathbb{E}\left[x(\mathcal{B})\right]$ the expected length of the time period until the machine becomes idle for the first time after r'_j. Under the condition $\mathcal{B}$, for any realization p of the processing times, conditioned on $\mathcal{B}$, some job $\ell(p)$ is in process at time r'_j (in fact, $\ell(p)$ might have lower or higher priority than j). Any such job ℓ was available at time $r'_\ell < r'_j$, and by definition of the modified release dates, we therefore know that $\mathbb{E}\left[P_\ell\right] \le (1/\alpha) r'_\ell < (1/\alpha) r'_j$ for any such job ℓ. Moreover, letting $t = r'_j - S_\ell^\alpha$, the expected remaining processing time of such job ℓ, conditioned on the fact that it is indeed in process at time r'_j, is $\mathbb{E}\left[P_\ell - t \mid P_\ell > t\,\right]$. Due to the assumption of δ-NBUE processing times, we thus know that

$$\mathbb{E}\left[P_\ell - t \mid P_\ell > t\,\right] \;\le\; \delta\,\mathbb{E}\left[P_\ell\right] \;\le\; (\delta/\alpha) r'_j .$$

Therefore, the expected remaining processing time of any job ℓ that might be in process at time r'_j is bounded by $(\delta/\alpha) r'_j$, and thus

$$\mathbb{E}\left[x(\mathcal{B})\right] \;\le (\delta/\alpha) r'_j .$$

Repeating the same argument as above, we can now conclude that

$$\mathbb{E}\left[S_j^\alpha \mid \mathcal{B}\,\right] \;\le\; (1+\delta/\alpha)\, r'_j \;+ \sum_{k\in H(j)\setminus\{j\}} \mathbb{E}\left[P_k\right] . \tag{1}$$

As each of the two conditional expectations $\mathbb{E}\left[S_j^\alpha \mid \mathcal{I}\,\right]$ and $\mathbb{E}\left[S_j^\alpha \mid \mathcal{B}\,\right]$ is bounded by the right hand side of (1), we obtain that

$$\mathbb{E}\left[S_j^\alpha\right] \;\le\; (1+\delta/\alpha)\, r'_j \;+ \sum_{k\in H(j)\setminus\{j\}} \mathbb{E}\left[P_k\right] ,$$

and the fact that $\mathbb{E}\left[C_j^\alpha\right] = \mathbb{E}\left[S_j^\alpha\right] + \mathbb{E}\left[P_j\right]$ concludes the proof. □

In fact, it is quite straightforward to use Lemma 3.1 in order to show the following.

THEOREM 3.1 *The α-SHIFT-WSEPT algorithm is a $(\delta+2)$-approximation for the stochastic online single machine problem $1|r_j|\mathbb{E}\left[\sum w_j C_j\right]$, for δ-NBUE processing times. The best choice of the parameter α is $\alpha = 1$.*

PROOF. With Lemma 3.1 and the definition of modified release dates $r'_j = \max\{r_j, \alpha\,\mathbb{E}\left[P_j\right]\}$ we can bound the expected value of a schedule obtained by α-SHIFT-WSEPT.

$$\begin{aligned}
\sum_j w_j\,\mathbb{E}\left[C_j^\alpha\right] \quad &\le \quad (1+\delta/\alpha)\sum_j w_j\ \max\left\{r_j, \alpha\,\mathbb{E}\left[P_j\right]\right\} + \sum_j w_j \sum_{k\in H(j)} \mathbb{E}\left[P_k\right] \\
&= \quad \sum_j w_j\ \max\left\{\,(1+\delta/\alpha)\, r_j,\ (\alpha+\delta)\,\mathbb{E}\left[P_j\right]\,\right\} + \sum_j w_j \sum_{k\in H(j)} \mathbb{E}\left[P_k\right] \\
&= \quad \max\left\{\,(1+\delta/\alpha)\,,\ (\alpha+\delta)\,\right\} \sum_j w_j\,(r_j + \mathbb{E}\left[P_j\right]) + \sum_j w_j \sum_{k\in H(j)} \mathbb{E}\left[P_k\right] .
\end{aligned}$$

We can now apply the trivial lower bound $\sum_j w_j(r_j + \mathbb{E}[P_j]) \le \mathbb{E}\left[Z^{\mathrm{OPT}}\right]$, and exploit the fact that $\sum_j w_j \sum_{k\in H(j)} \mathbb{E}[P_k] \le \mathbb{E}\left[Z^{\mathrm{OPT}}\right]$ by Lemma 2.1 (for $m = 1$), and obtain

$$\sum_j w_j \,\mathbb{E}\left[C_j^{\alpha}\right] \;\le\; (1 + \max\{\,(1+\delta/\alpha)\,,\;(\alpha+\delta)\,\}\,)\;\mathbb{E}\left[Z^{\mathrm{OPT}}\right].$$

Simple analysis shows that $\alpha = 1$ minimizes the expression above, independently of δ, and the theorem follows. □

Note that for NBUE processing times, the result matches the best known performance bound of 3 derived by Möhring et. al in [20] for the traditional stochastic scheduling model. In contrast to our policy, their LP-based policy requires an a priori knowledge of all jobs. In the (deterministic) online setting, the best possible algorithm is 2-competitive as is shown in [12]. This leaves a gap of 1 compared to our result, which, however, holds for stochastic (NBUE) processing time distributions as well.

4. Stochastic online scheduling on parallel machines. In this section, we define stochastic online scheduling policies for the problem on parallel machines. In order to motivate our policies, we first consider the relaxed version of the problem, in which all jobs have the same release date, $r_j = 0$. Later, we generalize the policy for the problem with nontrivial release dates.

4.1 Scheduling jobs without release dates. In the case that all jobs arrive at time 0, the problem effectively turns into a traditional stochastic scheduling problem, $\mathrm{P}\,|\,|\,\mathbb{E}[\sum w_j\,C_j]$. For that problem, it is known that the WSEPT rule yields a $(1 + (m-1)(\Delta+1)/(2m))$-approximation, Δ being an upper bound on the squared coefficients of variation of the processing time distributions [20].

In this paper, however, we consider an online variant of the problem $\mathrm{P}\,|\,|\,\mathbb{E}[\sum w_j\,C_j]$, that resembles the online-list model from online optimization. We assume that the jobs are presented to the scheduler sequentially, and each job must immediately and irrevocably be assigned to a machine: a *fixed-assignment* policy. In particular, during this assignment phase, the scheduler does not know anything about the jobs that are still about to come. We show that an intuitive and simple fixed-assignment policy exists in this model that eventually yields the same performance bound as the one proved in [20] for the WSEPT rule. Note, however, that the WSEPT rule is not a feasible policy in the online-list model we consider here.

We first introduce the following notation: If a job j is assigned to machine i, this is denoted by $j \to i$. Now we can define the MinIncrease policy as follows.

Algorithm 2: MinIncrease

When a job j is presented to the scheduler, it is assigned to the machine i that minimizes the expression

$$\mathrm{incr}\,(j,i) \;=\; w_j \cdot \sum_{k\in H(j),\,k\le j,\,k\to i} \mathbb{E}[P_k] \;\;+\;\; \mathbb{E}[P_j] \cdot \sum_{k\in L(j),\,k<j,\,k\to i} w_k \;+ w_j \mathbb{E}[P_j]\,.$$

On each machine, the jobs assigned to this machine are sequenced in WSEPT order.

Since WSEPT is known to be optimal on a single machine, MinIncrease in fact assigns each job j to that machine where it causes the least increase in the expected value, given the previously assigned jobs.

Theorem 4.1 *Consider the stochastic online scheduling problem on parallel machines,* $\mathrm{P}|\,|\mathbb{E}[\sum w_j\,C_j]$, *as described above. Given that* $\mathrm{Var}[P_j]/\mathbb{E}[P_j]^2 \le \Delta$ *for all jobs* j *and some constant* $\Delta \ge 0$, *the* MinIncrease *policy is a* ρ*–approximation, where*

$$\rho = 1 + \frac{(m-1)(\Delta+1)}{2m}\,.$$

Proof. Consider some job j, and denote by $\mathbb{E}[\mathrm{incr}\,(j)\,]$ the increase in the expected objective value caused by fixing the assignment of job j using MinIncrease. Since MinIncrease chooses the machine

i on which j causes the least expected increase, the expected increase is

$$\begin{aligned}
\mathbb{E}\left[\operatorname{incr}(j)\right] &= \min_i\{\mathbb{E}\left[\operatorname{incr}(j,i)\right]\} \\
&= \min_i\left\{ w_j \sum_{k\in H(j),\, k\to i,\, k<j} \mathbb{E}\left[P_k\right] \;+\; \mathbb{E}\left[P_j\right] \sum_{k\in L(j),\, k\to i,\, k<j} w_k \right\} + w_j\mathbb{E}\left[P_j\right] \\
&\le \frac{1}{m}\left(w_j \sum_{k\in H(j),\, k<j} \mathbb{E}\left[P_k\right] \;+\; \mathbb{E}\left[P_j\right] \sum_{k\in L(j),\, k<j} w_k \right) + w_j\mathbb{E}\left[P_j\right] ,
\end{aligned}$$

where the inequality holds because the least expected increase is not more than the average expected increase over all machines. By summing up these quantities over all jobs we obtain the expected performance $\mathbb{E}\left[Z^{\mathrm{MI}}\right]$ of the MinIncrease policy.

$$\begin{aligned}
\mathbb{E}\left[Z^{\mathrm{MI}}\right] &= \sum_j \mathbb{E}\left[\operatorname{incr}(j)\right] \\
&\le \frac{1}{m}\sum_j \left(w_j \sum_{k\in H(j),\, k<j} \mathbb{E}\left[P_k\right] \;+\; \mathbb{E}\left[P_j\right] \sum_{k\in L(j),\, k<j} w_k \right) + \sum_j w_j\mathbb{E}\left[P_j\right] \\
&= \frac{1}{m}\sum_j w_j \sum_{k\in H(j)} \mathbb{E}\left[P_k\right] \;+\; \frac{m-1}{m}\sum_j w_j\mathbb{E}\left[P_j\right] ,
\end{aligned}$$

where the last equality holds by an index rearrangement, since

$$\sum_j w_j \sum_{k\in H(j),\, k>j} \mathbb{E}\left[P_k\right] \;=\; \sum_j \mathbb{E}\left[P_j\right] \sum_{k\in L(j),\, k<j} w_k \,.$$

Now, we plug in the inequality of Lemma 2.1, and using the trivial fact that $\sum_j w_j\mathbb{E}\left[P_j\right]$ is a lower bound for the expected performance $\mathbb{E}\left[Z^{\mathrm{OPT}}\right]$ of an optimal policy, we obtain

$$\begin{aligned}
\mathbb{E}\left[Z^{\mathrm{MI}}\right] &\le \mathbb{E}\left[Z^{\mathrm{OPT}}\right] + \frac{(m-1)(\Delta-1)}{2m}\sum_j w_j\mathbb{E}\left[P_j\right] + \frac{m-1}{m}\sum_j w_j\mathbb{E}\left[P_j\right] \\
&\le \left(1 + \frac{(m-1)(\Delta+1)}{2m}\right)\cdot \mathbb{E}\left[Z^{\mathrm{OPT}}\right] .
\end{aligned}$$

□

As mentioned above, this performance guarantee matches the currently best known performance guarantee for the traditional stochastic setting, which was derived for the performance of the WSEPT rule in [20]. The WSEPT rule, however, requires the knowledge of all jobs with their weights w_j and expected processing times $\mathbb{E}\left[P_j\right]$ at the outset. In contrast, the MinIncrease policy decides on machine assignments online, without any knowledge of the jobs to come. Finally, it is worthy to note that simple instances show that these two policies are indeed different.

Applied to deterministic processing times, where $\Delta = 0$, the MinIncrease policy thus achieves an approximation guarantee of $3/2 - 1/(2m)$. The WSPT rule has a slightly better competitive ratio of $(1+\sqrt{2})/2 \approx 1.21$ as shown in [14].

Lower bound for fixed assignment policies. The requirement of a fixed assignment of jobs to machines beforehand may be interpreted as ignoring the additional information that evolves over time in the form of the actual realization of processing times. In the following, we therefore give a lower bound on the expected performance $\mathbb{E}\left[Z^{\mathrm{FIX}}\right]$ of an optimal stochastic scheduling policy FIX that assigns jobs to machines beforehand. A fortiori, this lower bound holds for the best possible SoS policy, too.

Theorem 4.2 *For stochastic parallel machine scheduling with unit weights and i.i.d. exponential processing times,* $\mathrm{P}|p_j \sim \exp(1)|\mathbb{E}\left[\sum C_j\right]$*, there exist instances such that*

$$\mathbb{E}\left[Z^{\mathrm{FIX}}\right] \ge 3(\sqrt{2}-1)\cdot \mathbb{E}\left[Z^{\mathrm{OPT}}\right] - \varepsilon ,$$

for any $\varepsilon > 0$*. Here,* $3(\sqrt{2}-1) \approx 1.24$*. Hence, no policy that uses fixed assignments of jobs to machines can perform better in general.*

Notice that the theorem is formulated for the special case of exponentially distributed processing times. Stronger bounds can be obtained for arbitrary distributions. However, since our performance guarantees, as in [20], depend on the coefficient of variation of the processing times, we are particularly interested in lower bounds for classes of distributions where this coefficient of variation is small. The coefficient of variation of exponentially distributed random variables equals 1. For example, for the case of $m = 2$ machines, we get a lower bound of $8/7 \approx 1.14$ on the performance of any fixed-assignment policy, and for that case the performance bound of MinIncrease equals $2 - 1/m = 1.5$.

Proof of Theorem 4.2. For simplicity, we will prove a slightly worse lower bound. Let us consider an instance with m machines and $n = m + \lceil m/2 \rceil$ exponentially distributed jobs, $P_j \sim \exp(1)$. The optimal stochastic scheduling policy is SEPT, shortest expected processing time first [4, 35], and the expected performance is (see, e.g., [31, Cor. 3.5.17])

$$\mathbb{E}\left[Z^{\mathrm{OPT}}\right] = \mathbb{E}\left[Z^{\mathrm{SEPT}}\right] = \sum_j \mathbb{E}\left[C_j^{\mathrm{SEPT}}\right] = m + \sum_{j=m+1}^{n} \frac{j}{m}\,.$$

The best fixed assignment policy assigns 2 jobs each to $\lceil m/2 \rceil$ of the machines, and 1 job each to $\lfloor m/2 \rfloor$ of the machines. Hence, there are m jobs with $\mathbb{E}[C_j]=1$, and $\lceil m/2 \rceil$ jobs with $\mathbb{E}[C_j]=2$. The expected performance for the best fixed assignment policy FIX is

$$\mathbb{E}\left[Z^{\mathrm{FIX}}\right] = \sum_j \mathbb{E}\left[C_j^{\mathrm{FIX}}\right] = m + 2 \cdot \lceil m/2 \rceil\,.$$

For small values $m = 2, 3, 4 \ldots$, we calculate $\mathbb{E}\left[Z^{\mathrm{FIX}}\right]/\mathbb{E}\left[Z^{\mathrm{OPT}}\right] = 8/7$, $7/6, 32/27, \ldots$. It is easy to see that

$$\frac{\mathbb{E}\left[Z^{\mathrm{FIX}}\right]}{\mathbb{E}\left[Z^{\mathrm{OPT}}\right]} = \frac{16m^2}{13m^2 + \mathrm{o}(m^2)}\,,$$

and for $m \to \infty$ we get a lower bound of $16/13 \approx 1.23$. Now the claim of the theorem follows along the same lines if we redefine the number of jobs as $n = m + \lceil \sqrt{2}m \rceil$. □

Lower bound for MinIncrease**.** The lower bound on the performance ratio for *any* fixed assignment policy given in Theorem 4.2 holds for the MinIncrease policy, too. Hence, MinIncrease cannot be better than 1.24-approximative. For general (i.e., non-exponential) probability distributions we obtain a lower bound of 3/2 on the expected performance of MinIncrease relative to the expected performance of an optimal scheduling policy, as shown by the following instance.

Example 4.1 *The instance consists of $n-1$ deterministic unit length jobs and one job with a stochastic two-point distributed processing time. There are $m = 2$ machines, and we assume that n, the number of jobs is even. The $n-1$ deterministic jobs have unit weight $w_j = 1$; they appear first in the online sequence. The final job in the online sequence is the stochastic job. It has processing time $p_j = n^2/4$ with probability $2/n$, and $p_j = 1$ with probability $1 - 2/n$. The weight w_j of the stochastic job equals the value of its expected processing time, i.e. $1 - 2/n + n/2$.*

The MinIncrease policy assigns $n/2 - 1$ deterministic jobs to one machine, and $n/2$ deterministic jobs to the other. The stochastic job is assigned to the machine with $n/2 - 1$ deterministic jobs. Hence, the expected objective value of the schedule under MinIncrease is $\mathbb{E}[\sum w_j C_j] = 3n^2/4 + o(n^2)$. An optimal stochastic policy would start the uncertain job and one deterministic job at time 0. At time $t = 1$ it is known if the stochastic job has completed, or if it blocks the machine for another $n^2/4 - 1$ time units. If the stochastic job has completed then the remaining unit jobs are distributed equally on both machines, otherwise all deterministic jobs are scheduled on the same machine. Thus, the expected objective value of an optimal schedule is $\mathbb{E}\left[Z^{\mathrm{OPT}}\right] = n^2/2 + o(n^2)$. The ratio of both values tends to 3/2 if the number of jobs goes to infinity.

Notice, however, that this result is less meaningful in comparison to the performance bound of Theorem 4.1, which depends on an upper bound Δ on the squared coefficient of variation.

4.2 Scheduling jobs with nontrivial release dates. In this section, we consider the setting where jobs arrive over time, that is, the stochastic online version of $\mathrm{P}\,|\,r_j\,|\,\mathbb{E}\left[\sum w_j\,C_j\right]$. The main idea is to adopt the MinIncrease policy to this setting. However, the difference is that we are no longer equipped with an optimal policy (i.e., WSEPT) to schedule the jobs that are assigned to a single machine. Even worse, even if we knew such a policy for a single machine, it would not be straightforward how to use it in the setting with parallel machines to define a feasible online scheduling policy. However, we show that we can in fact use the α-Shift-WSEPT rule as introduced in Section 3 to sequence the jobs that are assigned to the same machine.

The modified MinIncrease policy behaves as follows: A job j, released at time r_j, is immediately assigned to machine i_j, where machine i_j is chosen exactly as previously in the case without release dates. In a sense, in assigning jobs to machines, we thus ignore completely the existence of release dates. The crucial observation is that the α-Shift-WSEPT policy on machine i_j learns about job j's existence also at time r_j. Hence, for each single machine, it is indeed feasible to use the α-Shift-WSEPT rule, and the so-defined policy is a feasible SoS policy.

Theorem 4.3 *Consider the stochastic online scheduling problem on parallel machines with release dates,* $\mathrm{P}|r_j|\mathbb{E}\left[\sum w_j\,C_j\right]$. *Given that all processing times are δ-NBUE, the* MinIncrease *policy running α-*Shift-WSEPT *on each single machine is a ρ–approximation, where*

$$\rho = 1 + \max\left\{1+\frac{\delta}{\alpha}\ ,\ \alpha+\delta+\ \frac{(m-1)(\Delta+1)}{2m}\right\}\ ,$$

Here, Δ is such that $\mathrm{Var}[P_j]/\mathbb{E}\left[P_j\right]^2 \le \Delta$ *for all jobs j. In particular, since all processing times P_j are δ-NBUE, we know that $\Delta \le 2\delta - 1$, hence $\rho \le 1 + \max\left\{1+\delta/\alpha,\ \alpha+\delta(2m-1)/m\right\}$.*

Proof. Let i_j be the machine to which job j is assigned. Then, by Lemma 3.1 we know that

$$\mathbb{E}\left[C_j\right] \le (1+\frac{\delta}{\alpha})r'_j + \sum_{k\in H(j),\,k\to i_j} \mathbb{E}\left[P_k\right], \tag{2}$$

and the expected value of MinIncrease can be bounded by

$$\mathbb{E}\left[Z^{\mathrm{MI}}\right] \le \left(1+\frac{\delta}{\alpha}\right)\sum_j w_j\,r'_j + \sum_j w_j \sum_{k\in H(j),\,k\to i_j} \mathbb{E}\left[P_k\right]\ . \tag{3}$$

The double sum over all jobs and their higher priority jobs can be split depending on the position of jobs in the online sequence.

$$\begin{aligned}\sum_j w_j \sum_{k\in H(j),\,k\to i_j} \mathbb{E}\left[P_k\right] &= \sum_j w_j \left(\sum_{k\in H(j),\,k\to i_j,k<j} \mathbb{E}\left[P_k\right] \ +\ \sum_{k\in H(j),\,k\to i_j,\,k>j} \mathbb{E}\left[P_k\right] \ +\ \mathbb{E}\left[P_j\right]\right)\\ &= \sum_j \left(w_j \sum_{k\in H(j),\,k\to i_j,\,k<j} \mathbb{E}\left[P_k\right] \ +\ \mathbb{E}\left[P_j\right] \sum_{k\in L(j),\,k\to i_j,\,k<j} w_k \ +\ w_j\mathbb{E}\left[P_j\right]\right)\ ,\end{aligned}$$

where the second equality holds by the same index rearrangement as in the proof of Theorem 4.1.

By definition of MinIncrease, we know that job j is assigned to the machine which minimizes the sums in parenthesis of the right hand side of this equation. Hence, by an averaging argument, we know that

$$\begin{aligned}\sum_j w_j \sum_{k\in H(j),\,k\to i_j} \mathbb{E}\left[P_k\right] \ &\le\ \sum_j \left(w_j \sum_{k\in H(j),\,k<j} \frac{\mathbb{E}\left[P_k\right]}{m} + \mathbb{E}\left[P_j\right] \sum_{k\in L(j),\,k<j} \frac{w_k}{m} + w_j\mathbb{E}\left[P_j\right]\right)\\ &= \sum_j w_j \sum_{k\in H(j)} \frac{\mathbb{E}\left[P_k\right]}{m} \ +\ \frac{m-1}{m}\sum_j w_j\mathbb{E}\left[P_j\right]\ ,\end{aligned}$$

where the last equality follows from index rearrangement. Plugging this into (3), leads to the following bound on the expected performance of MinIncrease.

$$\mathbb{E}\left[Z^{\mathrm{MI}}\right] \le \left(1+\frac{\delta}{\alpha}\right)\sum_j w_j r'_j + \sum_j w_j \sum_{k\in H(j)} \frac{\mathbb{E}\left[P_k\right]}{m} + \frac{m-1}{m}\sum_j w_j\mathbb{E}\left[P_j\right]\ .$$

Plugging the bound of Lemma 2.1 into the above inequality, we obtain

$$\begin{aligned}\mathbb{E}\left[Z^{\mathrm{MI}}\right] &\le \left(1+\frac{\delta}{\alpha}\right)\sum_j w_j r'_j + \mathbb{E}\left[Z^{\mathrm{OPT}}\right] + \frac{(m-1)(\Delta+1)}{2m}\sum_j w_j \mathbb{E}\left[P_j\right] \\ &= \mathbb{E}\left[Z^{\mathrm{OPT}}\right] + \sum_j w_j \left(\left(1+\frac{\delta}{\alpha}\right)r'_j + \frac{(m-1)(\Delta+1)}{2m}\mathbb{E}\left[P_j\right]\right), \qquad (4)\end{aligned}$$

where again, Δ is an upper bound on the squared coefficient of variation of the processing time distributions P_j. By bounding r'_j by $r_j + \alpha\mathbb{E}\left[P_j\right]$, we obtain the following bound on the term in parenthesis of the sum in the right hand side of inequality (4).

$$\begin{aligned}&\left(1+\frac{\delta}{\alpha}\right)r'_j + \frac{(m-1)(\Delta+1)}{2m}\mathbb{E}\left[P_j\right] \\ &\quad\le \left(1+\frac{\delta}{\alpha}\right)r_j + \left(\alpha+\delta+\frac{(m-1)(\Delta+1)}{2m}\right)\mathbb{E}\left[P_j\right] \\ &\quad\le \left(r_j+\mathbb{E}\left[P_j\right]\right)\max\left\{1+\frac{\delta}{\alpha}\ ,\ \alpha+\delta+\frac{(m-1)(\Delta+1)}{2m}\right\}.\end{aligned}$$

By using this inequality in equation (4), and applying the trivial lower bound $\sum_j w_j(r_j + \mathbb{E}\left[P_j\right]) \le \mathbb{E}\left[Z^{\mathrm{OPT}}\right]$ on the expected optimum performance, we get the claimed performance bound of

$$\rho = 1 + \max\left\{1+\frac{\delta}{\alpha}\ ,\ \alpha+\delta+\frac{(m-1)(\Delta+1)}{2m}\right\}.$$

Since all processing times are δ-NBUE, we know that $\Delta \le 2\delta - 1$ and thus the second claim of the theorem follows, namely $\rho \le 1 + \max\{1+\delta/\alpha, \alpha+\delta+\delta(2m-1)/m\}$. □

For NBUE processing times, where $\Delta = \delta = 1$, Theorem 4.3 yields a performance bound of

$$\rho = 2 + \max\{\frac{1}{\alpha},\ \alpha + \frac{m-1}{m}\}.$$

This term is minimal for $\alpha = (\sqrt{5m^2-2m+1} - m + 1)/(2m)$, which yields a ratio of $\rho = 2 + (\sqrt{5m^2-2m+1} + m - 1)/(2m)$. This is less than $(5+\sqrt{5})/2 - 1/(2m) \approx 3.62 - 1/(2m)$ improving upon the previously best known bound of $4-1/m$ from [20] for the traditional stochastic problem. More generally, for δ-NBUE processing times, optimizing the term $\max\{1+\delta/\alpha,\ \alpha+\delta+\delta(2m-1)/m\}$ for α yields $\rho < 3/2 + \delta(2m-1)/2m + \sqrt{4\delta^2+1}/2$.

Moreover, for deterministic processing times, where $\Delta = 0$ and $\delta = 1$, Theorem 4.3 yields a performance bound of

$$\rho = 2 + \max\{\frac{1}{\alpha},\ \alpha + \frac{m-1}{2m}\}.$$

Optimizing for α yields $\alpha = (\sqrt{17m^2-2m+1} - m + 1)/(4m)$, which yields a ratio of $\rho = 2 + (\sqrt{17m^2-2m+1} + m - 1)/(4m)$. This is less than 3.281 for any value of m, matching the currently best known bound from [18] for (deterministic) online scheduling.

4.3 Randomized job assignment. In fact, the MinIncrease policy can be interpreted as the derandomized version of a policy that assign jobs uniformly at random to the machines. Even though randomly assigning jobs to machines ignores much information, it is nevertheless known to be quite powerful as has been observed already by, e.g., Schulz and Skutella [25]. They apply a random assignment strategy, based on the solution of an LP-relaxation, for scheduling jobs with deterministic processing times on unrelated machines. For the special case of identical machines, their approach boils down to assigning jobs uniformly at random to the machines. The random assignment strategy for the stochastic online scheduling problem at hand is as follows.

Algorithm 3: RandAssign

When a job is presented to the scheduler, it is assigned to machine i with probability $1/m$ for all $i = 1, \ldots, m$. The jobs assigned to machine i are scheduled according to the α-Shift-WSEPT policy.

THEOREM 4.4 *Consider the stochastic online scheduling problem on parallel machines with release dates,* $P|r_j|\mathbb{E}\left[\sum w_j C_j\right]$. *Given that all processing times are δ-NBUE, the* RANDASSIGN *policy running α-*SHIFT-WSEPT *on each single machine is a ρ–approximation, where*

$$\rho = 1 + \max\left\{1 + \frac{\delta}{\alpha}\ ,\ \alpha + \delta + \frac{(m-1)(\Delta+1)}{2m}\right\}\ ,$$

Here, Δ is such that $\mathrm{Var}[P_j]/\mathbb{E}\left[P_j\right]^2 \le \Delta$ *for all jobs j. In particular, since all processing times P_j are δ-NBUE, we know that $\Delta \le 2\delta - 1$, hence $\rho \le 1 + \max\{1 + \delta/\alpha,\ \alpha + \delta(2m-1)/m\}$.*

PROOF. Consider a job j and let i denote the machine to which it has been assigned. Let $\Pr[j \to i]$ be the probability for job j being assigned to machine i. Then, by Lemma 3.1 we know that

$$\begin{aligned}\mathbb{E}\left[C_j \,|\, j \to i\right] &\le (1 + \frac{\delta}{\alpha})r'_j + \sum_{k \in H(j)} \Pr\left[k \to i \,|\, j \to i\right] \cdot \mathbb{E}\left[P_k \,|\, j \to i\right] \\ &= (1 + \frac{\delta}{\alpha})r'_j + \sum_{k \in H(j)} \Pr\left[k \to i \,|\, j \to i\right] \cdot \mathbb{E}\left[P_k\right]\ .\end{aligned}$$

The probability that a job is assigned to a certain machine is equal for all machines, i.e., $Pr[j \to i] = 1/m$ for all $i = 1, \ldots, m$, and for any job j. Unconditioning the expected value of jth completion time yields

$$\begin{aligned}\mathbb{E}\left[C_j\right] &= \sum_{i=1}^{m} \Pr\left[j \to i\right] \cdot \mathbb{E}\left[C_j \,|\, j \to i\right] \\ &\le (1 + \frac{\delta}{\alpha})r'_j + \sum_{i=1}^{m} \Pr\left[j \to i\right] \cdot \sum_{k \in H(j)} \Pr\left[k \to i \,|\, j \to i\right] \cdot \mathbb{E}\left[P_k\right] \\ &= (1 + \frac{\delta}{\alpha})r'_j + \sum_{k \in H(j)} \frac{\mathbb{E}\left[P_k\right]}{m} \ +\ \frac{m-1}{m}\ \mathbb{E}\left[P_j\right]\ ,\end{aligned}$$

where the last equality is due to the independence of the job assignments to the machines. Then, the expected objective value of RANDASSIGN, $\mathbb{E}\left[Z^{\mathrm{RA}}\right]$, can be bounded by

$$\begin{aligned}\mathbb{E}\left[Z^{\mathrm{RA}}\right] &= \sum_j w_j \mathbb{E}\left[C_j\right] \\ &\le \left(1 + \frac{\delta}{\alpha}\right) \sum_j w_j r'_j + \sum_j w_j \sum_{k \in H(j)} \frac{\mathbb{E}\left[P_k\right]}{m} + \frac{m-1}{m} \sum_j w_j \mathbb{E}\left[P_j\right]\ .\end{aligned}$$

This bound equals the upper bound that we achieved on the expected performance of the MININCREASE policy in the proof of Theorem 4.3. Hence, we conclude the proof in the same way and with the same result as for MININCREASE. □

Acknowledgments. The authors thank Martin Skutella for pointing out that the MININCREASE algorithm can be interpreted as the derandomized version of an algorithm that randomly distributes jobs to machines. Thanks to Rolf H. Möhring for helpful discussions, and Andreas S. Schulz and Sven O. Krumke for pointing out some misinterpretations in an earlier version of this paper.

References

[1] F. N. Afrati, E. Bampis, C. Chekuri, D. R. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko, *Approximation schemes for minimizing average weighted completion time with release dates*, Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS), 1999, pp. 32–43.

[2] E. J. Anderson and C. N. Potts, *On-line scheduling of a single machine to minimize total weighted completion time*, Mathematics of Operations Research **29** (2004), 686–697.

[3] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.

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

[5] S. Chakrabarti and S. Muthukrishnan, *Resource scheduling for parallel database and scientific applications*, Proc. 8th Ann. ACM Symp. on Parallel Algorithms and Architectures (Padua, Italy), 1996, pp. 329–335.

[6] C. Chekuri, R. Johnson, R. Motwani, B. Natarajan, B. Rau, and M. Schlansker, *An analysis of profile-driven instruction level parallel scheduling with application to super blocks*, Proc. 29th IEEE/ACM Int. Symp. on Microarchitecture (Paris, France), 1996, pp. 58–69.

[7] M. A. H. Dempster, J. K. Lenstra, and A. H. G. Rinnooy Kan (editors), *Deterministic and stochastic scheduling*, D. Reidel Publishing Company, Dordrecht, 1982.

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

[9] A. Fiat and G. J. Woeginger, *On-line scheduling on a single machine: Minimizing the total completion time*, Acta Informatica **36** (1999), 287–293.

[10] 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** (1979), 287–326.

[11] W. J. Hall and J. A. Wellner, *Mean residual life*, Proc. Int. Symp. on Statistics and Related Topics (Ottawa, ON, Canada) (M. Csörgö, D. A. Dawson, J. N. K. Rao, and A. K. Md. E. Saleh, eds.), North-Holland, Amsterdam, The Netherlands, 1981, pp. 169–184.

[12] H. Hoogeveen and A. P. A. Vestjens, *Optimal on-line algorithms for single-machine scheduling*, Proceedings of the Fifth Conference on Integer Programming and Combinatorial Optimization IPCO (Berlin) (W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds.), Lecture Notes in Computer Science, vol. 1084, Springer, 1996, pp. 404–414.

[13] A. Karlin, M. Manasse, L. Rudolph, and D. Sleator, *Competitive snoopy paging*, Algorithmica **3** (1988), 70–119.

[14] T. Kawaguchi and S. Kyan, *Worst case bound of an LRF schedule for the mean weighted flow-time problem*, SIAM Journal on Computing **15** (1986), 1119–1129.

[15] E. Koutsoupias and C. H. Papadimitriou, *Beyond competitive analysis*, SIAM Journal on Computing **30** (2000), 300–317.

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

[17] X. Lu, R. Sitters, and L. Stougie, *A class of on-line scheduling algorithms to minimize total completion time*, Operations Research Letters **31** (2003), 232–236.

[18] N. Megow and A. S. Schulz, *On-line scheduling to minimize average completion time revisited*, Operations Research Letters **32(5)** (2004), 485–490.

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

[20] 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** (1999), 924–942.

[21] C. A. Phillips, C. Stein, and J. Wein, *Minimizing average completion time in the presence of release dates*, Mathematical Programming **82** (1998), 199–223.

[22] K. Pruhs, J. Sgall, and E. Torng, *Online scheduling*, Handbook of Scheduling: Algorithms, Models, and Performance Analysis (J. Leung, ed.), CRC Press, 2004.

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

[24] M. Scharbrodt, T. Schickinger, and A. Steger, *A new average case analysis for completion time scheduling*, Proc. 34th Ann. ACM Symp. on the Theory of Computing (Montréal, QB, Canada), 2002, pp. 170–178.

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

[26] M. Skutella and M. Uetz, *Stochastic machine scheduling with precedence constraints*, SIAM Journal on Computing (2005), to appear.

[27] M. Skutella and G.J. Woeginger, *A PTAS for minimizing the total weighted completion time on identical parallel machines*, Mathematics of Operations Research **25** (2000), 63–75.

[28] D. Sleator and R. Tarjan, *Amortized efficiency of list update and paging rules*, Communications of the ACM **28** (1985), 202–208.

[29] W. Smith, *Various optimizers for single-stage production*, Naval Research Logistics Quarterly **3** (1956), 59–66.

[30] A. Souza and A. Steger, *The expected competitive ratio for weighted completion time scheduling*, Proc. 21st Symp. on Theoretical Aspects of Computer Science (Montpellier, France), Lecture Notes in Computer Science, vol. 2996, Springer, 2004, pp. 620–631.

[31] M. Uetz, *Algorithms for deterministic and stochastic scheduling*, Cuvillier Verlag, Göttingen, Germany, 2002.

[32] A. P. A. Vestjens, *On-line machine scheduling*, Ph.D. thesis, Eindhoven University of Technology, Netherlands, 1997.

[33] G. Weiss, *Approximation results in parallel machines stochastic scheduling*, Annals of Operations Research **26** (1990), 195–242.

[34] _____, *Turnpike optimality of Smith's rule in parallel machines stochastic scheduling*, Mathematics of Operations Research **17** (1992), 255–270.

[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** (1980), 187–202.