*Journal of Global Optimization* **27:** 97–103, 2003.



# Competitive Analysis of a Better On-line Algorithm to Minimize Total Completion Time on a Single-machine

JAIRO R. MONTOYA-TORRES
*300mm Simulation and Scheduling, Central CAM and Automation, ST Microelectronics, Z.I. de Rousset, 13106 Rousset Cedex, France (e-mail: montoya@emse.fr)*

(Received 13 November 2002; accepted in revised form 28 February 2003)

**Abstract.** We consider the problem of scheduling jobs on-line on a single machine with the objective of minimizing total completion time. We assume that jobs arrive over time and that release dates are known in advance, but not the processing times. The most important result we are given in this paper is the competitive analysis of a new clairvoyant on-line algorithm for this scheduling problem. We are proving that this deterministic semi-online algorithm, called ST-$\alpha$, is $\sqrt{3}$-competitive, which beats the existing lower bound for non-clairvoyant online algorithms.

**Key words:** competitive analysis, completion time, on-line scheduling, single-machine

## 1. Introduction

Scheduling is a theoretical area of combinatorial optimization that has been inspired by applications in production systems and practical computer science to develop optimization models with very interesting results. In these models, one of the basic assumptions made in deterministic scheduling was, until few years ago, that all information needed to define the problem instance is well known at the initial time. However, this assumption could be not valid in practice. More specifically, when the scheduling models take into account the restriction of jobs arriving over time, jobs do not have to be scheduled immediately as they arrive, and the on-line algorithm has to decide, at each time the machine is idle, either to process one of the unscheduled jobs that have been released or to leave the machine idle so that if an important job arrives it can be processed immediately (Hoogeveen and Vestjens, 1996).

Formally, we consider the problem of scheduling $n$ jobs on a single machine. Job $J_j$ becomes available for processing at its non-negative release date $r_j$ and has processing time $p_j$, for $j$=1,...,$n$. Pre-emption is not permitted, so that no interruption in the processing of a job is allowed. The goal is to find a schedule that minimizes the total completion time $\sum_{j=1}^{n} C_j$, where $C_j$ is the completion time of job $J_j$. This problem is noted in the literature as $1/r_j/\Sigma C_j$.

In the off-line version of the problem, a scheduling algorithm (called an off-line algorithm) has access to complete information of all jobs when constructing a schedule. In this context, the shortest processing time first (STP) rule schedules jobs in non-decreasing order of processing times $p_j$, and it provides the optimal solution where jobs arrive at identical release dates (Smith, 1956). For arbitrary release dates and pre-emption allowed, the Shortest Remaining Processing Time (SRPT) rule gives also an optimal solution (Schrage, 1968). However, for our case, pre-emption is not allowed and the problem is known to be strongly NP-hard (Lenstra et al., 1977).

Until now, in an online environment, jobs arrive over time, as defined by their release dates. The number of jobs is not known in advance, and no information is known about any future job. However, immediately a job $J_j$ arrives in the system at time $r_j$, its processing time $p_j$ is known. These on-line scheduling models consider that no future information is given to the algorithm when scheduling a job (Sgall, 1999). So that, those scheduling algorithms are called non-clairvoyants. For this version of our problem, several authors have proposed very interesting algorithms (Phillips et al., 1995; Hoogeveen and Vestjens, 1996, Lu et al., 2002), with competitive ratio of 2, and they are optimal (Hoogeveen and Vestjens, 1996). A randomized online algorithm was proposed by Chekuri et al. (1997) with an expected competitive ratio of $e/(e\text{-}1)$ and this is also optimal (Vestjens, 1997).

In this paper, we are interested in analyzing a new on-line scheduling problem where partial knowledge of future information is known (in advance). Little is known about clairvoyant on-line (or semi-online) scheduling algorithms. Some other models are given by Liu et al. (1996), Kellerer et al. (1997), Azar and Regev (1998), and Seiden et al. (1998), who analyzed the problem of scheduling jobs within different shop configurations to maximize productivity (or minimize the makespan) as the objective function. We analyze here the semi-online scheduling problem on a single machine where the objective function is the total completion time, or equivalently, total flow time, total waiting time, and latency (Conway et al., 1967). Particularly, we are given the competitive analysis of a new clairvoyant on-line algorithm, called SPT-$\alpha$, where decision strategies are based on the well-known SPT rule anticipating the possibility of processing a job before the next arrival (Montoya Torres, 2002).

The remainder of this paper is arranged as follows. In Section 2 we present the algorithm SPT-$\alpha$, for which the competitive analysis is given in Section 3. Finally, Section 4 presents some concluding remarks.

## 2. The SPT-$\alpha$ Algorithm

In this section, we are describing a clairvoyant on-line algorithm for the scheduling problem on a single machine where jobs arrive over time and release dates are well-known at the initial time of scheduling. This algorithm, called SPT-$\alpha$ and proposed firstly by Montoya Torres (2002), is inspired from the idea of $\alpha$-schedules (or frac-

tional schedules) introduced by Phillips et al. (1995) to schedule jobs on parallel machines. This idea was later used by several authors in different one-machine scheduling problems (Hall et al., 1997; Schulz and Skutella, 1999; Goemans et al., 2002). A $\alpha$-schedule is a schedule given by an algorithm $A$ that schedules jobs according to the shortest remaining processing time (SRPT) rule and then establishes a non-preemptive scheduling list according to the order in which jobs were executed during $\alpha \times p_j$ units of time (i.e. $\alpha \times p_1 \leqslant \alpha \times p_2 \leqslant ... \leqslant \alpha \times p_n$) in this SRPT– pseudo-schedule.

In this context, a $\alpha$-scheduler takes non-preemptive scheduling decisions once the $\alpha \times p_j$ fraction of a job has been finished on the machine. Hence, we say that these decisions are taken *a posteriori*. Scheduling decisions could be taken *a priori* if an on-line algorithm knew future release dates in advance. So that, for this new semi-online algorithm, the SPT rule is used to create the priority list of jobs, and $\alpha$-points are used as the execution strategy, if and only if a job can be executed during at least $\alpha \times p_j$ units of time before the next arrival.

Algorithm SPT-$\alpha$:

1. Chose a value of $\alpha \in (0,1]$ before the first input.
2. Classify arrived jobs in a list $L$ according to the SPT rule (i.e. $p_{[1]} \leqslant p_{[2]} \leqslant \ldots \leqslant p_{[n]}$).
3. Let $t$ be any time the machine is idle and $r_k$ be the next release date (i.e. $r_k = \min r_j / r_j > t$). If $(t + \alpha \times p_{[1]} \leqslant r_k)$, then schedule $p_{[1]}$ on the machine at time $t$. If there is more than one job, then schedule the one with $\min\{r_i\}$.
4. If there is no candidate to be scheduled, then wait for the release date $r_k$.
5. If no more jobs arrive before $t$, schedule jobs according to SPT rule.

SPT-$\alpha$ turns in $O(n^2 \Sigma p_j)$ time. So that, it is a pseudo-polynomial time algorithm. However, computational experiments have been showed that SPT-$\alpha$ is, in average, as competitive as the non-clairvoyant *optimal* on-line algorithms existing in the literature. It has been also showed that the experimental worst-case analysis gives a lower value compared against those algorithms and others semi-online heuristics (Montoya Torres, 2002), proving the practical effectiveness of this algorithm.

## 3. Competitive Analysis of the SPT-$\alpha$ Algorithm

The analysis of on-line algorithms is habitually presented by evaluating the performance of an algorithm against an optimal off-line solution for the worst-case analysis. Formally, for a scheduling problem $J$ of size $n$, let $C_A(J)$ denote the value of the total completion time given by the on-line algorithm $A$. The optimal clairvoyant off-line algorithm OPT minimizes $C(J)$ for each $J$. The performance

ratio of algorithm $A$ is defined as

$$R_A(n) = \sup C_A(J)/C_{\mathrm{OPT}}(J).$$

An online scheduling algorithm is said $f(n)$– competitive if $R_A(n) \leqslant f(n)$, and the algorithm is said to be competitive is there exists a constant $k$ for which it is $k$–competitive (Motwani et al., 1994). It means that the solution obtained by applying an on-line algorithm $A$, is at most $k$ times the value given by the optimal off-line algorithm for any instance of the problem.

For the problem of scheduling jobs on-line on a single machine in order to minimize total completion time, Hoogeveen and Vestjens (1996) showed that no deterministic non-clairvoyant on-line algorithm can have a competitive ratio smaller than 2. In this section we will show that it exists an on-line (clairvoyant) algorithm for which the competitive ratio is smaller than this bound. More precisely, we show that the semi-online scheduling algorithm SPT-$\alpha$ is a $\sqrt{3}$-competitive algorithm, and this result beats the existing lower bound for non-clairvoyant online algorithms.

From now on, we call $\sigma$ the schedule given by the SPT-$\alpha$ algorithm and we assume that jobs are numbered according to their position in this schedule $\sigma$ (i.e., $C_1 \leqslant C_2 \leqslant \ldots \leqslant C_n$). Actually, this schedule $\sigma$ is composed of a sequence of blocks, where each block is a set of jobs executed continually on the machine without any idle time. Between two blocks, the idle time of the machine is due to SPT-$\alpha$'s decisions to not to schedule a job and to wait for the next arrival. So that, sequencing decisions taken inside each block do not influence the sequencing decisions concerning the jobs scheduled after the idle time, and *vice-versa*. Hence, any instance can be split into several independent smaller instances, where jobs are ordered according to the SPT rule, and such that the last job of a block is larger than the first job of the succeeding block, if it exists. We denote these blocks by $B_1,\ldots,B_k$. Block $B_{i+1}$ consists of the jobs $J_{b(1)+1},\ldots,J_{b(i+1)}$, where the indexes *b(i)* are determined recursively as b(i)= $\min\{j > b(i-1) | p_j > p_{j+1}\}$ and the number $k$ of blocks in which the schedule is partitioned, is given by the recursion scheme.

Let $m$ be the index of the job that has the largest processing in the first $i$ blocks (i.e., $p_m = \max_{1 \leqslant j \leqslant b(j)}\{p_j\}$). We define a pseudo-schedule $\psi$ for the schedule $\sigma$ as follows. The order in $\psi$ is the same that in $\sigma$ but the job $J_j$ in $B_{i+1}$ starts at time $s_j(\psi) = s_j(\sigma) - 2\alpha p_m$, for $0 < \alpha \leqslant 1$. We can easy verify that $\psi$ is not an acceptable schedule because some jobs overlap and some jobs begin before their release dates. It is also important to note that the amount that each job is shifted backward only increases with time and hence $\psi$ contains no idle time for the machine, where the idea behind is that all jobs that are not in SPT order due to the greediness decisions of the heuristic SPT-$\alpha$, are now scheduled before or at the time they would have been scheduled if the algorithm would have decided to postpone to execute the largest job. We are using some ideas given by Vestjens (1997) to show in lemmas 1 and 2 that $\psi$ is comparable to both the optimal preemptive schedule for any problem instance, noted hereinafter by $\phi$, and the schedule $\sigma$ given by

SPT-$\alpha$. Using these relations, we will then prove the competitiveness ratio of this algorithm.

LEMMA 1. *$C_j(\sigma)$-$C_j(\psi)\leqslant 2\alpha C_j(\phi)$, for all $J_j$ belonging to any instance of the problem.*

*Proof.* Consider an arbitrary job $J_j$ and suppose that $J_j \in B_{i+1}$. Because of the definition of the pseudo-schedule $\psi$, we have that $C_j(\sigma) - C_j(\psi) = 2\alpha p_m$. If $p_j < p_m$, then $r_j > s_m(\sigma) \geqslant p_m$, because the algorithm SPT-$\alpha$ always schedule a job if it is possible to execute it for at least $\alpha p_j$ units of time on the machine. Therefore, either $p_j \geqslant p_m$ or $r_j > p_m$, which implies that $p_m \leqslant r_j + p_j \leqslant C_j(\phi)$. Hence, $C_j(\sigma) - C_j(\psi) \leqslant 2\alpha C_j(\phi)$.

LEMMA 2. *$\Sigma C_j(\psi) \leqslant \Sigma C_j(\phi)$.*

*Proof.* Consider any instance, say $J$, of the problem. To prove this lemma, we use this instance $J$ to create a new instance $J'$ which consists of all jobs in $J$ with the same processing times and with release dates defined as $r_j' = \min\{r_j, s_j(\psi)\}$.

Let $\phi'$ be the optimal preemptive schedule for the instance $J'$. We will show that no job will start at an earlier time in $\phi'$ than in $\psi$. Suppose the contrary, that at least one job starts in $\phi'$at an earlier time than in $\psi$, and let $J_j$ be the first such job. Suppose also that $J_j \in B_{i+1}$ in $\sigma$. If $p_j \geqslant p_m$, then all jobs scheduled before $J_j$ in $\psi$ have a higher priority than $J_j$ (i.e. either they have a smaller processing time or they have equal processing time but a smaller release date). This implies that in $\phi'$ these jobs also have a higher priority and hence they will be scheduled before $J_j$, contradicting the fact that $s_j(\phi') < s_j(\psi)$.

Now consider the case $p_j < p_m$. because of the assumption, we have that $r_j' \leqslant s_j(\phi') < s_j(\psi)$, which implies again $r_j < s_j(\psi)$.

Consider this time the jobs within the interval $[r_j, s_j(\psi)]$. Because $J_j$ was available at the beginning of these jobs, those must have a higher priority than $J_j$, which means that the SRPT algorithm prefers them to the job $J_j$. Since we have assumed that $J_j$ was the first job starting earlier in $\phi'$ than in $\psi$, these jobs must occupy the machine in $\phi'$ until time $s_j(\psi)$, from which we deduce that $s_j(\phi') \geqslant s_j(\psi)$. Therefore, no job starts earlier in $\phi'$ than in $\psi$, which implies that $C_j(\phi') \geqslant C_j(\psi)$ $\forall\, j=1,\ldots,n$.

Because of the definition of the instance $J'$, release dates $r_j'$ are smaller than or equal to release dates $r_j$ of the original instance $J$, we have that $\Sigma C_j(\phi') \leqslant \Sigma C_j(\phi)$, that implies also that $\Sigma C_j(\psi) \leqslant \Sigma C_j(\phi') \leqslant \Sigma C_j(\phi)$.

THEOREM 3. *For the problem of scheduling jobs arriving over time to minimize total completion time when ready times are well known at the initial time, the deterministic online algorithm SPT-$\alpha$ is $(2\alpha+1)$–competitive.*

*Proof.* Combining Lemmas 1 and 2 we obtain $\Sigma C_j(\sigma) \leqslant 2\alpha\ \Sigma C_j(\phi) + \Sigma C_j(\psi)$. Since the value of an optimal preemptive schedule is a lower bound of the value of an optimal non-preemptive schedule, we have that $\Sigma C_j(\sigma) \leqslant (2\alpha+1)\ \Sigma C_j(\phi) \leqslant (2\alpha+1)\ \Sigma C_j(\pi)$

THEOREM 4. *For the on-line scheduling problem where jobs arrive over time and release dates are known in advance, there are some instances for which the deterministic algorithm SPT-$\alpha$ is 3/(2$\alpha$+1)–competitive.*

*Proof.* Consider an instance $J$ of $n$ jobs. The first job arrives at time $r_1$=0 and has processing time $p_j$=$n$, and the ($n$-1) other jobs arrive at times $r_j$=$nj\alpha$ and have processing times $p_j$=1, for $j$=2,...,$n$. For this problem instance, SPT-$\alpha$ schedules the first job followed by all the unitary jobs, yielding to a total completion time of

$$n + \sum_{j=1}^{n}(n + j) = 3(n^2 + n)/2.$$

The optimal off-line solution will wait for the second arrival to schedule all the unitary jobs followed by the largest one. So that, the optimal solution yields to a total completion time of

$$\sum_{j=1}^{n}(\alpha n + j) + \alpha n + 2n = ((2\alpha + 1)n^2 + (2\alpha + 5)n)/2.$$

Hence, the performance ratio of SPT-$\alpha$ for this instance is: $R_A(J)$= $(3(n^2+n))/((2\alpha+1)n^2+(2\alpha+5)n)$.

If we let $n$ tend to infinity, SPT-$\alpha$ is 3/(2$\alpha$+1)– competitive algorithm.

LEMMA 5. *The SPT-$\alpha$ algorithm is 3–competitive.*

*Proof.* Combining Theorems 3 and 4, simple algebra shows that for $\alpha$=($\sqrt{3}$-1)/2; the SPT-$\alpha$ algorithm is 3 – competitive.

## 4. Concluding Remarks

In this paper, we analyzed the problem of scheduling jobs arriving over time on a single machine, where the objective function was the minimization of the total completion time. More precisely, we studied a new on-line problem, where release dates are all known at the initial time. We analyzed the competitiveness of the semi-online algorithm SPT-$\alpha$, that is $\sqrt{3}$-competitive, which beats the existing lower bound for non-clairvoyant online algorithms. The main idea of the scheduling strategy of this algorithm is to anticipate the possibility of scheduling a job to not to wait for a long period of time before the next arrival time, as well as the use of the well-known SPT rule to establish the priority order of job's execution.

## References

Azar, Y. and Regev, O. (1998), Online bin stretching. In *Proceedings of the International Workshop on Randomization and Approximation Techniques in Computer Science*, Barcelona, Spain.

Chekuri, C., Motwani, R., Natarajan, B. and Stein, C. (1997), Approximation techniques for average completion time scheduling. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA.

Conway, R., Maxwell, W. and Miller, L.W. (1967), *Theory of Scheduling*. Addison-Wesley, Reading, MA.

Goemnas, M.X., Queyranne, M., Schulz, A., Skutella, M. and Wang, Y (2002). Single machine scheduling with release dates, *Journal on Discrete Mathematics* 15, 65–192.

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

Hoogeveen, J.A. and Vestjens, A.P.A. (1996), Optimal on-line algorithms for single-machine scheduling. In: Cunningham, W.H., McCornick, S.T. and Queyranne, M. (Eds.), *Integer Programming and Combinatorial Optimization*, Springer, Berlin.

Kellerer, H., Kotov, V., Speranza M. and Tuza, Z. (1997), Semi-online algorithms for the partition problem. *Operations Research Letters* 21, 235–242.

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

Liu, W.P., Sidney, J.B. and van Vliet, A. (1996). Ordinal algorithms for parallel machine scheduling. *Operations Research Letters* 18, 223–232.

Lu, X. Sitters, R., and Stougie, L. (2002). A class of on-line scheduling algorithms to minimize total completion time. *$10^{th}$ European Symposium on Algorithms*. Rome, Italy, Submitted.

Montoya Torres, J.R. (2002), *Une etude de l'influence de l'information anticipee en ordonnancement dynamique*. Master's thesis. Institut National Polytechnique de Grenoble, France.

Motwani, R., Phillips, S. and Torng, E. (1994). Non-clairvoyant scheduling. *Theoretical Computer Science* 130, 17–47.

Phillips, C., Stein, C. and Wein, J. (1995). Minimizing average completion time in the presence of release dates. *Mathematical Programming* 82, 199–223.

Schrage, L.E. (1968). A proof of the optimality of the shortest remaining processing time discipline, *Operations Research* 16, 678–690.

Schulz, A. and Skutella, M. (1999), The power of $\alpha$-points in preemptive single machine scheduling. *Technical Report Department of Mathematics*, Preprint 639/1999. Technical University of Berlin, Berlin.

Seiden, S., Sgall J. and Woeginger, G. (1998), Semi-online scheduling with decreasing job sizes, *Operations Research Letters* 27(5), 215–221.

Sgall, J. (1999), On-line scheduling – A survey. In: Fiat, A. and Woeginger, G.I. (Eds.), *On-line algorithms: The State of the Art*, Springer-Verlag, New York.

Smith, W.E. (1956), Various optimizers for single-stage production. *Naval Research and Logistics Quarterly* 3, 59–66.

Vestjens, A.P.A. (1997), *On-line machine scheduling*. PhD thesis. Eindhoven University of Technology, The Netherlands.