A class of on-line scheduling algorithms to minimize total completion time

A class of on-line scheduling algorithms to minimize total completion time

X. Lu*† R.A. Sitters* L. Stougie*‡

Abstract

We consider the problem of scheduling jobs on-line on a single machine and on identical machines with the objective to minimize total completion time. We assume that the jobs arrive over time. We give a general 2-competitive algorithm for the single machine problem. The algorithm is based on delaying the release time of the jobs, i.e., making the jobs artificially later available to the on-line scheduler than the actual release times. Our algorithm includes two known algorithms for this problem that apply delay of release times. The proposed algorithm is interesting since it gives the on-line scheduler a whole range of choices for the delays, each of which leading to 2-competitiveness.

We also show that the algorithm is 2α competitive for the problem on identical machines where α is the performance ratio of the Shortest Remaining Processing Time first rule for the preemptive relaxation of the problem.

1 Introduction

Scheduling jobs on a single machine with the objective to minimize the *total completion time* (also sometimes called latency) is a fundamental problem in scheduling theory. Many off-line as well as on-line algorithms have been proposed for this problem. We consider the on-line model where the jobs arrive over time, i.e., jobs are available for processing from their given release times, but the jobs are unknown to the algorithm before their release times. Hoogeveen and Vestjens [5] show that no deterministic on-line algorithm can have a competitive ratio smaller than 2. Several algorithms with competitive ratio matching this lower bound have been given in the literature. Phillips, Stein and Wein [7] presented a 2-competitive algorithm based on the optimal preemptive schedule. Hoogeveen and Vestjens used the idea of shifted release times to obtain a 2-competitive algorithm. The same idea was used by Stougie (cited in Vestjens [10]) who obtained a third algorithm. However, 2-competitveness of this algorithm was never proved.

Our algorithm, which we call SSPT, contains these last two algorithms as a special case. SSPT is described in Section 2. The proof that the algorithm is 2-competitive is found in Section 3. The proof implies directly 2-competitiveness of the algorithms of Hoogeveen and Vestjens and of Stougie. It is simpler and more elegant than the competitiveness proof in [5], and the first competitiveness proof of Stougie's algorithm.

Although the corresponding off-line problem is already strongly NP-hard for the single machine [6], polynomial time approximation schemes for the problem on identical machines have recently been given independently by many people. The approximation scheme given by Afrati et al.[1] even applies if the objective is the total sum of weighted completion times and for a fixed number of unrelated machines.

*Department of Mathematics, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, The Netherlands. e-mail: {x.lu, r.a.sitters, l.stougie}@tue.nl.

†East China University of Science and Technology, Shanghai 200237, China. e-mail: xwlu@ecust.edu.cn.

‡CWI, P.O. Box 94079, 1090GB Amsterdam, The Netherlands. e-mail: stougie@cwi.nl.

In the absence of release times the problem on identical machines is easily solved by scheduling the jobs in order of increasing processing times [3]. This algorithm is well-known as the Shortest Processing Time first rule (SPT).

In the competitive analysis of our algorithm the preemptive version of the problem plays an important role. In this version the processing of any job may be preempted at any time and continued any time later. Schrage [8] showed that, if jobs have release times, the Shortest Remaining Processing Time first rule (SRPT), always produces an optimal preemptive schedule for the single machine problem. The problem with release times and preemption on identical machines is NP-hard [4]. SRPT is an on-line approximation algorithm for this problem. We do not know of a bound smaller than 2 on the worst-case performance ratio of SRPT. The bound of 2 has been shown by Phillips et al. [7].

For the on-line problem of scheduling jobs on m identical parallel machines (without preemption), Chekuri et al. [2] gave an on-line algorithm that is $3 - 1/m$-competitive. They construct a preemptive schedule on one machine, and use the order of completion times of the jobs in this schedule to make a non-preemptive schedule on identical machines. In Section 3 we show, very simply, that, when applied to identical machines, our SSPT-algorithm is 4-competitive for any number of machines, which is worse than the $3 - 1/m$. However, we give a nice property of our algorithm which suggests that this ratio may be improved considerably. More specifically, we show that if the SRPT-rule for the preemptive version of the problem is α-competitive, then SSPT is 2α-competitive for the non-preemptive problem. This makes it interesting to try to improve the bound of 2 on the competitive ratio of SRPT in [7]. We do show that this bound cannot be lower than 12/11 for the preemptive problem on 2 machines. Vestjens [10] showed that for an arbitrary number of machines no preemptive algorithm can be better than 22/21 competitive. We can also show that SSPT has a competitive ratio of at least 2 on any number of machines, which leaves an interesting gap with the lower bound on the competitive ratio of any on-line algorithm for the problem of 1.309 proved by Vestjens [10]. An on-line algorithm with competitive ratio smaller than 2, which we conjecture to exist, would be a divergence from the general phenomenon in on-line scheduling in which competitive ratio's of algorithms for multiple machine problems are higher than those for their single machine counterparts (see [9]).

2 The SSPT-algorithm

Formally, an instance of the on-line scheduling problem is given by n jobs J_j $(j = 1, \ldots, n)$ and m machines M_i $(i = 1, \ldots, m)$. Each job J_j $(j = 1, \ldots, n)$ has a given processing time p_j and release time r_j. A machine can process only one job at a time and a job can be processed by only one machine at a time. The processing of a job cannot start before its release time. If preemption is not allowed, then a job must be processed without interruption on one machine. If preemption is allowed we may interrupt the processing of a job and continue it at the same time on another machine or at a later moment on any machine. The first moment at which job J_j is processed is referred to as its starting time S_j, and the time at which a job is completed is referred to as its completion time C_j. We say that a machine is idle at time t if it is not processing any job during an open interval containing t.

The problems that we study here concern finding non-preemptive schedules for which the sum, taken over all jobs, of completion times is minimal. The sum of completion times is also called the latency in the literature. The algorithm is to construct a feasible schedule on-line, meaning that for any $t > 0$, the schedule restricted to the interval $[0, t]$ must be constructed without knowledge of the jobs that are not released before time t, i.e., those jobs with $r_j > t$.

The idea behind our algorithm is to delay the time at which jobs become available for processing to the algorithm, by increasing the release times and to apply the SPT rule to the available jobs. Since the release times are shifted before applying SPT, we call this algorithm Shifted SPT (SSPT). The idea was used before by Hoogeveen and Vestjens [5], and Stougie (cited in [10]) who delayed the release time of a job j until time $\max(r_j, p_j)$ and $r_j + p_j$, respectively. Both algorithms are 2-competitive for a single machine, although a 2-competitiveness of the algorithm by Stougie

has never been proved before this paper. We generalize their idea and show that the release time of any job may be delayed, independently of any other job, until any moment between these two values without reducing the competitive ratio of the algorithm.

Algorithm SSPT
Make job j available for processing at a time r'_j , where r'_j is an arbitrary number in the interval $[\max\{r_j, p_j\}\ ,\ r_j + p_j]$. At any moment a machine becomes available, schedule from among the available jobs the one with shortest processing time.

Hoogeveen and Vestjens [5] showed that no deterministic on-line algorithm can be better than 2-competitive for the single machine problem. We shall use the idea behind their proof later to give a lower bound on the competitive ratio of our SSPT-algorithm for identical machines. For that reason we include their result.

Theorem 1 *(Hoogeveen and Vestjens [5])*
There is no deterministic on-line algorithm A and $\epsilon > 0$ such that A is $(2-\epsilon)$-competitive for a single machine.

Proof. Assume that some algorithm A is $2 - \epsilon$-competitive for some small number $\epsilon > 0$. We define the following job sequence. Job J_1 with processing time $p_1 = 1$ is released at time 0. If $S_1 > 1 - \epsilon$, then no more jobs are given and the competitive ratio becomes $(S_1 + 1)/1 > 2 - \epsilon$. A contradiction. So we may assume that $S_1 \leq 1 - \epsilon$. Then $n - 1$ jobs are presented at time $1/2 + S_1/2$, each with processing time 0.
The total completion time of the schedule produced by A is at least $n(S_1+1)$, whereas the optimal schedule has total completion time $n(1/2 + S_1/2) + 1$. We see that the competitive ratio tends to 2 if n tends to infinity, which contradicts the assumption that A is $2 - \epsilon$-competitive. □

Vestjens [10] showed that for any number of identical machines, no algorithm can be better than 1.309-competitive.

3 Competitive analysis of SSPT

To prove competitiveness results for the SSPT algorithm we give a few more preliminaries. Assume that σ is a schedule produced by algorithm SSPT for some instance I. Let S_j and C_j be, respectively, the starting and completion time of job J_j ($j \in \{1, \ldots, n\}$) in this schedule.

Given I and σ we define another instance $I(\sigma)$ as follows. For each job J_j of instance I we define a job, also denoted by J_j, for instance $I(\sigma)$ with the same processing time p_j but with shifted release time $\bar{r}_j = \min\{S_j, 2r_j + p_j\}$. Thus, the instance $I(\sigma)$ is obtained from I by shifting the release times over an appropriate length of time. It is not hard to prove that the optimal value of instance $I(\sigma)$ is at most twice the optimal value of instance I.

Lemma 1 *Let $C^*(I)$ and $C^*(I(\sigma))$ be the total completion time of an optimal non-preemptive schedule for, respectively, I and $I(\sigma)$. Then $C^*(I(\sigma)) \leq 2C^*(I)$.*

Proof. Let σ^* be an optimal schedule for I. We define the schedule $\bar{\sigma}^*$ by multiplying all completion times by a factor of 2. To be precise, we define the starting time of job J_j by $\bar{S}^*_j = 2C^*_j - p_j$. Notice that $C^*_j \geq r_j + p_j$. Therefore, $\bar{S}^*_j \geq 2r_j + p_j$, and the schedule $\bar{\sigma}^*$ is feasible for $I(\sigma)$ and has total completion time $2C^*(I)$. □

If preemption is allowed, then the Shortest Remaining Processing Time first rule (SRPT) can be used to construct a feasible preemptive schedule on-line. At any moment the SRPT-rule chooses to process the jobs that have smallest remaining processing time, where the remaining processing time of job j at time t is defined by p_j minus the time that job j has been processed until time t.

We say that a schedule σ is an SRPT schedule for an instance I if it can be obtained by applying the SRPT-rule to instance I.

Our results on the competitiveness of SSPT follow almost directly from the following theorem.

Theorem 2 *For any instance I on identical machines, any schedule σ, produced by algorithm SSPT, is an SRPT-schedule for the corresponding instance $I(\sigma)$.*

PROOF. First we notice that no job in σ starts before its release time in $I(\sigma)$ (by the definition of the release times in $I(\sigma)$). Therefore, σ is a feasible schedule for instance $I(\sigma)$.

We define $J^a(t) = \{j | r'_j \leq t < S_j\}$, and $\bar{J}^a(t) = \{j | \bar{r}_j \leq t < S_j\}$. Notice that $r'_j \leq \bar{r}_j$ and therefore $\bar{J}^a(t) \subseteq J^a(t)$ for all $t \geq 0$.

We are now ready to prove that the SRPT-rule is satisfied, i.e. at any moment t, under the SSPT-schedule σ the machines either process all jobs that are available but not yet completed, or they process the m jobs that have smallest remaining processing time among all jobs that are available at time t in instance $I(\sigma)$.

If $\bar{J}^a(t) = \emptyset$ then the SRPT-rule is clearly satisfied. Now take an arbitrary job $h \in \bar{J}^a(t)$ and an arbitrary job k that is being processed at time t. Such a job k exists since $h \in \bar{J}^a(t) \subseteq J^a(t)$ and therefore no machine is idle at time t. We have to show that $C_k - t \leq p_h$. If job h was available to SSPT at time S_k, i.e. if $h \in J^a(S_k)$, then by definition of SSPT we have $C_k - t \leq p_k \leq p_h$. So now assume $h \notin J^a(S_k)$. By definition of SSPT we have $r_h + p_h > S_k$. On the other hand we have $C_k = S_k + p_k \leq 2S_k$, since job k is not released before time p_k by SSPT. We obtain $C_k - t < 2r_h + 2p_h - t = \bar{r}_h + p_h - t \leq p_h$. □

Schrage [8] showed that the SRPT-rule, produces an optimal preemptive schedule for the single machine problem. Hence, together, Theorem 2 and Lemma 1 imply that SSPT is 2-competitive on a single machine.

Theorem 3 *Algorithm SSPT is 2-competitive for a single machine.*

Unfortunately, SRPT is not optimal for the problem with preemption on parallel identical machines (see Proposition 1 below). Phillips et al. [7] showed that SRPT is 2-competitive for the preemptive problem on identical machines. This bound together with Theorem 2 and Lemma 1 implies immediately that SSPT is 4-competitive for the on-line scheduling problem (without preemption) on identical machines. More generally, we have the following theorem.

Theorem 4 *Algorithm SSPT is 2α-competitive for the on-line scheduling problem on identical parallel machines, where α is the competitive ratio of the SRPT-rule for the preemptive relaxation of the problem.*

We are able to improve the bound to $4 - 1/m$, which is still larger than the bound given by Chekuri et al. [2]. However, finding the performance ratio of SRPT on identical machines seems a very interesting problem on its own. We conjecture that this ratio is much smaller than 2, which, if true, would decrease the competitive ratio of SSPT considerably. We also remark that we are actually interested in the performance of SRPT with respect to the optimal non-preemptive schedule. Notice that the value of a preemptive schedule is a lower bound on the value of an optimal non-preemptive schedule. The following proposition gives a lower bound on the performance of SRPT, which also applies on the ratio between the value of the preemptive SRPT-schedule and the value of an optimal non-preemptive schedule.

Proposition 1 *The algorithm SRPT on identical machines is not α-competitive for $\alpha < \frac{12}{11}$.*

PROOF. We define the instance with 2 machines and jobs 1,2,3,4 and 5 having processing times $p_1 = p_2 = p_4 = p_5 = 1$ and $p_3 = 2 + \epsilon$ and release times $r_1 = r_2 = r_3 = 0$, and $r_4 = r_5 = 2$. SRPT gives completion times $C_1 = 1, C_2 = 1, C_3 = 4 + \epsilon$ and $C_4 = C_5 = 3$, summing to $12 + \epsilon$, whereas the non-preemptive schedule induced by the starting times $S_1 = S_3 = 0$, $S_2 = 1$, $S_4 = 2$, and $S_5 = 2 + \epsilon$ has total completion time $11 + 2\epsilon$. The lemma follows by choosing ϵ sufficiently small. □

Notice that the optimal schedule in the proof is a non-preemptive schedule. The proof is given for the problem on 2 machines. Making m copies of the instance in the proof yields the same lower bound for $2m$ machines. For odd numbers of machines we get a slightly weaker lower bound.

Proposition 1 implies that, through Theorem 4, we will not be able to prove a competitive ratio lower than 24/11 for SSPT. This does not mean that SSPT can not be better than 24/11 competitive. On the other hand, an instance of the scheduling problem with only one job released at time 0 and having processing time 1, shows that SSPT can not have a competitive ratio lower than 2 on any number of machines. Even if SSPT would be 2-competitive, the challenging question remains if an on-line algorithm for the problem on multiple machines exists with competitive ratio strictly less than 2. We recall that Vestjen's lower bound is 1.309 [10]. A competitive ratio strictly smaller than 2 would be a divergence from the general phenomenon in on-line scheduling in which competitive ratio's of algorithms for multiple machine problems are higher than those for their single machine counterparts (see [9]).

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, M. Sviridenko, Approximation schemes for minimizinge average weighted completion time with release dates, Proceedings of the 40th IEEE Symposium on Foundations of Computer Science, New York City, NY, October 1999.

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

[3] R.W. Conway, W.L. Maxwell, L.W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, Massachusetts, 1967.

[4] J. Du, J.Y-T. Leung, G.H. Young, Minimizing mean flow time with release time constraint, *Theoretical Computer Science 75*, 1990, 347–355.

[5] J.A.Hoogeveen, A.P.A. Vestjens, Optimal on-line algorithms for single-machine scheduling, Proceedings 5th International Conference on Integer Programming and Combinatorial Optimization, Vancouver, British Columbia, Canada, June 3-5, 1996, Lecture Notes in Computer Science 1084, Springer, Berlin, 404–414.

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

[7] C. Phillips, C. Stein, J. Wein, Minimizing average completion time in the presence of release dates, Networks and matroids; Sequencing and scheduling, *Mathematical Programming 82*, (1998), 199–223.

[8] L. Schrage, A proof of the shortest remaining processing time processing discipline, *Operations Research 16* (1968), 687–690.

[9] J. Sgall, On-line scheduling, in A. Fiat, G.J. Woeginger (eds.), *Online algorithms: the state of the art*, Lecture Notes in Computer Science 1442, Springer, Berlin, 1998, 196–231.

[10] A.P.A.Vestjens, *On-line Machine Scheduling*, Ph.D. thesis, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, Eindhoven, the Netherlands, 1997.