STOCHASTIC MACHINE SCHEDULING WITH PRECEDENCE CONSTRAINTS*

MARTIN SKUTELLA‡ AND MARC UETZ§

Abstract. We consider parallel, identical machine scheduling problems where the jobs are subject to precedence constraints, release dates, and the processing times of jobs are governed by independent probability distributions. The objective is to minimize the expected value of the total weighted completion time. Building upon a linear programming relaxation by Möhring, Schulz, and Uetz (J. ACM 46, 1999, 924–942) and a delayed list scheduling algorithm by Chekuri, Motwani, Natarajan, and Stein (SIAM J. Comput. 31, 2001, 146–166), we derive the first constant-factor approximation algorithms for this model.

Key words. approximation algorithms, stochastic scheduling, parallel machines, precedence constraints, release dates, list scheduling algorithms, LP relaxation

AMS subject classifications. 68M20, 68Q25, 68W25, 68W40, 90B36, 90C05

1. Introduction.

This paper addresses stochastic parallel machine scheduling problems with the objective to minimize the total weighted completion time in expectation. Machine scheduling problems have attracted researchers for decades since they play an important role in various applications from the areas of operations research, management science, and computer science. The total weighted completion time objective is of particular importance in scheduling environments where many jobs are to be scheduled on a limited number of machines, and a good average performance is desired. Prominent examples for such a scheduling situation are problems that arise, e.g., in compiler optimization [4] and in parallel computing [2]. The main characteristic of *stochastic* scheduling problems is the fact that the processing times of the jobs may be subject to random fluctuations. Hence, the effective processing times are not known with certainty in advance. This assumption is of particular practical relevance in many applications.

Problem Definition. Denote by $V = \{1, \ldots, n\}$ a set of jobs which must be scheduled on m parallel, identical machines. Each machine can handle only one job at a time, and the jobs can be scheduled on any of the machines. Once the processing of a job is started on one machine, it must be processed without preemption on this machine. Precedence constraints are given by an acyclic digraph $G = (V, A)$, where any arc $(i, j) \in A$ restricts the start time of job j to be not earlier than the completion time of job i. We consider problems with and without release dates r_j for the jobs, with the intended meaning that job j must not start earlier than r_j. In the classical (deterministic) setting, the objective is to minimize the total weighted completion time

*This work was done while both authors were with Technische Universität Berlin, Germany. An extended abstract [20] appeared in the Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2001.

‡Max-Planck Institut für Informatik, Stuhlsatzenhausweg 85, D-66123 Saarbrücken, Germany (skutella@mpi-sb.mpg.de). Supported in part by the EU Thematic Networks APPOL I+II, Approximation and Online Algorithms, IST-1999-14084 and IST-2001-30012, and by the DFG Research Center 'Mathematics for key technologies: Modelling, simulation and optimization of real-world processes'.

§Faculty of Economics and Business Administration, Quantitative Economics, Universiteit Maastricht, NL-6200 MD Maastricht, The Netherlands (m.uetz@ke.unimaas.nl). Supported in part by the German-Israeli Foundation for Scientific Research and Development (GIF), grant I 246-304.02/97, and by the Deutsche Forschungsgemeinschaft (DFG), grant Mo 446/3-4.

$\sum_{j\in V} w_j\, C_j$, where w_j is a non-negative weight and C_j denotes the completion time of job j. In the stochastic model, it is assumed that the processing time p_j of a job j is not known in advance. It becomes known only upon completion of the job. However, the distribution of the corresponding random variable P_j is given beforehand. Let $P = (P_1, \ldots, P_n)$ denote the vector of random variables for the processing times, and denote by $p = (p_1, \ldots, p_n)$ a particular realization of the processing times. By $\mathrm{E}[P_j]$ we denote the expected processing time of a job j. We assume throughout that the processing times of the jobs are stochastically independent. In the classical $\alpha \,|\, \beta \,|\, \gamma$ notation of Graham et al. [9], the problem of minimizing the expected total weighted completion time can be denoted by $\mathrm{P}|\, \mathit{prec}, r_j|\mathrm{E}\,[\sum w_j\, C_j]$. Here, P stands for the parallel machine environment, *prec* and r_j for the existence of precedence constraints and release dates, respectively, and $\mathrm{E}\,[\sum w_j\, C_j]$ denotes the objective to minimize the expected total weighted completion time.

Dynamic view on stochastic scheduling. The twist from deterministic to stochastic processing times changes the nature of the scheduling problem considerably. The solution of a stochastic scheduling problem is no longer a simple schedule, but a so-called *scheduling policy*. We adopt the notion of scheduling policies as defined by Möhring, Radermacher, and Weiss [14]. In the following, we briefly summarize what that means.

Apart from the data that specifies the input of the problem, the *state* of the system at any time $t \geqslant 0$ is determined by the time t itself, as well as the (conditional) probability distributions of the jobs' processing times. At any time $t > 0$, the state thus depends on the observed *past* up to time t. This is the start and completion times of the jobs already completed by t, together with the start times of the jobs in process at time t. The *action* of a scheduling policy at time t is given by a set of jobs $B(t) \subseteq V$ that is started at t, together with a tentative next decision time $t_{\mathrm{tent}} > t$. The tentative decision time t_{tent} is the latest point in time when the next action of the policy takes place, subject to the condition that no other job is released or ends before t_{tent}. Notice that $B(t)$ may be empty, and $t_{\mathrm{tent}} = \infty$ implies that the next action of the policy takes place when the next job is released or some job ends, whatever occurs first. Of course, the definition of $B(t)$ must respect potential release dates and precedence constraints, and the number of available machines. A policy is required to be *non-anticipatory*, meaning that the action of a policy at any time t must only depend on the state of the system at time t (together with the given input data, of course). The time instances when a policy takes its actions are called *decision times*. Given an action of a policy at a decision time t, the next decision time is t_{tent}, or the time of the next job completion, or the time when the next job is released, whatever occurs first. Depending on the action of the policy, the state at the next decision time is realized according to the (conditional) probability distributions of the jobs' processing times.

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

Approximation. It follows from simple examples that, in general, a scheduling policy cannot yield the optimal schedule for each possible realization of the processing times, see e.g. [21]. Hence, our goal is to find a policy Π which minimizes the objective, say $Z^{\Pi}(P)$, in expectation. But even under this mild notion of optimality, few spe-

cial cases exist for which optimal scheduling policies are known to be efficiently computable. One example is the optimality of list scheduling according to SEPT (shortest expected processing time first) for the problem without precedence constraints or release dates, with unit weights, and with exponentially distributed processing times, $\mathrm{P}|\, p_j \sim \exp(\lambda_j)|\mathrm{E}\left[\sum C_j\right]$ [1, 24]. This result was extended by Kämpke [12] to the case where the weights w_j are compliant with the expected processing times. In general, however, there exist examples which show that optimal policies can be rather complicated in the sense that they must indeed utilize the full information on the conditional distributions of the jobs' processing times, see e.g. [22]. In this paper, we therefore concentrate on approximation algorithms. In stochastic scheduling, a scheduling policy Π is said to be an α-*approximation* if its expected performance $\mathrm{E}[Z^{\Pi}(P)]$ is within a factor of α of the expected performance $\mathrm{E}[Z^{\Pi^*}(P)]$ of an optimal (non-anticipatory) scheduling policy Π^*. The value α is called the *performance guarantee*.

List scheduling policies. There exist essentially three different classes of list scheduling policies, all of which have in common that there is a fixed priority list L of the jobs which determines the order in which the jobs are considered. We call a job j *available* with respect to a partial schedule at time t if all predecessors of j are completed by t and if $r_j \leqslant t$.

Graham's list scheduling. This is perhaps the most natural class of policies, often referred to as *the* list scheduling algorithm of Graham [7]. Iterating over decision times, it greedily starts as many available jobs as possible, always in the order of the list L. It always holds that $t_{\mathrm{tent}} = \infty$, and jobs are thus started only at release dates or upon completion of other jobs. If precedence constraints or release dates exist, it may happen that the order of start times of jobs differs from the order of the jobs in the priority list L; the jobs are scheduled 'out of order' with respect to the priority list L. For the deterministic problem with *makespan* objective, $\mathrm{P}|prec|C_{\max}$, it is well known that Graham's list scheduling achieves a performance guarantee of $2 - 1/m$ for any priority list of the jobs [7]. This result straightforwardly extends to stochastic processing times and the expected makespan objective, $\mathrm{P}|\, prec|\mathrm{E}\left[C_{\max}\right]$ [3]. For the expected total weighted completion time, Graham's list scheduling in the WSEPT order[1] yields a constant-factor approximation for the problem without precedence constraints or release dates, $\mathrm{P}||\mathrm{E}\left[\sum w_j C_j\right]$ [15]. In the presence of release dates or precedence constraints, even in the deterministic setting, there are examples which show that the performance of Graham's algorithm can be arbitrarily bad. For an example with precedence constraints, see [18].

Job-based list scheduling. This is in fact the same list scheduling policy as before, only with the additional constraint that no job is started earlier than any of its predecessors in the priority list L. Hence, this policy preserves the order of the jobs in the priority list L, at the cost of deliberate idle times on the machines. For the deterministic problem $\mathrm{P}|r_j, prec|\sum w_j C_j$, the currently best known performance guarantee of 4 relies on (a slight variation of) job-based list scheduling, and the priority list is defined on the basis of an optimal solution to an LP-relaxation [16]. For the stochastic problem without precedence constraints, $\mathrm{P}|r_j|\mathrm{E}\left[\sum w_j C_j\right]$, job-based list scheduling yields a constant performance guarantee, too. This result is also based on a priority list that is defined on the basis of an optimal solution to an LP-relaxation [15].

Delayed list scheduling. As a matter of fact, approximation results for stochastic parallel machine scheduling were previously known only for problems without prece-

[1] In the WSEPT order, jobs appear in non-increasing order of the ratios $w_j/\mathrm{E}[P_j]$.

dence constraints [23, 15]. In this paper, we close this gap, relying on yet another class of list scheduling policies which generalizes both Graham's and job-based list scheduling. It has been suggested in a paper by Chekuri et al. [5] to obtain a 5.828-approximation for the deterministic problem $\mathrm{P}|r_j, prec|\sum w_j C_j$. We consider the analogous stochastic variant of this algorithm. The basic idea is to extend Graham's list scheduling in such a way that a job may be scheduled out of order only if a certain amount of deliberate idle time has accumulated before. Thus, in this algorithm we use values $t_{\text{tent}} < \infty$, and jobs may be started at times different from release dates or completion times of other jobs. The algorithm is parametric on a parameter $\beta \geqslant 0$ that controls the tradeoff between the amount of 'out of order' processing of jobs and the desire to adhere to the order of the given priority list L. For $\beta = 0$ and $\beta = \infty$ we get Graham's and job-based list scheduling, respectively. The algorithm will be described in more detail in § 2.

Contribution of this paper. We derive the first constant performance guarantees for stochastic parallel machine scheduling with precedence constraints. The results are derived by heavily borrowing from two previous approaches. On the one hand, we use (an appropriate adaption of) the delayed list scheduling algorithm of Chekuri, Motwani, Natarajan, and Stein [5]. On the other hand, the priority list is derived from an optimal solution for (a generalized version of) the LP-relaxation by Möhring, Schulz, and Uetz [15]. It seems, however, that only this combination of the previous techniques is capable of yielding the desired approximation results.

Table 1 gives an overview of performance guarantees for stochastic parallel machine scheduling problems with the total weighted completion time objective. The last column indicates which of the results are proved in [15]; the asterisk [∗] indicates that the results are derived in this paper. The term Δ denotes some common upper bound on the values $\mathrm{Var}[P_j]/(\mathrm{E}[P_j])^2$, for all jobs $j \in V$. In other words, $\sqrt{\Delta}$ is a common upper bound on the coefficient of variation $\mathrm{CV}[P_j] = \sqrt{\mathrm{Var}[P_j]}/\mathrm{E}[P_j]$ for all processing time distributions P_j, $j \in V$. Moreover, the number of machines is denoted by m, and β is the non-negative parameter used to control the delayed list scheduling algorithm. The third column shows the respective performance bounds for processing time distributions where the coefficient of variation is bounded by 1, which is the case for exponential, uniform, or Erlang distributions, to name a few.

TABLE 1
Performance bounds for stochastic scheduling problems. Asterisks [∗] mark results of this paper.

scheduling model	performance guarantee		
	arbitrary P_j	$\mathrm{CV}[P_j] \leqslant 1$	
$1\|prec\|\mathrm{E}[\sum w_j C_j]$	2	2	[15]
$1\|r_j, prec\|\mathrm{E}[\sum w_j C_j]$	3	3	[15]
$\mathrm{P}\|\,\|\mathrm{E}[\sum w_j C_j]$	$1 + \frac{(m-1)(\Delta+1)}{2m}$	$2 - \frac{1}{m}$	[15]
$\mathrm{P}\|r_j\|\mathrm{E}[\sum w_j C_j]$	$3 - \frac{1}{m} + \max\{1, \frac{m-1}{m}\Delta\}$	$4 - \frac{1}{m}$	[15]
$\mathrm{P}\|in\text{-}forest\|\mathrm{E}[\sum w_j C_j]$	$2 - \frac{1}{m} + \max\{1, \frac{m-1}{m}\Delta\}$	$3 - \frac{1}{m}$	[∗]
$\mathrm{P}\|prec\|\mathrm{E}[\sum w_j C_j]$	$(1+\beta)\big(1 + \frac{m-1}{m\,\beta} + \max\{1, \frac{m-1}{m}\Delta\}\big)$	$3 + 2\sqrt{2} - \frac{1+\sqrt{2}}{m}$	[∗]
$\mathrm{P}\|r_j, prec\|\mathrm{E}[\sum w_j C_j]$	$(1+\beta)\big(1 + \frac{1}{\beta} + \max\{1, \frac{m-1}{m}\Delta\}\big)$	$3 + 2\sqrt{2}$	[∗]

Relations to online optimization and other models. Compared to the model described above, *on-line optimization* is another way of coping with the fact that the future is uncertain. We refer to Fiat and Woeginger [6] for details on online optimization. There is, however, a significant difference between the underlying paradigms of the above described analysis and the usual competitive analysis that is prevailing in on-line optimization. First, competitive analysis is based upon the ex-post comparison *'What was achieved under uncertainty about the future, and what could have been achieved if the future would not have been uncertain?'*. This is expressed by the fact that the *adversary* is generally an oracle that knows the optimal solution. In contrast, stochastic scheduling addresses the ex-ante question *'What is the best that can be achieved under the given uncertainty about the future?'*. Here, the underlying adversary is much weaker: the adversary must not anticipate future information, just like the policy itself. Second, in competitive analysis the adversary is allowed to determine, to a certain extent, the input distribution. This is not the case in the stochastic model considered here, since the input distributions are considered fixed. It is interesting to note that two generalized on-line frameworks were suggested by Koutsoupias and Papadimitriou [13]. They restrict the adversary's power in two ways: Its ability to choose an input distribution, and its ability to find an optimal solution. To some extent, the stochastic scheduling model incorporates both ideas, too. We refer to [13] for details, and to the PhD thesis [21] for a brief discussion. Another type of analysis for stochastic models has been proposed recently by Scharbrodt, Schickinger, and Steger [17]. If $Z^{\mathrm{OPT}}(p)$ is the optimal solution value for a realization p, they analyze the *expected competitive ratio* $\mathrm{E}[Z^{\Pi}(P)/Z^{\mathrm{OPT}}(P)]$. In this type of analysis the adversary is again an oracle that knows the optimal solution. We refer to [17] for a more detailed discussion of the benefits of their approach in comparison to the approach of this paper.

2. List scheduling with deliberate idle times. We start with a few preliminaries that will be used later in the analysis. First, recall that we call a job j *available* with respect to a partial schedule at time t if all predecessors of j are completed by t and if $r_j \leqslant t$.

ASSUMPTION 2.1. *For any instance of* $\mathrm{P}|r_j, prec|\,\gamma$, *assume that* $r_j \geqslant r_i$ *whenever job* i *is a predecessor of job* j *in the precedence constraints.*

(Here, γ is used to denote an arbitrary objective function.) Obviously, Assumption 2.1 can be made without loss of generality. Additionally, we use the following definitions.

DEFINITION 2.2 (critical predecessor). *Let some realization* p *of the processing times and a feasible schedule be given. For any job* j, *a critical predecessor of* j *is a predecessor* i *of* j (*with respect to the precedence constraints*) *with* $C_i > r_j$ *and* C_i *maximal among all predecessors.*

DEFINITION 2.3 (critical chain). *Let some realization* p *of the processing times and a feasible schedule be given. For a given job* j, *a critical chain for job* j *and its length* $\ell_j(p)$ *is defined backwards recursively: If* j *has no critical predecessor,* j *is the only job in the critical chain, and* $\ell_j(p) = r_j + p_j$. *Otherwise,* $\ell_j(p) = p_j + \ell_k(p)$, *where job* k *is a critical predecessor of job* j.

Definition 2.3 is illustrated in Figure 1. Notice that the critical chain as well as its length $\ell_j(p)$ depend on both the realization of the processing times p and the underlying schedule. Moreover, since a critical predecessor is not necessarily unique, the critical chain and its length also depend on a tie-breaking rule for choosing critical predecessors. This is not relevant for our analysis, but in order to make the above

definition unique, let us suppose that some arbitrary but fixed tie-breaking rule is used. Notice further that the first job j_1 of a critical chain is available at its release date r_{j_1}. This follows directly from the definition.

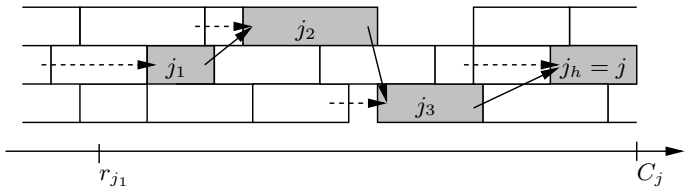

FIGURE 1. *Example of a critical chain for job j. Its length is $\ell_j(p) = r_{j_1} + \sum_{i=1}^{h} p_{j_i}$.*

Like Graham's list scheduling, the algorithm we use iterates over decision times until all jobs have been scheduled. Assume a priority list L is given. Like with job-based list scheduling, the algorithm strives to schedule the jobs in the order of the list L by leaving deliberate idle times. But if the accumulating deliberate idle time exceeds a certain threshold, the algorithm 'panics' and schedules the first available job from the list. The algorithm is parametric on a parameter $\beta \geqslant 0$ that controls the tradeoff between the amount of 'out of order' processing of jobs, and the desire to adhere to the order of the given priority list L. At each stage of the algorithm, the sub-list of L containing all jobs that are not scheduled yet is referred to as the *residual list*. The following is a direct adaptation of the algorithm introduced by Chekuri, Motwani, Natarajan, and Stein [5].

> **Algorithm CMNS[2].** Whenever a machine is idle and the first job in the residual list is available, the job is scheduled. Otherwise, if the first job is not available, the first available job j in the residual list (if any) is *deliberately delayed*. If j was deliberately delayed for an accumulated time of $\beta \mathrm{E}[P_j]$, it is scheduled *out of order*.

We emphasize that *deliberate idle time* accumulates m' times faster when a job is deliberately delayed while m' machines are idle. For the purpose of analyzing the performance of Algorithm CMNS, any job j gets *charged* the deliberate idle time that accumulates during time intervals when j is deliberately delayed. An alternative interpretation is the following: whenever a job j is deliberately delayed, the tentative next decision time t_{tent} is that point in time where the accumulated deliberate idle time charged to job j would equal $\beta \mathrm{E}[P_j]$.

Analogous to [5], we introduce some additional notation. For a given job j, denote by B_j and A_j the sets of jobs that come before and after job j in the priority list L, respectively; by convention, B_j also includes job j. For the remaining definitions, we consider a fixed realization p of the processing times and the resulting schedule constructed by Algorithm CMNS. Then, $r_j(p) \geqslant r_j$ denotes the earliest point in time when job j becomes available; let $j_1, j_2, \ldots, j_h = j$ be the critical chain for job j and define $B_j(p) := B_j \setminus \{j_1, \ldots, j_h\}$. That is, the set $B_j(p)$ contains all jobs that come before job j in the priority list L, except for those which belong to the critical chain. Moreover, let $O_j(p) \subseteq A_j$ be the jobs in A_j that are started out of order, that is, before j.

[2]The only difference from the algorithm presented in [5] is the use of the threshold $\beta \mathrm{E}[P_j]$ instead of βp_j. The algorithm coincides with the classical list scheduling algorithm of Graham [7] if we choose $\beta = 0$, and it coincides with the job-based list scheduling algorithm if we choose $\beta = \infty$.

The following observation is the analog to the results for the deterministic setting by Chekuri et al. [5, Fact 4.6 & Lemma 4.7].

OBSERVATION 2.4. *For any realization p of the processing times and any job j:*

(i) job j is charged no more than $\beta\,\mathrm{E}[P_j]$ deliberate idle time;
(ii) the deliberate idle time in $[r_j(p), S_j(p)[$ is charged only to jobs in B_j;
(iii) there is no uncharged deliberate idle time.

Proof. Part (i) follows by construction of the algorithm and part (iii) by definition of deliberate idle time. Finally, for (ii), observe that no job from A_j is the first available job from the residual list in the time interval $[r_j(p), S_j(p)[$, since job j is available from $r_j(p)$ on, and j has higher priority than any job in A_j. □

The following analysis of Algorithm CMNS closely resembles the analysis performed in [5] for the deterministic case. We first derive an upper bound on the completion time of any job for a fixed realization p.

LEMMA 2.5. *Consider the schedule constructed by Algorithm CMNS for any $\beta \geqslant 0$, any realization p of the processing times, and any priority list L which is a linear extension of the precedence constraints. Let $C_j(p)$ denote the resulting completion time of any job j, and let $\ell_j(p)$ denote the length of the critical chain for job j. Then*

$$C_j(p) \;\leqslant\; \frac{m-1}{m}\,\ell_j(p) \;+\; \frac{1}{m}\,r_j \;+\; \frac{1}{m}\left(\sum_{i\in B_j}\left(p_i + \beta\,\mathrm{E}[P_i]\right) \;+\; \sum_{i\in O_j(p)} p_i\right). \qquad (2.1)$$

Proof. The basic idea resembles Graham's analysis for the makespan objective [7]. Consider the critical chain for job j with total length $\ell_j(p)$, consisting of jobs $j_1, j_2, \ldots, j_h = j$. Now partition the interval $[r_{j_1}, C_j(p)[$ into time intervals where some job from the critical chain is in process and the remaining time intervals. The latter are exactly $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$. (Recall that $r_{j_1} = r_{j_1}(p)$ due to the definition of the critical chain). By definition,

$$C_j(p) \;=\; \ell_j(p) + \sum_{i=j_1}^{j_h}\big(S_i(p) - r_i(p)\big)\,. \qquad (2.2)$$

To bound the total length of the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$, observe that in each of these intervals there is no idle time except (possibly) deliberate idle time, since job i is available in $[r_i(p), S_i(p)[$. Hence, the total processing in these intervals can be partitioned into three categories:

– processing of jobs from B_j which do not belong to the critical chain, i.e., jobs in $B_j(p)$,
– deliberate idle time,
– processing of jobs from A_j which are scheduled 'out of order', i.e., jobs in $O_j(p)$.

Due to Observation 2.4 (ii), all deliberate idle time in the interval $[r_i(p), S_i(p)[$ is charged only to jobs in B_i, $i = j_1, \ldots, j_h$. Since the priority list L is a linear extension of the precedence constraints, we have $B_{j_1} \subset B_{j_2} \subset \cdots \subset B_{j_h} = B_j$. Hence, all deliberate idle time in the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$, is charged only to jobs in B_j. Since there is no uncharged deliberate idle time (Observation 2.4 (iii)), and since each job $i \in B_j$ gets charged no more than $\beta\,\mathrm{E}[P_i]$ idle time (Observation 2.4 (i)), the total amount of deliberate idle time in the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$,

is bounded from above by $\beta \sum_{i \in B_j} \mathrm{E}[P_i]$. This yields

$$\sum_{i=j_1}^{j_h} \big(S_i(p) - r_i(p)\big) \;\leqslant\; \frac{1}{m} \left(\sum_{i \in B_j(p)} p_i + \sum_{i \in B_j} \beta\, \mathrm{E}[P_i] + \sum_{i \in O_j(p)} p_i \right) \;. \tag{2.3}$$

Finally, due to Assumption 2.1 we have $r_{j_1} \leqslant r_j$, thus

$$\sum_{i \in B_j(p)} p_i \;\leqslant\; \sum_{i \in B_j} p_i - \big(\ell_j(p) - r_j\big) \;. \tag{2.4}$$

Now put (2.4) into (2.3), and then (2.3) into (2.2), and the claim follows. □

Before we take expectations in (2.1), we concentrate on the term $\sum_{i \in O_j(p)} p_i$. The following lemma shows that the expected total processing of the jobs in $O_j(p)$ — the jobs that are scheduled out of order with respect to j (and p) — is independent of their actual processing times.

LEMMA 2.6. $$\mathrm{E}\Big[\sum_{i \in O_j(P)} P_i \Big] = \mathrm{E}\Big[\sum_{i \in O_j(P)} \mathrm{E}[P_i] \Big] .$$

Proof. We can write $\sum_{i \in O_j(P)} P_i$ equivalently as $\sum_{i \in A_j} \delta_i(P)\, P_i$, where $\delta_i(P)$ is a binary random variable which is 1 if and only if $i \in O_j(p)$. Linearity of expectation yields

$$\mathrm{E}\Big[\sum_{i \in O_j(P)} P_i \Big] \;=\; \mathrm{E}\Big[\sum_{i \in A_j} \delta_i(P)\, P_i \Big] \;=\; \sum_{i \in A_j} \mathrm{E}[\delta_i(P)\, P_i] \;.$$

Notice that $\delta_i(P)$ is dependent on $\beta\mathrm{E}[P_i]$ but stochastically independent of P_i as the decision to process job i out of order is made before it is actually processed. (Here we require that the processing times are stochastically independent, and that policies are non-anticipatory.) Hence,

$$\sum_{i \in A_j} \mathrm{E}[\delta_i(P) P_i] \;=\; \sum_{i \in A_j} \mathrm{E}[\delta_i(P)] \mathrm{E}[P_i] \;=\; \sum_{i \in A_j} \mathrm{E}\big[\delta_i(P) \mathrm{E}[P_i]\big] \;=\; \mathrm{E}\Big[\sum_{i \in O_j(P)} \mathrm{E}[P_i] \Big] \;.$$

This concludes the proof. □

The following lemma bounds the expected amount of processing of jobs from A_j which are scheduled 'out of order' in terms of the expected length of the critical chain for job j; compare [5, Lemma 4.8].

LEMMA 2.7. $$\frac{1}{m}\, \mathrm{E}\Big[\sum_{i \in O_j(P)} \mathrm{E}[P_i] \Big] \;\leqslant\; \frac{1}{\beta}\, \mathrm{E}[\ell_j(P)] \;.$$

Proof. Consider a fixed realization p of the processing times. If some job $i \in A_j$ is scheduled out of order, i gets charged exactly $\beta\, \mathrm{E}[P_i]$ deliberate idle time. Hence, the total amount of deliberate idle time in $[0, S_j(p)[$ that is charged to jobs in $O_j(p)$ is $\beta \sum_{i \in O_j(p)} \mathrm{E}[P_i]$. Now consider the critical chain $j_1, \ldots, j_h = j$ for job j with total length $\ell_j(p)$. From the proof of Lemma 2.5, we know that all deliberate idle time in the intervals $[r_i(p), S_i(p)[$, $i = j_1, \ldots, j_h$, is charged only to jobs in B_j. In other words, all deliberate idle time in $[0, S_j(p)[$ that is charged to jobs in A_j lies in the complementary intervals $[0, r_{j_1}[$ and $[S_i(p), C_i(p)[$, $i = j_1, \ldots, j_{h-1}$. (Recall that $r_{j_1} = r_{j_1}(p)$ due to the definition of a critical chain.) The total length of these intervals is exactly $\ell_j(p) - p_j$. Hence, the total amount of deliberate idle time in $[0, S_j(p)[$ that is

charged to jobs in A_j is at most $m\,\ell_j(p)$. (In fact, it is at most $m\,(\ell_j(p) - p_j)$, but this is not essential.) Hence, we obtain $\beta \sum_{i \in O_j(p)} \mathrm{E}[P_i] \leqslant m\,\ell_j(p)$, for any realization p of the processing times. Taking expectations yields the claimed result. □

Finally, we obtain an upper bound on the expected completion time of any job under Algorithm CMNS; compare [5, Theorem 4.9].

THEOREM 2.8. *For any instance of a stochastic scheduling problem* $\mathrm{P}|r_j, prec|\,\gamma$ *and any priority list L which is a linear extension of the precedence constraints, the expected completion time of any job j under Algorithm CMNS (with parameter $\beta \geqslant 0$) fulfills*

$$\mathrm{E}[C_j(P)] \;\leqslant\; \Big(\frac{m-1}{m} + \frac{1}{\beta}\Big)\mathrm{E}[\ell_j(P)] \;+\; \frac{1+\beta}{m} \sum_{i \in B_j} \mathrm{E}[P_i] \;+\; \frac{1}{m}\, r_j\,. \tag{2.5}$$

(Again, γ is used to denote an arbitrary objective function.)

Proof. Taking expectations in (2.1) together with Lemma 2.6 yields

$$\mathrm{E}[C_j(P)] \leqslant \frac{m-1}{m}\mathrm{E}[\ell_j(P)] + \frac{1}{m}\bigg(r_j \;+\; (1+\beta) \sum_{i \in B_j} \mathrm{E}[P_i] \;+\; \mathrm{E}\Big[\sum_{i \in O_j(P)} \mathrm{E}[P_i]\Big]\bigg)\;.$$

Plugging in the inequality from Lemma 2.7 gives the desired result. □

3. Linear programming relaxation. To obtain a priority list L as input for Algorithm CMNS, and to obtain a lower bound on the optimum, Chekuri et al. [5] use a single machine relaxation. This approach does not help in the stochastic setting, since the single machine problem does not necessarily provide a lower bound for the parallel machine problem; see [15, Ex. 4.1] for an example. Instead, we use LP-relaxations which extend those used by Möhring, Schulz and Uetz [15], adding inequalities which represent the precedence constraints. First, define $f : 2^V \to \mathbb{R}$ by

$$f(W) \;:=\; \frac{1}{2m}\Big(\Big(\sum_{j \in W} \mathrm{E}[P_j]\Big)^2 + \sum_{j \in W} \mathrm{E}[P_j]^2\Big) - \frac{(m-1)(\Delta-1)}{2m}\Big(\sum_{j \in W} \mathrm{E}[P_j]^2\Big) \tag{3.1}$$

for $W \subseteq V$. Here, $\Delta \geqslant 0$ is a common upper bound on $\mathrm{Var}[P_j]/\mathrm{E}[P_j]^2$ for all jobs $j \in V$, where $\mathrm{Var}[P_j] = \mathrm{E}[P_j^2] - \mathrm{E}[P_j]^2$ is the variance of P_j. In other words, the *coefficient of variation*

$$\mathrm{CV}[P_j] \;:=\; \frac{\sqrt{\mathrm{Var}[P_j]}}{\mathrm{E}[P_j]}$$

of the distributions P_j is bounded by $\sqrt{\Delta}$, for all $j \in V$. The following *load inequalities* are crucial for the derivation of our results.

THEOREM 3.1 ([15, Cor. 3.1]). *If* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all P_j and some $\Delta \geqslant 0$, the load inequalities*

$$\sum_{j \in W} \mathrm{E}[P_j]\,\mathrm{E}[C_j^{\Pi}(P)] \;\geqslant\; f(W) \tag{3.2}$$

are valid for all $W \subseteq V$ and any non-anticipatory scheduling policy Π.

In fact, as mentioned in [15], an upper bound on the coefficients of variation of the processing time distributions P_j can be a reasonable assumption for many scheduling

problems. For instance, assume that job processing times follow so-called *NBUE distributions*.

DEFINITION 3.2 (NBUE). *A non-negative random variable X is NBUE, 'new better than used in expectation', if* $\mathrm{E}[X - t | X > t] \leqslant \mathrm{E}[X]$ *for all* $t \geqslant 0$.

Here, $\mathrm{E}[X - t | X > t]$ is the conditional expectation of $X - t$ under the assumption that $X > t$. Roughly spoken, when processing times are NBUE, on average it is not disadvantageous to process a job. Examples for NBUE distributions are, among others, exponential, uniform, and Erlang distributions. A result of Hall and Wellner [11] states that the coefficient of variation $\mathrm{CV}[X]$ of any NBUE distribution X is bounded by 1. Hence, by choosing $\Delta = 1$ the second term of the right hand side of (3.2) can be neglected for NBUE distributions, which leads to simplified performance guarantees in Section 4.

Observe that under any scheduling policy Π the trivial inequalities

$$\mathrm{E}[C_j^\Pi(P)] \;\geqslant\; \mathrm{E}[C_i^\Pi(P)] + \mathrm{E}[P_j] \;, \qquad\qquad (i,j) \in A$$

and

$$\mathrm{E}[C_j^\Pi(P)] \;\geqslant\; \mathrm{E}[P_j] \;, \qquad\qquad j \in V$$

are valid, since they even hold point-wise for any realization of the processing times. Due to Theorem 3.1, the following is thus a linear programming relaxation for the problem $\mathrm{P}|r_j, prec|\mathrm{E}[\sum w_j C_j]$.

$$\begin{aligned}
\text{minimize} \quad & \sum_{j \in V} w_j \, C_j^{\mathrm{LP}} \\
\text{subject to} \quad & \sum_{j \in W} \mathrm{E}[P_j] \, C_j^{\mathrm{LP}} \geqslant f(W) \;, && W \subseteq V \,, \\
& C_j^{\mathrm{LP}} \geqslant C_i^{\mathrm{LP}} + \mathrm{E}[P_j] \;, && (i,j) \in A \,, \\
& C_j^{\mathrm{LP}} \geqslant \mathrm{E}[P_j] \;, && j \in V \,,
\end{aligned}$$

where $f : 2^V \to \mathbb{R}$ is the set function defined in (3.1). It is known that the load inequalities $\sum_{j \in W} \mathrm{E}[P_j] \, C_j^{\mathrm{LP}} \geqslant f(W)$, $W \subseteq V$, can be separated in time $\mathrm{O}(n \log n)$ [15, 21]. Hence, due to the fact that the remaining number of inequalities is polynomial in terms of n, this LP-relaxation can be solved in time polynomial in n by the equivalence of separation and optimization [10]. The following technical lemma of Möhring, Schulz, and Uetz [15] is required later in the analysis.

LEMMA 3.3 ([15, Lemma 4.2]). *Let $C^{LP} \in \mathbb{R}^n$ be any point that satisfies the first and the last set of inequalities from the linear programming relaxation. Assuming $C_1^{LP} \leqslant C_2^{LP} \leqslant \cdots \leqslant C_n^{LP}$ we then have*

$$\frac{1}{m} \sum_{k=1}^{j} \mathrm{E}[P_k] \;\leqslant\; \left(1 + \max\{1, \frac{m-1}{m} \Delta\}\right) C_j^{LP}$$

for all $j \in V$.

4. Results. We are now ready to prove approximation results for stochastic machine scheduling problems with precedence constraints.

General precedence constraints. We consider the general problem with precedence constraints and release dates, $\mathrm{P}|r_j, prec|\mathrm{E}\left[\sum w_j\, C_j\right]$. From an optimal solution for the LP-relaxation, we define a priority list L according to non-decreasing 'LP completion times' C_j^{LP}. It is perhaps interesting to note that inequalities $C_j^{\mathrm{LP}} \geqslant C_i^{\mathrm{LP}} + \mathrm{E}[P_j]$, $(i,j) \in A$, are only required to ensure that the order according to nondecreasing LP completion times C_j^{LP} is a linear extension of the precedence constraints. They are not required elsewhere in the analysis. Moreover, instead of the weaker inequalities $C_j^{\mathrm{LP}} \geqslant \mathrm{E}[P_j]$ we could as well use $C_j^{\mathrm{LP}} \geqslant r_j + E[P_j]$, but this does not yield an improvement of our results.

THEOREM 4.1. *Consider an instance of the stochastic machine scheduling problem* $\mathrm{P}|r_j, prec|\mathrm{E}\left[\sum w_j\, C_j\right]$ *with* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all processing times* P_j *and some* $\Delta \geqslant 0$. *Let* L *be a priority list according to an optimal solution* C^{LP} *of the linear programming relaxation. Then Algorithm CMNS (with parameter* $\beta > 0$*) is an* α*-approximation with*

$$\alpha \;=\; (1+\beta)\left(1 + \frac{1}{\beta} + \max\{1, \frac{m-1}{m}\Delta\}\right).$$

Proof. Since L is a linear extension of the precedence constraints, Theorem 2.8 yields

$$\mathrm{E}[C_j(P)] \;\leqslant\; \Big(\frac{m-1}{m} + \frac{1}{\beta}\Big)\mathrm{E}[\ell_j(P)] \;+\; \frac{1+\beta}{m}\sum_{i\in B_j}\mathrm{E}[P_i] \;+\; \frac{1}{m}\,r_j\,,$$

for any job $j \in V$. (Recall that B_j denotes the jobs that come before job j in the priority list L.) Lemma 3.3 yields

$$\frac{1}{m}\sum_{i\in B_j}\mathrm{E}[P_i] \;\leqslant\; \left(1 \;+\; \max\{1, \frac{m-1}{m}\Delta\}\right) C_j^{\mathrm{LP}}\,,$$

for all $j \in V$. Hence,

$$\begin{aligned}\sum_{j\in V} w_j\,\mathrm{E}[C_j(P)] \leqslant\; & \left(\frac{m-1}{m} + \frac{1}{\beta}\right)\sum_{j\in V} w_j\,\mathrm{E}[\ell_j(P)] \\ & + (1+\beta)\left(1 \;+\; \max\{1, \frac{m-1}{m}\Delta\}\right)\sum_{j\in V} w_j\, C_j^{\mathrm{LP}} + \frac{1}{m}\sum_{j\in V} w_j\, r_j\,.\end{aligned}$$

Now, for any job j and any realization p of the processing times, the length $\ell_j(p)$ of a critical chain for job j is a lower bound for job j's completion time, $\ell_j(p) \leqslant C_j(p)$. This is true by definition of a critical chain. Hence, the value $\mathrm{E}[\ell_j(P)]$ is a lower bound on the expected completion time $\mathrm{E}[C_j(P)]$ of any job j, for any scheduling policy. (Notice that the critical chain may be different for different realizations of the processing times, and thus the fact that $\mathrm{E}[\ell_j(P)] \leqslant \mathrm{E}[C_j(P)]$ cannot be derived from the precedence constraints in the LP-relaxation.) Thus, $\sum_{j\in V} w_j\,\mathrm{E}[\ell_j(P)]$ is a lower bound on the expected performance of an optimal scheduling policy. Moreover, both terms $\sum_{j\in V} w_j\, C_j^{\mathrm{LP}}$ and $\sum_{j\in V} w_j\, r_j$ are lower bounds on the expected performance of an optimal scheduling policy as well. This gives a performance bound of

$$\left(\frac{m-1}{m} + \frac{1}{\beta}\right) + (1+\beta)\left(1 \;+\; \max\{1, \frac{m-1}{m}\Delta\}\right) + \frac{1}{m}\;.$$

Rearranging the terms yields the desired result. □

Notice that Theorem 4.1 implies a performance bound of $3 + 2\sqrt{2} \approx 5.828$ if $\beta = 1/\sqrt{2}$ and if the jobs' processing times are distributed according to NBUE distributions (see Definition 3.2). This matches the performance guarantee achieved in [5] for the corresponding deterministic scheduling problem $\mathrm{P}|r_j, prec|\sum w_j C_j$. The performance bound in Theorem 4.1 can be slightly improved if release dates are absent.

THEOREM 4.2. *Consider an instance of the stochastic machine scheduling problem* $\mathrm{P}|prec|\mathrm{E}[\sum w_j C_j]$ *with* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all processing times* P_j *and some* $\Delta \geqslant 0$. *Let* L *be a priority list according to an optimal solution* C^{LP} *of the linear programming relaxation. Then Algorithm CMNS (with parameter* $\beta > 0$*) is an* α*-approximation with*

$$\alpha \;=\; (1+\beta)\left(1 + \frac{m-1}{m\,\beta} + \max\{1, \frac{m-1}{m}\Delta\}\right) \;.$$

The tighter bound follows from two modifications in the proof of Theorem 4.1. On the one hand, in the proof of Lemma 2.7, one can show that

$$\frac{1}{m}\,\mathrm{E}\Bigg[\sum_{i \in O_j(P)} \mathrm{E}[P_i]\Bigg] \;\leqslant\; \frac{m-1}{m\,\beta}\,\mathrm{E}[\ell_j(P)] \;.$$

The reason is that there are only $m-1$ machines available for the deliberate idle time that is charged to jobs which are scheduled out of order: Simultaneous to the deliberate idle time, at least one job from the critical chain $j_1, j_2, \ldots, j_h$ is in process. (This argument does not hold if release dates are present, since deliberate idle time could possibly accumulate before r_{j_1}.) On the other hand, it is immediate that the last term $(1/m)\,r_j$ on the right hand side of (2.5) disappears. With these modifications, the claim follows exactly as in Theorem 4.1.

In-forest precedence constraints. Let us now turn to the special case of the problem denoted by $\mathrm{P}|$*in-forest*$|\mathrm{E}[\sum w_j C_j]$. In-forest precedence constraints are characterized by the fact that each job has at most one successor. Moreover, we assume that there are no release dates. For this problem, the results of the preceding section can be further improved.

We start with the following observation which is also contained in [5, Lemma 4.16]; we nevertheless give a short proof for the sake of completeness.

LEMMA 4.3. *Consider the schedule constructed by Graham's list scheduling for an arbitrary priority list* L *which is a linear extension of the (in-forest) precedence constraints, and any realization* p *of the processing times. Then, in the interval* $[r_j(p), S_j(p)[$ *there is no processing of jobs in* A_j.

Proof. Suppose the claim is false and among all jobs which violate it, let job j be one that is scheduled earliest. Obviously, $S_j(p) > r_j(p)$, otherwise the claim is trivially true. In the interval $[r_j(p), S_j(p)[$ no job from A_j is started, since j is available from time $r_j(p)$ on. Hence, there must be some job $k \in A_j$ that has been started before $r_j(p)$ and that is still in process at $r_j(p)$. Thus $r_j(p) > 0$. Denote by h the number of jobs that are started at time $r_j(p)$. All of these jobs i have higher priority than j, and the fact that j is the first job that violates the claim yields $r_i(p) = r_j(p)$. (At this point it is crucial that the priority list extends the precedence constraints.) In other words, for each of these jobs a critical predecessor ends at time $r_j(p)$, and due to the fact that the precedence constraints form an in-forest, all of these predecessors

are different. Hence, including j's critical predecessor, $h+1$ different jobs end at time $r_j(p)$, but only h are started. This is a contradiction since job j is available at time $r_j(p)$. □

LEMMA 4.4. *For any instance of the stochastic scheduling problem* $\mathrm{P}|\textit{in-forest}|\,\gamma$ *and any priority list L which is a linear extension of the precedence constraints, the expected completion time of any job j under Graham's list scheduling fulfills*

$$\mathrm{E}[C_j(P)] \;\leqslant\; \frac{m-1}{m}\mathrm{E}[\ell_j(P)] + \frac{1}{m}\sum_{i\in B_j}\mathrm{E}[P_i] \;. \tag{4.1}$$

(Again, γ is used to denote an arbitrary objective function.)

Proof. Consider any realization p of the processing times. Given any job j, consider a critical chain for j, consisting of jobs $j_1, j_2, \ldots, j_h = j$ and with total length $\ell_j(p)$. The time interval $[0, C_j(p)]$ can be partitioned into time intervals where a job from a critical chain for j is in process, and the remaining time intervals. Due to Lemma 4.3, in each time interval $[r_i(p), S_i(p)[$ there is no job from A_i in process, for all $i = j_1, \ldots, j_h$. Moreover, there is no idle time on any of the machines in these time intervals (we consider Graham's list scheduling, and there are no release dates). Since $A_{j_1} \supset A_{j_2} \supset \cdots \supset A_{j_h} = A_j$, it follows that the only processing in these time intervals is the jobs in B_j, or more precisely, in $B_j(p)$. In other words, the total processing in these time intervals is at most $\sum_{i\in B_j} p_i - \ell_j(p)$. Hence,

$$C_j(p) \;\leqslant\; \frac{m-1}{m}\ell_j(p) + \frac{1}{m}\sum_{i\in B_j} p_i \;,$$

for any realization p. Taking expectations, the claim follows. □

THEOREM 4.5. *Consider an instance of* $\mathrm{P}|\textit{in-forest}|\mathrm{E}[\sum w_j\, C_j]$ *with* $\mathrm{CV}[P_j] \leqslant \sqrt{\Delta}$ *for all processing times P_j and some $\Delta \geqslant 0$. Let L be a priority list according to an optimal solution C^{LP} of the linear programming relaxation. Then Graham's list scheduling is an α-approximation with*

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

Proof. Graham's list scheduling coincides with Algorithm CMNS for $\beta = 0$. The proof is therefore exactly the same as the one of Theorem 4.1, except that Lemma 4.4 is used instead of Theorem 2.8. □

For NBUE distributions (see Definition 3.2), Theorem 4.5 yields a performance guarantee of $3 - 1/m$.

Single machine problems. Theorem 4.2 implies a 2-approximation for the special case of a single machine: In this case the term $(m-1)/(m\,\beta)$ disappears, and we can choose $\beta = 0$ to obtain performance guarantee 2. (For $\beta = 0$, the algorithm corresponds to Graham's list scheduling.) This holds for arbitrarily distributed, independent processing times. In fact, this matches the best bound currently known in the deterministic setting; see Open Problem 9 in the collection of Schuurman and Woeginger [19].

5. Further Remarks. A scheduling policy defines a mapping of processing times to start times of jobs. This mapping has to be universally measurable in order

to grant existence of the expected objective function value [14]. Without going into further details we just mention that the scheduling policies discussed in this paper fulfill this requirement, and we refer to [21, Cor. 3.6.15] for further details.

We point out that, apart from the expected processing times of the jobs, a uniform upper bound on their coefficients of variation is the sole stochastic information required as input for the presented scheduling policy. Nevertheless, in our analysis we compare its performance to a lower bound on the performance of *any* non-anticipatory scheduling policy. This refers to the broadest possible sense of scheduling policies as defined by Möhring, Radermacher, and Weiss [14]. In particular, an optimal scheduling policy is therefore allowed to take advantage of the *complete* knowledge of the conditional distributions of the processing times, at any time.

Finally, we mention that our analysis indeed requires policies to be non-anticipatory, because the linear programming lower bound does not hold otherwise. This can be seen from the observation that an anticipatory 'scheduling policy' could, for instance, compute an optimal schedule for any realization of the processing times. Theorem 3.1, however, is no longer valid in this case; see [21]. In other words, our analysis is based upon an adversary that is just as powerful as the scheduling policy itself. This constitutes a major difference compared to the rather 'unfair' competitive analysis known from on-line optimization.

Acknowledgments. We thank the anonymous referee for many valuable comments and suggestions for improvements. In particular, the compact formulation of the algorithm in § 2 was kindly proposed by the referee.

REFERENCES

[1] 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 Association for Computing Machinery, 28 (1981), pp. 100–113.

[2] S. Chakrabarti and S. Muthukrishnan, *Resource scheduling for parallel database and scientific applications*, in Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures, Padua, Italy, 1996, pp. 329–335.

[3] K. M. Chandy and P. F. Reynolds, *Scheduling partially ordered tasks with probabilistic execution times*, Operating Systems Review, 9 (1975), pp. 169–177.

[4] C. Chekuri, R. Johnson, R. Motwani, B. Natarajan, B. Rau, and M. Schlansker, *An analysis of profile-driven instruction level parallel scheduling with application to super blocks*, in Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture, Paris, France, 1996, pp. 58–69.

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

[6] A. Fiat and G. J. Woeginger (eds.), *Online Algorithms: The State of the Art*, Lecture Notes in Computer Science 1442, Springer, Berlin, 1998.

[7] R. L. Graham, *Bounds for certain multiprocessing anomalies*, Bell System Technical Journal, 45 (1966), pp. 1563–1581. see also [8].

[8] R. L. Graham, *Bounds on multiprocessing timing anomalies*, SIAM Journal on Applied Mathematics, 17 (1969), pp. 416–429.

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

[10] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, vol. 2 of Algorithms and Combinatorics, Springer, Berlin, 1988.

[11] W. J. Hall and J. A. Wellner, *Mean residual life*, in Proceedings of the International Symposium on Statistics and Related Topics, M. Csörgö, D. A. Dawson, J. N. K. Rao, and A. K. Md. E. Saleh (eds.), Ottawa, ON, 1981, North-Holland, Amsterdam, pp. 169–184.

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

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

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

[15] R. H. Möhring, A. S. Schulz, and M. Uetz, *Approximation in stochastic scheduling: The power of LP-based priority policies*, Journal of the Association for Computing Machinery, 46 (1999), pp. 924–942.

[16] A. Munier, M. Queyranne, and A. S. Schulz, *Approximation bounds for a general class of precedence constrained parallel machine scheduling problems*, in Proceedings of the 6th International Conference on Integer Programming and Combinatorial Optimization, Houston, TX, 1998, R. Bixby, E. A. Boyd, and R. Z. Rios Mercado (eds.), Lecture Notes in Computer Science 1412, Springer, Berlin, pp. 367–382.

[17] M. Scharbrodt, T. Schickinger, and A. Steger, *A new average case analysis for completion time scheduling*, in Proceedings of the 34th ACM Symposium on Theory of Computing, Montréal, QB, 2002, pp. 170–178.

[18] A. S. Schulz, *Polytopes and Scheduling*, PhD thesis, Institut für Mathematik, Technische Universität Berlin, Germany, 1996.

[19] P. Schuurman and G. J. Woeginger, *Polynomial time approximation algorithms for machine scheduling: Ten open problems*, Journal of Scheduling, 2 (1999), pp. 203–213.

[20] M. Skutella and M. Uetz, *Scheduling precedence-constrained jobs with stochastic processing times on parallel machines*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, pp. 589–590.

[21] M. Uetz, *Algorithms for Deterministic and Stochastic Scheduling*, PhD thesis, Institut für Mathematik, Technische Universität Berlin, Germany, 2001. Published: Cuvillier Verlag, Göttingen, Germany.

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

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

[24] G. Weiss and M. Pinedo, *Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions*, Journal of Applied Probability, 17 (1980), pp. 187–202.