When greediness fails: Examples from stochastic scheduling

Marc Uetz
Faculty of Economics and Business Administration, Quantitative Economics, Universiteit Maastricht, 6200 MD Maastricht, The Netherlands, m.uetz@ke.unimaas.nl

Abstract

The purpose of this paper is to present examples which show that deterministic and stochastic scheduling problems often have a surprisingly different behavior. In particular, it demonstrates some seemingly counterintuitive properties of optimal scheduling policies for stochastic machine scheduling problems.

1 Introduction

The paper addresses stochastic parallel machine scheduling problems with the objective to minimize the total weighted completion time in expectation. Machine scheduling problems 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. Examples for such a scheduling situation are problems that arise, e.g., in compiler optimization (Chekuri, Johnson, Motwani, Natarajan, Rau, and Schlansker 1996) and in parallel computing (Chakrabarti and Muthukrishnan 1996).

Denote by $V = \{1,\ldots,n\}$ a set of jobs with processing requirements p_j, $j = 1,\ldots,n$, 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. In addition to the limited number of available machines, sometimes also precedence constraints must be respected. In that case a partial order $(V,\prec)$ is given, and whenever $i \prec j$ the start of job j must not occur earlier then the completion of job i. We consider the objective to minimize the total weighted completion time $\sum_{j\in V} w_j C_j$, 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 upon completion of the job. Only 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 that the processing times of the jobs are stochastically independent.

In fact, 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 proposed by Möhring, Radermacher, and Weiss (1984). Roughly spoken, a scheduling policy makes scheduling decisions at certain decision times t, and these decisions are based upon the observed past up to time t as well as the a priori knowledge of the input data of the problem. The policy, however, must not anticipate information about the future, such as the actual realizations p_j of the processing times of the jobs which have not yet been completed by time t. A scheduling policy is called *optimal* if it minimizes the objective function value in expectation. In the classical three-field notation of Graham, Lawler, Lenstra, and Rinnooy Kan (1979), the problem of minimizing the expected total weighted completion time can be denoted by $\mathrm{P}|prec|\mathrm{E}\left[\sum w_j C_j\right]$.

The purpose of the paper is to present examples which demonstrate some seemingly counterintuitive properties of stochastic scheduling problems. This particularly includes an example which shows that it may

be even beneficial to *not* use the available machine capacity to its full extent, but rather wait and leave machines *deliberately idle*. The reason for this phenomenon is the gain of information that occurs over time, counter-balancing the loss of efficiency.

In fact, the examples shed a somehow discouraging light on stochastic scheduling problems: The analysis of deterministic counterparts is often not very helpful, and optimal scheduling policies for stochastic models may be hopelessly complex, let alone their precise analysis in terms of typical performance criteria such as the expected objective function value[1].

2 Preliminaries

Let us specify the above sketched dynamic view on scheduling policies more precisely. The *state* of the system at any time t is given by the time t itself as well as the conditional distributions of the jobs' processing times, which depend on the observed *past* up to time t. The past at a time t is given by the set of jobs which have already been completed by t, together with their start and processing times, and the set of jobs which have been started before t but have not been completed by t, together with their start times. The *action* of a scheduling policy at a time t consists of a set of jobs $B(t) \subseteq V$ and a tentative decision time $t_{\text{tent}} > t$. The set $B(t)$ is the set of jobs that are started at time t. The tentative decision time t_{tent} is the latest point in time when the next action of the policy takes place, given that no job ends before t_{tent}. Notice that $B(t)$ may be empty, and $t_{\text{tent}} = \infty$ implies that the next action of the policy takes place when the next job ends. The action of a policy at any time t must only depend on the state of the system at time t; this is the *non-anticipatory* constraint. The definition of $B(t)$ must respect potential precedence constraints and the number of available machines at t. The times 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, whatever occurs first. Depending on the action of the policy, the state at the next decision time is realized according to the probability distributions of the jobs' processing times. A scheduling policy Π can be seen as a function which maps processing times $p = (p_1, \ldots, p_n)$ to start times of jobs $S(p) = (S_1(p), \ldots, S_n(p))$,

$$\Pi : \mathbb{R}_+^n \to \mathbb{R}_+^n, \qquad p \xrightarrow{\Pi} S(p)\,.$$

It turns out that the dynamic properties of scheduling policies can also be described analytically; we refer to Möhring et al. (1984) for more details.

Let us briefly fix some additional notation. A job is called *available* at a time t if all predecessors have been completed by t. A policy that starts jobs only at completion times of other jobs (or at time 0) is called *elementary*; it is characterized by the fact that $t_{\text{tent}} = \infty$ at any decision time. The simplest type of elementary scheduling policies are *list scheduling* policies. Given is a priority list of jobs, and at any time as many available jobs as possible are scheduled greedily in the order given by the list. In a deterministic setting, Graham (1966) analyzed this algorithm for the makespan objective, in fact the earliest paper on worst case analysis of a polynomial time algorithm for an NP-hard combinatorial optimization problem. Algorithm 1 gives the precise description for the stochastic setting.

Algorithm 1: Graham's list scheduling algorithm.

```
initialize t ← 0;
while there are unscheduled jobs in list L do
    let j be the first unscheduled job in list L
    which is available at time t (if any);
    if such a job j exists and a machine is idle at
    time t then
        schedule job j at time t on any of the idle
        machines;
    else
        augment t to the next time when a ma-
        chine falls idle (if necessary, update L);
```

If the list is the same over the whole planning horizon the list scheduling policy is called *static*, otherwise it is called *dynamic* (Pinedo 2002). Prominent instances of static list scheduling policies are LEPT and SEPT, longest respectively shortest expected processing time first, as well WSEPT, the weighted version of SEPT where the priorities of jobs are according to non-decreasing ratios $\mathrm{E}[P_j]/w_j$. It follows, e.g. from an example with exponentially distributed processing times by Kämpke (1987, Ex. 2), that dynamic list scheduling policies may yield a better expected performance than any static list scheduling policy.

[1] The difficulty of the latter problem was impressively underlined by Hagstrom (1988), who showed among other things that the computation of the expected makespan $\mathrm{E}[C_{\max}]$ is a #P-hard problem even for a simple class of stochastic, finite-discrete PERT problems.

Irrespective of the fact wether a list scheduling policy is static or dynamic, it is always *greedy* in the sense that the machines are never left deliberately idle. The following is thus a folklore observation.

Observation 2.1. *Any scheduling policy that avoids deliberate idle times is a (possibly dynamic) list scheduling policy of the Graham type.*

In particular, any policy which avoids deliberate idle times is elementary. Let us call policies which avoid deliberate idle times *idle time free*.

3 Deterministic and stochastic scheduling

Instead of analyzing a stochastic problem, it is tempting to consider a corresponding deterministic counterpart instead by letting $p_j := \mathrm{E}[P_j]$ be the deterministic processing times of the jobs. Let us give two simple examples which demonstrate that this is not necessarily helpful.

Example 3.1 (Möhring and Radermacher 1989). *Consider a family of instances of* $\mathrm{P}|prec|\mathrm{E}[\sum w_j C_j]$ *with* $n = m+1$ *jobs. All jobs* $j = 1,\ldots,n-1$ *have independent, exponentially distributed processing times with parameters* $\lambda_j = 1$. *Job* n *has a deterministic processing time of* $1/m$. *Moreover,* $w_j = 0$ *for all* $j = 1,\ldots,n-1$, *and* $w_n = 1$. *There are precedence constraints* $j \prec n$ *for all* $j = 1,\ldots,n-1$. *It is obviously optimal to start every job as early as possible. Elementary calculations yield* $\mathrm{E}[\sum w_j C_j] = \left(\sum_{j=1}^{m} 1/j\right) + 1/m \to \infty$ *as* $m \to \infty$, *but* $\sum w_j C_j = 1 + 1/m$ *for the corresponding deterministic counterpart.* □

In other words, the deterministic counterpart underestimates the the expected objective function value of the stochastic problem by an arbitrarily large factor as m increases.

Example 3.1 suggests that the deterministic counterpart of a stochastic scheduling problem could at least provide a lower bound on the expected objective value. Due to Jensen's inequality this is in fact the case if the machine restrictions do not take effect, like in the above example, and if the objective function is convex (here it is even linear); see Fulkerson (1962) or Möhring and Radermacher (1989). The reason is that the completion times of jobs are random variables which are composed of summation and/or maxima of the random variables for the jobs' processing times. Thus the objective function is a convex function of the jobs' processing times. If machine restrictions take effect, however, this need no longer be true. The following example illuminates this effect.

Example 3.2. *Consider a family of instances of* $\mathrm{P}|prec|\mathrm{E}[\sum w_j C_j]$ *with* $n = m+2$ *jobs. Jobs* $j = 1,\ldots,n-1$ *have independent, identically distributed processing times according to the following two-point distribution:* $p_j = 1$ *with probability* $1-(\log m)/m$, *and* $p_j = 1/m$ *with probability* $(\log m)/m$. *Job* n *has a deterministic processing time of* $1/m$. *There are precedence constraints* $j \prec n$ *for all* $j = 1,\ldots,n-1$. *Moreover,* $w_j = 0$ *for all* $j = 1,\ldots,n-1$, *and* $w_n = 1$. *Then consider the policy that schedules* m *jobs at time* 0 *and the remaining two jobs as early as possible. With probability* $(1-(\log m)/m)^{m+1}$ *the objective function value is* $2+1/m$. *With probability* $((\log m)/m)^{m+1}$ *the objective function value is* $3/m$. *In all other cases it is at most* $1+2/m$. *Hence,*

$$\begin{aligned}\mathrm{E}[\textstyle\sum w_j C_j] \quad \leqslant \quad & \left(2+\frac{1}{m}\right)\left(1-\frac{\log m}{m}\right)^{m+1} \\ & + \frac{3}{m}\left(\frac{\log m}{m}\right)^{m+1} \\ & + \left(1+\frac{2}{m}\right)\left(1-\left(1-\frac{\log m}{m}\right)^{m+1} - \left(\frac{\log m}{m}\right)^{m+1}\right).\end{aligned}$$

The right hand side converges to 1 *for* $m \to \infty$, *since for any* $x > 0$ *and* $m > e^x$, *we have* $0 \leqslant (1-(\log m)/m)^m \leqslant (1-x/m)^m \leqslant e^{-x}$. *For the corresponding deterministic instance, however,* $\sum w_j C_j \to 2$ *for* $m \to \infty$. □

Hence, the deterministic counterpart does not even yield a lower bound on the expected objective function value of the stochastic problem. In fact, Examples 3.1 and 3.2 suggest that the analysis of deterministic counterparts of stochastic scheduling problems is generally of limited value.

4 When greediness fails

The presence of precedence constraints can make deliberate idle times indeed necessary in stochastic scheduling. The following example illustrates this.

Example 4.1. *Consider an instance of* $\mathrm{P}|prec|\mathrm{E}[\sum w_j C_j]$ *with* 4 *jobs and 2 machines. All jobs have exponentially distributed processing times, jobs* $1-3$ *with parameter* 1, *and job* 4 *with parameter* $1/k$, $k > 0$. *There are precedence constraints* $1 \prec 2$ *and* $1 \prec 3$. *Moreover,* $w_1 = w_4 = 0$, *while* $w_2 = w_3 = 1$.

The only optimal scheduling policy is to start with job 1 at time 0, leave the second machine deliberately

idle, and start jobs 2 and 3 at the end of job 1. Eventually job 4 is scheduled when a machine falls idle. This policy yields $\mathrm{E}[\sum w_j C_j] = 4$. If one uses Graham's list scheduling algorithm instead, jobs 1 and 4 will be started at time 0, irrespective of the priority list. The expected start time of the job scheduled latest, w.l.o.g. assume that this is job 3, is $1+(k/(k+1))^2$. (This follows from elementary calculations: 3 is started at time C_1 if $C_1 \geqslant C_4$ and at time $\min\{C_2, C_4\}$ if $C_1 < C_4$.) This yields $\mathrm{E}[\sum w_j C_j] = 4+(k/(k+1))^2 \to 5$ for $k \to \infty$. Observe that the greedy list scheduling algorithm also performs worse for the case that $0 < k < 1$, so even if job 4 has a smaller expected processing time than job 1 it is better not scheduled in the beginning on the idle machine. (It is not hard to extend this example to more than 2 machines, yet the idea should be clear.)

As long as there are no precedence constraints, however, there seems to be little reason to not use the full machine capacity at any time. Nevertheless, an example by Möhring et al. (1985, Ex. 4.2.5), also reproduced in Möhring and Radermacher (1985, Ex. 5.7) and Kämpke (1987, Ex. 3), shows that an optimal idle time free policy need not exist even for problems *without* precedence constraints. Their example uses exponentially distributed processing times, however, it builds on a somewhat artificial objective function that is not as simple as the total weighted completion time considered here. The objective function in their example is constructed in such a way that certain pairs of jobs are better not scheduled in parallel. Although this objective function is specifically designed to force the policy to use deliberate idle times, it is nevertheless regular and additive[2]. For such objective functions, given exponentially distributed processing times, it is known that an optimal policy always exists within the class of so-called *set policies* (Möhring et al. 1985). This is a subclass of elementary policies where the action at any time t may only depend on the set of jobs completed by t and the set of jobs that is still in process at t, but not on the realizations of processing times p_j or the time t itself (Möhring et al. 1985). Their example thus demonstrates that, although an optimal set policy exists, it is not necessarily idle time free.

We next present a family of instances for the total weighted completion time objective. It uses finite discrete processing times and shows that an optimal idle time free policy, and even an optimal elementary policy, does not exist. Admittedly, the example builds on somewhat artificial finite discrete processing time distributions. Nevertheless, to the best of the author's knowledge it is the first example showing that optimal idle time free policies need not exist for the total weighted completion time objective. The following is the crucial building block of the instance.

Example 4.2. *There are* 4 *jobs. Jobs* 1 *and* 2 *have weights* $w_1 = w_2 = 1$ *and* (*deterministic*) *processing times* $p_1 = p_2 = 1$. *Jobs* 3 *and* 4 *have weights* $w_3 = w_4 = k$, $k \geqslant 5$, $k \in \mathbb{N}$. *The processing times* p_3 *and* p_4 *of these jobs are either* 1, *with probability* $1 - 1/k^3$, *or* $2k^4 - k^3 + 1$ *with probability* $1/k^3$, *independent of each other.* (*The expected processing time of jobs* 3 *and* 4 *is* $2k$.) *The objective is to minimize the expected total weighted completion time* $\mathrm{E}[\sum w_j C_j]$. □

Lemma 4.3. *For the* 4 *jobs in Example 4.2, the order in which the jobs are started in an optimal scheduling policy is different in dependence on the number of available machines.*

Proof. First consider the situation that the jobs are to be scheduled on a single machine. Then, any optimal scheduling policy is according to the WSEPT rule (Rothkopf 1966), say in the order $1,2,3,4$. The expected objective value is $6k^2+4k+3$. If one of jobs 3 or 4 is scheduled first, this leads to an expected objective value of at least $6k^2+6k+3$; a difference of $2k$. If jobs 1 – 4 are to be scheduled on two machines, a case analysis of Graham's list scheduling according to the order $3,4,1,2$ shows that the expected objective value is $4k^2+4+\mathrm{o}(1)$. A tedious but straightforward case analysis shows that any policy which does not start with jobs 3 and 4 leads to an expected objective value of at least $4k^2+k+3-\mathrm{o}(1)$; a difference of $k-1-\mathrm{o}(1)$. In other words, in dependence on the available number of machines, the optimal scheduling policies for the jobs are in fact oppositional. □

Next follows a family of instances for $\mathrm{P}|\,|\mathrm{E}[\sum w_j C_j]$ with finite discrete processing times which shows that an optimal idle time free policy, and even an optimal elementary policy, need not exist.

Example 4.4. *Consider a family of instances of* $\mathrm{P}|\,|\mathrm{E}[\sum w_j C_j]$ *with* 5 *jobs and* $m = 2$ *machines. Jobs* 1 – 4 *are the same as in Example 4.2, using the same parameter* $k \geqslant 5$. *The processing time of job* 5 *is either* $1/k^5$

[2] An objective function $\gamma : \mathbb{R}^n_+ \to \mathbb{R}_+$ is *regular* if $C \geqslant C'$ yields $\gamma(C) \geqslant \gamma(C')$, where $C, C' \in \mathbb{R}^n_+$ are vectors of completion times. It is *additive* if it can be described by a non-decreasing set function $g : 2^V \to \mathbb{R}_+$, the cost rate, where $g(W)$ is the holding cost for any time when the subset of jobs W is not yet completed. The total weighted completion time $\sum w_j C_j$ is regular and additive, with $g(W) = \sum_{j \in W} w_j$, $W \subseteq V$.

or k^5, each with probability $1/2$. Moreover, job 5 has a weight of $2k^3$. The objective is again to minimize the expected total weighted completion time $\mathrm{E}[\sum w_j C_j]$. □

The idea of this example is the following: Due to its comparatively large weight, job 5 must be scheduled at time $t = 0$. It either blocks one machine almost forever (time k^5), or idles the machine after a negligible small amount of time (time $1/k^5$), each with probability $1/2$. On a single machine, any optimal policy of the 4 other jobs must schedule 1 and 2 before 3 and 4. On two machines, however, any optimal policy of the 4 other jobs must schedule 3 and 4 before 1 and 2. In this situation it is the only optimal policy to schedule job 5 at time $t = 0$ and leave the second machine idle. From time $1/k^5$ on the actual processing time of job 5 is known. In other words, it is known if one or two machines are available for the remaining 4 jobs, hence they can be scheduled optimally from time $1/k^5$ on.

Theorem 4.5. *For the jobs in Example 4.4, any optimal scheduling policy is not idle time free, and not even elementary.*

Proof. Consider the given instance and a policy that schedules job 5 at time 0, leaves a machine idle until time $1/k^5$, and then, depending on the realization of the processing time of job 5, optimally schedules the remaining 4 jobs. The maximal total processing time of the remaining 4 jobs is $4k^4 - 2k^3 + 4$. Hence, if job 5 turns out to be long (time k^5), irrespective of the realizations of the processing times of the remaining jobs, there is only one machine available for them, since $k \geqslant 5$. If job 5 turns out to be short (time $1/k^5$), there are two machines available. With the above observations, the expected total weighted completion time of this policy is $k^8 + 5k^2 + 2k + 7/2 + \mathrm{o}(1)$.

Next, observe that any idle time free or elementary policy that schedules job 5 not at time 0, can only schedule it at time 1 or later. This, however, yields an expected objective value of at least $k^8 + 2k^3 - \mathrm{o}(1)$. Since $k \geqslant 5$, this cannot be optimal. So consider a scheduling policy that greedily starts job 5 and some other job at time 0. Then, with probability $1/2$ it turns out at time $1/k^5$ that this decision was not the optimal one. According to the above argumentation, this yields an expected total weighted completion time of at least $k^8 + 5k^2 + (5/2)k + 3 - \mathrm{o}(1)$, which cannot be optimal either. In other words, any optimal policy is not idle time free, since it must leave one machine idle until time $1/k^5$ to make the right decision. It is now obvious that an optimal policy cannot be elementary either, since the scheduling of the remaining jobs must start at time $1/k^5$, irrespective of the processing time of job 5. □

The main idea of Example 4.4 is on the one hand the gain of information which can be achieved by leaving one machine idle and on the other hand the fact that this information is meaningful for the future: The gain of information allows better scheduling decisions which reduce the expected objective value. Moreover, the additional cost that is caused by the deliberate idle time is negligible compared to the gain. It can be conjectured that such nasty examples do not exist if the processing times are exponential: the only information that can be gathered is the question which of the jobs ends next, and this random event is again exponentially distributed. In the case of 2 machines, the gain of information by leaving one machine idle is obviously useless, since there is only one job in process. This even holds for any regular and additive objective function $\gamma : \mathbb{R}_+^n \to \mathbb{R}_+$.

Proposition 4.6. *Consider any instance of* $\mathrm{P}||\mathrm{E}[\gamma]$ *where the jobs have exponentially distributed processing times and γ is any regular and additive objective function. If the number of machines is* 2*, then there exists an optimal policy that is idle time free, and particularly elementary.*

Proof. Let us give the main idea only; see also (Uetz 2001). It can be shown that a policy Π that leaves a machine deliberately idle at time 0 can be replaced by another policy Π', with expected objective value at most as large, which does not leave a machine deliberately idle before first job completion. The claim then follows by induction. Say Π schedules some job i at time 0 and leaves the second machine deliberately idle. Upon completion of job i at least one job is scheduled by Π, say job j. Policy Π' schedules both job i and job j at time 0 and simulates Π henceforth. This is indeed possible due to the fact that the processing times are exponential and thus memory-less. It remains to be shown that the expected objective value of Π' is no larger than the one of Π. This is a bit technical but straightforward; the proof is based on the memory-less property of the exponential distribution and it requires that the objective function is regular and additive. □

Notice that the proposition is not necessarily true for problems with precedence constraints, as was demonstrated in Example 4.1. Moreover, it follows directly from Möhring et al. (1985) that among all optimal idle time free policies there is also a set policy. Finally, it follows from the Example of Kämpke (1987, Ex. 2) that in

general, the optimal (set-type and idle time free) policy must be dynamic.

5 Final remark

Given that the processing times are exponential, it is still an open problem to characterize the objective functions for which Proposition 4.6 holds if number of machines is more than 2. It was conjectured by Möhring et al. (1985) that convexity of the objective function γ, which is equivalent to submodularity of the underlying cost rate, suffices to guarantee the existence of optimal (set) policies which are idle time free. However, the problem is still open even for the linear objective function $\sum w_j C_j$. In the words of Möhring and Radermacher (1985, p. 127), this *deliberate idleness* problem

> " [. . .] seems to be one of the deep open problems in nonpreemptive models in the framework of stochastic scheduling problems. "

Acknowledgements. The author is partially supported by a grant of METEOR, the Maastricht research school of Economics of TEchnology and ORganizations.

References

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

Chekuri, C., R. Johnson, R. Motwani, B. Natarajan, B. Rau, and M. Schlansker (1996). 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), pp. 58–69.

Fulkerson, D. R. (1962). Expected critical path length in PERT networks. *Operations Research 10*, 808–817.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal 45*, 1563–1581. see also (Graham 1969).

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

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

Hagstrom, J. N. (1988). Computational complexity of PERT problems. *Networks 18*, 139–147.

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

Möhring, R. H. and F. J. Radermacher (1985). Introduction to stochastic scheduling problems. In K. Neumann and D. Pallaschke (Eds.), *Contributions to Operations Research — Proceedings of the Conference on Operations Research*, Oberwolfach, Germany, Volume 240 of *Lecture Notes in Economics and Mathematical Systems*, pp. 72–130. Berlin: Springer.

Möhring, R. H. and F. J. Radermacher (1989). The order-theoretic approach to scheduling: The stochastic case. In R. Słowiński and J. Węglarz (Eds.), *Advances in Project Scheduling*, Volume 9 of *Studies in Production and Engineering Economics*, Chapter 4, pp. 497–531. Amsterdam: Elsevier.

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

Möhring, R. H., F. J. Radermacher, and G. Weiss (1985). Stochastic scheduling problems II: Set strategies. *ZOR - Zeitschrift für Operations Research 29*, 65–104.

Pinedo, M. (2002). *Scheduling: Theory, Algorithms, and Systems* (2 ed.). Upper Saddle River (NJ): Prentice-Hall.

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

Uetz, M. (2001). *Algorithms for Deterministic and Stochastic Scheduling*. Ph. D. thesis, Institut für Mathematik, Technische Universität Berlin, Berlin, Germany. Published: Cuvillier Verlag, Göttingen, Germany.