Sequencing Stochastic Jobs with a Single Sample

Puck te Rietmole ✉ 🏠
Department of Mathematics, Utrecht University, Utrecht, The Netherlands

Marc Uetz ✉ 🏠 🆔
Mathematics of Operations Research, University of Twente, Enschede, The Netherlands

──── **Abstract** ────────────────────────────

This paper revisits the well known single machine scheduling problem to minimize total weighted completion times. The twist is that job sizes are stochastic from unknown distributions, and the scheduler has access to only a *single* sample from each of the distributions. For this restricted information regime, we analyze the simplest and probably only reasonable scheduling algorithm, namely to schedule by ordering the jobs by weight over sampled processing times. In general, this algorithm can be tricked by adversarial input distributions, performing in expectation arbitrarily worse even in comparison to choosing a random schedule. The paper suggests notions to capture the idea that this algorithm, on reasonable inputs, should exhibit a provably good expected performance. Specifically, we identify three natural classes of input distributions, such that for these classes, the algorithm performs better than random on any input.

1 Introduction, Motivation, and Model

A classical result in the literature on scheduling algorithms is due to Smith [18], showing how to minimize the total weighted completion time $\sum_j w_j C_j$ of n independent, non-preemptive jobs with weights w_j and processing times p_j on a single machine. They should be scheduled in the so-called WSPT order, that is, jobs must be ordered by non-increasing ratios w_j/p_j. The same is still true when the jobs' processing times are not known in advance, but instead governed by independent random variables P_j. Indeed, when minimizing the expected total weighted completion times, $\mathbb{E}[\sum_j w_j C_j] = \sum_j w_j \mathbb{E}[C_j]$, the same exchange argument shows that scheduling the jobs in order of non-increasing ratios $w_j/\mathbb{E}[P_j]$ is optimal in expectation [15]. Likewise, also for more general scheduling models, specifically on more than a single machine, some of the algorithmic framework that has been developed for computing approximately optimal solutions to deterministic scheduling problems, e.g. [6, 3], can be generalized to the more general setting with stochastic processing times, e.g. [12, 17, 10, 4, 5, 7]. However basically all of the previously cited approximation algorithms for stochastic scheduling problems, and also Rothkopf's result [15] need to assume that the expected processing times $\mathbb{E}[P_j]$ are known.

This paper asks the simple question what can be done if this is *not* the case. The paper therefore revisits the well understood single machine scheduling problem to minimize the total weighted completion times of jobs $\sum_j w_j C_j$. We ask how much of the optimality of Smith's result [18] can be recovered if the processing times p_j that the scheduler uses to determine the schedule, is in fact a *sample from an unknown distribution P_j*. In that setting, one should naturally minimize the expected total weighted completion times of jobs $\mathbb{E}[\sum_j w_j C_j]$. We can equivalently see the problem as a *stochastic* single machine scheduling problem, but with

the assumption that the scheduler has access to only a *single sample* of the processing time distributions.

This question relates to some recent work on prophet inequalities. There, it is known that a single sample per item (instead of knowing the whole distribution per item) is sufficient to get the same performance guarantee, namely 1/2, for a simple online threshold policy, relative to the benchmark (the prophet) that knows which item will turn out to have the maximum value. This is true for the classical prophet inequality setting where one item is chosen in an online fashion [16], and also for two items [13]. One major difference with the scheduling problem considered in this paper is that we do not have a natural notion of a prophet, which means that we have to come up with a reasonable benchmark to compare to. This is discussed in Sections 2 and 3.

The question can also be interpreted from the perspective of learning augmented algorithms. There, the goals is to find improved algorithms under the assumption that certain problem parameters can be learned by some hypothetical (machine) learning device. The better the learner, the better the resulting approximation guarantee, yet recovering a reasonable guarantee if the learner is bad. As to scheduling, examples of work in this direction are [8, 9, 2, 20]; see [1] for more references. Our information regime about learning stochastic job sizes is minimalistic, namely through one sample p'_j from P_j only, for all jobs j.

It is maybe no surprise that, without any further assumptions, one can define malicious input distributions that yield "wrong" samples with high probability, rendering an algorithm that follows these samples to have arbitrarily bad performance. Such an example is given below in Section 2. Our main result is the identification of sufficient conditions on the input distributions that rule out these undesirable effects. This is formalized using a notion of *relative optimality gap*, a scaled variation on the usual notion of approximation ratio, and in comparison to an adversary that chooses a schedule uniformly at random. We also argue that our positive results are tight in a mild sense, meaning that without any of these conditions, there is a corresponding counterexample.

2   Preliminaries: Single Machine Scheduling by Samples

We consider n jobs with independent processing time distributions P_j and weights w_j, $j = 1, \ldots, n$. The jobs have to be scheduled non-preemptively on a single machine with the goal to minimize $\sum_j w_j C_j$, where C_j denotes the completion time of job j. Let $p := (p_1, \ldots, p_n)$ denote possible realizations of $P := (P_1 \ldots P_n)$. We denote by $I = (w, P)$ an instance. If processing time realizations $p \sim P$ are known, the only optimal solutions are the sequences in so-called WSPT order, ratios w_j/p_j non-increasing [18].

If the processing times are stochastic, the solution is a stochastic scheduling policy that is non-anticipatory in the sense of [11], meaning that it cannot use information about actual realizations $p_j \sim P_j$ before scheduling a job j. If Π denotes such a scheduling policy, let $C_j^{\Pi}(p)$ be job j's completion time under policy Π for realization p, then the expected cost of policy Π on instance $I = (w, P)$ is

$$\text{cost}_I(\Pi) := \mathbb{E}_{p \sim P} \left[\sum_j w_j C_j^{\Pi}(p) \right] .$$

Also let WSPT(p) be the minimal cost for realization p, achieved by a WSPT order. Our goal is to minimize the *expected regret of* Π, which is minimizing

$$\mathbb{E}_{p \sim P} \left[\sum_j w_j C_j^{\Pi}(p) - \text{WSPT}(p) \right] = \text{cost}_I(\Pi) - \mathbb{E}_{p \sim P} \left[\text{WSPT}(p) \right] .$$

In other words, we effectively seek a policy Π minimizing $\text{cost}_I(\Pi)$. Since the processing times are independent across jobs, it is clear (for the single machine setting) that it suffices to consider so-called static list scheduling policies [14], meaning that jobs are scheduled in the order of a fixed priority list which is determined ex ante, using the given information about processing time distributions P_j. We will therefore simply refer to a scheduling policy as (scheduling) algorithm. If the distributions P_j are known, `WSEPT`, that is scheduling in order of non-increasing ratios $w_j/\mathbb{E}[P_j]$ is the policy that minimizes the expectation of the total weighted completion times [15].

For convenience let us write $\mathbb{E}[\,]$ instead of $\mathbb{E}_{p\sim P}[\,]$, if no ambiguity arises. This paper addresses the information regime where the random variables P_j are unknown, hence also $\mathbb{E}[P_j]$ is unknown. The only information available to the scheduler is one sample $p'_j \sim P_j$, for all jobs $j = 1, \ldots, n$. In this regime, the only reasonable algorithm that exploits the given information is -arguably- to take the sample p'_j as a proxy for $\mathbb{E}[P_j]$.

▶ **Definition 2.1.** *Algorithm* `SAM` *schedules jobs in non-increasing order weight over sampled processing time* w_j/p'_j.

Our goal is to analyze this algorithm's expected cost, resp. expected regret. The following simple example gives a malicious input instance showing that, in general, the expected cost as well as expected regret achieved by `SAM` can be unbounded.

▶ **Example 2.2.** Consider instance I with $n-1$ jobs with deterministic processing time $\varepsilon > 0$ and job n with processing time

$$P_n = \begin{cases} 0 & \text{with probability } 1 - \frac{1}{M}\,, \\ M^2 & \text{with probability } \frac{1}{M}\,, \end{cases}$$

where M is large. Let $w_j = 1$ for all jobs.

For the following discussion let n be fixed and consider $\varepsilon \to 0$. As $\mathbb{E}[P_n] = M$, the algorithm that minimizes $\mathbb{E}[\sum_j C_j]$ is to schedule all ε-jobs first, with expected cost M and expected regret equal to $(1-1/M)(n-1)\varepsilon \to 0$. As to `SAM`, observe that with probability $1 - 1/M$ the sample for job n will be $p'_n = 0$, which yields `SAM` to schedule job n before all ε-jobs. In this case, the expected cost of `SAM` is nM, while `WSPT`(p) has expected cost M. With probability only $1/M$, the sample $p'_n = M^2$ and `SAM` schedules job n last, and just like `WSPT`(p) achieves expected cost M. Therefore, `SAM`'s expected cost equals

$$\text{cost}_I(\texttt{SAM}) = \mathbb{E}_{p'\sim P}\mathbb{E}_{p\sim P}\left[\sum_j w_j C_j^{\texttt{SAM}(p')}(p)\right]$$

$$= \left(1 - \frac{1}{M}\right)nM + \frac{1}{M}M = nM - (n-1) \;=\; \Theta(nM)\,,$$

and the expected regret over the offline optimum `WSPT`(p) is of the same order of magnitude, as

$$\text{expected regret } \texttt{SAM} = \mathbb{E}_{p'\sim P}\mathbb{E}_{p\sim P}\left[\sum_j w_j C_j^{\texttt{SAM}(p')}(p) - \texttt{WSPT}(p)\right]$$

$$= \left(1 - \frac{1}{M}\right)[nM - M] \;=\; (n-1)(M-1) \;=\; \Theta(nM)\,.$$

Observe that the expectations are over the samples $p' \sim P$ to define `SAM`, and over the actual realizations of processing times $p \sim P$. If there is no ambiguity, we will omit the subscripts.

Example 2.2 also shows more, however. If we make the assumption that the adversary should be required to be non-anticipatory as in [12], then the omniscient adversary $\mathtt{WSPT}(p)$ is ruled out. The fundamental question is which is the right adversary to compare with, under the given information regime. In lack of reasonable alternatives, the least to ask for is an algorithm that compares favourably to an adversary that does *not* make use of the given processing time samples. In other words, we ask for an algorithm that is at least as good as the adversary $\mathtt{RND}$ that picks *any schedule uniformly at random.*

▶ **Definition 2.3.** *Algorithm* $\mathtt{RND}$ *schedules jobs in any of the n*! *sequences uniformly at random.*

We argue below that the choice of this adversary has several advantages. However, the bad news is that Example 2.2 is built so that algorithm $\mathtt{SAM}$ has large expected regret even in comparison to the much weaker adversary $\mathtt{RND}$, because

$$
\mathbb{E}\left[\sum_j w_j C_j^{\mathtt{SAM}(p')}(p) - \mathtt{RND}(p)\right] \;=\; \left(1 - \frac{1}{M}\right)\left[nM - \frac{n+1}{2}M\right] + \frac{1}{M}\left[M - \frac{n+1}{2}M\right]
$$
$$
= \frac{1}{2}(n-1)(M-2) \;=\; \Theta(nM)\,.
$$

To formalize the question that we aim to answer, let us define a partial order on the set of algorithms as follows.

▶ **Definition 2.4.** *If $\mathcal{I}$ is a set of problem instances, and Π and Π' are two scheduling algorithms, write $\Pi \leq_{\mathcal{I}} \Pi'$ if $\mathrm{cost}_I(\Pi) \leq \mathrm{cost}_I(\Pi')$ for every problem instance $I \in \mathcal{I}$.*

Example 2.2 then shows that not even $\mathtt{SAM} \leq_{\mathcal{I}} \mathtt{RND}$ holds, if $\mathcal{I}$ contains all problem instances. We mainly intend to give an answer to the following question about the potential benefit of having access to samples $p' \sim P$.

> Are there conditions on the input instances $\mathcal{I}$ under which $\mathtt{SAM} \leq_{\mathcal{I}} \mathtt{RND}$?

3   Uniform Randomization as Adversary

The adversary $\mathtt{RND}$ seems logical for Example 2.2, because all job weights w_j are the same. In general, one might define other adversaries that take into account the job weights. We argue that even in the presence of job weights the uniform random schedule $\mathtt{RND}$ is favourable.

The motivation lies in the fact that the expected cost achieved by $\mathtt{RND}$ is the average of the expected cost of the cheapest schedule and the expected cost of the most expensive schedule for any problem instance. In this relative sense, the expected cost of $\mathtt{RND}$ is independent of the considered problem instance. This will be made rigorous in Section 4. Comparing favourably against this instance independent benchmark shows that we have successfully leveraged the fact that we have access to samples.

Other, seemingly reasonable adversaries, e.g. the algorithm that schedules jobs by largest weight first, or randomized variations thereof, depend non-trivially on the problem instance. Since we do not have access to the P_j's, however, showing that $\mathtt{SAM}$ outperforms such an adversary would generally not allow to distinguish between $\mathtt{SAM}$ performing well, or the adversary performing poorly. The latter is ruled out when considering $\mathtt{RND}$ as a benchmark.

4   Relative Optimality Gap and Better Than Random Schedules

The partial order on algorithms $\leq_{\mathcal{I}}$ defined in Section 2 has the shortcoming that it does not give a quantitative measure of how well an algorithm performs. In this section, we therefore define a scaled version of the usual notion for approximation ratios.

For a given problem instance I, denote by L_I the lowest possible expected cost achieved by an optimal sequence, and by H_I the highest possible expected cost (achieved by the reverse sequence). For simplicity, let us exclude for the rest of the paper instances I where $L_I = H_I$, in which case trivially $\text{cost}_I(\texttt{SAM}) = \text{cost}_I(\texttt{RND}) = L_I = H_I$. One might consider to prove that $\texttt{SAM}$ (or any other algorithm) is an α-approximation algorithm for some $\alpha > 1$, meaning $\text{cost}_I(\texttt{SAM}) \leq \alpha L_I$, for all $I \in \mathcal{I}$ and given $\mathcal{I}$. However, depending on the problem instance it might be that $\alpha L_I \geq H_I$, trivializing such a statement. So, if we want to assess if using the processing time samples improves upon scheduling at random, showing that an algorithm is an α-approximation algorithm does not necessarily yield an answer for all instances. This motivates the following definition.

▶ **Definition 4.1.** *The relative optimality gap* (rog) *of an algorithm* Π *on a problem instance* $I \in \mathcal{I}$ *is defined as*

$$\text{rog}_I(\Pi) := \frac{\text{cost}_I(\Pi) - L_I}{H_I - L_I} \ .$$

For a set of problem instances $\mathcal{I}$, *define* $\text{rog}_{\mathcal{I}}(\Pi) := \sup_{I \in \mathcal{I}} \text{rog}_I(\Pi)$.

Note that $\text{rog}_{\mathcal{I}}(\Pi) \leq \alpha$, for some $\alpha \leq 1$, means that there is no instance I where $\Pi's$ expected cost exceeds that of an optimal solution by more than $\alpha(H_I - L_I)$. It corresponds to a β-approximation algorithm, with $\beta = \beta(I) = 1 + \alpha(H_I/L_I - 1)$. Roughly speaking, the rog is more meaningful than the notion of approximation ratios whenever $H_I - L_I$ is small, relative to L_I. (However this argument works vice versa, too.) One advantage of working with the rog is that it yields a meaningful statement for every single instance. Moreover, if $\text{rog}_{\mathcal{I}}(\Pi) \leq \alpha$, it implies that the expected regret of Π is at most an α-fraction of the maximum expected regret. In that sense, the rog can be also seen as a "regret version" of the usual notion of approximation ratios.

As long as the set of input instances $\mathcal{I}$ does include instances such as Example 2.2, we have that (again, considering $\varepsilon \to 0$)

$$\text{rog}_{\mathcal{I}}(\texttt{SAM}) \geq \frac{nM - (n-1) - M}{nM - M} \to 1 \ (\text{for } M \to \infty \text{ and any } n \geq 2) \ .$$

The following lemma defines the benchmark that we aim for in our subsequent analysis of algorithm $\texttt{SAM}$.

▶ **Lemma 4.2.** *For any instance I we have* $\text{rog}_I(\texttt{RND}) = 1/2$. *Moreover, for scheduling algorithm* Π *and any set of input instances* $\mathcal{I}$, *we have* $\Pi \leq_{\mathcal{I}} \texttt{RND} \Leftrightarrow \text{rog}_{\mathcal{I}}(\Pi) \leq \text{rog}_{\mathcal{I}}(\texttt{RND})$.

Proof. For given instance $I = (w, P)$ assume w.l.o.g. that $w_1/\mathbb{E}[P_1] \geq \cdots \geq w_n/\mathbb{E}[P_n]$. The first claim follows because $L_I = \sum_{j=1}^{n} \sum_{k=j}^{n} w_k \mathbb{E}[P_j]$, and any two jobs j, k have probability $1/2$ of being ordered either way by $\texttt{RND}$, so by canceling the terms $w_j \mathbb{E}[p_j]$

$$\text{cost}_I(\texttt{RND}) - L_I = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{1}{2} w_j \mathbb{E}(P_k) - \frac{1}{2} w_k \mathbb{E}(P_j) = \frac{1}{2}(H_I - L_I) \ .$$

The second claim then follows directly by using the definition of rog, since $\text{rog}_I(\Pi) \leq \frac{1}{2}$ precisely means that $\text{cost}_I(\Pi) \leq \frac{1}{2}(L_I + H_I) = \text{cost}_I(\texttt{RND})$.   ◀

That means that we may proceed to find conditions on $\mathcal{I}$ under which $\mathrm{rog}_{\mathcal{I}}(\mathtt{SAM}) \leq 1/2$. Note that the second statement in Lemma 4.2 is not true when comparing two arbitrary scheduling algorithms Π and Π', as $\mathrm{rog}_{\mathcal{I}}(\Pi) \leq \mathrm{rog}_{\mathcal{I}}(\Pi')$ does not imply that $\Pi \leq_{\mathcal{I}} \Pi'$.

For what follows it is convenient to realize that the relative optimality gap may be expressed in terms of the extra cost for scheduling pairs of jobs in the incorrect order. To that end, for a given instance $I = (w, P)$, we denote this extra cost by

$$\Delta_{jk} := w_j \mathbb{E}[\, P_k \,] - w_j \mathbb{E}[\, P_j \,],$$

and write $\mathbb{P}[\, \Pi : j \to k \,]$ for the probability that Π schedules job j before k.

▶ **Lemma 4.3.** *Let Π be a scheduling algorithm for instance $I = (w, P)$, and assume w.l.o.g. that $w_1/\mathbb{E}[\, P_1 \,] \geq \cdots \geq w_n/\mathbb{E}[\, P_n \,]$, then*

$$\mathrm{rog}_I(\Pi) = \sum_{j,k:j<k} \mathbb{P}[\, \Pi : k \to j \,] \frac{\Delta_{jk}}{\sum_{h,\ell:h<\ell} \Delta_{h\ell}} \,. \tag{1}$$

Proof. The proof follows from observing that $H_I - L_I = \sum_{h,\ell:h<\ell} \Delta_{h\ell}$, and since, again by canceling the terms $w_j \mathbb{E}[\, P_j \,]$,

$$\mathrm{cost}_I(\Pi) - L_I = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} (\mathbb{P}[\, \Pi : j \to k \,] - 1)\, w_k \mathbb{E}[\, P_j \,] + \mathbb{P}[\, \Pi : k \to j \,] w_j \mathbb{E}[\, P_k \,]$$

$$= \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \mathbb{P}[\, \Pi : k \to j \,] \Delta_{jk} \,.$$

◀

Now we can derive a simple and intuitive bound on the rog by bounding the probability of scheduling jobs in the incorrect order.

▶ **Theorem 4.4.** *If Π is an algorithm for instance I, if there exists some $\kappa \leq 1$, so that for all pairs of jobs $\mathbb{P}[\, \Pi : j \to k \,] \geq \kappa$ whenever $w_j/\mathbb{E}[\, p_j \,] > w_k/\mathbb{E}[\, p_k \,]$, then $\mathrm{rog}_I(\Pi) \leq 1 - \kappa$.*

Proof. The proof follows from Lemma 4.3 and $\mathbb{P}[\, \Pi : k \to j \,] \leq (1 - \kappa)$, and because $\Delta_{jk} = 0$ whenever $w_j/\mathbb{E}[\, p_j \,] = w_k/\mathbb{E}[\, p_k \,]$. ◀

One way of interpreting this result, in view of the second part of Lemma 4.2, is that $\mathrm{rog} \leq \alpha$ means that the expected regret is no worse than the expected regret of an algorithm that schedules each pair of jobs in the correct order with probability $1 - \alpha$. Also note that the inverse statement does not hold, namely, $\mathrm{rog} \leq \alpha$ does not necessarily imply that all pairs of jobs are scheduled in the correct order with probability at least $1 - \alpha$.

5 Well Behaved Input Distributions

Here we derive our main results, namely three classes of rather natural assumptions on input distributions that allow to make use of Theorem 4.4 to obtain performance guarantees for sampling based algorithm $\mathtt{SAM}$. We will argue in Section 6 that these results are tight in a mild sense. An assumption that we make henceforth is that the distributions for P_j all have a density f_j. The same ideas and results also work for discrete distributions, however.

If p'_j, p'_k are the sampled processing times, then $\mathtt{SAM}$ schedules j before k if $p'_j \leq \frac{w_j}{w_k} p'_k$. By independence of processing times we have the following.

▶ **Observation 5.1.** *Consider scheduling algorithm* SAM, *then the probability for scheduling two jobs j, k in order $j \to k$ is*

$$\mathbb{P}[\,\text{SAM} : j \to k\,] = \int_0^\infty f_k(y) \int_0^{\frac{w_j}{w_k}y} f_j(x)dxdy\,. \tag{2}$$

5.1 Symmetric Processing Time Distributions

Intuitively speaking, if all distributions P_j are symmetric around their means $\mathbb{E}[\,P_j\,]$, then for each sampled pair p'_j, p'_k that gives rise to an incorrect ordering of j and k, by symmetry, there exist samples p''_j, p''_k that appear with equal probability and give rise to the correct ordering. The proof of the following theorem confirms that this intuition is essentially correct. Let us first recall what symmetry of non-negative random variables means.

▶ **Definition 5.2.** *Consider a random variable P on $\mathbb{R}$ with density f and positive and finite expected value E, then P is symmetric if $f(E - x) = f(E + x)$ for all x.*

Note that in our context $P \geq 0$, which implies that $f(x) = 0$ for all $x \leq 0$ and all $x \geq 2E$.

▶ **Theorem 5.3.** *Consider instances $\mathcal{I}$ so that all processing time distributions P_j are symmetric for all jobs $j = 1, \ldots, n$, then* SAM $\leq_\mathcal{I}$ RND.

Proof. Pick an instance I with symmetric processing time distributions, and recall by Theorem 4.4 and Lemma 4.2 that it suffices to prove that for any pair of jobs j, k with $w_j/\mathbb{E}[\,P_j\,] > w_k/\mathbb{E}[\,p_k\,]$, we have $\mathbb{P}[\,\text{SAM} : j \to k\,] \geq \frac{1}{2}$, since then $\text{rog}_\mathcal{I}(\text{SAM}) \leq \frac{1}{2}$ by Theorem 4.4. For convenience write E_j and E_k for $\mathbb{E}[\,P_j\,]$ and $\mathbb{E}[\,P_k\,]$. Making use of (2), we apply linear variable substitutions. Substituting x by $x' = w_k(x - E_j)$, y by $y' = w_j(y - E_k)$, and noting that $dx' = w_k\,dx$ and $dy' = w_j\,dy$, we get

$$\mathbb{P}[\,\text{SAM} : j \to k\,] = \frac{1}{w_j w_k} \int_{-w_j E_k}^\infty f_k\left(E_k + \frac{y'}{w_j}\right) \int_{-w_k E_j}^{y' + w_j E_k - w_k E_j} f_j\left(E_j + \frac{x'}{w_k}\right)dx'dy'\,.$$

From symmetry of P_k it follows that $f_k(E_k + \frac{y'}{w_j}) = 0$ for $y' > w_j E_k$. Thus, we can replace the upper boundary of the outer integral by $w_j E_k$. Renaming x' back to x and y' to y, gives

$$\mathbb{P}[\,\text{SAM} : j \to k\,] = \frac{1}{w_j w_k} \int_{-w_j E_k}^{w_j E_k} f_k\left(E_k + \frac{y}{w_j}\right) \int_{-w_k E_j}^{y + w_j E_k - w_k E_j} f_j\left(E_j + \frac{x}{w_k}\right)dxdy\,.$$

And symmetrically also

$$\mathbb{P}[\,\text{SAM} : k \to j\,] = \frac{1}{w_j w_k} \int_{-w_k E_j}^{w_k E_j} f_j\left(E_j + \frac{x}{w_k}\right) \int_{-w_j E_k}^{x + w_k E_j - w_j E_k} f_k\left(E_k + \frac{y}{w_j}\right)dydx\,.$$

The final two expressions can be interpreted as surface integrals in $\mathbb{R}^2$. The integration domain in both cases is an isosceles right-angled triangle. These triangles are shown in Figure 1. We denote the triangle corresponding to $\mathbb{P}[\,\text{SAM} : j \to k\,]$ with T_{jk}, and analogously for T_{kj}. Due to the linear transformations, the symmetry of P_j and P_k translates into the fact that the integrals over these triangles remain the same after reflection through either of the coordinate axes. Mirroring the triangle T_{kj} twice, once in the x-axis, and once in the y-axis, aligns its right-angle corner with the right-angle corner of the triangle T_{jk}. This is illustrated in Figure 1. This geometric operation can also be done formally by changing the order of integration in the term for $\mathbb{P}[\,\text{SAM} : k \to j\,]$, using Fubini's theorem. It results in

$$\mathbb{P}[\,\text{SAM} : k \to j\,] = \frac{1}{w_j w_k} \int_{w_j E_k - 2w_k E_j}^{w_j E_k} f_k\left(E_k + \frac{y}{w_j}\right) \int_{-w_k E_j}^{y - w_j E_k + w_k E_j} f_j\left(E_j + \frac{x}{w_k}\right)dxdy\,.$$
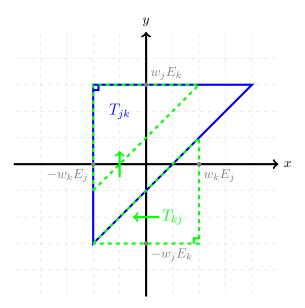
■ **Figure 1** Two triangles representing the integral domains for the probabilities of `SAM` scheduling in each pairwise order. The diagonal strip where the triangles do not overlap corresponds to the difference in the integral domains, proving that $\mathbb{P}[\,\mathtt{SAM}:j\to k\,]\geq\mathbb{P}[\,\mathtt{SAM}:k\to j\,]$.

Geometrically, the assumption that $w_jE_k>w_kE_j$ yields that triangle T_{kj} has shorter side-lengths than triangle T_{jk}. This means that the integration domains of the two otherwise identical integrals differ by the diagonal strip in Figure 1, and since the integrand is non-negative, $\mathbb{P}[\,\mathtt{SAM}:j\to k\,]\geq\mathbb{P}[\,\mathtt{SAM}:k\to j\,]$. Therefore, as required, $\mathbb{P}[\,\mathtt{SAM}:j\to k\,]\geq\frac{1}{2}$.   ◀

Note that the proof also shows that whenever the difference between the terms w_j/E_j and w_k/E_k gets large, also $\mathbb{P}[\,\mathtt{SAM}:j\to k\,]$ gets larger relative to $\mathbb{P}[\,\mathtt{SAM}:k\to j\,]$. However without knowledge about the precise processing time distributions, this does generally not yield qualitatively better bounds than rog $\leq 1/2$. We will again pick up this idea in Section 5.4.

5.2   Identical Processing Time Distributions up to Scaling

Here we show that if all distributions P_j are identical up to scaling, then algorithm `SAM` performs better than random.

▶ **Definition 5.4.** *Let $g(\cdot)$ be any probability density function over $\mathbb{R}_{\geq 0}$ with expectation 1. Define for $\lambda_j>0$, $f_j(x):=\lambda_jg(\lambda_jx)$. Call an instance of the scheduling problem* shape-uniform *if the processing time distributions P_j all have densities $f_j(\cdot)$, for the same $g(\cdot)$ and $\lambda_j>0$, $j=1,\ldots,n$.*

Note that $\mathbb{E}[\,P_j\,]=1/\lambda_j$, $j=1,\ldots,n$, for any shape-uniform instance. Also note that exponentially distributed processing times where $f_j(x)=\lambda_je^{-\lambda_jx}$ are contained as a special case of shape-uniformity via $g(x)=e^{-x}$.

▶ **Theorem 5.5.** *Consider instances $\mathcal{I}$ so that all processing time distributions P_j are shape-uniform with the same underlying probability density function $g(\cdot)$, then $\mathtt{SAM}\leq_\mathcal{I}\mathtt{RND}$.*

Proof. Consider any two jobs j,k and assume w.l.o.g. that $w_j/\mathbb{E}[\,P_j\,]\geq w_k/\mathbb{E}[\,P_k\,]$, which is equivalent to $w_j\lambda_j\geq w_k\lambda_k$. Again we make use of Theorem 4.4 and Lemma 4.2, and show

that $\mathbb{P}[\,\texttt{SAM} : j \rightarrow k\,] \geq \mathbb{P}[\,\texttt{SAM} : k \rightarrow j\,]$. Making use of (2), and substituting x by $\lambda_j x$ and y by $\lambda_k y$, we get

$$\mathbb{P}[\,\texttt{SAM} : j \rightarrow k\,] = \int_0^\infty g(y) \int_0^{\frac{w_j \lambda_j}{w_k \lambda_k} y} g(x) dx dy \,.$$

Symmetrically, we get the same term for $\mathbb{P}[\,\texttt{SAM} : k \rightarrow j\,]$, and subtracting gives

$$\mathbb{P}[\,\texttt{SAM} : j \rightarrow k\,] - \mathbb{P}[\,\texttt{SAM} : k \rightarrow j\,] = \int_0^\infty g(y) \int_{\frac{w_k \lambda_k}{w_j \lambda_j} y}^{\frac{w_j \lambda_j}{w_k \lambda_k} y} g(x) dx \, dy \,.$$

Now since $w_j \lambda_j \geq w_k \lambda_k$, and since the integrand is non-negative everywhere, this last term must be non-negative, which yields the claim. ◄

5.3 Translations of Identical Processing Time Distributions

Here we show that that $\texttt{SAM}$ performs better than random if all processing time distributions are identical up to having different expectations. This result is restricted to the special case of uniform weights $w_j = 1$, $j = 1, \ldots, n$, however.

▶ **Definition 5.6.** *Let $g(\cdot)$ be the density of some random variable with expectation $E > 0$, so that $g(x) = 0$ for all $x < 0$. Call the processing times* translation-identical *for $g(\cdot)$, if processing time P_j has density $f_j(x) = g(x - E_j + E)$, for $E_j \geq E$.*

Note that by definition, $\mathbb{E}[\,P_j\,] = E_j$ for $j = 1, \ldots, n$.

▶ **Theorem 5.7.** *Consider instances $\mathcal{I}$ with uniform weights $w_j = 1$ and translation-identical processing time distributions P_j, for some $g(\cdot)$, then $\texttt{SAM} \leq_\mathcal{I} \texttt{RND}$.*

Proof. A final time we use Theorem 4.4 and Lemma 4.2, and show that $\mathbb{P}[\,\texttt{SAM} : j \rightarrow k\,] \geq \mathbb{P}[\,\texttt{SAM} : k \rightarrow j\,]$ for any two jobs j, k with $\mathbb{E}[\,P_j\,] \leq \mathbb{E}[\,P_k\,]$. For the sake of the proof assume w.l.o.g. that $f_j(x) = g(x)$ and $f_k(x) = g(x - a)$, $a = E_k - E \geq 0$, are the density functions for P_j and P_k, so that $\mathbb{E}[\,P_k\,] = a + \mathbb{E}[\,P_j\,] = a + E$. By the same arguments as in the proof of Theorem 5.5, and using the substitution $y - a$ for y, we easily see that

$$\mathbb{P}[\,\texttt{SAM} : j \rightarrow k\,] - \mathbb{P}[\,\texttt{SAM} : k \rightarrow j\,] = \int_0^\infty g(y) \int_{y-a}^{y+a} g(x) dx \, dy \,.$$

As the integrand is non-negative, and as $a \geq 0$, the term is non-negative. ◄

The following example shows, maybe surprisingly, that the above result does not generalize to the setting with arbitrary weights. (The example uses finite discrete distributions for simplicity, but could be adapted accordingly.)

▶ **Example 5.8.** Consider an instance I with $n = 2$ jobs, weights $w_1 = 1$ and $w_2 = 2$, and processing time distributions

$$P_1 = \begin{cases} 1, & \text{with probability } 1 - \frac{1}{M} \,, \\ M^2 & \text{with probability } \frac{1}{M} \,, \end{cases}$$

and P_2 identically distributed as P_1, but independently and translated by $1 + \varepsilon$ so that $\mathbb{E}[\,P_2\,] = \mathbb{E}[\,P_1\,] + 1 + \varepsilon$.

Here, again assume that M is large and take limits for $\varepsilon \to 0$. Observe that Algorithm `SAM` schedules job 1 first whenever the sampled processing time $p'_1 = 1$, independent of the sample for P_2. This happens with probability $(1 - \frac{1}{M})$. Scheduling job 1 first has maximal expected cost which equals $5M^2$. The optimal solution is to schedule job 2 first, with expected cost $4M^2$. One readily verifies that

$$\mathrm{rog}_I(\mathtt{SAM}) = \frac{(1 - \frac{1}{M})5M^2 + \frac{1}{M}4M^2 - 4M^2}{5M^2 - 4M^2} = \frac{M^2 - M}{M^2} \to 1 \ (\text{for } M \to \infty) \,.$$

5.4   Exponential Distributions and α-separated Priorities

So far we showed that using the processing time samples is at least as good as random, arguably the least one would hope for. It is not hard to come up with example instances showing that these results cannot be substantially improved without making additional assumptions on the input. For example, to drive the rog to $1/2$ one can always add two jobs with large weights and almost identical input distributions, so that the gap between $\mathrm{cost}_I(\mathtt{SAM}) - L_I$ and $H_I - L_I$ gets arbitrarily close to $1/2$. Here we show how to derive a qualitatively better result, yet under rather strong assumptions on the input.

We consider the single machine scheduling problem with arbitrary weights w_j and exponentially distributed processing times, so P_j has a distribution with density $f_j(x) = \lambda_j e^{-\lambda_j x}$, $\lambda_j > 0$. Let us define $\pi_j := w_j \lambda_j = w_j / \mathbb{E}[P_j]$ for all $j = 1, \dots, n$, which can be seen as the priority of job j. We know from Section 5.2 that $\mathrm{rog}_{\mathcal{I}}(\mathtt{SAM}) \leq 1/2$ if $\mathcal{I}$ are the instances with exponentially distributed processing times. Under an additional assumption on the priorities this can be improved. Using (2) and the definition of the exponential distribution, elementary calculus yields the following. (The basic proof is in the appendix.)

▶ **Lemma 5.9.** *If processing times are exponentially distributed, then for any pair of jobs j, k*

$$\mathbb{P}[\mathtt{SAM} : j \to k] = \frac{\pi_j}{\pi_j + \pi_k} \,.$$

Then call the priorities $\pi_1, \dots, \pi_n$ of an instance with weights w_j and exponentially distributed processing times α-*separated*, whenever the following is true for all pairs or jobs j, k: Either $\pi_j = \pi_k$, or π_j and π_k are at least a factor α apart, that is, $\max\{\pi_j/\pi_k, \pi_k/\pi_j\} \geq \alpha$. In other words, we have groups of jobs with identical priorities in each group, and the priorities across groups are at least a factor α apart. The intuition is that either $\pi_j = \pi_k$, in which case the order of these two jobs should not matter, or π_j and π_k are far apart, in which case algorithm `SAM` should have a high(er) probability for scheduling these two jobs in the correct order, hence leading to a better rog bound. This intuition is indeed correct.

▶ **Theorem 5.10.** *Consider instances $\mathcal{I} = (w, P)$ with exponentially distributed processing times and α-separated priorities π_j, for $\alpha \geq 1$, then $\mathrm{rog}_{\mathcal{I}}(\mathtt{SAM}) \leq \frac{1}{1+\alpha}$.*

Proof. To be able to use Theorem 4.4 for an instance I, observe that it suffices to consider pairs of jobs j, k with $\pi_j \neq \pi_k$, because otherwise the contribution of this pair of jobs j, k to $\mathrm{rog}_I(\mathtt{SAM})$ is indeed zero in (1). So take any pair of jobs j, k with $\pi_j > \pi_k$, meaning that $w_j / \mathbb{E}[P_j] > w_k / \mathbb{E}[P_k]$ and the order $j \to k$ is optimal, then by Lemma 5.9, $\mathbb{P}[\mathtt{SAM} : j \to k] = \frac{\pi_j}{\pi_j + \pi_k} = \frac{1}{1 + \pi_k / \pi_j} \geq \frac{1}{1 + 1/\alpha} = \frac{\alpha}{\alpha + 1}$, where the inequality is true because $\pi_j \geq \alpha \pi_k$. The claim now follows by using Theorem 4.4. ◀

Admittedly, this result is still unsatisfactory in the sense that the condition of being α-separated is not natural, and moreover, it may even seem unnecessary: Whenever a pair of

jobs is α-separated, we are good (by the above proof), and whenever it is not, we should not care too much about the order of these two jobs. Indeed this intuition can be worked out, and it gives improvements over the general bound $1/2$, in the sense that one can get a convex combination of the two bounds $1/2$ and $1/(1+\alpha)$. To substantiate this to an improved bound $< 1/2$ for *all* exponential instances, however, requires additional knowledge on the input, and turns out to be both tedious and the improvement quite marginal [19]. The reason is that the rog may be non-continuous at $\Delta_{jk} = 0$ in (1).

6   Conclusions

The three classes of processing time distributions for which we show that sample-based scheduling is better than random scheduling are quite reasonable from a practical viewpoint. More interestingly, in light of Example 2.2, the results can be called tight in a mild sense: First note that Example 2.2 can clearly be tweaked from discrete to continuous distributions. The counterexample does its job because there are processing time distributions with (recall Example 2.2 had unit weights $w_j = 1$)
  (i) two different shapes, one of them asymmetric,
  (ii) two different means.
Moreover, as can be seen from Example 5.8, as soon as we have an instance with non-uniform weights and different means, not even the *same* distributions around different means allow for positive results, as long as these distributions are allowed to be asymmetric and are not shape-uniform. One may wonder about instances with identical expected processing times, yet different distributions. But this is either not interesting because all schedules yield the same expected cost (in case of unit weights), or it would allow to replicate an example exactly analogous to Example 2.2 by exchanging the roles of weights and expected processing times, which yields the same lower bound for instances with with arbitrary weights and identical expected processing times.

There are some open ends to take this work further. That includes the analysis for adversaries other than uniformly at random, the identification of other classes of distributions that would allow for positive results, and a less restrictive information regime with two or more samples, which would allow for algorithms that use a proxy for both expected processing time and variance. However to break Example 2.2, note that not even a polynomial number of samples is sufficient. Finally, one could explore if nontrivial notions for expressing the quality of the "learner", i.e., the samples, would allow to get improved performance bounds, in the spirit of learning augmented algorithms.

─── **References** ───

1 Algorithms with predictions. `https://algorithms-with-predictions.github.io/`. Accessed: 2023-08-02.
2 E. Bamas, A. Maggiori, L. Rohwedder, and O. Svensson. Learning augmented energy minimization via speed scaling. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33. NeurIPS, 2020.
3 C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. *SIAM Journal on Computing*, 31:146–166, 2001.
4 V. Gupta, B. Moseley, M. Uetz, and Q. Xie. Greed works—online algorithms for unrelated machine stochastic scheduling. *Mathematics of Operations Research*, 45(2):497–516, 2020.
5 V. Gupta, B. Moseley, M. Uetz, and Q. Xie. Corrigendum: Greed works—online algorithms for unrelated machine stochastic scheduling. *Mathematics of Operations Research*, 46(3):1230–1234, 2021.

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

7   Sven Jäger. An improved greedy algorithm for stochastic online scheduling on unrelated machines. *Discrete Optimization*, 47:100753, 2023.

8   R. Kumar, M. Purohit, and Z. Svitkina. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*, volume 31. NeurIPS, 2018.

9   S. Lattanzi, T. Lavastida, B. Moseley, and S. Vassilvitskii. Online scheduling via learned weights. In *Symposium on Discrete Algorithms (SODA 2020)*, page 1859–1877. ACM-SIAM, 2020.

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

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

12   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.

13   K. Pashkovich and A. Sayutina. Single sample prophet inequality for uniform matroids of rank 2, 2023. `arXiv:2306.17716`.

14   M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 5th edition, 2016.

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

16   A. Rubinstein, J.Z. Wang, and S.M. Weinberg. Optimal single-choice prophet inequalities from samples, 2019. `arXiv:1911.07945`.

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

18   W. E. Smith. Various optimizers for single-stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1956.

19   Puck te Rietmole. Sampling-based stochastic single-machine scheduling. Master's thesis, Utrecht University, 2023.

20   A. Wei and F. Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33. NeurIPS, 2020.

A   Proof Lemma 5.9

▶ **Lemma 5.9.** *If processing times are exponentially distributed, then for any pair of jobs j, k*

$$\mathbb{P}[\, \mathtt{SAM} : j \rightarrow k \,] = \frac{\pi_j}{\pi_j + \pi_k} \, .$$

Proof. Using (2) and the definition of the exponential distribution, we get that $\mathbb{P}[\, \mathtt{SAM} : j \rightarrow k \,]$ equals

$$\int_0^\infty e^{-y} \int_0^{\frac{\lambda_j w_j}{\lambda_k w_k} y} e^{-x} dx \, dy \; = \; \int_0^\infty e^{-y} \left[-e^{-x} \right]_{x=0}^{x=\frac{\lambda_j w_j}{\lambda_k w_k} y} dy$$

$$= \int_0^\infty e^{-y} \left[-e^{-\frac{\lambda_j w_j}{\lambda_k w_k} y} + 1 \right] dy \; = \; \left[\left(\frac{\lambda_j w_j}{\lambda_k w_k} + 1 \right)^{-1} e^{-\left(\frac{\lambda_j w_j}{\lambda_k w_k} + 1 \right) y} - e^{-y} \right]_{y=0}^{y=\infty}$$

$$= 1 - \left(\frac{\lambda_j w_j + \lambda_k w_k}{\lambda_k w_k} \right)^{-1} \; = \; \frac{\pi_j}{\pi_j + \pi_k} .$$

◀