Approximating preemptive stochastic scheduling

Maastricht University

Nicole Megow, Tjark Vredeveld

Approximating Preemptive Stochastic Scheduling

RM/09/054

METEOR

Maastricht University School of Business and Economics
Maastricht Research School of Economics
of Technology and Organization

P.O. Box 616
NL - 6200 MD Maastricht
The Netherlands

Approximating Preemptive Stochastic Scheduling*

Nicole Megow† Tjark Vredeveld‡

November 4, 2009

Abstract

We present constant approximative policies for preemptive stochastic scheduling. We derive policies with a guaranteed performance ratio of 2 for scheduling jobs with release dates on identical parallel machines subject to minimizing the sum of weighted completion times. Our policies as well as their analysis apply also to the recently introduced more general model of stochastic online scheduling. The performance guarantee we give matches the best result known for the corresponding deterministic online problem.

In contrast to previous results for non-preemptive stochastic scheduling, our preemptive policies yield an approximation guarantee that is independent of the processing time distributions. However, our policies extensively utilize information on the distributions other than the first (and second) moments. To obtain our results, we 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 policies.

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., [25]. Only recently has research also focussed on approximative policies for less restrictive problem settings [8, 19, 20, 23, 29, 35]. All these results apply to non-preemptive scheduling, and we are not aware of any approximation results when job preemption is allowed.

In this paper, we consider the stochastic version of the classical problem of scheduling jobs preemptively, with or without release dates, on identical parallel machines. Our goal is to minimize the expected sum of weighted completion times. We present policies with an approximation guarantee of 2. These policies, as well as their analysis, are based on the celebrated Gittins index priority rule [9, 10]. This dynamic allocation rule was originally proposed for optimal control in the multi-armed bandit problem and since then has found several applications in other areas. Compared to previous approximation results in non-preemptive stochastic scheduling, our results have the advantage that the guarantee is constant and does not depend on the properties of the underlying probability distributions for the processing times. On the other hand, our policies need to have complete information about the probability distribution, whereas previous approximative policies for non-preemptive stochastic scheduling only require information about the first and the second moments.

Our algorithms and their analysis apply also in the more general model of stochastic online scheduling [5, 19]. It is worth mentioning that our approximation result of 2 exactly matches the best performance guarantee known for the deterministic online version of this problem [18]. This result not only justifies the

†Max-Planck-Institut für Informatik, Saarbrücken, Germany, nmegow@mpi-inf.mpg.de.

‡Department of Quantitative Economics, Maastricht University, The Netherlands, t.vredeveld@maastrichtuniversity.nl.

*Parts of this work appeared previously in *Proceedings of the 14th European Symposium on Algorithms*, 2006, Zurich, Switzerland. The first author's research was partially supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin. The second author's research was partially supported by METEOR, the Maastricht Research School of Economics of Technology and Organizations.

general model for scheduling with incomplete information, it also shows—for this particular scheduling problem—that general policies that can handle stochastic *and* online information may achieve the same performance guarantee as specialized policies handling only one type of limited information.

Model and problem definition. Let $J = \{1,2,\dots,n\}$ be a set of jobs which 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 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 $\Omega_j \subseteq \{1,2,\dots,M_j\}$, 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 $\mathrm{P}\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$.

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, Radermacher, and Weiss [21, 22]. Roughly speaking, a scheduling policy makes scheduling decisions at certain *decision time points* t, and 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, such as the actual realizations p_j of processing times of jobs that have not yet been completed by time t.

In this paper, we concentrate on approximation policies as defined by Möhring, Schulz, and Uetz in [23].

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

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

where $\mathbb{E}[\Pi(I)]$ *and* $\mathbb{E}[\mathsf{Opt}(I)]$ *denote the expected values that the policy* Π *and an optimal non-anticipatory 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 non-anticipatory. We refer to [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 non-anticipatory *online* policy with the expected outcome of an optimal non-anticipatory offline policy.

Previous work. The classical deterministic variant of our scheduling problem which seeks to minimize the weighted sum of completion times is well-known to be NP-hard [15, 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 forty years. We refer the reader to Pinedo's book [25] for an overview. Some of the first results on *preemptive scheduling* that can be found in the literature are by Chazan, Konheim, and Weiss [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 [32] developed an intuitive method for creating optimal schedules (in expectation). He introduces a priority policy that relies on a dynamic index which 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 [9, 10]. Later in 1995, Weiss [38] 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 [3, 14, 32]: 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 [24] 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 even on parallel machines: Rothkopf [27] showed that for increasing hazard rate (failure rate) distributions, no finite number of preemptions can outperform a non-preemptive 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 [37] 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 non-preemptive *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 [39], and Bruno, Downey, and Frederickson [2]. In the case when processing times are not drawn from a distribution with monotone hazard rates, Coffman, Hofri, and Weiss [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 [38] 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, $\mathbb{E}[P_i] < \mathbb{E}[P_j]$ implies $w_i \leq w_j$. In fact, under such an assumption, WSEPT in fact coincides with SEPT.

While optimal policies have been found only for very special problem settings, research has focussed lately on obtaining approximation algorithms. Such investigations have been successful in the non-preemptive setting. Möhring, Schulz, and Uetz [23] derived the first constant-factor approximations for the non-preemptive 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. Their results were slightly improved later by Megow et al. [19] and Schulz [29] for a more general setting. Skutella and Uetz [35] complemented the first approximation results by approximation policies for scheduling with precedence constraints. In general, all known performance guarantees for non-preemptive policies are derived using the same technique for deriving LP-based lower bounds in [23], and they all 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 [5, 19, 20, 29]. All obtained results which include asymptotic optimality [5] and approximation guarantees for deterministic as well as randomized policies [19, 29] including precedence constraints [20] address non-preemptive scheduling.

Several scheduling algorithms were designed and analyzed for deterministic problem variants of this online model. For a general survey of online scheduling models and results, we refer the reader to Sgall [33] and Pruhs, Sgall, and Torng [26]. In the context of preemptive scheduling, Sitters [34] gave a 1.56-competitive algorithm for the single-machine problem. This is the best result currently known. It improved upon an earlier result by Schulz and Skutella [30], who generalized the classical *Smith Rule* [36] 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 by Megow and Schulz [18]. Even when considering randomized algorithms, there is no better guarantee known than $2 - 1/m$ for this problem on parallel machines [7]. For the single-machine problem, Schulz and Skutella [31] provide a randomized $4/3$-competitive algorithm.

Our contribution. We present constant approximative policies for preemptive stochastic scheduling. For jobs with arbitrary processing time distributions and individual release dates, we give two different 2-approximative policies for multiple machines. In comparison to previously known results for non-preemptive variants of this model, our results stand out by being constant and independent of the probability distribution of processing times. Our policies, as well as their analysis, apply also to the more general model of stochastic online scheduling. The performance guarantee of 2 for preemptive stochastic online scheduling matches the best result known in deterministic online scheduling [18], although we consider a more general model.

We present two different policies and their analysis. Both policies are motivated by Gipp, the optimal policy for the single-machine problem without release dates [14, 32, 38]. However, they differ in the degree of similarity to Gipp and in the tightness of the result we prove. The first policy, FOLLOW-Gipp (F-Gipp), is a parallel-machine policy which 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 for quite an easy analysis. However, the performance guarantee of 2 is tight, and this is true even on a single machine. Our second policy is actually a single-machine policy which we see as the natural generalization of Gipp to the setting with arbitrary release dates; we call it GENERALIZED-Gipp (Gen-Gipp). The analysis is more involved, but yields the same guarantee of 2. However, we conjecture that the true approximation ratio is much lower. The best upper bound we can give on Gen-Gipp's approximation guarantee is 1.21. 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.

The Gittins index not only inspires the design of our scheduling policies, but it is also crucial for bounding the optimal value. We derive a new nontrivial 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.

In general, our policies are not optimal. However, under restricted problem settings, they coincide 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 [2, 11, 39] or, more generally, if they are agreeable [13]. If there is only a single machine available and jobs have arbitrary release dates, then F-Gipp, which coincides with preemptive WSEPT, is optimal [24]. If there are no release dates, then both F-Gipp as well as Gen-Gipp solve the weighted single-machine problem optimally for arbitrary processing time distributions because, in that case, both coincide with the optimal policy Gipp [14, 32, 38]. Finally, we discuss the behavior of F-Gipp and Gen-Gipp under deterministic input on a single machine. In the unweighted setting with release dates, Gen-Gipp yields an optimal solution, since it obtains the same schedule as Schrage's optimal *Shortest Remaining Processing Time* (SRPT) rule [28]. In contrast, this is not true for F-Gipp, which coincides with the suboptimal *Shortest Processing Time* (SPT) rule in that case. However, in the case of arbitrary job weights without release dates, Gen-Gipp as well as F-Gipp are optimal since they coincide with the *Weighted Shortest Processing Time* (WSPT) rule, also known as *Smith's Rule* [36]. This folkloric

algorithm processes at any time the unfinished job with the highest ratio w_j/p_j; however, its preemptive variant has an approximation guarantee of 2 for arbitrary release dates, which is tight [30]. In a way, our stochastic policies, F-Gipp and Gen-Gipp, can be seen as generalizations of the deterministic algorithms WSPT and WSRPT (weighted variant of SRPT; see [17]), respectively. Similar to our conjecture that Gen-Gipp may outperform F-Gipp, [17] proposed WSRPT as a promising candidate for improving on the tight performance result for WSPT on a single machine with arbitrary job weights and release dates.

Organization of the paper. In Section 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. In Section 3, we derive our new lower bound on the expected value of an unknown optimal policy for the preemptive stochastic scheduling problem. In Section 4, we introduce a simple parallel-machine policy, F-Gipp, with an approximation factor of exactly 2. In Section 5, there follows a more natural 2-approximative policy for the single machine, Gen-Gipp, that uses more information about the current status of a job. Using an immediate randomized extension to multiple machines, it also yields an approximation guarantee of 2. For the last two policies, we cannot show an improved performance guarantee. However, there is well-founded hope that their approximation factor is less than we prove here. In Section 6, we comment on the feasibility of the techniques presented for use in the more general model of stochastic online scheduling.

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 non-trivial release dates.

Given that a job j has been processing 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

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

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 sub-job of length q of job j, after it has completed y units of processing:

$$R_j(q,y) = \frac{w_j \Pr\left[P_j - y \leq q \,|\, P_j > y\right]}{I_j(q,y)}.$$

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

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

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 1 ([14,32,38]). *The Gittins index priority policy* (Gipp) *optimally solves the stochastic scheduling problem* $1\,|\,\mathrm{pmtn}\,|\,\mathbb{E}[\sum w_j C_j]$ *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 [10,38]. In parts, they have been derived earlier in the scheduling context by Konheim [14] and Sevcik [32]. They prove useful to analyze Gipp as well as the more general policies in the following sections.

Proposition 2 ([10,38]). *Consider a job j that has been processed for y time units. Then, for any $0 < \zeta < q_j(y)$ it holds that*

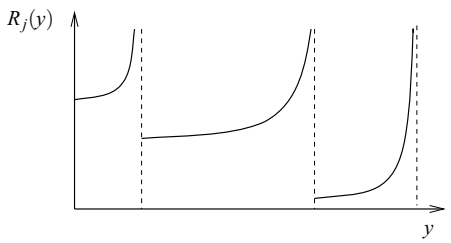
$$R_j(y) \;\leq\; R_j(y+\zeta)\,, \tag{a}$$
$$q_j(y+\zeta) \;\leq\; q_j(y)-\zeta\,, \tag{b}$$
$$R_j(y+q_j(y)) \;\leq\; R_j(y)\,. \tag{c}$$

Denote the sub-job of length $q_j(y)$ that causes the maximal rank $R_j(y)$, a *quantum* of job j. 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 (a), we know that, while processing a quantum, the rank of the job does not decrease, whereas Proposition 2 (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 non-preemptively in non-increasing 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.

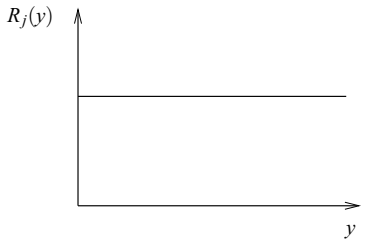
Algorithm 2: (Reformulated) Gittins index priority policy (Gipp)

For each job, recursively compute the partition into quanta of maximal rank. Schedule job quanta of unfinished jobs in non-decreasing 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.



(a) The maximum rank of a job with processing time that follows a three-point distribution. The breakpoints occur after each quantum in the recursive partition.



(b) The maximum rank of a job with exponential processing time.

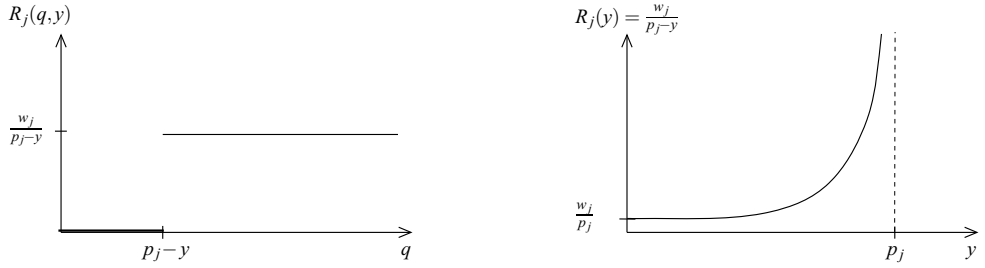
Figure 1: The maximum rank of jobs with certain processing time distributions depending on the amount of time, y, that the job has been processing.

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.

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

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 [36].



(a) The rank of a sub-job of length q after y units of processing.

(b) The maximum rank of a job after it has been processing for y units of time.

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

For the analysis of our policies in the following sections, Proposition 2 (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 (a), at least as large as R and, as Proposition 2 (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 non-increasing 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, $Q = \{(k,\ell) \,|\, k = 1,\ldots,n,\ l = 1,\ldots,n_k\}$, then

$$\begin{aligned} H(j,i) \; &= \left\{ (k,\ell) \in Q \;\middle|\; R_k(y_{k\ell}) > R_j(y_{ji}) \right\} \\ &\quad \cup \left\{ (k,\ell) \in Q \;\middle|\; R_k(y_{k\ell}) = R_j(y_{ji}) \;\wedge\; k \le j \right\}. \end{aligned}$$

Since the Gittins index of a job is decreasing with every finished quantum (Prop. 2 (c)), we know that $H(j,h) \subseteq H(j,i)$, for $h \le i$. In order 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 $H(j,0) = \emptyset$. Note that the quantum (j,i) is also 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.

Lemma 3. *The optimal policy for* $1 \,|\, pmtn \,|\, \mathbb{E}[\sum w_j C_j]$, Gipp, *achieves an expected objective value of*

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

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, $y_{ji_p} < p_j \leq y_{ji_p} + q_{ji_p}$. Gipp processes quanta of jobs that have not completed non-preemptively in non-increasing order of their ranks. Hence,

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

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

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

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

The equalities follow from an index rearrangement and the facts that, by definition, $H(j,h) = \cup_{i=1}^{h} H'(j,i)$ 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 $\xi_{k\ell}$ for any i,ℓ. Using linearity of expectation, we rewrite (2) as

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

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

3 A new lower bound on the optimum on parallel machines

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 $\mathbb{E}[\mathsf{Opt}]$ of an unknown optimal policy Opt. The trivial bound $\mathbb{E}[\mathsf{Opt}] \geq \sum_{j\in J} w_j(r_j + \mathbb{E}[P_j])$ is unlikely to suffice proving constant approximation guarantees. However, we are not aware of any other bounds for the general preemptive problem. LP-based approaches are used in the non-preemptive setting [5, 19, 23, 29, 35], but it is unclear if and how they transfer.

In this section, we derive a new non-trivial 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 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 stochastic setting as well.

Let I denote a scheduling instance of the parallel-machine scheduling problem $\mathrm{P}\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_jC_j]$, 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 Opt_1 denote an optimal single-machine policy that yields an expected value $\mathbb{E}[\mathsf{Opt}_1(I')]$ on instance I'.

Lemma 4. *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* Opt_1 *on instance* I' *on a fast single machine, that is,*

$$\mathbb{E}[\Pi(I)] \;\geq\; \mathbb{E}\left[\mathsf{Opt}_1(I')\right].$$

Proof. Given a parallel-machine policy Π, we provide a policy Π' for the fast single machine that yields an expected objective value $\mathbb{E}[\Pi'(I')] \leq \mathbb{E}[\Pi(I)]$ for any instance I. Then the lemma follows since an optimal policy Opt_1 yields on the single machine an expected objective value $\mathbb{E}[\mathsf{Opt}_1(I')] \leq \mathbb{E}[\Pi'(I')]$.

We construct policy Π' by letting its first decision point coincide with the first decision point of policy Π (the earliest release date). At any of its decision points, Π' can compute the jobs to be scheduled by policy Π and, due to the fact that the processing times of all jobs are discrete random variables, it computes the earliest possible completion time of these jobs, in the parallel-machine schedule. The next decision point of Π' is the minimum of these possible completion times and the next decision point of Π. Between two consecutive decision points of Π', the policy schedules the same set of jobs that Π schedules, for the same amount of time. This is possible because the single machine, on which Π' operates, works m times as fast.

In this way, we ensure that all job completions in the parallel-machine schedule obtained by Π coincide with a decision point of policy Π'. Moreover, as Π' schedules the same set of jobs as Π between two decision points, any job that completes its processing at a certain time t in the schedule of Π, will also be completed by time t in the schedule of Π'. □

With this relaxation, we derive a lower bound on the expected optimal value.

Theorem 5. *The expected value of an optimal policy* Opt *for the parallel-machine problem I is bounded by*

$$\mathbb{E}[\mathsf{Opt}(I)] \;\geq\; \frac{1}{m}\sum_{j\in J} w_j \sum_{i=1}^{n_j} \sum_{(k,\ell)\in H'(j,i)} \Pr[P_j > y_{ji} \wedge P_k > y_{k\ell}]\cdot I_k(q_{k\ell}, y_{k\ell}).$$

Proof. We consider the fast single-machine instance I' introduced above and relax it further to instance I'_0 by setting all release dates equal. By Theorem 1, the resulting problem can be solved optimally by Gipp. Then, with Lemma 4 we have

$$\mathbb{E}[\mathsf{Opt}(I)] \;\geq\; \mathbb{E}\left[\mathsf{Opt}_1(I')\right] \;\geq\; \mathbb{E}\left[\mathsf{Gipp}(I'_0)\right]. \tag{3}$$

From Lemma 3 we know that

$$\mathbb{E}\left[\mathsf{Gipp}(I'_0)\right] \;=\; \sum_{j\in J} w_j \sum_{i=1}^{n_j} \sum_{(k,\ell)\in H'(j,i)} \Pr\left[P'_j > y'_{ji} \wedge P'_k > y'_{k\ell}\right]\cdot I'_k(q'_{k\ell}, y'_{k\ell}), \tag{4}$$

where the primes indicate the modified variables in the fast single-machine instance I'_0. By definition, $P'_j = P_j/m$ holds for any job j; this is also true for $\Pr[P_j > x] = \Pr\left[P'_j > x/m\right]$. Furthermore, the probability distribution for the remaining processing time after y units of processing, $\Pr[P_j - y = x \,|\, P_j > y]$, remains the same on the fast machine. Therefore, the expected investment $I'_j(q',y')$ for any sub-job of length $q' = q/m$ of job j after it has received $y' = y/m$ units of processing coincides with

$$\begin{aligned} I'_j(q',y') &= \mathbb{E}\left[\min\{P'_j - y', q'\} \,|\, P'_j > y'\right] \\ &= \frac{1}{m}\,\mathbb{E}\left[\min\{P_j - y, q\} \,|\, P_j > y\right] = \frac{1}{m} I_j(q,y)\,. \end{aligned}$$

We conclude that the partition of jobs into quanta in instance I immediately gives the partition for the fast single-machine instance I'. Each quantum (j,i) of job j maximizes the rank $R_j(q,y_{ji})$ and thus $q' = q/m$ maximizes the rank $R'_j(q/m, y/m) = R_j(q,y)/m$ on the single machine; hence, the quanta are simply shortened to an m-fraction of the original length, $q'_{ji} = q_{ji}/m$ and, therefore, $y'_{ji} = \sum_{l=1}^{i-1} q'_{jl} = y_{ji}/m$.

Combining these observations with (3) and (4) yields

$$\mathbb{E}[\mathsf{Opt}(I)] \;\geq\; \frac{1}{m} \sum_{j\in J} w_j \sum_{i=1}^{n_j} \sum_{(k,\ell)\in H'(j,i)} \Pr[P_j > y_{ji} \wedge P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell})\,.$$

□

Theorem 5 above and Lemma 3 directly imply the following lower bound.

Corollary 6. *The lower bound on the optimal preemptive policy for parallel-machine scheduling on an instance I equals an m-fraction of the expected value achieved by* Gipp *on the relaxed instance I_0 without release dates but with the same processing times to be scheduled on one machine, that is,*

$$\mathbb{E}[\mathsf{Opt}(I)] \;\geq\; \frac{\mathbb{E}[\mathsf{Gipp}(I_0)]}{m}\,. \tag{5}$$

4 A simple algorithm 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 non-trivial release dates, $\mathrm{P}\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$. 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 processing for some time less than q_{ji}. Informally speaking, F-Gipp updates the ranks only after quantum completions and then follows Gipp.

Theorem 7. F-Gipp *is a* 2*-approximation for the preemptive stochastic problem* $\mathrm{P}\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$ *of scheduling jobs with non-trivial release dates on parallel machines.*

Proof. Fix a realization $p \in \Omega$ of processing times and consider a job j and its completion time $C_j^{\mathsf{F\text{-}Gipp}}(p)$. Job j is processing in the time interval $[r_j, C_j^{\mathsf{F\text{-}Gipp}}(p)]$. We split this interval into two disjunctive sets of sub-intervals, $T(j,p)$ and $\overline{T}(j,p)$. Let $T(j,p)$ denote the set of sub-intervals in which job j is processing

and $\overline{T}(j,p)$ contains the remaining sub-intervals. Denoting the total length of all intervals in a set T by $|T|$, we have

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

The total length of intervals in $T(j,p)$ is, by definition, p_j. In intervals of the set $\overline{T}(j,p)$, 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, $|\overline{T}(j,p)|$ 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 $|\overline{T}(j,p)|$ which is the sum of all realized quantum lengths on m machines. That yields

$$C_j^{\text{F-Gipp}}(p) \;\leq\; r_j + p_j + \frac{1}{m} \sum_{\substack{(k,\ell)\in H(j,i_p):\\ p_k > y_{k\ell}}} \min\{q_{k\ell}, p_k - y_{k\ell}\}\,.$$

Following the same arguments as in Lemma 3, weighted summation over all jobs and taking expectations on both sides gives:

$$\sum_{j\in J} w_j \mathbb{E}\left[C_j^{\text{F-Gipp}}\right] \;\leq\; \sum_{j\in J} w_j\,(r_j + \mathbb{E}[P_j]) \;+\; \frac{1}{m} \sum_{j\in J} w_j \sum_{i=1}^{n_j} \sum_{(k,\ell)\in H'(j,i)} \Pr[P_j > y_{ji} \wedge p_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell})\,.$$

Finally, we apply the trivial lower bound $\mathbb{E}[\mathsf{Opt}] \geq \sum_{j\in J} w_j(r_j + \mathbb{E}[P_j])$ and Theorem 5, and the approximation result follows. □

In the case of exponentially distributed processing times and absent release dates, our F-Gipp policy coincides with the WSEPT rule. This rule is known to be optimal if all weights are equal [14, 32, 38] or, more generally, if they are agreeable [13]. On a single machine, our policy coincides again with the WSEPT rule if all processing times are drawn from exponential distributions, and it is, thus, optimal [24]. In the absence of release dates and for general processing times, our policy coincides on a single machine with Gipp and is in that case optimal as well (Theorem 1), even if jobs have arbitrary weights.

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

Theorem 8 ([18])**.** *The approximation ratio of* F-Gipp *is not better than* 2 *for the problem* $\mathrm{P}\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$*, for any given number of machines.*

5 A more natural alternative algorithm

In this section, we present another policy for which we prove the approximation guarantee of 2. Again, our policy is based on classical Gipp. In contrast to the previous policy F-Gipp, we now deviate less from the original Gittins index priority rule and, thus, we use more information on the actual state of the set of known, unfinished jobs. While the analysis of F-Gipp in the previous section is tight, we conjecture that our new policy has a better performance than we prove here.

5.1 Single machine: generalized Gittins index priority policy

Consider the problem of preemptive scheduling on a single processor. As mentioned earlier, Gipp is not an optimal policy even in this single-machine setting due to release dates. A straightforward extension of Gipp is to choose at any time the job with the highest rank among the set of available jobs.

Algorithm 4: Generalized Gittins Index Priority Policy (Gen-Gipp)

At any time t, process an available job j with the current highest rank, $R_j(y_j(t))$, depending on the amount of processing $y_j(t)$ that the job j has completed by time t.

In principle, the jobs are still processed in non-increasing order of maximum ranks as in Gipp and F-Gipp. Applied to an instance with equal release dates, all three policies, Gipp, F-Gipp and Gen-Gipp, yield the same schedule. As in F-Gipp, the generalization in Gen-Gipp concerns the fact that we incorporate release dates and cause preemptions of quanta whereas Gipp preempts jobs only after completion of a quantum. Due to different arrival times, F-Gipp and Gen-Gipp preempt jobs within the processing of a quantum if a job with a higher rank is released. The crucial difference between these two policies concerns the time for updating the rank of a preempted quantum: while F-Gipp recomputes the rank of a job only after a quantum has completed, Gen-Gipp considers at any time the actual maximum rank of a job. In that sense, it is an immediate generalization of the original Gittins index priority policy. The interesting question addresses the effect of these rank updates in case of job preemption on the ordering of quanta.

From Proposition 2 (a), we know that if a quantum (j,k) is preempted after $\zeta < q_{jk}$ units of processing, the rank of job j has not decreased, that is, $R_j(y_{jk}+\zeta) \geq R_j(y_{jk})$. Hence, all quanta with a lower priority than the original priority of (j,k) available at or after the time that (j,k) is preempted will not be processed before quantum (j,k) is completed.

Consider a realization of processing times $p \in \Omega$ and a job j in the schedule obtained by Gen-Gipp. Let i_p be the index of the quantum in which job j finishes, that is, $y_{ji_p} < p_j \leq y_{ji_p} + q_{ji_p}$. Then the completion time $C_j^{\text{Gen-Gipp}}$ of job j can be bounded by its release date plus the total length of the quanta that have a higher rank than (j,i_p) at time r_j. This also includes quanta of jobs k with $r_k > r_j$ since they have rank $R_k(0)$, even though they are not available for scheduling.

However, this set of quanta contains not only quanta in $H(j,i_p)$ with a higher priority than (j,i_p). In the presence of release dates the following situation is possible: A part of quantum $(k,\ell) \in L(j,i_p)$ is scheduled before quantum (j,i_p), which has higher rank than (k,ℓ), even though job j is available. This happens when job k has been processed for $\gamma \in (y_{k\ell}, y_{k,\ell+1})$ units of time before time r_j and its rank improved (increased) such that $R_k(\gamma) > R_j(y_{ji_p})$. We call this event $\mathscr{E}_{k,ji_p}(\gamma)$ and we say that *job k* or *one of its quanta disturbs* (j,i_p). Formally, we define

$$\mathscr{E}_{k,ji}(\gamma) \;=\; \{\,\text{by time } r_j\,,\, k \text{ has been processed for } \gamma \text{ units of time and } R_k(\gamma) > R_j(y_{ji})\,\}\,.$$

For $y_{k\ell} < \gamma \leq y_{k,\ell+1}$, the amount of time that the quantum $(k,\ell) \in L(j,i)$ disturbs (j,i) is given by

$$q_{k\ell}^{ji}(\gamma) = \max\{q \,:\, R_k(\gamma+q) > R_j(y_{ji})\}.$$

Because $(k,\ell) \in L(j,i)$, we know by Proposition 2(b) and (c) that $q_{k\ell}^{ji} \leq y_{k,\ell+1} - \gamma$. Note that the event $\mathscr{E}_{k,ji}(\gamma)$ only depends on the (partial) realizations of jobs that have been processed before r_j and is, therefore, independent of P_j.

Now, let us come back to the completion time of a job j in a realization $p \in \Omega$. As stated above, it can be bounded by r_j plus the total sum of quanta that have a higher rank at time r_j. These are contained in:

(i) all quanta in $H(j,i_p)$ except those for which event $\mathscr{E}_{j,k\ell}(\gamma)$ occurs with $p_j \in (\gamma,\, \gamma + q_{ji_p}^{k\ell}(\gamma)\,]$, that is, quanta that are disturbed by (j,i_p) with j completing while it is disturbing, and

(ii) the quanta $(k,\ell) \in L(j,i_p)$ for which an event $\mathscr{E}_{k,ji_p}(\gamma)$ occurs for some $\gamma > y_{k\ell}$, that is, quanta that disturb (j,i_p).

Expressed formally, we get the following proposition.

Proposition 9. *Given a realization of processing times $p \in \Omega$ and a job j, let i_p be the index of the quantum in which this job finishes, that is, $y_{ji_p} < p_j \leq y_{j,i_p+1}$. Then, the completion time of job j in the* Gen-Gipp

schedule can be bounded by

$$
\begin{aligned}
C_j^{\text{Gen-Gipp}}(p) \;\le\; & r_j \\
& + \sum_{(k,\ell)\in H(j,i_p)\,:\,p_k>y_{k\ell}} \min\{q_{k\ell}, p_k - y_{k\ell}\} && (6)\\
& - \sum_{\substack{(k,\ell)\in H(j,i_p):\\ p_k>y_{k\ell}}} \sum_{\substack{\gamma:\mathscr{E}_{j,k\ell}(\gamma),\\ p_j\in(\gamma,\gamma+q_{ji_p}^{k\ell}(\gamma)]}} \min\{q_{k\ell}, p_k - y_{k\ell}\} && (7)\\
& + \sum_{\substack{(k,\ell)\in L(j,i_p):\\ p_k>y_{k\ell}}} \sum_{\substack{\gamma:\mathscr{E}_{k,ji_p}(\gamma),\\ y_{k,\ell}<\gamma<y_{k,\ell+1}}} \min\{q_{k\ell}^{ji_p}(\gamma), p_k - \gamma\}\,. && (8)
\end{aligned}
$$

Given the above bound for a particular realization, we compute the expected completion time of job j.

Lemma 10. *The expected completion time of job j under* Gen-Gipp *can be bounded by*

$$
\begin{aligned}
\mathbb{E}\left[C_j^{\text{Gen-Gipp}}\right] \;\le\; & r_j + \sum_{i=1}^{n_j} \sum_{(k,\ell)\in H'(j,i)} \Pr[P_j > y_{ji} \wedge P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell}) \\
& - \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in H(j,i)}} \sum_{\gamma=y_{ji}}^{y_{j,i+1}} \Pr\left[\mathscr{E}_{j,k\ell}(\gamma) \wedge \gamma < P_j \le \gamma + q_{ji}^{k\ell}(\gamma)\right] \cdot \Pr[P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell}) \\
& + \sum_{i=1}^{n_j} \sum_{(k,\ell)\in L(j,i)} \sum_{\gamma=y_{k\ell}}^{y_{k,\ell+1}} \Pr\left[y_{ji} < P_j \le y_{j,i+1}\right] \cdot \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot I_k(q_{k\ell}^{ji}(\gamma), \gamma).
\end{aligned}
$$

Proof. The bound in Proposition 9 holds for each realization $p \in \Omega$. Taking the expectation over all realizations on both sides, we get an upper bound on the expected completion time of a job j scheduled by Gen-Gipp. By linearity of expectation, we can consider the sum of expected values of the summands (6), (7), and (8) separately.

Recall that $\chi(\mathscr{E})$ is an indicator random variable which equals 1 if and only if the event $\mathscr{E}$ occurs; furthermore, $\xi_{k\ell}$ denotes the special indicator random variable for the event $P_k > y_{k\ell}$. We show how to transform the expected values of (6) to (8) such that their sum plus $\mathbb{E}[r_j]$ equals the claimed expression. The term (6) equals exactly the right-hand side of equation (1) in the proof of Lemma 3. In that proof, we showed that

$$
\mathbb{E}[(1)] \;=\; \sum_{i=1}^{n_j} \sum_{(k,\ell)\in H'(j,i)} \Pr[P_j > y_{ji} \wedge P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell}) \;=\; \mathbb{E}[(6)]\,.
$$

Similarly, we transform the expected value of

$$
\sum_{\substack{1\le i\le n_j:\\ y_{ji}<P_j\le y_{j,i+1}}} \sum_{\substack{(k,\ell)\in H(j,i):\\ P_k>y_{k\ell}}} \sum_{\substack{\gamma:\mathscr{E}_{j,k\ell}(\gamma),\\ P_j\in(\gamma,\gamma+q_{ji}^{k\ell}(\gamma)]}} \min\{q_{k\ell}, P_k - y_{k\ell}\}
$$

from (7) to

$$
\begin{aligned}
& \mathbb{E}\left[\sum_{i=1}^{n_j} \chi(y_{ji} < P_j \le y_{j,i+1}) \sum_{\substack{(k,\ell)\in H(j,i):\\ P_k>y_{k\ell}}} \sum_{\substack{\gamma:\mathscr{E}_{j,k\ell}(\gamma),\\ P_j\in(\gamma,\gamma+q_{ji}^{k\ell}(\gamma)]}} \min\{q_{k\ell}, P_k - y_{k\ell}\}\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in H(j,i)}} \sum_{\gamma=y_{ji}}^{y_{j,i+1}} \chi\left(\gamma < P_j \le \gamma + q_{ji}^{k\ell}(\gamma) \wedge P_k > y_{k\ell} \wedge \mathscr{E}_{j,k\ell}(\gamma)\right) \min\{q_{k\ell}, P_k - y_{k\ell} \mid P_k > y_{k\ell}\}\right] \\
&= \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in H(j,i)}} \sum_{\gamma=y_{ji}}^{y_{j,i+1}} \Pr\left[\mathscr{E}_{j,k\ell}(\gamma) \wedge \gamma < P_j \le \gamma + q_{ji}^{k\ell}(\gamma)\right] \cdot \Pr[P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell}).
\end{aligned}
$$

Finally, the expected value of Term (8) can be reformulated in a similar way and, therefore, we omit the details:

$$\mathbb{E}[(8)] = \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in L(j,i)}} \sum_{\gamma=y_{k\ell}}^{y_{k,\ell+1}} \Pr\left[y_{ji} < P_j \le y_{j,i+1}\right] \cdot \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot I_k(q_{k\ell}^{ji}(\gamma), \gamma).$$

The summation of expected values (6) to (8) concludes the proof. □

Using the previous lemmata, we can prove the following approximation guarantee.

Theorem 11. Gen-Gipp *is a* 2*-approximation for the stochastic single-machine problem* $1\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$ *with non-trivial release dates.*

Proof. Denote by I an instance of the problem $1\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$, and let I_0 be the relaxation of I in which we assume all release dates are zero. With Lemmata 10 and 3, we have prepared the ground for bounding the expected objective value, $\mathbb{E}[\,\mathsf{Gen\text{-}Gipp}\,]$, of a schedule that has been obtained by Gen-Gipp.

$$\begin{aligned}\mathbb{E}[\,\mathsf{Gen\text{-}Gipp}(I)\,] &= \sum_{j\in J} w_j \mathbb{E}\left[C_j^{\mathsf{Gen\text{-}Gipp}(I)}\right] \\ &\le \sum_{j\in J} w_j r_j + \mathbb{E}[\,\mathsf{Gipp}(I_0)\,] + \sum_{j\in J} w_j\,(O_j - N_j),\end{aligned}$$

where

$$O_j = \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in L(j,i)}} \sum_{\gamma=y_{k\ell}}^{y_{k,\ell+1}} \Pr\left[y_{ji} < P_j \le y_{j,i+1}\right] \cdot \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot I_k(q_{k\ell}^{ji}(\gamma), \gamma)$$

$$N_j = \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in H(j,i)}} \sum_{\gamma=y_{ji}}^{y_{j,i+1}} \Pr\left[\mathscr{E}_{j,k\ell}(\gamma) \wedge \gamma < P_j \le \gamma + q_{ji}^{k\ell}(\gamma)\right] \cdot \Pr[P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell}).$$

We claim that $\sum_{j\in J} w_j\,(O_j - N_j) \le 0$ and give the proof in Lemma 12 below. This implies the theorem due to the trivial lower bound on the expected value of an optimal policy Opt, $\mathbb{E}[\,\mathsf{Opt}(I)\,] \ge \sum_{j\in J} w_j r_j$, and the fact that Gipp is an optimal policy for the relaxed problem instance without release dates I_0 (Theorem 1), which gives $\mathbb{E}[\,\mathsf{Opt}(I)\,] \ge \mathbb{E}[\,\mathsf{Gipp}(I_0)\,]$. □

Lemma 12. *Let*

$$O_j = \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in L(j,i)}} \sum_{\gamma=y_{k\ell}}^{y_{k,\ell+1}} \Pr\left[y_{ji} < P_j \le y_{j,i+1}\right] \cdot \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot I_k(q_{k\ell}^{ji}(\gamma), \gamma)$$

and

$$N_j = \sum_{i=1}^{n_j} \sum_{\substack{(k,\ell)\\ \in H(j,i)}} \sum_{\gamma=y_{ji}}^{y_{j,i+1}} \Pr\left[\mathscr{E}_{j,k\ell}(\gamma) \wedge \gamma < P_j \le \gamma + q_{ji}^{k\ell}(\gamma)\right] \cdot \Pr[P_k > y_{k\ell}] \cdot I_k(q_{k\ell}, y_{k\ell}),$$

then

$$\sum_{j\in J} w_j\,(O_j - N_j) \;\le\; 0\,.$$

Proof. First note that $(j,i) \in H(k,\ell)$, for jobs $k \ne j$, implies $(k,\ell) \in L(j,i)$ and vice versa. Moreover, the event $\mathscr{E}_{j,ji}(\gamma)$ is empty for all i and γ. Thus, we can transform $\sum_{k\in J} w_k N_k$ by rearranging indices:

$$\begin{aligned}\sum_{k\in J} w_k N_k &= \sum_{k\in J} \sum_{\ell=1}^{n_k} \sum_{\substack{(j,i)\\ \in H(k,\ell)}} \sum_{\gamma=y_{k\ell}}^{y_{k,\ell+1}} w_k \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge \gamma < P_k \le \gamma + q_{k\ell}^{ji}(\gamma)\right] \cdot \Pr[P_j > y_{ji}] \cdot I_j(q_{ji}, y_{ji}) \\ &= \sum_{j\in J} \sum_{i=1}^{n_j} \sum_{\substack{(j,i)\\ \in L(k,\ell)}} \sum_{\gamma=y_{k\ell}}^{k,\ell+1} w_k \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge \gamma < P_k \le \gamma + q_{k\ell}^{ji}(\gamma)\right] \cdot \Pr[P_j > y_{ji}] \cdot I_j(q_{ji}, y_{ji}).\end{aligned}$$

By definition of the conditional probability it holds that

$$\Pr\left[y_{ji} < P_j \le y_{j,i+1}\right] = \Pr\left[P_j > y_{ji}\right] \cdot \Pr\left[P_j \le y_{j,i+1} \,|\, P_j > y_{ji}\right],$$

for any quantum (j,i). Moreover, due to the independence of the processing times, we know that

$$\Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge \gamma < P_k \le y\right] = \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot \Pr\left[P_k \le y \,|\, P_k > \gamma\right]$$

for any y. With these arguments we have

$$\begin{aligned}
\sum_{j\in J} w_j\left(O_j - N_j\right) &= \sum_{j\in J}\sum_{i=1}^{n_j}\sum_{\substack{(k,\ell)\\ \in L(j,i)}}\sum_{\gamma=y_{k\ell}}^{y_{k,\ell+1}}\Bigg(w_j \Pr\left[y_{ji} < P_j \le y_{j,i+1}\right]\Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot I_k(q_{k\ell}^{ji}(\gamma),\gamma) \\
&\qquad\qquad - w_k \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge \gamma < P_k \le \gamma + q_{k\ell}^{ji}(\gamma)\right] \cdot \Pr\left[P_j > y_{ji}\right] \cdot I_j(q_{ji},y_{ji})\Bigg) \\
&= \sum_{j\in J}\sum_{i=1}^{n_j}\sum_{\substack{(k,\ell)\\ \in L(j,i)}}\sum_{\gamma=y_{k\ell}}^{k,\ell+1} \Pr\left[\mathscr{E}_{k,ji}(\gamma) \wedge P_k > \gamma\right] \cdot \Pr\left[P_j > y_{ji}\right] \cdot \\
&\qquad \left(w_j \Pr\left[P_j \le y_{j,i+1} \,|\, P_j > y_{ji}\right] \cdot I_k(q_{k\ell}^{ji}(\gamma),\gamma) - w_k \Pr\left[P_k \le \gamma + q_{k\ell}^{ji}(\gamma) \,|\, P_k > \gamma\right] \cdot I_j(q_{ji},y_{ji})\right) \\
&\le 0\,.
\end{aligned}$$

The final inequality holds because $R_k(q_{k\ell}^{ji}(\gamma),\gamma) \ge R_j(q_{ji},y_{ji})$ if event $\mathscr{E}_{k,ji}(\gamma)$ occurs and thus,

$$\frac{w_k \Pr\left[P_k \le \gamma + q_{k\ell}^{ji}(\gamma) \,|\, P_k > \gamma\right]}{I_k(q_{k\ell}^{ji}(\gamma),\gamma)} \;=\; R_k(q_{k\ell}^{ji}(\gamma),\gamma) \ge\; R_j(q_{ji},y_{ji}) \;=\; \frac{w_j \Pr\left[P_j \le y_{j,i+1} \,|\, P_j > y_{ji}\right]}{I_j(q_{ji},y_{ji})}\,.$$

□

Theorem 13. Gen-Gipp *has no approximation guarantee of* 1.21 *or less for preemptive stochastic scheduling on a single machine.*

Proof. Consider the following deterministic instance with $k+2$ jobs: a high priority job h with unit weight and processing requirement, a low priority job ℓ of length p_ℓ and unit weight and k small jobs of length ε. The job ℓ and the first small job are released at time 0 followed by the remaining small jobs at times $r_j = (j-1)\varepsilon$ for $j = 2,\ldots,k$ and the high priority job is released at time $r_h = p_\ell - 1$. The weights of the small jobs are $w_j = \varepsilon/(p_\ell - (j-1)\varepsilon)$ for $j = 1,\ldots,k$. We choose ε such that all small jobs could be processed until r_h, that is, $\varepsilon = r_h/k = (p_\ell - 1)/k$.

W.l.o.g. we can assume that Gen-Gipp starts processing job ℓ at time 0. Note that the weights of the small jobs are chosen such that the ratio of weight over remaining processing time of job ℓ at the release date of a small job equals the ratio of the newly released small job, and thus Gen-Gipp does not preempt ℓ until time $r_h = p_\ell - 1$, when job h is released and starts processing. After its completion, job j is finished, followed by the small jobs $\ell, \ell-1,\ldots,1$. The value of the achieved schedule is

$$2p_\ell + 1 + \sum_{i=1}^{k}(p_\ell + 1 + i\varepsilon)\frac{\varepsilon}{p_\ell - (k-i)\varepsilon}\,.$$

An optimal policy, instead, first processes all small jobs followed by the high priority job and finishes with the job ℓ. The value of such a schedule is

$$\sum_{i=1}^{k} i\varepsilon \frac{\varepsilon}{p_\ell - (i-1)\varepsilon} + 3p_\ell\,.$$

If the number of small jobs, k, tends to infinity then the performance ratio of Gen-Gipp is no less than

$$\frac{p_\ell(3-\log\frac{1}{p_\ell-1}+\log\frac{p_\ell}{p_\ell-1})}{1+2p_\ell+p_\ell\log p_\ell},$$

which gives a lower bound of 1.21057 for $p_\ell \approx 5.75$. □

This rather large gap between lower and upper bounds raises hope that the true approximation ratio of Gen-Gipp is less than 2, in which case it would improve on the performance of F-Gipp. We underpin this conjecture further with the arguments that Gen-Gipp adapts more dynamically to the actual job rank. Moreover, it solves deterministic problem instances with equal weights optimally; in such cases, Gen-Gipp coincides with Schrage's [28] optimal SRPT rule. In contrast, F-Gipp fails to be optimal as simple examples show.

5.2 A randomized extension to parallel machines

In this section we derive a randomized policy for multiple machines that utilizes the single-machine policy Gen-Gipp in a straightforward way. It yields again an approximation guarantee of 2.

Algorithm 5: Randomized Gittins Index Priority Policy (Rand-Gipp)

Assign a job at its release date to any of the m machines by choosing one with probability $1/m$. On each of the machines, run Gen-Gipp for the set of jobs assigned to it

Theorem 14. Rand-Gipp *is a* 2*-approximation for the preemptive scheduling problem on parallel machines,* $P\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_jC_j]$.

Proof. The policy uses the single machine policy Gen-Gipp and parts of the performance analysis from the previous section can also be recycled. Therefore, we avoid repeating rather complex terms and ask the reader to follow the references.

Consider a realization $p \in \Omega$ of processing times and a job j. Denote by $j \to m_x$ that job j is assigned to machine m_x in the considered realization. Since the single machine policy Gen-Gipp runs on each machine m_x, the completion time of job j, given that it is processing on machine m_x, is given in Corollary 9 with a minor modification for our current setting, that is,we sum only over jobs that are assigned to the same machine m_x as job j. We denote the corresponding value by $(6)' + (7)' + (8)'$. Thus, the expected completion time of j over all realizations is

$$\begin{aligned}
\mathbb{E}\left[C_j^{\text{Rand-Gipp}}\,\middle|\, j\to m_x\right] &\le r_j + \mathbb{E}\left[\,(6)'+(7)'+(8)'\,|\,j\to m_x\right]\\
&\le r_j + \sum_k \Pr[k\to m_x\,|\,j\to m_x]\cdot\mathbb{E}\left[\,(6)+(7)+(8)\,|\,j\to m_x\right]\\
&= r_j + \sum_k \Pr[k\to m_x\,|\,j\to m_x]\cdot\mathbb{E}\left[\,(6)+(7)+(8)\,\right].
\end{aligned}$$

Unconditioning the expected completion time from the fixed machine assignment and using the fact that all jobs are assigned to m_x with the same probability $1/m$, independently of each other, yields

$$\begin{aligned}
\mathbb{E}\left[C_j^{\text{Rand-Gipp}}\right] &= \sum_{x=1}^{m}\Pr[j\to m_x]\cdot\mathbb{E}\left[C_j^{\text{Rand-Gipp}}\,\middle|\, j\to m_x\right]\\
&\le \sum_{x=1}^{m}\Pr[j\to m_x]\left(r_j+\sum_k\Pr[k\to m_x\,|\,j\to m_x]\,\mathbb{E}\left[(6)+(7)+(8)\right]\right)\\
&\le r_j+\frac{1}{m}\,\mathbb{E}\left[\,(6)+(7)+(8)\,\right].
\end{aligned}$$

The total expected value of the schedule is then

$$\begin{aligned}
\mathbb{E}[\textsf{Rand-Gipp}] &= \sum_{j\in J} w_j \mathbb{E}\left[C_j^{\textsf{Rand-Gipp}}\right] \\
&\leq \sum_{j\in J} w_j r_j + \frac{1}{m}\sum_{j\in J} w_j \mathbb{E}[\,(6)+(7)+(8)\,] \\
&\leq \sum_{j\in J} w_j r_j + \frac{1}{m}\sum_{j\in J} w_j \mathbb{E}\left[C_j^{\textsf{Gipp}}\right] \\
&\leq 2\cdot\mathbb{E}[\textsf{Opt}]\,.
\end{aligned}$$

The second inequality follows from Lemma 10 and Theorem 11. Finally, the third inequality is derived from the trivial lower bound on the optimum $\mathbb{E}[\textsf{Opt}] \geq \sum_{j\in J} w_j r_j$ and from the bound in Corollary 6. □

6 Stochastic online scheduling

We consider now the preemptive stochastic scheduling problem in an online environment where jobs are not known in advance but arrive online over time. We argue that the results in the previous sections on traditional stochastic (offline) scheduling transfer to the more general stochastic online scheduling model. In this model, an online policy is compared to an optimal offline policy in the traditional stochastic scheduling model. Therefore, the Gittins index-based lower bound on an expected optimal value in Section 3 still holds in this generalized model. Further, notice that the presented stochastic policies are feasible policies in the online model where jobs arrive over time. They employ Gipp in different ways; roughly speaking, their decisions are based 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, Gipp itself and each of the proposed extensions, F-Gipp, Gen-Gipp, and Rand-Gipp, are not only non-anticipatory, 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. Thus, Theorem 5.1 directly implies the following result.

Corollary 15. Gen-Gipp *is a* 2*-approximation for the online version of the single-machine problem* $1\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$.

Furthermore, from Theorems 7 and 14, we immediately obtain the following result.

Corollary 16. *Consider the online version of the stochastic scheduling problem* $\mathrm{P}\,|\,r_j, pmtn\,|\,\mathbb{E}[\sum w_j C_j]$ *with individual release dates on parallel machines. Both policies,* F-Gipp *and* Rand-Gipp*, yield an approximation ratio of* 2.

This performance guarantee of 2 matches the best result currently known in deterministic online scheduling on parallel machines for deterministic algorithms by Megow and Schulz [18] and nearly the guarantee of $2-1/m$ for randomized algorithms by Correa and Wagner [7], even though we consider a more general model.

Thus, our algorithms are not only the first approximations for preemptive stochastic scheduling with non-trivial release dates and/or multiple machines, but they also imply that one does not give up (provable) performance by using general stochastic online policies for solving a purely online or stochastic problem instance.

References

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

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

[3] D. Chazan, A. G. Konheim, and B. Weiss. A note on time sharing. *Journal of Combinatorial Theory*, 5:344–369, 1968.

[4] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. *SIAM Journal on Computing*, 31:146–166, 2001.

[5] C.-F. M. Chou, H. Liu, M. Queyranne, and D. Simchi-Levi. On the asymptotic optimality of a simple on-line algorithm for the stochastic single machine weighted completion time problem and its extensions. *Operations Research*, 54(3):464–474, 2006.

[6] E. G. Coffman Jr., M. Hofri, and G. Weiss. Scheduling stochastic jobs with a two point distribution on two parallel machines. *Probability in the Engineering and Informational Sciences*, 3:89–116, 1989.

[7] J. Correa and M. Wagner. LP-based online scheduling: From single to parallel machines. *Mathematical Programming*, 119:109–136, 2009.

[8] B. C. Dean. *Approximation Algorithms for Stochastic Scheduling Problems*. PhD thesis, Massachusetts Institute of Technology, 2005.

[9] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, 41:148–177, 1979.

[10] J. C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley, New York, 1989.

[11] K. Glazebrook. Scheduling tasks with exponential service times on parallel processors. *Journal of Applied Probability*, 16:658–689, 1979.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[13] T. Kämpke. Optimal scheduling of jobs with exponential service times on identical parallel processors. *Operations Research*, 37(1):126–133, 1989.

[14] A. G. Konheim. A note on time sharing with preferred customers. *Probability Theory and Related Fields*, 9:112–130, 1968.

[15] J. Labetoulle, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Preemptive scheduling of uniform machines subject to release dates. In W. R. Pulleyblank, editor, *Progress in Combinatorial Optimization*, pages 245–261. Academic Press Canada, Waterloo, ON, Canada, 1984.

[16] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:243–362, 1977.

[17] N. Megow. *Coping with Incomplete Information in Scheduling – Stochastic and Online Models*. PhD thesis, Technische Universität Berlin, 2006. Published: Cuvillier Verlag, Göttingen, Germany, 2007.

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

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

[20] N. Megow and T. Vredeveld. Stochastic online scheduling with precedence constraints. Submitted, 2009.

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

[22] R. H. Möhring, F. J. Radermacher, and G. Weiss. Stochastic scheduling problems II: Set strategies. *Zeitschrift für Operations Research*, 29(3):A65–A104, 1985.

[23] 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.

[24] M. Pinedo. Stochastic scheduling with release dates and due dates. *Operations Research*, 31:559–572, 1983.

[25] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, third edition, 2008.

[26] K. R. Pruhs, J. Sgall, and E. Torng. Online scheduling. In J. Y.-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 15. Chapman & Hall/CRC, 2004.

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

[28] L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:687–690, 1968.

[29] A. S. Schulz. Stochastic online scheduling revisited. In B. Yang, D.-Z. Du, and C. A. Wang, editors, *Proceedings of the 2nd International Conference on Combinatorial Optimization and Applications*, volume 5165 of *Lecture Notes in Computer Science*, pages 448–457. Springer, 2008.

[30] A. S. Schulz and M. Skutella. The power of α-points in preemptive single machine scheduling. *Journal of Scheduling*, 5:121–133, 2002.

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

[32] K. C. Sevcik. Scheduling for minimum total loss using service time distributions. *Journal of the ACM*, 21:65–75, 1974.

[33] J. Sgall. On-line scheduling – a survey. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 196–231. Springer, Berlin, 1998.

[34] R. A. Sitters. *Complexity and Approximation in Routing and Scheduling*. PhD thesis, Technische Universiteit Eindhoven, 2004.

[35] M. Skutella and M. Uetz. Stochastic machine scheduling with precedence constraints. *SIAM Journal on Computing*, 34(4):788–802, 2005.

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

[37] R. R. Weber. Scheduling jobs with stochastic processing requirements on parallel machines to minimize makespan or flow time. *Journal of Applied Probability*, 19:167–182, 1982.

[38] G. Weiss. On almost optimal priority rules for preemptive scheduling of stochastic jobs on parallel machines. *Advances in Applied Probability*, 27:827–845, 1995.

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