MIT Sloan School of Management

Working Paper 4435-03
September 2003

Scheduling to Minimize Average Completion Time Revisited: Deterministic On-line Algorithms

Nicole Megow, Andreas S. Schulz

Scheduling to Minimize Average Completion Time Revisited: Deterministic On-line Algorithms

Nicole Megow*
Institut für Mathematik
Technische Universität Berlin
Sekr. MA 6-1
Strasse des 17. Juni 136
10623 Berlin, Germany
nmegow@math.tu-berlin.de

Andreas S. Schulz
Sloan School of Management
Massachusetts Institute of Technology
Office E53-361
77 Massachusetts Avenue
Cambridge, MA 02139-4307, USA
schulz@mit.edu

September 2003

Abstract

We consider the scheduling problem of minimizing the average weighted completion time on identical parallel machines when jobs are arriving over time. For both the preemptive and the nonpreemptive setting, we show that straightforward extensions of Smith's ratio rule yield smaller competitive ratios compared to the previously best-known deterministic on-line algorithms, which are $(4+\varepsilon)$-competitive in either case. Our preemptive algorithm is 2-competitive, which actually meets the competitive ratio of the currently best randomized on-line algorithm for this scenario. Our nonpreemptive algorithm has a competitive ratio of 3.28. Both results are characterized by a surprisingly simple analysis; moreover, the preemptive algorithm also works in the less clairvoyant environment in which only the ratio of weight to processing time of a job becomes known at its release date, but neither its actual weight nor its processing time. In the corresponding nonpreemptive situation, every on-line algorithm has an unbounded competitive ratio.

1 Introduction

Model. We consider the problem of scheduling jobs arriving over time on-line on identical parallel machines to minimize the sum of weighted completion times. Each of the m machines can process only one of the n jobs at a time. Each job j of a given instance has a positive processing time $p_j > 0$ and a nonnegative weight $w_j \geq 0$. We learn about these job data only at the job's release date $r_j \geq 0$, which is not known in advance, either. If C_j denotes the completion time of job j in a feasible schedule, the corresponding objective function value is $\sum_{j=1}^{n} w_j C_j$. We consider both the preemptive and the nonpreemptive machine environments. In the preemptive setting, the processing of a job may be suspended and resumed later on any machine at no extra cost. In contrast, once a

*Supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin. Part of this research was performed while this author was visiting the Operations Research Center at the Massachusetts Institute of Technology.

job is started in the nonpreemptive mode, it must be processed on the same machine without any interruption until its completion. In scheduling notation [6], the corresponding off-line problems are denoted by $P \,|\, r_j, \text{pmtn} \,|\, \sum w_j C_j$ and $P \,|\, r_j \,|\, \sum w_j C_j$, respectively. Already the analogous single-machine problems are NP-hard [9, 10].

However, instances of $1 \,|\, | \,\sum w_j C_j$ are optimally solved by Smith's Weighted Shortest Processing Time (WSPT) rule, which sequences jobs in nonincreasing order of their weight-to-processing-time ratios [17]. For convenience, we assume that the jobs are indexed in this order so that $w_1/p_1 \geq w_2/p_2 \geq \cdots \geq w_n/p_n$. Moreover, we say that a job with a smaller index has higher priority than one with a larger index.

The quality of on-line algorithms is typically assessed by their worst-case performance, expressed as the competitive ratio [16]. A ρ-competitive algorithm provides for any instance a solution with an objective function value of at most ρ times the value of an optimal off-line solution.

Main Results. We show in Section 2 that a natural extension of the WSPT rule to preemptive scheduling on identical parallel machines with release dates is 2-competitive, and this bound is tight. The idea is to interrupt currently active jobs of lower priority whenever new high-priority jobs arrive and not enough machines are available to accommodate the arrivals.

When preemption is not allowed, a straightforward extension of this scheme is to start the currently available job of highest priority whenever a machine becomes idle. However, this rule does not directly lead to a bounded competitive ratio. In fact, consider a single-machine instance in which a job of high priority is released right after the start of a long lower-priority job. Therefore, we first modify the release date of each job such that it is equal to a certain fraction of its processing time, if necessary. If we now start a long job j and a high-priority job becomes available shortly thereafter, the ill-timed choice of starting j can be accounted for by the fact that the high-priority job has a release date or processing time at least as large as a fraction of p_j. Therefore, its delay is bounded by its own contribution to the objective function in any feasible schedule. We consider a family of alike algorithms in Section 3 and show that the best one is 3.28-competitive. In this case, we cannot show that our analysis is tight, but the remaining gap is at most 0.5.

Related Work. Lu, Sitters and Stougie [11] introduced a related class of 2-competitive algorithms, which use similar waiting strategies for the on-line variant of the single-machine problem $1 \,|\, r_j \,|\, \sum C_j$. In fact, the idea of boosting release dates was used before by Hoogeveen and Vestjens [8] and Stougie (cited in [18]), who delayed the release date of each job j until time $\max\{r_j, p_j\}$ and $r_j + p_j$, respectively. Anderson and Potts [2] extended both, Hoogeveen and Vestjen's algorithm and its competitive ratio of 2, to the setting of arbitrary nonnegative job weights. These results are best possible since Hoogeveen and Vestjens also proved that no nonpreemptive deterministic on-line algorithm can achieve a competitive ratio better than 2.

Phillips, Stein and Wein [12] presented another on-line algorithm for $1 \,|\, r_j \,|\, \sum C_j$, which converts a preemptive schedule into a nonpreemptive one of objective function value at most twice that of the preemptive schedule. Since Schrage's Shortest Remaining Processing Time (SRPT) rule [13] works on-line and produces an optimal preemptive schedule for the single-machine problem, it follows that Phillips, Stein and Wein's algorithm has competitive ratio 2 as well. The conversion factor is $3 - 1/m$ if applied to identical parallel machines, but the corresponding preemptive problem is NP-hard in this case. However, Chekuri, Motwani, Natarajan and Stein [3] noted that by sequencing jobs nonpreemptively in the order of their completion times in the optimal preemptive schedule on

a single machine of speed m times as fast as that of any one of the m parallel machines, one obtains a $(3 - 1/m)$-competitive algorithm for the on-line version of $\mathrm{P} \,|\, r_j \,|\, \sum C_j$. For the same problem, Lu, Sitters and Stougie [11] gave a 2α-competitive algorithm, where α is the competitive ratio of the direct extension of the SRPT rule to identical parallel machines. Phillips, Stein and Wein [12] showed that this rule is 2-competitive, but a smaller value of α has not been ruled out.

In any case, the hitherto best known deterministic on-line result for the corresponding scheduling problems with arbitrary job weights, $\mathrm{P} \,|\, r_j, \mathrm{pmtn} \,|\, \sum w_j C_j$ and $\mathrm{P} \,|\, r_j \,|\, \sum w_j C_j$ was a $(4 + \varepsilon)$-competitive algorithm by Hall, Schulz, Shmoys and Wein [7], which was given as part of a more general on-line framework. For $1 \,|\, r_j, \mathrm{pmtn} \,|\, \sum w_j C_j$, Goemans, Wein and Williamson (cited as personal communication in [14]) noted that the preemptive version of the WSPT rule is 2-competitive; it schedules at any point in time the highest-priority job, possibly preempting jobs of lower priority. (A proof of this result is given in [14].) Our preemptive parallel-machine algorithm is the direct extension of this variation of Smith's rule. Schulz and Skutella [14] and Goemans, Queyranne, Schulz, Skutella and Wang [5] give comprehensive reviews of the development of on-line algorithms for the preemptive and nonpreemptive single-machine problems, respectively; Hall, Schulz, Shmoys and Wein [7] do the same for the parallel machine counterparts.

On the side of negative results, Vestjens [18] proved a universal lower bound of 1.309 for the competitive ratio of any deterministic on-line algorithm for $\mathrm{P} \,|\, r_j \,|\, \sum C_j$. In the preemptive case, the currently known lower bound is just 22/21, also given by Vestjens.

Let us eventually mention that the currently best randomized on-line algorithms for the two problems considered here have (expected) competitive ratio 2; see [15]. Moreover, the off-line versions of these problems are well understood; in fact, both problems have a polynomial-time approximation scheme [1].

2 Preemptive Parallel Machine Scheduling

We consider the following extension of the single-machine preemptive WSPT rule to identical parallel machines.

Algorithm 1: P-WSPT

At any point in time, schedule the m jobs with the highest priorities among the available, not yet completed jobs (or fewer if less than m incomplete jobs are available). Interrupt the processing of currently active jobs, if necessary.

The algorithm works on-line since the decision about which job to run at any given point in time t is just based on the set of available jobs at time t. In fact, it only depends on the priorities of the available jobs. In particular, Algorithm P-WSPT also operates in an environment in which actual job weights and processing times may not become available before the completion of the jobs, as long as the jobs' priorities are known at their respective release dates.

Theorem 2.1. *The Algorithm* P-WSPT *produces a solution of objective function value at most twice the optimal value for the off-line problem* $P \,|\, r_j, pmtn \,|\, \sum w_j C_j$.

Proof. Consider the time interval $(r_j, C_j]$ of an arbitrary but fixed job j. We partition this interval into two disjunctive sets of subintervals: $I(j)$ contains the subintervals in which job j is being processed, and $\bar{I}(j)$ denotes the set of remaining subintervals, in which other jobs than j are being processed. Note that no machine can be idle during the subintervals belonging to $\bar{I}(j)$. Since the algorithm processes job j after its release date r_j whenever a machine is idle, we obtain

$$C_j \leq r_j + |I(j)| + |\bar{I}(j)| \ ,$$

where $|\cdot|$ denotes the sum of the lengths of the subintervals in the corresponding set.

The overall length of $I(j)$ is clearly p_j. Only jobs with a higher ratio of weight to processing time than j can be processed during the intervals of the set $\bar{I}(j)$, because the algorithm gives priority to j before scheduling jobs with lower ratio. In the worst case, that is when $|\bar{I}(j)|$ is maximal, all jobs with higher priority than j are being processed in the subintervals of this set. Then $|\bar{I}(j)| = (\sum_{k<j} p_k)/m$, and we can bound the value of the P-WSPT schedule as follows:

$$\sum_j w_j C_j \ \leq \ \sum_j w_j (r_j + p_j) + \sum_j w_j \sum_{k<j} \frac{p_k}{m} \ .$$

Since the completion time C_j of a job j is always at least as large as its release date plus its processing time, $\sum_j w_j(r_j + p_j)$ is obviously a lower bound on the value of an optimal schedule. Moreover, $\sum_j w_j \sum_{k \leq j} p_k/m$ is the objective function value of an optimal solution to the corresponding instance of the relaxed problem $1\,|\,|\sum w_j C_j$ on a single machine with speed m times the speed of any of the identical parallel machines. As this problem is indeed a relaxation of the scheduling problem considered here, we can conclude that the P-WSPT algorithm is 2-competitive. □

A family of instances provided by Schulz and Skutella [14] shows that this result cannot be improved. In fact, for $m = 1$, P-WSPT coincides with the preemptive single-machine algorithm studied in their paper. Taking m copies of Schulz and Skutella's instance yields the following result.

Lemma 2.2. *The competitive ratio of the Algorithm* P-WSPT *is not better than* 2 *for the on-line problem* $P\,|\,r_j, pmtn\,|\sum w_j C_j$*, for any given number of machines.*

Proof. We include a proof for the sake of completeness. We consider an instance that is slightly different from the one given in [14]. It consists of m copies of $n+1$ jobs with $w_j = 1$, $p_j = n - j/n$ and $r_j = jn - j(j+1)/(2n)$ for all $0 \leq j \leq n$. Algorithm P-WSPT preempts any job when it has left just $1/n$ units of processing time and finishes it only after all jobs with a larger release date have been completed. The value of this schedule is $m \cdot (\sum_{j=0}^{n}(r_n + p_n + j/n))$. An optimal off-line algorithm does not preempt any job and yields a schedule of value $m \cdot (\sum_{j=0}^{n}(r_j + p_j + j/n))$. A simple calculation shows that the ratio of the values of the P-WSPT schedule and the optimal schedule goes to 2 when n goes to infinity. □

Of course, Theorem 2.1 subsumes the scheduling problem $\mathrm{P}\,|\,r_j, \mathrm{pmtn}\,|\sum C_j$ as a special case. Thus, this extension of the 2-competitive single-machine Shortest Processing Time (SPT) rule to the parallel machine setting has the same competitive ratio as the analogous extension of Schrage's optimal single-machine SRPT rule [12].

3 Nonpreemptive Parallel Machine Scheduling

Every reasonable on-line algorithm for nonpreemptive scheduling has to make use of some kind of waiting strategy. We refer the reader to [18, Chapter 2] and [11] for comprehensive discussions of related techniques for the single machine. Here, we extend the idea of delaying release dates to the parallel machine problem.

Algorithm 2: SHIFTED WSPT

Modify the release date of every job j to r'_j, where r'_j is some value between $\max\{r_j, \alpha\ p_j\}$ and $r_j + \alpha\ p_j$, for some $\alpha \in (0, 1]$. Whenever a machine becomes idle, choose among the available jobs a job j with highest priority and schedule it on the idle machine.

Note that this is indeed an on-line algorithm; we will later choose α so that we minimize the corresponding competitive ratio. Moreover, for $m = 1$ and $\alpha = 1$, Algorithm SHIFTED WSPT is identical to the algorithm proposed in [11] for $1 \,|\, r_j \,|\, \sum C_j$.

In the analysis of the SHIFTED WSPT algorithm, we make use of the following lower bound on the optimal value of the relaxed problem with trivial release dates, which is due to Eastman, Even and Isaacs [4].

Lemma 3.1. *The value of an optimal schedule for an instance of the scheduling problem* $P \,||\, \sum w_j C_j$ *is bounded from below by*

$$\sum_{j=1}^{n} w_j \sum_{k \le j} \frac{p_k}{m} + \frac{m-1}{2m} \sum_{j=1}^{n} w_j p_j.$$

Let us now analyze the performance of the SHIFTED WSPT algorithm.

Theorem 3.2. *The Algorithm* SHIFTED WSPT *has a competitive ratio of less than* $2 + \max\{1/\alpha, \alpha + \frac{m-1}{2m}\}$ *for the on-line problem* $P \,|\, r_j \,|\, \sum w_j C_j$.

Proof. The algorithm schedules a job j at time r'_j if a machine is idle and j is the job with the highest ratio of weight to processing time among all available jobs; otherwise, j has to wait for some time. The waiting time for j after r'_j is caused by two types of jobs: jobs with lower priority but which started before time r'_j, and jobs with higher priority. Let $L(j)$ denote the set of lower-priority jobs $\ell > j$ with $r'_\ell < r'_j$ and $C_\ell > r'_j$, and let $H(j)$ denote the set of higher-priority jobs $h < j$ that postpone the start of j. Then,

$$C_j \le r'_j + W(L(j)) + W(H(j)) + p_j\ , \tag{1}$$

where $W(L(j))$ and $W(H(j))$ denote the waiting time for j caused by jobs in the sets $H(j)$ and $L(j)$, respectively. Note that the algorithm does not insert idle time on any machine in the time interval between r'_j and the start of job j. Let us analyze the maximal waiting time that can be caused by jobs in the set $L(j)$ and in the set $H(j)$, separately.

1. *Jobs in $L(j)$.* Each machine has at most one job $\ell \in L(j)$. By definition, every one of these jobs satisfies $\alpha\, p_\ell \le r'_\ell < r'_j$. Thus, $p_\ell < r'_j/\alpha$.

2. *Jobs in $H(j)$.* The maximal waiting time for job j arises if all jobs $h < j$ are processed after the completion of jobs in the set $L(j)$ and before the start of job j. If all jobs in $H(j)$ are simultaneously processed on m machines, then at least one machine finishes processing jobs in $H(j)$ after a time period of length at most $\sum_{h\in H(j)} p_h/m$. Hence, the maximal waiting time for job j caused by jobs in $H(j)$ is $\sum_{h<j} p_h/m$.

Inequality (1) and the observations on the sets causing waiting time imply:

$$\begin{aligned} C_j &< (1+\frac{1}{\alpha})\; r'_j + \sum_{h<j} \frac{p_h}{m} + p_j \\ &\le (1+\frac{1}{\alpha})(r_j + \alpha p_j) + \frac{m-1}{2m} p_j + \sum_{h\le j} \frac{p_h}{m} + \frac{m-1}{2m} p_j \\ &= (\; \frac{r_j}{\alpha} + (\alpha + \frac{m-1}{2m})\; p_j \;) + (r_j + p_j) + \sum_{h\le j} \frac{p_h}{m} + \frac{m-1}{2m}\; p_j \;. \end{aligned}$$

Thus, Algorithm Shifted WSPT generates a schedule of value

$$\sum_j w_j C_j < \Big(1 + \max\{\frac{1}{\alpha}, \alpha + \frac{m-1}{2m}\}\Big) \sum_j w_j (r_j + p_j) + \sum_j w_j \big(\; \sum_{h\le j} \frac{p_h}{m} + \frac{m-1}{2m}\; p_j\big) \;.$$

The proof is completed by applying two lower bounds on the optimal value: first, the trivial lower bound $\sum_j w_j(r_j + p_j)$, and second, the lower bound presented in Lemma 3.1. □

A simple calculation shows that the minimum of $\max\{\frac{1}{\alpha}, \alpha + \frac{m-1}{2m}\}$ is attained at $\alpha = (1 - m + \sqrt{16m^2 + (m-1)^2})/(4m) =: \alpha_m$. In particular, $\alpha_1 = 1$.

Corollary 3.3. *The Algorithm* Shifted WSPT *with $\alpha = \alpha_m$ is $(2+1/\alpha_m)$-competitive. The value of this increasing function of m is 3 for the single-machine case and has its limit at $(9+\sqrt{17})/4 \approx 3.28$ for $m \to \infty$.*

Lemma 3.4. *Algorithm* Shifted WSPT *cannot achieve a better competitive ratio than $\max\{2 + \alpha, 1 + \frac{1}{\alpha}\} \ge 2 + \frac{\sqrt{5}-1}{2}$ for $\alpha \in (0, 1]$, on any number of machines.*

Proof. We give two instances from which the lower bound follows. Consider $2m$ jobs released at time 0; half of the jobs are of type I and have unit processing times and weights ε, whereas the other half of the jobs, type II, have processing time $1 + \varepsilon$ and unit weight. The algorithm modifies the release dates and schedules jobs of type I at time $t = \alpha$ first, one on each machine, followed by jobs of type II. The value of the schedule is $m(\alpha + 1)\varepsilon + m(\alpha + 2 + \varepsilon)$. In the optimal schedule, jobs of type II start processing first, at time $t = 0$, such that the value of the schedule is $m(1+\varepsilon)+m(2+\varepsilon)\varepsilon$. For $\varepsilon \to 0$, the ratio between the value of the Shifted WSPT schedule and the optimal schedule goes to $2 + \alpha$.

The second instance consists again of $2m$ jobs: half of the jobs are of type I and have release dates $1 + \varepsilon$, processing times ε and weights $1/m$, whereas the other half of the jobs, type II, are released at time 0 and have processing time $1/\alpha$ and zero weight. Shifted WSPT starts scheduling jobs at time 1 and obtains a solution with a value of at least $1 + 1/\alpha + \varepsilon$. The value of the optimal schedule is $1 + 2\varepsilon$. □

For the special choice $\alpha = \alpha_m$, the first lower bound is tighter and it follows more concretely:

Corollary 3.5. *The Algorithm* SHIFTED WSPT *with* $\alpha = \alpha_m$ *is not better than* $(1 + 7m + \sqrt{16m^2 + (m-1)^2})/(4m)$ *for instances with* m *machines. This means that our performance analysis has a gap of at most* $(m-1)/2m < 0.5$*, and it is tight for the single-machine problem.*

References

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

[2] E.J. Anderson and C.N. Potts. On-line scheduling of a single machine to minimize total weighted completion time. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, San Francisco, CA, pages 548–557, 2002.

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

[4] W.L. Eastman, S. Even, and I.M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, 11:268–279, 1964.

[5] M.X. Goemans, M. Queyranne, A.S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics*, 15:165–192, 2002.

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

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

[8] J.A. Hoogeveen and A.P.A. Vestjens. Optimal on-line algorithms for single-machine scheduling. In W.H. Cunningham, S.T. McCormick, and M. Queyranne, editors, *Proceedings of the Fifth Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 1084 of *Lecture Notes in Computer Science*, pages 404–414, Springer, Berlin, 1996.

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

[10] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[11] X. Lu, R.A. Sitters, and L. Stougie. A class of on-line scheduling algorithms to minimize total completion time. *Operations Research Letters*, 31:232–236, 2003.

[12] C.A. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199–223, 1998.

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

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

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

[16] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

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

[18] A.P.A. Vestjens. *On-line Machine Scheduling*. PhD thesis, Eindhoven University of Technology, Netherlands, 1997.