A note on "An optimal online algorithm for single machine scheduling to minimize total general completion time"

Sheng Yu*
School of Management
Xi'an Jiaotong University
a.sheng@stu.xjtu.edu.cn

Prudence W. H. Wong
Department of Computer Science
University of Liverpool
pwong@liverpool.ac.uk

August 25, 2011

Abstract

In this paper we study the problem of online scheduling of jobs with release times on a single machine to minimize the total general completion time $\sum C_j^\alpha$, where C_j is the completion time of job J_j and $\alpha \geq 1$ is a constant. It has been shown in the paper by Liu et al., namely, "An optimal online algorithm for single machine scheduling to minimize total general completion time" [2] that the lower bound on the competitive ratio of any online algorithm is 2^α. The authors also attempted to prove that the online algorithm DSPT (Delayed Shortest Processing Time) is 2^α-competitive. Unfortunately, Lemma 2 in the paper, which is crucial in the proof, is indeed incorrect. This leaves the performance of DSPT as an open question. The contribution of this paper is two-fold. First, we give a counterexample to this lemma and an explanation of the flaw in the argument. Second, we give a proof that DSPT is 2^α-competitive. Together with the lower bound by Liu et al., this implies that DSPT is an optimal online algorithm for minimizing the total general completion time.

Keywords: Online scheduling, total general completion time, delayed shortest processing time, competitive analysis

1 Introduction

Scheduling jobs on a single machine in order to minimize the total completion time is a fundamental scheduling problem. Here preemption is not allowed and online algorithms are considered [5]. The input is a set $\mathcal{J}$ of n jobs $J_1, J_2, \cdots, J_n$. Each job J_j is released at time r_j with processing time p_j. The job has to be processed contiguously for p_j time units. An online algorithm has to determine at any time which job to execute without future information about jobs that have not been released yet. In a schedule σ, we denote the start time and completion time of J_j by $S_j(\sigma)$ and $C_j(\sigma)$, respectively. When the context is clear, we simply use S_j and C_j. The total completion time is defined as $\sum_{j=1}^n C_j$. Furthermore, the general completion time [2] attempts to characterize the property that dissatisfaction increases with delay in processing in a manner of a power function. The *general completion time* of J_j, is defined as C_j^α where $\alpha \geq 1$ is a constant. The objective of the problem is to minimize the total general completion time, i.e., $\sum_{j=1}^n C_j^\alpha$. The performance of an online algorithm is typically measured by competitive analysis [1]. An online algorithm is said to be c-competitive if for all input job set, its total general completion time is at most c times that of the optimal offline algorithm.

*The work is done while the author is visiting University of Liverpool.

The problem of minimizing total completion time $\sum C_j$ has been studied by Vestjens [5] (Chapter 2), who proposed the algorithm DSPT (Delayed Shortest Processing Time) and showed that it is an optimal online algorithm with competitive ratio 2. Other 2-competitive algorithms have also been proposed [3, 4]. Liu et al. [2] extended the work by Vestjens [5] to total general completion time and showed a lower bound of 2^α. They also showed that DSPT is 2^α-competitive. However, the proof relies on an intermediate result (Lemma 2 in [2]), which is not correct. In this note, we give a counterexample to Lemma 2 in [2]. We also show that DSPT is indeed 2^α-competitive. Our proof also follows the same line as [5] and [2], and we produce a different argument to complete the proof by [2].

In Section 2, we repeat the description of DSPT and outline the main arguments in [2, 5]. In Section 3, we give a counterexample to Lemma 2 in [2]. In Section 4, we prove that DSPT is 2^α-competitive.

2 Algorithm DSPT and its analysis

Algorithm. For the sake of completeness, we first describe the online algorithm DSPT (Delayed Shortest Processing Time). Consider a job instance $\mathcal{J}$. DSPT runs as follows.

Step 1: Consider time t when the machine is idle. If an unscheduled job is available at that time, let J_j be the one with the shortest processing time. Ties are broken by taking the one with the earliest release time.

Step 2: If $t \geq p_j$, then schedule J_j at t; otherwise, wait until time p_j or until a new job arrives, whichever happens first;

Step 3: If all jobs are scheduled, stop; otherwise, go to Step 1.

Analysis in [2, 5]. We now outline the main ideas of the proof in [5] for $\sum C_j$ and how [2] extends the proof to $\sum C_j^\alpha$. Let σ and π be the DSPT schedule and the optimal offline schedule, respectively. First of all, Vestjens [5] showed that we only need to consider instances such that DSPT schedules jobs contiguously as a block without idle time (stated as Lemma 1 below). The main observation for this is that if there is a job instance $\mathcal{J}$ such that the total completion time of DSPT is c times that of the optimal offline algorithm, one can prove that there is another job instance $\mathcal{J}'$ such that DSPT schedules jobs contiguously without idle time and its total completion time is at most c times that of the optimal offline algorithm. This is also true for total general completion time as well [2]. Note that such schedule may start with an idle time which is due to Step 2 of DSPT .

Lemma 1 ([2, 5])**.** *The* DSPT *schedule σ consists of a single block: it possibly starts with an idle time after which all jobs are executed contiguously.*

The proof [5] proceeds with defining a new job instance $\mathcal{J}'$ based on $\mathcal{J}$ and σ, and relating σ with the optimal offline preemptive schedules for $\mathcal{J}$ and $\mathcal{J}'$, and in turn with the optimal offline non-preemptive schedule for $\mathcal{J}$. We first describe how $\mathcal{J}'$ is defined. We number the jobs according to the execution order in DSPT schedule σ. To simplify the argument, we define a dummy job J_0 with $p_0 = S_1(\sigma)$, which will not be scheduled, yet we define $S_0(\sigma) = p_0$. We partition the schedule σ into subblocks $B_1, B_2, \cdots, B_k$ such that a subblock is a maximal contiguous sequence of jobs ordered from shortest to longest processing time, i.e., the last job of a subblock (except the last subblock) has longer processing time than the first job of the following subblock. We denote by $b(i)$ the index of the last job in block B_i. With $b(0) = 0$, $b(i)$ is formally defined as $b(i) = \min\{j > b(i-1) \mid p_j > p_{j+1}\}$, and for the last block B_k, $b(k)$ is defined as n. In other words, B_i consists of jobs $J_{b(i-1)+1}, \cdots, J_{b(i)}$. Furthermore, we define $m(i)$ to be the largest

index of the job that has the longest processing time in the first i subblocks, i.e., $p_{m(i)} = \max_{0 \le j \le b(i)} p_j$ or $p_{m(i)} = \max_{0 \le j \le i} p_{b(j)}$.

We now define a *pseudo-schedule* ψ based on schedule σ and from that define a new instance $\mathcal{J}'$. Vestjens [5] showed that we can bound the total completion time of σ in terms of the optimal offline schedule for $\mathcal{J}'$ and $\mathcal{J}$. The order of the jobs in ψ is the same as in σ, but J_j in B_{i+1} starts at time $S_j(\psi) = S_j(\sigma) - p_{m(i)}$. Then, $C_j(\psi) = C_j(\sigma) - p_{m(i)}$. Note that ψ is not a genuine schedule and the execution of jobs may overlap with each other. Based on the instance $\mathcal{J}$ and the pseudo-schedule ψ for $\mathcal{J}$, we define a new instance $\mathcal{J}'$ which consists of all jobs in $\mathcal{J}$. The processing times of the jobs remain the same but the release time of job J_j is defined as $r'_j = \min\{r_j, S_j(\psi)\}$. Let ϕ and ϕ' be the optimal preemptive schedules for $\mathcal{J}$ and $\mathcal{J}'$, respectively. Vestjens [5] showed a number of properties which we capture in the following lemma.

Lemma 2 ([5])**.** (i) *For any job* J_j, $C_j(\sigma) \le C_j(\phi) + C_j(\psi)$. (ii) *For any job* J_j, $C_j(\psi) \le C_j(\phi')$. (iii) $\sum C_j(\phi') \le \sum C_j(\phi)$. (iv) $\sum C_j(\phi) \le \sum C_j(\pi)$.

Roughly speaking, Statement (i) is due to the fact that DSPT does not start a job J_j in B_{i+1} until p_j and noting that $p_{m(i)} \le r_j + p_j$. Statement (ii) is proved by a claim that no job will start earlier in ϕ' than in ψ (see Lemma 2.4 in [5]). Statement (iii) holds because any valid schedule for $\mathcal{J}$ including ϕ is a valid schedule for $\mathcal{J}'$ (the release times in $\mathcal{J}'$ is at most that in $\mathcal{J}$). Statement (iv) is because the cost of an optimal preemptive schedule is always at most that of an optimal non-preemptive schedule.

Combining all the statements in Lemma 2, one can derive the following corollary, implying that DSPT is 2-competitive for minimizing total completion time $\sum C_j$.

Corollary 3 ([5])**.** $\sum C_j(\sigma) \le 2 \sum C_j(\pi)$.

To analyze the performance of DSPT with respect to the total general completion time $\sum C_j^\alpha$, a claim is made in [2] (Lemma 2) on the relationship of $C_j(\psi)$ and $C_j(\phi)$ for any job J_j, which we state as Claim 4. If correct, Claim 4 would have implied that $\sum C_j^\alpha(\sigma) \le 2^\alpha \sum C_j^\alpha(\pi)$. Unfortunately, as we will show in Section 3, this claim is incorrect and hence the competitive ratio of DSPT for $\sum C_j^\alpha$ remains unknown.

Claim 4 (incorrect Lemma 2 in [2])**.** *For any job* J_j, $C_j(\psi) \le C_j(\phi)$.

3 Counterexample to Claim 4

In this section, we give a counterexample $\mathcal{J}$ to Claim 4, in particular, we give an instance $\mathcal{J}$ and show that $C_j(\psi) > C_j(\phi)$ for some $J_j \in \mathcal{J}$. Let ε be a sufficiently small positive number. The release times and the processing times of jobs in $\mathcal{J}$ are listed in Table 1.

job J_j	release time r_j	processing time p_j
J_1	0	3
J_2	0	5
J_3	0	7
J_4	$11 + \varepsilon$	1
J_5	$18 + \varepsilon$	0.5
J_6	$11 + 2\varepsilon$	2

Table 1: Input job set $\mathcal{J}$ of the counterexample.

According to DSPT, the processor is idle for 3 time units before starting J_1 and then σ continues to execute J_2, J_3, J_4, J_5 and J_6 in that order (see Figure 1 (a)).

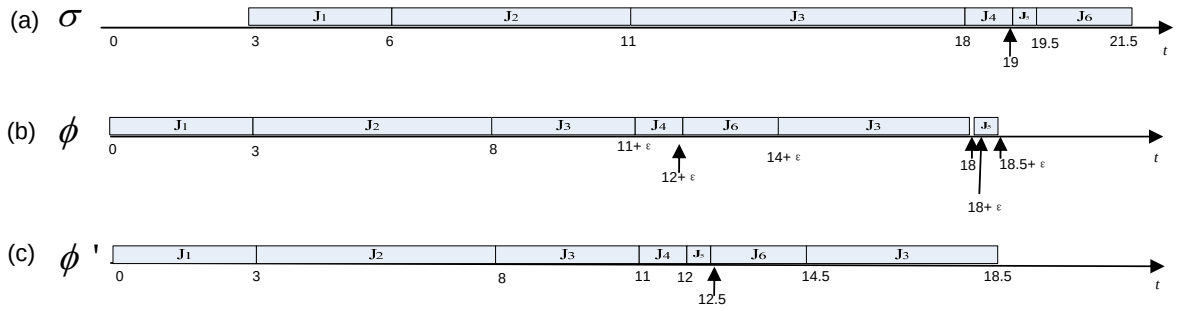

Figure 1: Schedules of σ, ϕ and ϕ'

J_j	r_j	$S_j(\sigma)$	$C_j(\sigma)$	$S_j(\psi) = S_j(\sigma) - P_{m(i)}$	$C_j(\psi)$	r'_j	$C_j(\phi)$	$C_j(\phi')$
J_1	0	3	6	0=3-3	3	0	3	3
J_2	0	6	11	3=6-3	8	0	8	8
J_3	0	11	18	8=11-3	15	0	18	18.5
J_4	$11+\varepsilon$	18	19	11=18-7	12	11	12+ε	12
J_5	$18+\varepsilon$	19	19.5	12=19-7	12.5	12	18.5+ε	12.5
J_6	$11+2\varepsilon$	19.5	21.5	12.5=19.5-7	14.5	11+ε	14+ε	14.5

Table 2: The values of $S_j(\psi)$, r'_j, $C_j(\phi)$ and $C_j(\phi')$ for the counterexample.

The optimal preemptive schedule ϕ first completes J_1 and J_2 at time 3 and 8, respectively. Then J_3 is executed for $3+\varepsilon$ time units until J_4 arrives and preempts J_3. Then ϕ completes J_4 and J_6 at time $12+\varepsilon$ and $14+\varepsilon$, respectively. Job J_3 is then resumed and completed at time 18. Finally, as soon as J_6 arrives at time $18+\varepsilon$, ϕ executes J_6 until it is completed at $18.5+\varepsilon$. Figure 1 (b) shows the schedule ϕ.

The start time and completion time of the infeasible schedule ψ are listed in Table 2. Based on that, one can compute r'_j for each job J_j (see Table 2). For the new instance $\mathcal{J}'$ whose jobs have release times r'_j, the optimal preemptive schedule ϕ' runs as follows. J_1 and J_2 are first executed and completed at time 3 and 8, respectively. Then J_3 is executed but it is now preempted earlier at time 11 when J_4 arrives. J_4, J_5 and J_6 are then executed consecutively to completion at time 12, 12.5 and 14.5, respectively. Finally, J_3 is resumed to completion at time 18.5. Figure 1 (c) shows the schedule ϕ'.

In Table 2, $C_6(\psi) = 14.5 > C_6(\phi) = 14+\varepsilon$. By picking an arbitrarily small ε, we see that Claim 4 does not hold. We now explain the flaw in the proof of this claim in [2]. The proof of the claim is based on proving two properties of any job J_j: (1) $C_j(\psi) \leq C_j(\phi')$ and (2) $C_j(\phi') \leq C_j(\phi)$. Property (1) is indeed correct but the argument of Property (2) (towards the end of the proof of Lemma 2 in [2]) that it is a consequence of $r'_j \leq r_j$ is incorrect. The completion time of the optimal preemptive algorithm is not an increasing function of the release time and therefore we cannot conclude Property (2) and neither Lemma 2 in [2]. As a result, the performance of DSPT for $\sum C_j^\alpha$ is still open for $\alpha > 1$.

4 Competitive ratio of DSPT

In this section, we give a proof that DSPT is 2^α-competitive for $\sum C_j^\alpha$. Note that in [2], α is assumed to be at least 1. The proof in this section actually works for any non-negative number. The idea of the proof is as follows. By definition of ψ, we know that $C_j(\sigma) \leq C_j(\psi) + p_{m(i)}$ for a job J_j in block B_{i+1}. In Lemma 5, we relate $C_j(\sigma)$ and $p_{m(i)}$ so that we can express $C_j(\sigma)$ in terms of $C_j(\psi)$ alone. In Lemma 2 (ii), it is shown that $C_j(\psi) \leq C_j(\phi')$. We then show in Lemma 6 that $\sum C_j^\alpha(\phi') \leq \sum C_j^\alpha(\pi)$ and as a result we can conclude the competitive ratio of DSPT in Theorem 7.

Lemma 5. *Consider a job J_j in subblock B_{i+1}. We have* (i) $C_j(\sigma) \geq 2p_{m(i)}$; (ii) $C_j(\sigma) \leq 2C_j(\psi)$.

Proof. (i) We prove the statement by induction. According to DSPT, σ schedules a job J_j to start at time at least p_j and completes at time at least $2p_j$. For any job $J_j \in B_1$, the completion time is at least that of J_1, which equals to $2p_1 = 2p_{m(0)}$. Thus, the base case holds.

Assume that the statement holds for all jobs in $B_1, \cdots, B_i$. This means that for any such job $J_{j'}$, $C_{j'} \geq 2p_{m(i-1)}$. We are going to show that the last job J_ℓ in B_i completes at time at least $2p_{m(i)}$, implying any job in B_{i+1} also completes at time $2p_{m(i)}$ or later. If $p_{m(i)} = p_{m(i-1)}$, then the statement holds for $i+1$. Otherwise, it means that J_ℓ is the job with the largest processing time in $B_1, \cdots, B_i$ and $p_{m(i)} = p_\ell$. In this case, $C_\ell \geq 2p_\ell = 2p_{m(i)}$ and the statement holds for $i+1$.

(ii) By definition of ψ, we know that $C_j(\sigma) \leq C_j(\psi) + p_{m(i)}$. Together with (i), we have $C_j(\sigma) \leq C_j(\psi) + \frac{1}{2}C_j(\sigma)$ and therefore, $C_j(\sigma) \leq 2C_j(\psi)$. □

The following lemma relates the optimal preemptive schedule for $\mathcal{J}'$ and the optimal (non-preemptive) schedule for $\mathcal{J}$. The argument is similar to the result by Vestjens [5] as stated in Lemma 2 (iii) and (iv).

Lemma 6. $\sum C_j^\alpha(\phi') \leq \sum C_j^\alpha(\phi) \leq \sum C_j^\alpha(\pi)$.

Proof. As the release time of a job in $\mathcal{J}'$ is at most that of the corresponding job in $\mathcal{J}$, any valid schedule for $\mathcal{J}$ including ϕ is a valid schedule for $\mathcal{J}'$. Therefore $\sum C_j^\alpha(\phi') \leq \sum C_j^\alpha(\phi)$.

On the other hand, $\sum C_j^\alpha(\phi) \leq \sum C_j^\alpha(\pi)$ because the cost of an optimal preemptive schedule is always at most that of an optimal non-preemptive schedule. The lemma follows. □

Combining Lemmas 5 and 6 with Lemma 2 (ii), we conclude the following theorem.

Theorem 7. *The online algorithm* DSPT *is 2^α-competitive for minimizing $\sum C_j^\alpha$.*

Proof. By Lemma 5 (ii), we have $\sum C_j^\alpha(\sigma) \leq 2^\alpha \sum C_j^\alpha(\psi)$. By Lemma 2 (ii), the latter is at most $2^\alpha \sum C_j^\alpha(\phi')$, which is in turn at most $2^\alpha \sum C_j^\alpha(\pi)$ by Lemma 6. □

It has been shown in [2] that the lower bound is 2^α. Therefore, our result implies that DSPT is an optimal online algorithm for minimizing $\sum C_j^\alpha$.

5 Acknowledgments

This work is partially supported by NSF of China under Grants 71071123 and 60921003.

References

[1] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, 1998.

[2] M. Liu, C. Chu, Y. Xu, and J. Huo. An optimal online algorithm for single machine scheduling to minimize total general completion time. *Journal of Combinatorial Optimization*, 2010. http://dx.doi.org/10.1007/s10878-010-9348-0.

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

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

[5] A. Vestjens. *On-line machine scheduling*. PhD thesis, Eindhoven University of Technology, 1997.