MATHEMATICS OF OPERATIONS RESEARCH
Articles in Advance, pp. 1–14
ISSN 0364-765X (print) | ISSN 1526-5471 (online)

http://dx.doi.org/10.1287/moor.2014.0653

A Tight 2-Approximation for Preemptive Stochastic Scheduling

Nicole Megow
Institut für Mathematik, Technische Universität Berlin, 10623 Berlin, Germany,
nmegow@math.tu-berlin.de

Tjark Vredeveld
Department of Quantitative Economics, Maastricht University, 6200 MD, Maastricht, The Netherlands,
t.vredeveld@maastrichtuniversity.nl

We consider dynamic stochastic scheduling of preemptive jobs with processing times that follow independent discrete probability distributions. We derive a policy with a guaranteed performance ratio of 2 for the problem of minimizing the sum of weighted completion times on identical parallel machines subject to release dates. The analysis is tight. Our policy as well as their analysis applies also to the more general model of stochastic online scheduling.

In contrast to previous results for nonpreemptive stochastic scheduling, our preemptive policy yields an approximation guarantee that is independent of the processing time distributions. However, our policy extensively utilizes information on the distributions other than the first (and second) moments. We also introduce a new nontrivial lower bound on the expected value of an unknown optimal policy. It relies on a relaxation to the basic problem on a single machine without release dates, which is known to be solved optimally by the Gittins index priority rule. This dynamic priority index is crucial to the analysis and also inspires the design of our policy.

Keywords: scheduling; stochastic; approximation algorithm; total weighted completion time
MSC2000 subject classification: Primary: 90B36; secondary: 68W40
OR/MS subject classification: Primary: production/scheduling; secondary: approximation/heuristic
History: Received August 15, 2011; revised March 9, 2014. Published online in *Articles in Advance*.

1. Introduction.

Stochastic scheduling problems have attracted researchers for about four decades. A full range of articles is concerned with criteria that guarantee the optimality of simple policies for special scheduling problems; see, e.g., Pinedo [26]. It is only recently that research also focuses on approximations for less restrictive problem settings (Möhring et al. [24], Skutella and Uetz [37], Megow et al. [19], Schulz [30], Dean et al. [8]). All these results apply to nonpreemptive scheduling, and we are not aware of any approximation results when job preemption is allowed and processing times are stochastic.

In this paper, we give the first constant factor approximation for the stochastic version of the classical problem of scheduling jobs preemptively, with or without release dates, on identical parallel machines. The objective is to minimize the expected sum of weighted completion times.

We provide a very natural scheduling policy and show that it has an expected objective value of at most twice the expected optimal value. The analysis of our policy is tight. Both, the policy and its analysis are also valid in the model of stochastic online scheduling (Megow et al. [19], Chou et al. [5], Vredeveld [39]). They rely on the celebrated Gittins index priority rule (Gittins [9, 10]), which is an optimal policy for the setting when there is only one machine and there are no nontrivial release dates (Sevcik [33], Weiss [41]). Under deterministic input, our policy corresponds to the preemptive *weighted shortest processing time* (WSPT) rule, which is known to have a tight performance guarantee of 2 on identical parallel machines (Megow and Schulz [18]). We may interpret our stochastic policy as a generalization of the deterministic preemptive WSPT algorithm to stochastic input by incorporating the dynamic job prioritization according to the Gittins index.

Whereas previous approximation results for nonpreemptive scheduling have performance guarantees dependent on the coefficients of variation of the underlying probability distributions, our performance guarantee is independent of parameters of the processing time distributions. Moreover, it matches the guarantee of its deterministic counterpart. On the other hand, our policy requires the complete information about the probability distribution, whereas the policies for nonpreemptive stochastic scheduling only utilize information about the first and second moments.

1.1. Model and problem definition.

Let $J = \{1, 2, \ldots, n\}$ be a set of jobs that must be scheduled on m identical parallel machines. Each of the machines can process at most one job at a time, and any job can be processed by no more than one machine at a time. Each job j has an associated positive weight w_j and an individual release date $r_j \geq 0$, before which it is not available for processing. We allow preemption, which means that the processing of a job may be interrupted and resumed later, on the same or a different machine.

The stochastic component in the model we consider is the uncertainty about processing times. Any job j must be processed for P_j units of time, where P_j is a random variable. By $\mathbb{E}[P_j]$ we denote the expected value of

the processing time of job j, and by p_j a particular realization of P_j. We assume that all random variables of processing times are stochastically independent and follow finite, discrete probability distributions. With the latter restriction and a standard scaling argument, we may assume w.l.o.g. that P_j attains integral values in the set $\Omega_j = \{1, 2, \ldots, M_j\}$, and that all release dates are integral. The sample space of all processing times is denoted by $\Omega = \Omega_1 \times \cdots \times \Omega_n$.

The objective is to schedule the processing of all jobs so as to minimize the total weighted completion time of the jobs, $\sum_{j \in J} w_j C_j$, in expectation, where C_j denotes the completion time of job j. Adopting the well-known three-field classification scheme by Graham et al. [12], we denote the problem by $\mathrm{P} \mid r_j, pmtn \mid \mathbb{E}[\sum w_j C_j]$.

The solution of a stochastic scheduling problem is not a simple schedule, but a so-called *scheduling policy*. We follow the notion of scheduling policies as proposed by Möhring et al. [22, 23]. A scheduling policy makes scheduling decisions at certain *decision time points* t. Such a decision consists of the set of jobs to be scheduled at decision time t and a *tentative decision time point* t'. The next decision time will be the tentative decision time t' or the first job completion after the current decision time point t, whichever comes first. These decisions are based on information on the observed past up to time t, as well as a priori knowledge of the input data. The policy, however, must not anticipate information about the future, i.e., it must not use the actual realizations p_j of processing times of jobs that have not yet been completed by time t. Such a policy is called *nonanticipatory*. An optimal stochastic scheduling policy is a nonanticipatory policy that minimizes the total weighted completion time in expectation.

In this paper, we concentrate on approximation policies as defined by Möhring et al. [24].

Definition 1.1. A stochastic policy Π is a ρ-approximation, for some $\rho \geq 1$, if for all problem instances I,

$$\mathbb{E}[\Pi(I)] \leq \rho \mathbb{E}[\textsc{Opt}(I)],$$

where $\mathbb{E}[\Pi(I)]$ and $\mathbb{E}[\textsc{Opt}(I)]$ denote the expected values that the policy Π and an optimal nonanticipatory policy, respectively, achieve on a given instance I. The value ρ is called the performance guarantee of policy Π.

The policies we consider belong to the more general class of policies for the *stochastic online scheduling* model. Here, a policy learns about the existence and the characteristics of a job j only at its individual release date r_j. This means that an online policy must not anticipate the arrival of a job at any time earlier than its release date. At this point in time, the job with the probability distribution of its processing time and its deterministic weight are revealed. Thus, stochastic online policies are required to be online and nonanticipatory. We refer to Megow et al. [19] for a more detailed discussion on stochastic online policies. As suggested in that paper, we use in this model a generalized definition of approximation guarantees from the stochastic scheduling setting by comparing the expected outcome of a nonanticipatory *online* policy with the expected outcome of an optimal nonanticipatory offline policy.

1.2. Previous work. The classical deterministic variant of our scheduling problem that seeks to minimize the weighted sum of completion times is well known to be NP-hard (Labetoulle et al. [15], Lenstra et al. [16]). This is true even on a single processor or if all release dates are equal. Polynomial time approximation schemes have been presented by Afrati et al. [1].

Stochastic scheduling has been under consideration for more than 40 years. We refer the reader to Pinedo's book (Pinedo [26]) for an overview. Some of the first results on *preemptive stochastic scheduling* that can be found in the literature are by Chazan et al. [3] and Konheim [14]. They formulated sufficient and necessary conditions for a policy to optimally solve the single-machine problem where all jobs become available at the same time. Later Sevcik [33] developed an intuitive method for creating optimal schedules (in expectation). He introduces a priority policy that relies on a dynamic index that can be computed for each job based on the properties of the job, but independently of other jobs.

Gittins [9] showed that this priority index is a special case of his Gittins index (Gittins [9, 10]). Later in 1995, Weiss [41] formulated Sevcik's priority index again in terms of the Gittins index, and named it a *Gittins index priority policy* (Gipp). He also provided a different proof of the optimality of this priority policy, based on the work conservation invariance principle. Weiss covers a more general problem than the one considered here and in Chazan et al. [3], Konheim [14], and Sevcik [33]: The holding costs (weights) of a job are not deterministic constants, but may vary during the processing of a job. At each state, these holding costs are random variables.

For more general scheduling problems with release dates and/or multiple machines, no optimal policies are known. Instead, the literature reflects a variety of research on restricted problems such as those with special probability distributions for processing times or special job weights. Pinedo [25] considered the single-machine problem in the presence of release dates, but restricted the processing times to be drawn from exponential

distributions. He showed the optimality of the preemptive variant of the *weighted shortest expected processing time* (WSEPT) policy, which is a policy that processes at any time the job with the highest ratio $w_j/\mathbb{E}[P_j]$ among all jobs that have been released and not yet completed. If all jobs become available at the same time, then preemption is not even necessary. This is also true for more general processing time distributions: Rothkopf [28] showed that for increasing hazard rate (failure rate) distributions, no finite number of preemptions can outperform a nonpreemptive policy.

For scheduling jobs with equal weights and equal release dates, an optimal policy is known for quite a large class of processing time distributions. Weber [40] showed that for processing times with monotone hazard rates, the dynamic *shortest expected remaining processing time* (SERPT) policy is optimal; this policy always gives highest priority to jobs with minimum expected remaining processing times. This policy heavily utilizes the option of preempting jobs when the hazard rate is decreasing. On the other hand, it reduces to the nonpreemptive *shortest expected processing time* (SEPT) policy when the hazard rate is increasing. The optimality of this static policy, SEPT, has been shown earlier for exponentially distributed processing times by Glazebrook [11], Weiss and Pinedo [42], and Bruno et al. [2]. In the case when processing times are not drawn from a distribution with monotone hazard rates, Coffman et al. [6] have shown that this policy is not optimal, even if all processing times follow the same two-point distribution and even if we deal with only two processors. On the other hand, for such a special distribution, they showed that the SEPT policy is asymptotically optimal and has a turnpike property: Asymptotically, for large n, most of the optimal decisions will be made according to this policy. In case of general probability distributions and an arbitrary number of machines, Weiss [41] showed that the Gittins index priority policy is asymptotically turnpike optimal and has an expected value that is only an additive constant away from the optimal value.

None of the results on multiple machines consider individual job weights. Under strong restrictions on weights and exponential processing times, Kämpke [13] proves the optimality of the WSEPT policy; in this setting, weights need to be *agreeable*, which means that for any two jobs, i and j, $w_i/\mathbb{E}[P_i] < w_j/\mathbb{E}[P_j]$ implies $w_i \le w_j$.

Although optimal policies have been found only for very special problem settings, research has focused lately on obtaining approximation algorithms. Such investigations have been successful in the nonpreemptive setting. Möhring et al. [24] derived the first constant-factor approximations for the nonpreemptive problem with and without release dates. Their results are based on a lower bound on the expected optimum value that is derived from a linear programming (LP) relaxation. The performance guarantees they prove are functions of a parameter that bounds the squared coefficient of variation of processing times. For specific types of probability distributions of the processing times, stochastic policies with improved performance ratios were given in Megow et al. [19] and Schulz [30]. In fact, these subsequent results also apply to the more general stochastic scheduling model with online job arrivals. Skutella and Uetz [37] complemented the first approximation results by policies for scheduling with precedence constraints. In general, all known performance guarantees for nonpreemptive policies depend on the distribution of processing times. This is also true for recent results for the online version of the stochastic scheduling model obtained by Megow et al. [19], Chou et al. [5], and Schulz [30]. All obtained results that include asymptotic optimality (Chou et al. [5]) and approximation guarantees for deterministic as well as randomized policies (Megow et al. [19], Schulz [30]) address nonpreemptive scheduling.

Several scheduling algorithms were designed and analyzed for deterministic online scheduling problems. For a general survey of online scheduling models and results, we refer the reader to Sgall [34] and Pruhs et al. [27]. We consider online scheduling when jobs become known at their release date. In the context of preemptive scheduling, Sitters [35] gave a 1.56-competitive algorithm for the single-machine problem in this model. This is the best result currently known. It improved upon an earlier result by Schulz and Skutella [31], who generalized the classical *Smith rule* (Smith [38]) to the problem of scheduling jobs with individual release dates, achieving a competitive ratio of 2. This algorithm has been generalized further to the multiple-machine problem without loss of performance in Megow and Schulz [18]. Sitters [36] presented a deterministic online algorithm with a performance guarantee of $1.791(1+1/\sqrt{m})^2$, which is less than 2 for $m \ge 311$ and drops to 1.791 if m tends to infinity. Combining it with the randomized $(2-1/m)$-competitive algorithm by Correa and Wagner [7], Sitters obtains an improved randomized online algorithm for parallel machines. However, for the single-machine problem, Schulz and Skutella [32] provided a randomized 4/3-competitive algorithm.

1.3. Our contribution. We present a constant factor approximation for preemptive stochastic scheduling. For jobs with arbitrary finite discrete processing time distributions and individual release dates, we give a 2-approximation for multiple machines. This performance guarantee distinguishes from previously known results for nonpreemptive variants of this problem as it is constant and independent of the probability distribution of processing times. Our policy, as well as its analysis, applies also to the more general model of stochastic online

scheduling. It can be seen as a stochastic generalization of a deterministic online algorithm analyzed in Megow and Schulz [18]. While all previous generalizations of deterministic algorithms for the nonpreemptive setting resulted in a loss in performance depending on the parameters of the probability distributions (Möhring et al. [24], Schulz [30], Skutella and Uetz [37], Megow et al. [19]), our policy has the same performance guarantee as the deterministic algorithm from Megow and Schulz [18].

The policy we study, is motivated by GIPP, an optimal policy for the single-machine problem without release dates (Konheim [14], Sevcik [33], Weiss [41]). Our policy, named FOLLOW-GIPP (F-GIPP), is a parallel-machine policy that follows GIPP in a somewhat lazy way. As we explain later in more detail, F-GIPP updates the dynamic GIPP-index only at certain time points. This allows us to give a tight analysis that yields the performance guarantee 2.

The Gittins index not only inspires the design of our scheduling policy, but it is also crucial for bounding the optimal value. We derive a closed-form lower bound on the expected objective value of an unknown optimal policy for the preemptive stochastic scheduling problem. First, we give a closed-form expression of the expected value that GIPP achieves on a single machine without release dates. Then, we employ a stochastic variant of a *fast single-machine relaxation*, which was originally introduced for deterministic scheduling by Chekuri et al. [4]. Since GIPP is an optimal policy for a relaxed version of our fast single-machine relaxation, we can give a closed-form expression for a lower bound on the expected value of an optimal policy for the original parallel-machine problem. The closed-form expression of our lower bound is not imperative for the analysis of our algorithms, but we believe that it is of independent interest and might be useful in other applications.

In general, our policy is not optimal. However, under restricted problem settings, it coincides with policies whose optimality is known. If processing times are exponentially distributed and release dates are absent, F-GIPP coincides with the preemptive WSEPT rule. As mentioned above, this classical policy is optimal if all weights are equal (Glazebrook [11], Weiss and Pinedo [42], Bruno et al. [2]) or, more generally, if they are agreeable (Kämpke [13]). If there is only a single machine available and jobs have arbitrary release dates, then F-GIPP coincides with preemptive WSEPT and is optimal (Pinedo [25]). If there are no release dates, then F-GIPP solves the weighted single-machine problem optimally for arbitrary processing time distributions because, in that case, it coincides with the optimal policy GIPP (Konheim [14], Sevcik [33], Weiss [41]). Finally, we discuss the behavior of F-GIPP under deterministic input on a single machine. In the case of arbitrary job weights without release dates, F-GIPP coincides with the optimal *weighted shortest processing time* rule, also known as *Smith's rule* (Smith [38]). This folkloric algorithm processes at any time the unfinished job with the highest ratio w_j/p_j. For arbitrary release dates, its preemptive variant has an approximation guarantee of 2, which is tight (Schulz and Skutella [31]). In a way, F-GIPP can be seen as a generalization of the deterministic algorithms WSPT.

1.4. Organization of the paper. In §2, we define the Gittins index priority policy (GIPP) and discuss useful properties of the index function. This allows us to reinterpret GIPP and to derive a closed-form expression for the expected objective value it obtains. For scheduling problems with nontrivial release dates and/or multiple machines, optimal policies and the corresponding expected objective values are unknown. Therefore, we use lower bounds on the optimal value to compare the expected outcome of a policy with the expected outcome $\mathbb{E}[\text{OPT}]$ of an unknown optimal policy OPT. The trivial bound $\mathbb{E}[\text{OPT}] \geq \sum_{j\in J} w_j(r_j + \mathbb{E}[P_j])$ does not suffice proving constant approximation guarantees as it may diverge from an optimal solution by a factor $\Omega(n)$. (Consider, e.g., instances in which all jobs are released at the same time.)

In §3, we use the concept of a fast single machine to obtain such a lower bound on the optimal solution value for the parallel-machine problem based on the optimal single-machine policy GIPP. In §4, we introduce a simple parallel-machine policy, F-GIPP, with an approximation factor of exactly 2 for arbitrary discrete processing time distributions.

Finally, we discuss in §5 a very similar alternative scheduling policy, the feasibility of our techniques for continuous processing time distributions, and a more general stochastic online scheduling model.

2. The Gittins index priority policy.

In this section, we describe the Gittins index priority policy (GIPP) and derive a closed-form expression for the expected total weighted completion time of this optimal single-machine policy when there are no nontrivial release dates.

Given that a job j has been processed for y time units and it has not yet been completed, we define the *expected investment* of processing this job for q time units or up to completion, whichever comes first, as

$$I_j(q, y) = \mathbb{E}[\min\{P_j - y, q\} \mid P_j > y].$$

The ratio of the weighted probability that this job is completed within the next q time units over the expected investment, is the basis of the Gittins index priority rule. We define it as the *rank* of a part of a job, called *sub-job*, of length q of job j, after it has completed y units of processing:

$$R_j(q, y) = \frac{w_j \Pr[P_j - y \le q \mid P_j > y]}{I_j(q, y)}.$$

This ratio is well defined if we assume that we compute the rank only for $q > 0$ and $P_j > y$, in which case the investment $I_j(q, y)$ has a value greater than zero.

For a given (unfinished) job j and attained processing time y, we are interested in the maximal rank it can achieve. We call this the Gittins index, or rank, of job j, after it has been processed for y time units:

$$R_j(y) = \max_{q \in \mathbb{R}^+} R_j(q, y).$$

The length of the sub-job achieving the maximal rank is denoted as

$$q_j(y) = \max\{q \in \mathbb{R}^+ \colon R_j(q, y) = R_j(y)\}.$$

With the definitions above, we define the Gittins index priority policy for minimizing the expected total weighted completion time on a single machine.

Algorithm 1 (Gittins index priority policy (Gipp))
At any time t, process an unfinished job j with the currently highest rank $R_j(y_j(t))$, where $y_j(t)$ denotes the amount of processing that has been done on job j by time t. Break ties by choosing the job with the smallest job index.

Theorem 2.1 (Konheim [14], Sevcik [33], Weiss [41]). *The Gittins index priority policy* (Gipp) *optimally solves the stochastic scheduling problem* $1 \mid pmtn \mid \mathbb{E}[\sum w_j C_j]$ *on a single machine without job release dates.*

The following properties of the Gittins indices and the lengths of sub-jobs achieving the Gittins index are well known; see Gittins [10] and Weiss [41]. In parts, they have been derived earlier in the scheduling context by Konheim [14] and Sevcik [33]. They prove useful to analyze Gipp as well as the more general policy that we present in this paper.

Proposition 2.1 (Gittins [10], Weiss [41]). *Consider a job j that has been processed for y time units. Then, for any $0 < \zeta < q_j(y)$ it holds that*

$$R_j(y) \le R_j(y + \zeta), \tag{a}$$

$$q_j(y + \zeta) \le q_j(y) - \zeta, \tag{b}$$

$$R_j(y) + q_j(y)) \le R_j(y). \tag{c}$$

A *quantum* of job j is defined as the sub-job of length $q_j(y)$ that causes the maximal rank $R_j(y)$. We now split a job j into a set of n_j quanta, denoted by tuples (j, i), for $i = 1, \ldots, n_j$. The processing time y_{ji} that a job j has attained up to a quantum (j, i), and the length of each quantum, q_{ji}, are recursively defined as $y_{j1} = 0$, $q_{ji} = q_j(y_{ji})$, and $y_{j,i+1} = y_{j,i} + q_{ji}$. By Proposition 2.1(a), we know that, while processing a quantum, the rank of the job does not decrease, whereas Proposition 2.1(c) and the definition of $q_j(y)$ tell us that the rank is strictly lower at the beginning of the next quantum. Hence, once a quantum has been started, Gipp will process it for its complete length or up to the completion of the job, whichever comes first; that means, Gipp preempts a job only at the end of a quantum. Obviously, the Gipp policy processes job quanta nonpreemptively in nonincreasing order of their ranks. In particular, Gipp does not need to recompute the maximum rank of a running job until the completion of the current quantum. Thus, we may rephrase Gipp in the following way.

Algorithm 2 ((Reformulated) Gittins index priority policy (Gipp))
For each job, recursively compute the partition into quanta of maximal rank. Schedule job quanta of unfinished jobs in nonincreasing order of their rank.

Before proceeding with the structural analysis of Gipp, we briefly discuss the behavior of the rank function of a (sub-)job and more implications of the properties above. Figure 1 illustrates the maximum rank of two jobs with particular processing time distributions (three-point and exponential distribution) as a function of the amount of time that the job has been processing.

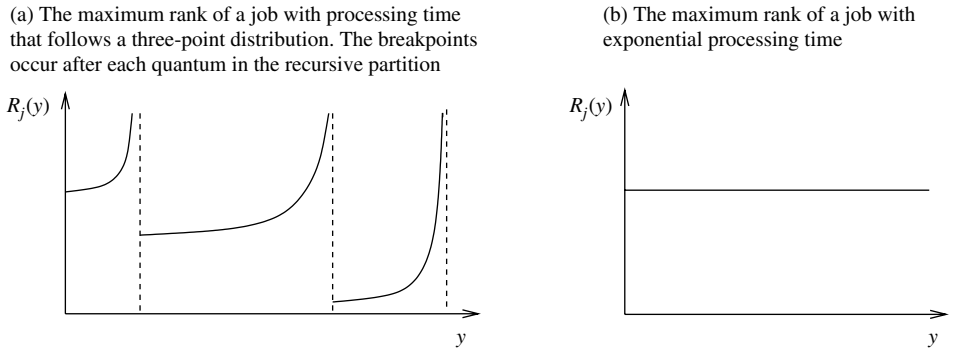

FIGURE 1. The maximum rank of jobs with certain processing time distributions depending on the amount of time, y, that the job has been processed.

The general assumption of stochastic job processing times subsumes deterministic processing times as a special case. Consider an incomplete job j with deterministic processing time p_j, of which y units already elapsed. The rank and the quantum lengths are deterministically predetermined by their definition:

$$R_j(q, y) = \frac{w_j \Pr[P_j - y \le q \mid P_j > y]}{I_j(q, y)} = \begin{cases} 0, & \text{iff } q < p_j - y; \\ w_j/(p_j - y), & \text{otherwise.} \end{cases}$$

The behavior of the rank function for deterministic job processing times is illustrated in Figure 2. The quantum length is infinite, which does not harm the policy since it processes only unfinished jobs. Note that for deterministic processing times, GIPP coincides with the WSPT rule, which is known to be optimal in the deterministic single-machine setting (Smith [38]).

For the analysis of our policy in §4, Proposition 2.1(b) is of particular importance. It bounds the length of a new quantum that causes maximum rank if a previous quantum got preempted. Suppose, at some time t, a quantum of length q that maximizes the job rank R begins processing. Now, consider some time $t' < t + q$. GIPP does not recompute the rank and the quantum until the completion of q, but in a more complex problem setting where jobs arrive at their individual release dates this might become essential. At time t', the new maximum job rank R' is, by Proposition 2.1(a), at least as large as R and, as Proposition 2.1(b) states, the new quantum that causes the new rank R' has length q', which is not greater than the remaining part of quantum q, that is, $q' \le q - (t' - t)$.

Turning back to the GIPP policy, recall that it runs job quanta in nonincreasing order of rank. We assume that quanta $(j, 1), (j, 2), \ldots (j, n_j)$ are naturally indexed in order of occurrence. Now, we define the set $H(j, i)$ of all quanta that are processed no later than quantum (j, i) in the GIPP order, assuming that the jobs have not already finished. Let Q be the set of all quanta, that is, $Q = \{(k, l) \mid k = 1, \ldots, n, l = 1, \ldots, n_k\}$, then

$$H(j, i) = \{(k, l) \in Q \mid R_k(y_{kl}) > R_j(y_{ji})\} \cup \{(k, l) \in Q \mid R_k(y_{kl}) = R_j(y_{ji}) \wedge k \le j\}.$$

Since the Gittins index of a job is decreasing with every finished quantum (Proposition 2.1(c)), we know that $H(j, h) \subseteq H(j, i)$, for $h \le i$. To uniquely relate higher priority quanta to exactly one quantum of a job, we introduce the notation $H'(j, i) = H(j, i) \setminus H(j, i-1)$, where we define $H(j, 0) = \varnothing$. Note that the quantum (j, i) is also

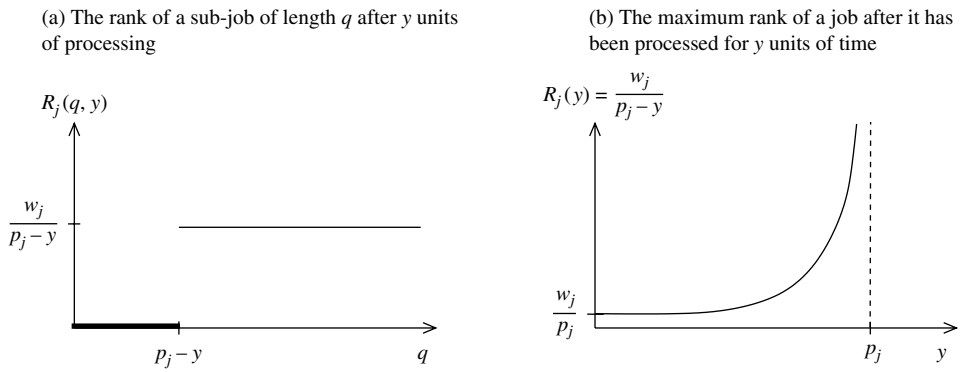

FIGURE 2. Rank functions in the special case of *deterministic processing times*.

contained in the set of its higher priority quanta $H'(j,i)$. In the same manner, we define the set of lower priority quanta as $L(j,i) = Q \setminus H(j,i)$.

With these definitions and the observations above, we can give a closed formula for the expected objective value of Gipp when scheduling an instance I_0 in which all jobs are released at time 0.

Lemma 2.1. *The optimal policy* Gipp *achieves for an instance* I_0 *of* $1 \mid pmtn \mid \mathbb{E}[\sum w_j C_j]$ *an expected objective value of*

$$\mathbb{E}[\text{Gipp}(I_0)] = \sum_{j \in J} w_j \sum_{i=1}^{n_j} \sum_{(k,l) \in H'(j,i)} \Pr[P_j > y_{ji} \wedge P_k > y_{kl}] \cdot I_k(q_{kl}, y_{kl}).$$

Proof. Consider a realization of processing times $p \in \Omega$ and a job j. Let i_p be the index of the quantum in which job j finishes, that is, $y_{ji_p} < p_j \le y_{ji_p} + q_{ji_p}$. Gipp processes quanta of jobs that have not completed nonpreemptively in nonincreasing order of their ranks. Hence,

$$C_j(p) = \sum_{(k,l) \in H(j,i_p):\, p_k > y_{kl}} \min\{q_{kl}, p_k - y_{kl}\}. \tag{1}$$

For an event $\mathscr{E}$, let $\chi(\mathscr{E})$ be an indicator random variable that equals 1 if and only if the event $\mathscr{E}$ occurs. The expected value of $\chi(\mathscr{E})$ equals the probability that the event $\mathscr{E}$ occurs, that is, $\mathbb{E}[\chi(\mathscr{E})] = \Pr[\mathscr{E}]$. Additionally, we denote by ξ_{kl} the special indicator random variable for the event $P_k > y_{kl}$.

We take expectations on both sides of Equation (1) over all realizations. This yields

$$\begin{aligned}
\mathbb{E}[C_j] &= \mathbb{E}\left[\sum_{h:\, y_{jh} < P_j \le y_{j,h+1}} \sum_{\substack{(k,l) \in H(j,h):\\ P_k > y_{kl}}} \min\{q_{kl}, P_k - y_{kl}\} \right] \\
&= \mathbb{E}\left[\sum_{h=1}^{n_j} \chi(y_{jh} < P_j \le y_{j,h+1}) \sum_{(k,l) \in H(j,h)} \xi_{kl} \cdot \min\{q_{kl}, P_k - y_{kl}\} \right] \\
&= \mathbb{E}\left[\sum_{h=1}^{n_j} \chi(y_{jh} < P_j \le y_{j,h+1}) \sum_{i=1}^{h} \sum_{(k,l) \in H'(j,i)} \xi_{kl} \cdot \min\{q_{kl}, P_k - y_{kl}\} \right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n_j} \sum_{h=i}^{n_j} \chi(y_{jh} < P_j \le y_{j,h+1}) \sum_{(k,l) \in H'(j,i)} \xi_{kl} \cdot \min\{q_{kl}, P_k - y_{kl}\} \right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n_j} \chi(y_{ji} < P_j) \sum_{(k,l) \in H'(j,i)} \xi_{kl} \cdot \min\{q_{kl}, P_k - y_{kl}\} \right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n_j} \xi_{ji} \sum_{(k,l) \in H'(j,i)} \xi_{kl} \cdot \min\{q_{kl}, P_k - y_{kl}\} \right].
\end{aligned} \tag{2}$$

The equalities follow from an index rearrangement and the fact that, by definition, $H(j,h) = \bigcup_{i=1}^{h} H'(j,i)$ for any $h = 1, 2, \ldots, n_j$ and that n_j is an upper bound on the actual number of quanta of job j.

For jobs $k \neq j$, the processing times P_j and P_k are independent random variables and, thus, the same holds for their indicator random variables ξ_{ji} and ξ_{kl} for any i, l. Using linearity of expectation, we rewrite (2) as

$$\begin{aligned}
&= \sum_{i=1}^{n_j} \sum_{(k,l) \in H'(j,i)} \mathbb{E}\left[\xi_{ji} \cdot \xi_{kl} \cdot \min\{q_{kl}, P_k - y_{kl}\}\right] \\
&= \sum_{i=1}^{n_j} \sum_{(k,l) \in H'(j,i)} \sum_{x} x \cdot \Pr\left[\xi_{ji} = \xi_{kl} = 1 \wedge \min\{q_{kl}, P_k - y_{kl}\} = x\right] \\
&= \sum_{i=1}^{n_j} \sum_{(k,l) \in H'(j,i)} \sum_{x} x \cdot \Pr\left[\xi_{ji} = \xi_{kl} = 1\right] \cdot \Pr\left[\min\{q_{kl}, P_k - y_{kl}\} = x \mid \xi_{kl} = 1\right] \\
&= \sum_{i=1}^{n_j} \sum_{(k,l) \in H'(j,i)} \Pr\left[P_j > y_{ji} \wedge P_k > y_{kl}\right] \cdot \mathbb{E}\left[\min\{q_{kl}, P_k - y_{kl}\} \mid P_k > y_{kl}\right] \\
&= \sum_{i=1}^{n_j} \sum_{(k,l) \in H'(j,i)} \Pr[P_j > y_{ji} \wedge P_k > y_{kl}] \cdot I_k(q_{kl}, y_{kl}),
\end{aligned}$$

where the third equality follows from conditional probability and the fact that either $j \neq k$, and thus ξ_{ji} and ξ_{kl} are independent, or $(j,i) = (k,l)$, and thus the variables ξ_{ji} and ξ_{kl} are the same. Weighted summation over all jobs concludes the proof.

2.1. A note on the running time complexity of Gipp. The time complexity of Gipp is essentially determined by the running time for computing the rank of a job for a given attained processing time and the corresponding quantum length. Expressing this running time as a function of the input requires the specification of the input encoding, in particular the encoding of the probability distributions of processing times.

Assume for each job j a finite discrete processing time distribution with a finite number of realizations M_j. Proposition 2.1 implies a partition of jobs into relevant quanta of maximum rank. For discrete distributions, the length of the ith quantum ($i = 1, 2, \ldots, M_j$) is the difference between the $(i-1)$st and ith processing time realization. Hence, we can compute all Gipp-relevant ranks of j and the corresponding quanta in time $O(M_j)$. This partition into quanta and rank computation can be done for each job individually and in advance—although, in an actual implementation one would avoid that. Now, Gipp as described in Algorithm 2, sorts these job quanta by their ranks, which takes $O(M \log M)$, where $M = \sum_j M_j$. Finally, Gipp runs through this sorted list of job quanta and processes each quantum of unfinished jobs, which takes linear time. Thus, the overall time complexity for arbitrary finite discrete distributions is $O(M \log M)$.

The most general type of such probability distributions is given explicitly by a list of possible realizations with corresponding probabilities. In that case the encoding length is $\Theta(M)$ and therefore Gipp runs in polynomial time in the input size.

Many (discrete) probability distributions can be encoded in a much more concise manner and Gipp may not run in polynomial time. Nevertheless, there are again special cases with special structures that allow to compute the rank of a job and the corresponding quantum in polynomial time. For example, if the processing times are uniformly distributed over the integer values in a certain range, then the length of the first quantum is equal to the maximum possible realization.

3. A stochastic fast single-machine relaxation. In this section, we derive a lower bound for preemptive stochastic scheduling on parallel machines. We utilize the knowledge about Gipp's optimality for the single-machine problem without release dates; see Theorem 2.1. To that end, we show first that the *fast single-machine relaxation* as introduced by Chekuri et al. [4] for the deterministic (online) scheduling environment applies in the preemptive stochastic setting as well. Note that the same is not true for nonpreemptive stochastic scheduling. This is in contrast to the deterministic scheduling environment, where the fast single-machine relaxation applies to both the preemptive and the nonpreemptive setting.

Let I denote a scheduling instance of the parallel-machine scheduling problem $\mathrm{P} \mid r_j, pmtn \mid \mathbb{E}[\sum w_j C_j]$, and let I' be the same instance to be scheduled on a single machine—called fast single machine—of speed m times the speed of the machines used for scheduling instance I. Let $\textsc{Opt}_1$ denote an optimal single-machine policy that yields an expected value $\mathbb{E}[\textsc{Opt}_1(I')]$ on instance I'.

Lemma 3.1. *The expected value of any parallel-machine policy* Π *applied to the parallel-machine scheduling instance* I *is bounded from below by the expected value of an optimal policy* $\textsc{Opt}_1$ *on instance* I' *on a fast single machine, that is,*

$$\mathbb{E}[\Pi(I)] \geq \mathbb{E}[\textsc{Opt}_1(I')].$$

Proof. Given a parallel-machine policy Π, we design a policy Π' for a fast single machine, that processes the same set of jobs as Π between any two consecutive decision points. In particular, we guarantee that at any such decision point, Π' has observed the same partial schedule as Π such that it can imitate Π's decisions. A crucial fact to do so is that for discrete processing time distributions, we can determine at any moment in time t the earliest possible completion time of the jobs that are scheduled by Π at time t.

Beginning at $t = 0$, we determine at any decision point t for Π' the next tentative decision point t' as the minimum of this earliest possible completion time and the first tentative decision point of Π after time t. Note that during (t, t') no job can complete in Π and each machine processes at most one job during this interval. Then in Π', we partition the interval $(t, t']$ into m equal-length intervals and in the ith interval we process the job that was processed on machine i by Π. This way, we ensure that Π' processes the same volume of the same set of jobs during this interval. Moreover, if in the schedule constructed by policy Π a job completes during this interval, then this happens at time t'. In Π' this job will then be completed at or before time t', depending on which machine this job was processed. As a job scheduled on the fast single machine according to Π' finishes not later than this job when scheduled on the parallel machines according to Π, we have that $\mathbb{E}[\Pi'(I')] \leq \mathbb{E}[\Pi(I)]$ for any

instance I. Then the lemma follows since an optimal policy OPT_1 yields on the single machine an expected objective value $\mathbb{E}[\text{OPT}_1(I')] \le \mathbb{E}[\Pi'(I')]$. □

Note that for processing times that are drawn from continuous probability distributions, the construction of a single-machine policy Π' from a given parallel-machine policy Π as described in the proof above fails. At some decision point t of Π' it may not be possible to determine the next tentative decision point $t' > t$ since the earliest possible completion time of the jobs that are scheduled by Π at time t may be arbitrarily close to t. In the appendix we show how we can cope with this issue at the cost of loosing an (arbitrarily small) constant factor on the lower bound.

We may relax the fast-single machine instance I' further by setting all release dates to 0. The optimal single-machine policy for the resulting instance is GIPP (Theorem 2.1). With the closed formula for the expected cost of GIPP (Lemma 2.1) and the above Lemma 3.1, we conclude a lower bound on the expected optimal value for the parallel-machine instance I with release dates. In fact, we directly get a relation between the expected value of $\text{GIPP}(I_0)$ and $\text{OPT}(I)$, where $\text{GIPP}(I_0)$ is the expected value of the single-machine policy GIPP (without speed modification) applied to instance I_0, which is obtained from I by relaxing all release dates to 0.

THEOREM 3.1. *The expected value of an optimal policy* OPT *for the parallel-machine problem I is bounded by*

$$\mathbb{E}[\text{OPT}(I)] \ge \frac{1}{m}\sum_{j\in J} w_j \sum_{i=1}^{n_j} \sum_{(k,l)\in H'(j,i)} \Pr\big[P_j > y_{ji} \wedge P_k > y_{kl}\big] \cdot I_k(q_{kl}, y_{kl}) = \frac{\mathbb{E}[\text{GIPP}(I_0)]}{m},$$

where I_0 is the same instance as I without release dates to be scheduled on a single machine (*of the same speed as machines in I*).

4. A 2-approximative policy for parallel machines. Simple examples show that GIPP is not an optimal policy for scheduling problems with release dates and/or multiple machines. The following policy, F-GIPP, is a coarse generalization of GIPP to the parallel-machine problem with nontrivial release dates, $\text{P} \mid r_j, pmtn \mid \mathbb{E}[\sum w_j C_j]$. We call a job *available* at time t, if it is released and has not been completed by t.

Algorithm 3 (Follow Gittins index priority policy (F-GIPP))
At any time t, process m available jobs j with highest rank $R_j(y_{j,k+1})$, where (j,k) is the last quantum of j that has been completed. If there are less than m jobs available, process all jobs. Define $k = 0$ if no quantum of job j has been completed.

Note that the decision time points in this policy are release dates and any time point when a quantum or a job is completed. In contrast to the original Gittins index priority policy, F-GIPP considers only the rank $R_j(y_{ji} = \sum_{k=1}^{i-1} q_{jk})$ that a job had before processing quanta (j,i) even if (j,i) has been processed for some time less than q_{ji}. Informally speaking, F-GIPP updates the ranks only after quantum completions and then follows GIPP.

THEOREM 4.1. F-GIPP *is a 2-approximation for the preemptive stochastic scheduling problem* $\text{P} \mid r_j, pmtn \mid \text{E}[\sum w_j C_j]$ *with nontrivial release dates on parallel machines.*

PROOF. For a given instance I, fix a realization $p \in \Omega$ of processing times and consider a job j and its completion time $C_j^{\text{F-GIPP}}(p)$. Job j is processing in the time interval $[r_j, C_j^{\text{F-GIPP}}(p)]$. We split this interval into two disjunctive sets of subintervals, $T(j,p)$ and $\bar{T}(j,p)$. Let $T(j,p)$ denote the set of subintervals in which job j is processing and $\bar{T}(j,p)$ contains the remaining subintervals. Denoting the total length of all intervals in a set T by $|T|$, we have

$$C_j^{\text{F-GIPP}}(p) = r_j + |T(j,p)| + |\bar{T}(j,p)|.$$

The total length of intervals in $T(j,p)$ is, by definition, p_j. In intervals of the set $\bar{T}(j,p)$, no machine is idle and F-GIPP schedules only quanta with a higher priority than (j, i_p), the final quantum of job j. Thus, $|\bar{T}(j,p)|$ is maximized if all these quanta are scheduled between r_j and $C_j^{\text{F-GIPP}}(p)$. This gives an upper bound on the overall length $|\bar{T}(j,p)|$, which is the sum of all realized quantum lengths on m machines. That yields

$$C_j^{\text{F-GIPP}}(p) \le r_j + p_j + \frac{1}{m} \sum_{\substack{(k,l)\in H(j,i_p):\\ p_k > y_{kl}}} \min\{q_{kl}, p_k - y_{kl}\} = r_j + p_j + \frac{1}{m} C_j^{\text{GIPP}(I_0)},$$

where $C_j^{\text{GIPP}(I_0)}$ denotes the completion time of job j when scheduled by GIPP on a single machine neglecting release dates. Weighted summation over all jobs and taking expectations on both sides gives

$$\sum_{j\in J} w_j \mathbb{E}\big[C_j^{\text{F-GIPP}}\big] \le \sum_{j\in J} w_j (r_j + \mathbb{E}[P_j]) + \frac{\mathbb{E}[\text{GIPP}(I_0)]}{m}.$$

Finally, we apply the trivial lower bound $\mathbb{E}[\textsc{Opt}] \geq \sum_{j\in J} w_j(r_j + \mathbb{E}[P_j])$ and Theorem 3.1, and the approximation result follows. □

For general input instances the approximation factor of 2 is best possible for F-Gipp. This follows directly from a deterministic worst-case instance in Megow and Schulz [18] since F-Gipp coincides for deterministic instances with a parallel-machine generalization of the Wspt rule considered in that paper.

Theorem 4.2 (Megow and Schulz [18]). *The approximation ratio of* F-Gipp *is not better than* 2 *for the problem* $\mathrm{P} \mid r_j, pmtn \mid \mathbb{E}[\sum w_j C_j]$, *for any given number of machines.*

5. Further remarks. In this paper, we presented a natural extension of the policy Gipp, which is an optimal policy for the single-machine scheduling problem with no nontrivial release dates, to the parallel-machine problem with nontrivial release dates. The policy F-Gipp that we presented has an approximation ratio of 2, which is tight even in the case that there is only a single machine and the processing times are deterministically known (Schulz and Skutella [31]).

An alternative preemptive policy. An alternative, even more straightforward generalization of Gipp is as follows: instead of recomputing the rank of a job only after a quantum has completed, as F-Gipp does, we consider at any time the actual rank of a job. We denote this policy as Generalized-Gipp (Gen-Gipp). It deviates much less from the original Gittins index priority rule, and thus it uses more information on the actual state of the set of known, unfinished jobs. The analysis of Gen-Gipp is much more involved, but yields the same guarantee of 2 for the single machine setting (Megow [17], Megow and Vredeveld [21]). However, we were not able to prove tightness and we conjecture that the true approximation ratio is much lower. As one supporting argument we note that with deterministic input and equal weights for all jobs, Gen-Gipp yields an optimal solution, since it obtains the same schedule as Schrage's optimal rule (Schrage [29]), which is not true for F-Gipp. A lower bound on Gen-Gipp's approximation guarantee is 1.215, which is again true even for deterministic input (Xiong and Chung [43]). Gen-Gipp can be applied to parallel machines using a random job-to-machine assignment; it then yields the same approximation guarantee of 2 in expectation.

Stochastic online scheduling. The policy presented in this paper is valid in an online setting in which jobs are not known in advance but arrive online over time, the so-called stochastic online scheduling model. F-Gipp employs Gipp in an online way: at any moment in time it bases its decisions on the Gittins index or rank of each job, which is a dynamic value that depends on the probability distribution of the job's processing time and information about the current status of the job in the schedule. Thus, F-Gipp is not only nonanticipatory, which is enforced by the stochastic scheduling model, but also online. At no point in time do any of the policies use information about jobs that will be released in the future.

The performance of a policy in the stochastic online scheduling model is assessed by comparing its expected value with an optimal offline policy in the traditional stochastic scheduling model. Therefore, the Gittins index-based lower bound on an expected optimal value in §3 still holds in this generalized model. Thus, Theorem 4.1 directly implies that F-Gipp yields an approximation ratio of 2 for the online version of the stochastic scheduling problem $\mathrm{P} \mid r_j, pmtn \mid \mathbb{E}[\sum w_j C_j]$.

Continuous distributions. We presented results assuming discrete probability distributions for processing times. The main reason is that our policy relies on the optimality of Gipp which is proven in Sevcik [33] under this assumption. Moreover, the alternative proof in Weiss [41] considers a slightly modified model in which jobs may be preempted only after a certain part of a job has been completed. For continuous probability distributions, we can easily show that $\textsc{Gipp} \leq \textsc{Opt} + \epsilon$, for any $\epsilon > 0$. Furthermore, our fast single-machine relaxation (Lemma 3.1) can be adopted to give a bound on the expected optimum value for arbitrary continuous distributions being only slightly weaker than its discrete equivalent in Theorem 3.1. Combining both, we conclude that F-Gipp yields a $(2+\epsilon)$-approximation for scheduling instances with job processing times that are drawn from continuous distributions. For the sake of completeness, we give the corresponding proofs in the appendix.

Acknowledgments. The authors thank the anonymous referees and the associate editor for helpful suggestions and motivation to discuss the time complexity of Gipp and the situation of nondiscrete probability distributions. An extended abstract with parts of this work appeared in the *Proceedings of the 14th Annual European Symposium on Algorithms* (Megow and Vredeveld [20]). The research of the first authors was partially supported by the DFG (Deutsche Forschungsgemeinschaft) Research Center Matheon *Mathematics for key technologies* in Berlin and the DFG Emmy Noether Programme (ME 3825/1).

Appendix A. Continuous probability distributions for processing times. We consider processing times that follow independent finite continuous probability distributions.

DEFINITION A.1. A scheduling policy Π is called a *nonretrospective policy* if at any decision moment t it makes its decisions based only on the knowledge it has about the jobs that have not yet been completed by time t. Thus, it uses neither the realizations of the processing times of already completed jobs nor the time t.

LEMMA A.1. *There exists an optimal policy to minimize the expected total weighted completion time on a single machine that is nonretrospective.*

PROOF. We first show that the decision that an optimal policy makes, given a certain partial schedule, does not depend on the actual time instant at which it is taken. Hereto, consider an instance I, an optimal policy Π_0 that can start scheduling all jobs in I from time 0 onward (denoted as the *time-0 case*) and an optimal policy Π_t for the case that the jobs in I can only be started at or after time $t > 0$ (denoted as the *t-shifted case*).

At time t, a policy for the t-shifted case has exactly the same information as a policy for the case in which jobs can start from time 0 on. Thus, we can construct a second policy for the t-shifted case, Π'_t, that follows policy Π_0. That means, at time $t+s$, Π'_t will process the same job as Π_0 does at time s. For the time-0 case, we can define in a similar way the policy Π'_0 that follows policy Π_t: at time s, Π'_0 processes the same job as policy Π_t does at time $t+s$. Then, the expected total weighted completion times of these policies are

$$\mathbb{E}[\Pi'_t(I)] = \sum_j w_j \mathbb{E}[C_j^{\Pi'_t}] = \sum_j w_j \mathbb{E}[C_j^{\Pi_0} + t] = \mathbb{E}[\Pi_0(I)] + \sum_j w_j t$$

$$\mathbb{E}[\Pi'_0(I)] = \sum_j w_j \mathbb{E}[C_j^{\Pi'_0}] = \sum_j w_j \mathbb{E}[C_j^{\Pi_t} - t] = \mathbb{E}[\Pi_t(I)] - \sum_j w_j t.$$

As Π_t is an optimal policy for the t-shifted case and Π_0 is an optimal policy for the time-0 case, we know that

$$\mathbb{E}[\Pi]_t(I) \le \mathbb{E}[\Pi'_t(I)] = \mathbb{E}[\Pi_0(I)] + \sum_j w_j t \le \mathbb{E}[\Pi'_0(I)] + \sum_j w_j t = \mathbb{E}[\Pi_t(I)].$$

Hence, both inequalities need to hold with equality and policy Π'_t that makes exactly the same decisions as policy Π_0, optimal for the time-0 case, is also optimal for the t-shifted case.

Secondly, we argue that the decisions made at a decision point t cannot be based on the processing time realizations of jobs that have been finished by time t. This follows directly from the fact that the processing time distributions are independent. Hence, by finishing a job we do not gain any information about other jobs.

As the decision made at a certain decision time by an optimal policy does not depend on the processing time realizations of jobs that have been completed by this time, nor does it depend on the time at which the decision is made, an optimal policy bases its decisions only on the information it has on the jobs that are still unfinished. Thus an optimal policy is nonretrospective. □

In the following lemma, we show that GIPP approximates an optimal policy arbitrarily well for instances with continuous probability distributions for processing times. Hereto, given a nonretrospective policy Π, an instance I_0 and $\epsilon > 0$, we construct a discretized instance I_Π, that has the same set of jobs, but the processing time distributions are discretized. To do so, we determine for each job j, for each possible realization of the processing times and for each tentative decision point of Π at which job j has not been completed, the amount of processing job j receives up to the next tentative decision point assuming that j does not complete before that. This way, we obtain distinct values $x_{j,1} < x_{j,2} < \cdots < x_{j,M_j}$ (some of which may occur multiple times) of possible attained processing at tentative decision points. To these values, we add the values y_{ji} of attained processing time up to quantum (j,i) as defined in §2. Moreover, whenever $x_{j,i+1} - x_{j,i} > \delta$, with $\delta = \epsilon/(n\sum_j w_j)$, we add some more values between $x_{j,i}$ and $x_{j,i+1}$ so that the maximum distance between two consecutive values is at most δ. Abusing notation, we denote these values still by $x_{j,1} < x_{j,2} < \cdots < x_{j,M_j}$. Note that, although the number of possible realizations can be unbounded because of the fact that the processing time distributions are continuous, we can still determine all values $x_{j,i}$ because the policy Π is nonretrospective: if a job j completes at or before a tentative decision point, the next decision made by the nonretrospective policy Π is independent of the realization of the processing time for j. Hence, all realizations for the processing time of job j within this particular interval can be considered as one realization for the determination of the $x_{j,i}$ values. The instance I_Π has the following probability distributions for processing times P'_j:

$$\Pr[P'_j = x_{j,i}] = \Pr[x_{j,i-1} < P_j \le x_{j,i}], \quad \text{for } i = 1, \ldots, M_j, \text{ and } x_{j,0} = 0.$$

LEMMA A.2. *Let I_0 denote an instance for the stochastic scheduling problem of minimizing the expected total weighted completion time on a single machine, and assume there are job processing times that follow finite continuous probability distributions. Then*

$$\text{GIPP}(I_0) \le \text{OPT}_1(I_0) + \epsilon,$$

for any $\epsilon > 0$, where OPT_1 is an optimal single-machine policy for instance I_0.

Proof. We convert any nonretrospective policy Π on instance I_0 into a policy Π' for instance I_Π. Policy Π' makes exactly the same decisions as policy Π. Because of the discretization of I_Π, we know that if a certain job j is completed by Π at or before a tentative decision point, its counterpart in I_Π will also be completed at or before the corresponding tentative decision point by Π'. Moreover, as $x_{j,i+1} - x_{j,i} \le \delta$, we also know that at each completion, the time for policy Π' is shifted by at most δ time units. Since the policy Π is nonretrospective, we know that at the completion time of the job, Π will base its decisions only on the set of jobs that is still available, that is, the same set of jobs that is available for Π', having the same attained processing time. Hence, Π' has indeed the same information as Π had at time of the job completion, and thus, Π' can imitate the decision of Π. There is a delay in this action by at most δ per completing job. Therefore, on every realization of processing times for I, we know that policy Π' on the corresponding realization of processing times of I_Π, finds completion times that are at most $n\delta$ larger. Hence, we have that

$$\mathbb{E}[\Pi'(I_\Pi)] = \sum_j w_j \mathbb{E}[C_j^{\Pi'}] \le \sum w_j(\mathbb{E}[C_j^{\Pi}] + n\delta) = \mathbb{E}[\Pi(I_0)] + \sum_j w_j n\delta.$$

By choosing, $\delta = \epsilon/(n\sum_j w_j)$, $\Pi(I) = \textsc{Opt}_1(I_0)$ and the fact that I_Π only has discrete probability distributions, for which Gipp is an optimal policy, we know that

$$\textsc{Gipp}(I) \le \textsc{Gipp}(I_\Pi) \le \mathbb{E}[\Pi'(I_\Pi)] \le \mathbb{E}[\textsc{Opt}(I_0)] + \epsilon.$$

To see the first inequality, note that because of the properties of the rank given in Proposition 2.1 and the fact that the values y_{ji} of attained processing time up to quantum (j,i) are possible realizations in I_Π, as long as no job completes $\textsc{Gipp}(I_\Pi)$ makes the same decisions as $\textsc{Gipp}(I)$ and we have that $\textsc{Gipp}(I) \le \textsc{Gipp}(I_\Pi)$. □

In the following lemma, we show that the lower bound on the value of any policy given in Lemma 3.1 holds with an additive ϵ, for any $\epsilon > 0$.

Lemma A.3. *For any $\epsilon > 0$, we have that the expected value of any parallel-machine policy Π applied to the parallel-machine scheduling instance I is bounded from below by the expected value of an optimal policy $\textsc{Opt}_1$ on instance I' on a fast single machine, that is*

$$\mathbb{E}[\Pi(I)] \ge \mathbb{E}[\textsc{Opt}_1(I')] - \epsilon,$$

where I' is the same instance as I, but to be scheduled on a single machine with speed m times the speed of the machines used for scheduling instance I.

Proof. In the proof of Lemma 3.1, we defined a policy Π' that schedules jobs on a fast single machine, that mimics a given policy Π. To ensure that Π' completes each job not later than the policy Π, we used the fact that at any time t we can determine the earliest possible completion time of any job that is scheduled at time t. However, this is not true anymore, when some of the processing times have continuous probability distributions. To overcome this issue, we set the time between two consecutive decision points in Π' to be at most δ, for some δ depending on ϵ. The time intervals between two consecutive tentative decision times in Π' are mapped one-to-one to intervals in the schedule produced by Π of at most the same length. Such a time interval in Π' is partitioned into m equal lengths subintervals and the ith subinterval is dedicated to the processing of the job that is processed on machine i during the corresponding interval of policy Π. If no job completes during a time interval of Π', then the corresponding interval of Π has the same length and Π' processes the same set of jobs as Π such that at the end of the interval these jobs have received the same amount of processing as in Π. On the other hand, if there is a job that completes in an interval between two consecutive tentative decision times, then the corresponding interval of Π will be shorter: it will be equal to the time between the start of the interval and the earliest job completion thereafter. During this interval, Π' processes the same set of jobs as Π processes during the corresponding interval. However, because of the fact that Π' may only learn at the end of its interval about the job completion, e.g., when the job that will finish during the interval is scheduled on machine m in the schedule produced by Π, at the end of the interval Π' may have given the jobs more processing than Π did at the end of its corresponding interval. Therefore, the end of the interval in Π' may be shifted by at most δ time units compared to the end of the corresponding interval in Π. Note that Π' still can follow what Π does, because Π' knows the exact time of the earliest job completion and can determine how much processing each of the other jobs would have attained until this time (as Π' may have processed some jobs a bit longer, but not shorter). As these shifts can only occur when there is a job completion, a time interval between two consecutive tentative decision times in Π' can only be at most $n\delta$ away from the corresponding time interval in the schedule produced by Π. Using this fact, and the fact that a job in Π may be completed at the beginning of an interval whereas in Π' it can be completed almost at the end of the corresponding interval, the completion time of a job j in Π' cannot be more than $(n+1)\delta$ away from its completion time in the execution of Π.

From the discussion above, we know that

$$\mathbb{E}[C_j^{\Pi'}(I')] \le \mathbb{E}[C_j^{\Pi}(I)] + (n+1)\delta.$$

By setting $\delta = \epsilon/((n+1)\sum_j w_j)$, we have

$$\mathbb{E}[\textsc{Opt}_1(I')] \le \mathbb{E}[\Pi'(I')] = \sum_j w_j \mathbb{E}[C_j^{\Pi'}(I')] \le \sum_j w_j \mathbb{E}[C_j^{\Pi}(I)] + \sum_j w_j(n+1)\delta = \mathbb{E}[\Pi(I)] + \epsilon. \quad \square$$

Using the same argumentation as in §3 for the discrete setting and applying Lemmas A.2 and A.3, we obtain the following lower bound on the expected optimal value for continuous distributions.

THEOREM A.1. *The expected value of an optimal policy* OPT *for the parallel-machine problem I with finite continuous processing time distributions is bounded by*

$$\mathbb{E}[\text{OPT}(I)] \geq \frac{\mathbb{E}[\text{GIPP}(I_0)]}{m} - 2 \cdot \epsilon,$$

where I_0 is the same instance as I without release dates to be scheduled on a single machine (of the same speed as machines in I).

From the proof of Theorem 4.1, we know that the value of F-GIPP is bounded by

$$\text{F-GIPP}(I) \leq \sum_{j \in J} w_j (r_j + \mathbb{E}[P_j]) + \frac{\mathbb{E}[\text{GIPP}(I_0)]}{m},$$

where I_0 is the instance obtained from instance I by setting the release dates of all jobs to $r_j = 0$. Now, Theorem A.1 implies directly the following result.

THEOREM A.2. F-GIPP *is a $(2+\epsilon)$-approximation for the preemptive stochastic problem $\text{P} \mid r_j, pmtn \mid \mathbb{E}[\sum w_j C_j]$ of scheduling jobs with nontrivial release dates on parallel machines, when the processing times are drawn from finite continuous distributions.*

References

[1] Afrati FN, Bampis E, Chekuri C, Karger DR, Kenyon C, Khanna S, Milis I, et al. (1999) Approximation schemes for minimizing average weighted completion time with release dates. *Proc. 40th IEEE Sympos. Foundations Comput. Sci.* (*FOCS*) (IEEE Computer Society, Washington, DC), 32–43.
[2] Bruno JL, Downey PJ, Frederickson GN (1981) Sequencing tasks with exponential service times to minimize the expected flowtime or makespan. *J. ACM* 28:100–113.
[3] Chazan D, Konheim AG, Weiss B (1968) A note on time sharing. *J. Combin. Theory* 5:344–369.
[4] Chekuri C, Motwani R, Natarajan B, Stein C (2001) Approximation techniques for average completion time scheduling. *SIAM J. Comput.* 31:146–166.
[5] Chou C-FM, Liu H, Queyranne M, Simchi-Levi D (2006) On the asymptotic optimality of a simple on-line algorithm for the stochastic single-machine weighted completion time problem and its extensions. *Oper. Res.* 54(3):464–474.
[6] Coffman EG Jr, Hofri M, Weiss G (1989) Scheduling stochastic jobs with a two-point distribution on two parallel machines. *Probab. Engrg. Informational Sci.* 3:89–116.
[7] Correa J, Wagner M (2009) LP-based online scheduling: From single to parallel machines. *Math. Programming* 119:109–136.
[8] Dean BC, Goemans MX, Vondrák J (2008) Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.* 33(4):945–964.
[9] Gittins JC (1979) Bandit processes and dynamic allocation indices. *J. Roy. Statist. Soc., Ser. B* 41:148–177.
[10] Gittins JC (1989) *Multi-Armed Bandit Allocation Indices* (Wiley, New York).
[11] Glazebrook KD (1979) Scheduling tasks with exponential service times on parallel processors. *J. Appl. Probab.* 16:658–689.
[12] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.* 5:287–326.
[13] Kämpke T (1989) Optimal scheduling of jobs with exponential service times on identical parallel processors. *Oper. Res.* 37(1):126–133.
[14] Konheim AG (1968) A note on time sharing with preferred customers. *Probab. Theory Related Fields* 9:112–130.
[15] Labetoulle J, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1984) Preemptive scheduling of uniform machines subject to release dates. Pulleyblank WR, ed. *Progress in Combinatorial Optimization* (Academic Press, New York), 245–261.
[16] Lenstra JK, Rinnooy Kan AHG, Brucker P (1977) Complexity of machine scheduling problems. *Ann. Discrete Math.* 1:243–362.
[17] Megow N (2006) Coping with incomplete information in scheduling—Stochastic and online models. Ph.D. thesis, Technische Universität, Berlin.
[18] Megow N, Schulz AS (2004) On-line scheduling to minimize average completion time revisited. *Oper. Res. Lett.* 32(5):485–490.
[19] Megow N, Uetz M, Vredeveld T (2006) Models and algorithms for stochastic online scheduling. *Math. Oper. Res.* 31(3):513–525.
[20] Megow N, Vredeveld T (2006) Approximation in preemptive stochastic online scheduling. Azar Y, Erlebach T, eds. *Proc. 14th Eur. Sympos. Algorithms* (*ESA*), Lecture Notes in Computer Science, Vol. 4168 (Springer, Berlin, Heidelberg), 516–527.
[21] Megow N, Vredeveld T (2009) Approximating preemptive stochastic scheduling. Technical report, METEOR Research Memorandum RM/09/054, Maastricht University, The Netherlands.
[22] Möhring RH, Radermacher FJ, Weiss G (1984) Stochastic scheduling problems I: General strategies. *Zeitschrift für Oper. Res.* 28:193–260.
[23] Möhring RH, Radermacher FJ, Weiss G (1985) Stochastic scheduling problems II: Set strategies. *Zeitschrift für Oper. Res.* 29(3):A65–A104.
[24] Möhring RH, Schulz AS, Uetz M (1999) Approximation in stochastic scheduling: The power of LP-based priority policies. *J. ACM* 46:924–942.
[25] Pinedo M (1983) Stochastic scheduling with release dates and due dates. *Oper. Res.* 31:559–572.
[26] Pinedo ML (2008) *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. (Springer, New York).
[27] Pruhs KR, Sgall J, Torng E (2004) Online scheduling. Leung JY-T, ed. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chap. 15 (Chapman & Hall/CRC, New York).
[28] Rothkopf MH (1966) Scheduling with random service times. *Management Sci.* 12:703–713.

[29] Schrage L (1968) A proof of the optimality of the shortest remaining processing time discipline. *Oper. Res.* 16:687–690.
[30] Schulz AS (2008) Stochastic online scheduling revisited. Yang B, Du D-Z, Wang CA, eds. *Proc. 2nd Internat. Conf. Combin. Optim. Appl.* (*COCOA*), Lecture Notes in Computer Science, Vol. 5165 (Springer, Berlin, Heidelberg), 448–457.
[31] Schulz AS, Skutella M (2002) The power of α-points in preemptive single machine scheduling. *J. Scheduling* 5:121–133.
[32] Schulz AS, Skutella M (2002) Scheduling unrelated machines by randomized rounding. *SIAM J. Discrete Math.* 15:450–469.
[33] Sevcik KC (1974) Scheduling for minimum total loss using service time distributions. *J. ACM* 21:65–75.
[34] Sgall J (1998) On-line scheduling—A survey. Fiat A, Woeginger GJ, eds. *Online Algorithms: The State of the Art*, Lecture Notes in Computer Science, Vol. 1442 (Springer, Berlin), 196–231.
[35] Sitters R (2010) Competitive analysis of preemptive single-machine scheduling. *Oper. Res. Lett.* 38(6):585–588.
[36] Sitters R (2010) Efficient algorithms for average completion time scheduling. Eisenbrand F, Shepherd FB, eds. *Proc. 14th Math. Programming Soc. Conf. Integer Programming Combin. Optim.* (*IPCO*), Lecture Notes in Computer Science, Vol. 6080 (Springer, Berlin, Heidelberg), 411–423.
[37] Skutella M, Uetz M (2005) Stochastic machine scheduling with precedence constraints. *SIAM J. Comput.* 34(4):788–802.
[38] Smith WE (1956) Various optimizers for single-stage production. *Naval Res. Logist. Quart.* 3:59–66.
[39] Vredeveld T (2012) Stochastic online scheduling. *Comput. Sci.—R&D* 27(3):181–187.
[40] Weber RR (1982) Scheduling jobs with stochastic processing requirements on parallel machines to minimize makespan or flow time. *J. Appl. Probab.* 19:167–182.
[41] Weiss G (1995) On almost optimal priority rules for preemptive scheduling of stochastic jobs on parallel machines. *Advances Appl. Probab.* 27:821–839.
[42] Weiss G, Pinedo M (1980) Scheduling tasks with exponential services times on nonidentical processors to minimize various cost functions. *J. Appl. Probab.* 17:187–202.
[43] Xiong B, Chung C (2012) Completion time scheduling and the WSRPT algorithm. Mahjoub AR, Markakis V, Milis I, Paschos VT, eds. *Proc. Second Internat. Sympos. Combin. Optim.* (*ISCO*), Lecture Notes in Computer Science, Vol. 7422 (Springer, Berlin, Heidelberg), 416–426.