New Old Algorithms for Stochastic Scheduling

Extended Abstract

Andreas S. Schulz

Sloan School of Management, Massachusetts Institute of Technology,
Office E53-361, 77 Massachusetts Ave., Cambridge, MA 02139, USA
`schulz@mit.edu`

Abstract We consider the stochastic identical parallel machine scheduling problem and its online extension, when the objective is to minimize the expected total weighted completion time of a set of jobs that are released over time. We give randomized as well as deterministic online and offline algorithms that have the best known performance guarantees in either setting, online or offline and deterministic or randomized. Our analysis is based on a novel linear programming relaxation for stochastic scheduling problems that can be solved online.

1 Introduction

We consider stochastic and online versions of the following deterministic, offline scheduling problem. There is a set of n jobs to be processed on m identical parallel machines. Each job j has a nonnegative weight w_j, processing time p_j, and release date r_j. After its release, each job has to be processed on any machine, and each machine can handle at most one job at a time. The objective is to minimize the total weighted completion time $\sum_{j=1}^{n} w_j C_j$, where C_j denotes the completion time of job j in the schedule. This problem is well understood: It is known to be strongly NP-hard (Lenstra, Rinnooy Kan, and Brucker 1977), and it has a polynomial-time approximation scheme (Afrati et al. 1999); a 2-approximation algorithm was earlier given by Schulz and Skutella (2002b).

In stochastic scheduling, job processing times are random, known in advance only as independent probability distributions (with expected values μ_j and standard deviations σ_j). The actual processing time of a job does not become known before it is completed. Research has traditionally focused on nonanticipative policies that aim at minimizing the objective function in expectation. A scheduling policy is nonanticipative if its decisions about which jobs to schedule at any given time t only depend on the jobs that are already completed by that time and on the conditional distributions of the remaining processing times of those jobs that are still active at time t. In particular, the optimal policy also has no knowledge about the actual processing times and has to make decisions based only on the information of the release dates, weights, and distributions of the processing times of the jobs. For a formal introduction of policies, we refer the reader to Möhring, Radermacher, and Weiss (1984, 1985). For the single-machine problem without nontrivial release dates ($m = 1$, $r_j = 0$ for all jobs j), Rothkopf (1966) showed that the WSEPT rule is optimal, which schedules the jobs in order of nonincreasing ratios of weight to expected processing time. For unit weights and exponentially distributed processing times, the Shortest Expected Processing Time rule remains optimal on identical parallel machines (Weiss and Pinedo 1980). In fact, Weber, Varaiya, and Walrand (1986) showed that it suffices when the processing time distributions are stochastically comparable in pairs. For arbitrary weights, WSEPT is optimal for exponentially distributed processing times if the WSEPT order of jobs coincides with sequencing the jobs in the order of nonincreasing weights (Kämpke 1987). Under minor technical assumptions, Weiss (1990) showed that the WSEPT rule is asymptotically optimal.

The problem considered here, when jobs may have individual release dates, was first addressed by Möhring, Schulz, and Uetz (1999). For processing time distributions whose coefficients of variation σ_j/μ_j are bounded from above by $\sqrt{\Delta}$, they gave a static priority policy whose expected objective function value is within a factor of $\max\{4, 3 + \Delta\}$ of that of an optimal policy.[1] In addition, they showed that the WSEPT rule has a performance guarantee of $1 + (\Delta + 1)/2$ for the problem with identical release dates. This also marked the first time that the use of approximation algorithms was proposed in the realm of stochastic scheduling. The analysis as well as the algorithm for the general case is based on a linear programming relaxation, which provides a lower bound on the expected value of an optimal policy.

A different way of dealing with incomplete information is that of online algorithms and competitive analysis. In our context, jobs arrive over time and are completely unknown prior to their arrival. However, a job's processing time and weight are fully revealed at the time of its arrival. The performance of an online algorithm is usually compared to that of an optimal offline algorithm, which has complete information beforehand. This value is known as the competitive ratio. For randomized online algorithms, we compare the expected objective function value of the solution generated by the algorithm to the value of an offline optimum. This corresponds to the so-called oblivious adversary model. We refer the reader to Borodin and El-Yaniv (1998) for a general introduction to online algorithms, and to Sgall (1998) for a survey of online scheduling models and results. In the context of the identical parallel machine scheduling problem considered here, online algorithms were designed and analyzed by Hall et al. (1997), Chakrabarti et al. (1996), Schulz and Skutella (2002b), Megow and Schulz (2004), and Correa and Wagner (2005). The currently best deterministic online algorithm has a competitive ratio of 2.618 (Correa and Wagner 2005), while the best known randomized algorithm is 2-competitive (Schulz and Skutella 2002b).

Chou, Liu, Queyranne, and Simchi-Levi (2001, 2004) proposed to look at stochastic online scheduling, where jobs arrive over time, as in online schedul-

[1] The performance guarantee of this algorithm is actually slightly better than this; to make for an improved reading, we generally suppress terms of order $1/m$ from this extended abstract.

ing, but when a job arrives only its weight and expected processing time become known. The expected total weighted completion time of the schedule computed by an online policy is then compared to that of the optimal stochastic policy, which has access to all job release dates, weights, and processing time distributions at time 0. In other words, the adversary controls the arrival of jobs, their weights, and their processing time distributions, but he cannot influence the actual realization of processing times. While Chou et al. (2001, 2004) considered single-machine and flow-shop problems, Megow, Uetz, and Vredeveld (2005) studied the identical parallel machine model considered here. They introduced δ-NBUE distributions[2] and gave a deterministic online algorithm with performance guarantee $3/2 + \delta + \sqrt{4\delta^2 + 1}/2$. Their analysis uses the linear programming relaxation introduced by Möhring, Schulz, and Uetz (1999), but Megow, Uetz, and Vredeveld are anxious to point out that their algorithm does not. In fact, they claim that "LP-based approaches to define list scheduling algorithms [for the problem considered here] do not seem to make sense at all."

In this paper, we present LP-based online algorithms for stochastic scheduling problems. The key is a new linear programming relaxation for the stochastic scheduling problem of minimizing the total weighted completion time on identical parallel machines. This linear program can be solved online. The resulting "LP schedule," which can be interpreted as a preemptive single-machine schedule, is then used to define an online policy for the original problem. This approach has previously been used successfully for various deterministic online problems, including nonpreemptive scheduling on a single machine (Goemans et al. 2002), preemptive single-machine scheduling (Schulz and Skutella 2002a), identical parallel machine scheduling (Schulz and Skutella 2002b), and uniform parallel machine scheduling (Chou, Queyranne, and Simchi-Levi 2001).

We present one randomized and one deterministic algorithm; both work online and run in polynomial time. Their respective performance ratios are $2 + \Delta$ and $\max\{3.618, 2.309 + 1.309\Delta\}$, respectively. The randomized algorithm can be derandomized, which results in a deterministic $(2 + \Delta)$-approximation algorithm for the stochastic (offline) scheduling problem. Table 1 compares the new results from this paper to earlier results; to simplify the exposition, we restrict ourselves in the table and throughout the remainder of this extended abstract to processing time distributions that satisfy $\Delta \leq 1$.

The algorithms proposed here are derived from earlier algorithms for deterministic scheduling problems, as were previous algorithms for stochastic scheduling. In our case, we convert a randomized online algorithm of Schulz and Skutella (2002b) as well as a deterministic online algorithm by Correa and Wagner (2005). Previously, Möhring, Schulz, and Uetz (1999) built on deterministic algorithms by Hall et al. (1997), Skutella and Uetz (2001) used techniques of Chekuri et al. (2001), and Megow, Uetz, and Vredeveld (2005) drew on ideas from Megow

[2] For $\delta = 1$, one recaptures the well-known NBUE distributions, "new better than used in expectation," which contain, among others, exponential, Erlang, uniform, and Weibull distributions. NBUE distributions satisfy $\Delta \leq 1$ (Hall and Wellner 1981). In general, $\Delta \leq 2\delta - 1$ (Megow, Uetz, and Vredeveld 2005).

Table 1. Summary of performance guarantees for instances for which the distributions of job processing times fulfill $\Delta \leq 1$. The relative order of the algorithms' performances remains the same for arbitrary distributions.

Model	Performance Guarantee		Reference
	deterministic	randomized	
deterministic online	$4+\varepsilon$		Hall et al. (1997)
		$2.885+\varepsilon$	Chakrabarti et al. (1996)
		2	Schulz and Skutella (2002b)
	3.281		Megow and Schulz (2004)
	2.618		Correa and Wagner (2005)
stochastic offline	4	–	Möhring, Schulz, and Uetz (1999)
	3.618	–	Megow, Uetz, and Vredeveld (2005)
	3	–	this paper
stochastic online	3.618		Megow, Uetz, and Vredeveld (2005)
	3.618	3	this paper

and Schulz (2004). The key is to refine the algorithms and their analyses such that they still work even though job processing times are random. In contrast to the previous approximation and online algorithms for stochastic scheduling problems, which all relied on the lower bounds presented by Möhring, Schulz, and Uetz (1999), we use a linear programming relaxation that is new in the context of stochastic scheduling.

2 A Linear Programming Relaxation

Möhring, Schulz, and Uetz (1999) showed that the vector of expected completion times of any given policy satisfies the following inequalities:

$$\sum_{j\in S} \mu_j\, C_j \geq \frac{1}{2m}\Big(\sum_{j\in S}\mu_j\Big)^2 \qquad \text{for all } S\subseteq N.$$

Here, N denotes the set of all jobs; i.e., $N=\{1,2,\dots,n\}$. One can actually strengthen these inequalities by observing that none of the jobs in a subset S can be processed before $r_{\min}(S) := \min\{r_j : j\in S\}$.

Lemma 1. *Let Π be any nonanticipative policy for the stochastic identical parallel machine scheduling problem. Then, the corresponding vector $E[C^{\Pi}]$ of expected completion times satisfies the following inequalities:*

$$\sum_{j\in S} \mu_j\, C_j \geq r_{\min}(S)\sum_{j\in S}\mu_j + \frac{1}{2m}\Big(\sum_{j\in S}\mu_j\Big)^2 \qquad \textit{for all } S\subseteq N. \tag{1}$$

This observation was made earlier in the context of deterministic scheduling; see Queyranne and Schulz (1995). Its relevance in our situation follows from the fact that the associated linear programming relaxation, when we minimize $\sum_{j\in N} w_j C_j$ over (1), is equivalent to that of a deterministic single-machine problem with processing times μ_j/m and mean busy time variables M_j:

$$\min \sum_{j\in N} w_j M_j \tag{2a}$$

$$\text{s.t.} \quad \sum_{j\in S} \frac{\mu_j}{m} M_j \geq \frac{\sum_{j\in S}\mu_j}{m}\left(r_{\min}(S) + \frac{\sum_{j\in S}\mu_j}{2m}\right) \quad \text{for all } S \subseteq N \tag{2b}$$

The mean busy time M_j of a job j is the average point in time at which the (single) machine is busy with processing j. In other words, if $I_j(t)$ is 1 if the machine is processing job j at time t, and 0 otherwise, then

$$M_j = \frac{1}{p_j}\int_{r_j}^{\infty} I_j(t)\, t\, dt \ .$$

Here and henceforth, we use p_j to denote the processing time of job j on the single machine; i.e., $p_j = \mu_j/m$. The following theorem is crucial for the design of an online LP-based policy.

Theorem 2 (Goemans et al. 2002). *The mean busy time vector of the preemptive single-machine schedule that is constructed by the following online algorithm is an optimal solution to the linear programming relaxation* (2)*:*

> *At any point in time, schedule from the jobs that are not yet completed the one with the highest ratio of weight to processing time.*

We refer to this preemptive schedule as *the LP schedule*. Let us emphasize that Theorem 2 implies that one can solve the linear programming relaxation of minimizing $\sum_{j\in N} w_j C_j$ over (1) online, and it still provides a lower bound on the expected value of the optimal offline policy.

3 A Randomized Algorithm for Stochastic Online Scheduling

In the spirit of previous approximation algorithms for stochastic scheduling problems, which adopted earlier algorithms for deterministic scheduling problems to the stochastic setting, we will now extend an algorithm of Schulz and Skutella (2002b) to stochastic online scheduling. Before we can describe the algorithm, we need to introduce the notion of α-points. For $0 < \alpha \leq 1$, the α-point $t_j(\alpha)$ of a job j is the first moment in time when an α-fraction of j has been completed in the LP schedule. The algorithm is described in Figure 1.

Let us elaborate a bit on each step of the algorithm. As we mentioned before, the LP schedule can be computed online. As soon as a job arrives, we assign it to one of the m machines, with each machine being equally likely. This is done

(1) Compute the LP schedule.
(2) Assign each job randomly to a machine.
(3) Draw α_j randomly from $(0, 1]$.
(4) Sequence the jobs in nondecreasing order of α_j-points $t_j(\alpha_j)$.

Figure 1. A randomized online algorithm for stochastic scheduling.

independently for all jobs. We also immediately draw α_j uniformly from $(0, 1]$, again independent from the drawings for other jobs. Then, on each machine, the jobs assigned to that machine are sequenced in nondecreasing order of their α-points. For this to work online, we just impose the additional restriction that no job j may start before time $t_j(\alpha_j)$.

Theorem 3. *The algorithm described in Figure 1 is a randomized online algorithm for stochastic scheduling on identical parallel machines to minimize the expected total completion time; its expected performance guarantee is* 3.

Proof. Let us consider an arbitrary but fixed job j. Initially, let us also fix the index i of the machine to which j has been assigned, as well as a value of α_j. Note that j is ready to start at time $t_j(\alpha_j)$; in particular, $r_j \leq t_j(\alpha_j)$. If j is not started at time $t_j(\alpha_j)$, then it is delayed by jobs with a smaller α-point that have been assigned to the same machine i. We denote by $E_{i,\alpha_j}[C_j]$ the conditional expected completion time of job j, where the expectation is taken both over the random choices of the algorithm, except for i and α_j, which are still fixed, and the processing times. We then have

$$\begin{aligned}
E_{i,\alpha_j}[C_j] &\leq t_j(\alpha_j) + \mu_j + \sum_{k \neq j} \mu_k \cdot P(k \text{ on } i \text{ before } j) \\
&\leq t_j(\alpha_j) + \mu_j + \sum_{k \neq j} \mu_k \, \frac{1}{m} \, \frac{1}{p_k} \int_0^{t_j(\alpha_j)} I_k(t) \, dt \\
&\leq t_j(\alpha_j) + \mu_j + t_j(\alpha_j) \\
&= 2 \, t_j(\alpha_j) + \mu_j \ .
\end{aligned}$$

In the first inequality, $P(k \text{ on } i \text{ before } j)$ is the probability that job $k \neq j$ is assigned to the same machine as j and will be started before j. The probability that k is assigned to machine i is $1/m$. The integral in the second inequality captures the fraction of job k that is processed in the LP schedule before $t_j(\alpha_j)$, which by the choice of α_k is precisely the probability of $t_k(\alpha_k)$ being smaller than $t_j(\alpha_j)$. The remaining two inequalities are straightforward. We finally get rid of the conditional expectation by noting that the average α_j-point is equal to the mean busy time M_j in the LP schedule (Goemans et al. 2002). Hence,

$$\begin{aligned}
E[C_j] &\leq 2 \int_0^1 t_j(\alpha_j) \, d\alpha_j + \mu_j \\
&= 2 \, M_j + \mu_j \ .
\end{aligned}$$

The result now follows from our earlier observation that $\sum_{j \in N} w_j M_j$ is a lower bound on the expected value of an optimal policy, and so is $\sum_{j \in N} w_j \mu_j$. □

The crucial observation which makes this proof work is that the set of jobs that is scheduled on machine i before j does not depend on the actual realization of processing times. The order of jobs is determined by the LP schedule, which only depends on the expected processing times.

Corollary 4. *There exists a deterministic* 3*-approximation algorithm for the stochastic (offline) problem of minimizing the weighted sum of completion times on identical parallel machines subject to release dates.*

We omit the proof from this extended abstract, but note that this algorithm can be obtained from the one in Figure 1 by the method of conditional probabilities. Of course, this implies that the derived algorithm does not work in an online context. This will be fixed, to some extent, in the next section.

4 A Deterministic Algorithm for Stochastic Online Scheduling

A simple, though somewhat less effective way of derandomizing the algorithm given in Figure 1, yet one that does not destroy its online nature, is to choose α_j deterministically and beforehand. The resulting algorithm is described in Figure 2. Step (2) requires some additional explanation. It essentially is a basic

```
(1) Compute the LP schedule.
(2) Schedule the jobs greedily in nondecreasing order of their $\alpha$-points.
```

Figure 2. A deterministic online algorithm for stochastic scheduling.

list scheduling algorithm, where one job after the other is assigned to the earliest possible start time, with a twist: No job j is started before its α-point $t_j(\alpha_j)$. Let ϕ denote the golden ratio, and let us choose $\alpha_j = \phi - 1$ for all $j \in N$.

Theorem 5. *The algorithm described in Figure 2 is a deterministic online algorithm for stochastic scheduling on identical parallel machines to minimize the expected total completion time; it has performance guarantee* $2 + \phi$.

Proof. The proof is saved for the full version of this paper; it mimics to a large extent that of Correa and Wagner (2005, Theorem 3.2), which itself is an extension of that of Goemans et al. (2002, Theorem 3.3). Apart from Lemma 1, the key insight is that the start of any job j is always delayed by the same set of jobs, regardless of the instantiation of actual processing times. □

Acknowledgments

I am grateful to the organizers of Dagstuhl Seminar 05031, where this work was conceived, and to José Correa for sending me a preliminary version of Correa and Wagner (2005), which inspired me to revisit stochastic scheduling problems.

References

Afrati, F., E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko (1999). Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pp. 32–43.

Borodin, A. and R. El-Yaniv (1998). *Online Computation and Competitive Analysis.* Cambridge University Press.

Chakrabarti, S., C. Phillips, A. Schulz, D. Shmoys, C. Stein, and J. Wein (1996). Improved scheduling algorithms for minsum criteria. In F. M. auf der Heide and B. Monien (Eds.), *Automata, Languages and Programming*, Volume 1099 of *Lecture Notes in Computer Science*, pp. 646–657. Springer.

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

Chou, C.-F., H. Liu, M. Queyranne, and D. Simchi-Levi (2001). On the asymptotic optimality of an on-line algorithm for the stochastic single machine & flow shop average weighted completion time problem. INFORMS Annual Meeting. Miami, FL.

Chou, C.-F., M. Queyranne, and D. Simchi-Levi (2001). The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates. In K. Aardal and B. Gerards (Eds.), *Integer Programming and Combinatorial Optimization*, Volume 2081 of *Lecture Notes in Computer Science*, pp. 45–59. Springer.

Chou, M., H. Liu, M. Queyranne, and D. Simchi-Levi (2004). On the asymptotic optimality of a simple on-line algorithm for the stochastic flow shop weighted completion time problem.

Correa, J. and M. Wagner (2005). LP-based online scheduling: From single to parallel machines. In *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science. Springer. To appear.

Goemans, M., M. Queyranne, A. Schulz, M. Skutella, and Y. Wang (2002). Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics 15*, 165–192.

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

Hall, W. and J. Wellner (1981). Mean residual life. In M. Csörgö, D. Dawson, J. Rao, and A. E. Saleh (Eds.), *Proceedings of the International Symposium on Statistics and Related Topics*, pp. 169–184. North-Holland.

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

Lenstra, J., A. Rinnooy Kan, and P. Brucker (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343–362.

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

Megow, N., M. Uetz, and T. Vredeveld (2005). Models and algorithms for stochastic online scheduling. Technical Report 003-2005, Institut für Mathematik, Technische Universität Berlin, Berlin, Germany. A preliminary version appeared in the Proceedings of the 2nd Workshop on Approximation and Online Algorithms (WAOA 2004).

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

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

Möhring, R., A. Schulz, and M. Uetz (1999). Approximation in stochastic scheduling: The power of LP-based priority policies. *Journal of the ACM 46*, 924–942.

Queyranne, M. and A. Schulz (1995). Scheduling unit jobs with compatible release dates on parallel machines with nonstationary speeds. In E. Balas and J. Clausen (Eds.), *Integer Programming and Combinatorial Optimization*, Volume 920 of *Lecture Notes in Computer Science*, pp. 307–320. Springer.

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

Schulz, A. and M. Skutella (2002a). The power of α-points in preemptive single machine scheduling. *Journal of Scheduling 5*, 121–133.

Schulz, A. and M. Skutella (2002b). Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics 15*, 450–469.

Sgall, J. (1998). On-line scheduling. In A. Fiat and G. Woeginger (Eds.), *Online Algorithms: The State of the Art*, Volume 1442 of *Lecture Notes in Computer Science*, Chapter 9, pp. 196–231. Springer.

Skutella, M. and M. Uetz (2001). Scheduling precedence-constrained jobs with stochastic processing times on parallel machines. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 589–590.

Weber, R., P. Varaiya, and J. Walrand (1986). Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. *Journal of Applied Probability 23*, 841–847.

Weiss, G. (1990). Approximation results in parallel machines stochastic scheduling. *Annals of Operations Research 26*, 195–242.

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