A Novel Way to Analyze Competitive Performance of Online Algorithms

Jiping Tao * Zhijun Chao Yugeng Xi

Abstract—**A competitive analysis method for online algorithms is developed based on the idea of instance transformation. The method is applied on a single machine online scheduling problem where an assumption is made that the longest processing time among the jobs in any instance is not longer than a constant, say γ, times the shortest processing time. An online algorithm is designed and its competitive ratio is proven to be $1+\frac{\gamma-1}{1+\sqrt{1+\gamma(\gamma-1)}}$ by the proposed analysis method.**

Keywords: online algorithm, competitive analysis, single machine scheduling, total completion time

1 Introduction

Many scheduling problems are intrinsically online in that they require immediate decisions to be made in real time. Ready-made examples include CPU scheduling in the multi-processor operating system and routing in communications networks. The corresponding algorithms solving these online problems have to be online. In contrast to the offline version, an online algorithm must produce a sequence of decisions based on past events without any information about the future. The lack of knowledge of the future does generally not guarantee the optimality of the obtained schedule. Thus a natural issue is how to evaluate different online algorithms that solve the problem.

A widely used approach to evaluate online algorithms is competitive analysis, where the quality of an online algorithm on each input instance is measured by comparing its performance with that of the optimal offline algorithm. An online algorithm is called ρ-*competitive* if, for any instance, the objective function value of the schedule obtained from this algorithm is no worse than ρ times the objective value of the optimal offline schedule [2]. Thus a ρ-competitive online algorithm guarantees that schedules by the algorithm are in some way not too far from the optimal. In this sense, we hope that ρ is as small as possible. For a given online algorithm, its competitive ratio is defined as the infimum of ρ such that the algorithm is ρ-competitive.

Competitive analysis of online algorithms is never a trivial routine. The difficulty lies in two aspects. The first one is that the set of all instance for a given problem is not a very structured set. The number of jobs, released dates, processing time can be extremely arbitrary. The difficulty is how to explore the arbitrariness and locate the worst case instance. Another difficulty is that no realizable optimal offline algorithm is available in most cases due to the NP-hard characteristic, a reasonable lower bound on the optimal schedule has to be constructed. For these twofold aspects of difficulty, most of the competitive analysis methods are typically constructive and problem dependent.

In this work, we focus on the first difficulty and develop a straightforward analysis method based on instance transformation. We use the widely used online model which assumes that jobs arrive at the machine over time, and any job's information is not revealed until it is released. For an instance I of the considered problem, Denote $ALG(I)$ as the objective value of the schedule by an online algorithm, and $L(I)$ as a lower bound on the optimal objective value. Our method is to modify I by an adjustment δ on the instance data, and to choose an appropriate δ according to the change tendency of both $ALG(I)$ and $L(I)$ with respected to δ, such that not only does the modified instance show a more special structure compared with before the modification, but also the ratio of $ALG(I)$ to $L(I)$ doesn't reduce.

We apply the proposed method to analysis online algorithms for the single machine scheduling problem with the objective of minimizing total completion time. This problem is denoted by $1|r_j|\sum C_j$ in the notation of Lawler et al.[7]. Numerous results have been obtained for the problem [3, 5, 6, 8, 9]. Hoogeveen and Vestjens [5] prove any online algorithm has a competitive ratio of at least 2 for the problem. They construct an online algorithm called by delayed SPT rule (D-SPT) which has exactly an competitive ratio of 2. The D-SPT algorithm delays the release date of each job j until time $\max\{r_j, p_j\}$, then implements the shortest processing time (SPT) rule on the modified instance.

In order to present our proof method more clearly, we

*This work is supported by the National Science Foundation of China (No.60504026) and Specialized Research Fund for the Doctoral Program of Higher Education (No.20070248004). Department of Automation, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai, China. Email: jipingtao@gmail.com first employ it to verify the competitive ratio of D-SPT algorithm. Then we generalize the problem by assuming that the ratio of the longest processing time among the jobs in any instance to the shortest is not greater than a constant, say γ. We notate the general problem by $1|r_j, \frac{p_{max}}{p_{min}} \leq \gamma|\sum C_i$ and call it a single machine scheduling problem with bounded processing time. For this problem, we construct a deterministic online algorithm and also apply our proposed analysis method to prove that it has a competitive ratio of $1 + \frac{\gamma-1}{1+\sqrt{1+\gamma(\gamma-1)}}$.

The paper is organized as follows. The competitive analysis method is presented by verifying the competitive ratio of D-SPT algorithm in Section 2. The same method is then applied on an online algorithm for a generalized single machine problem in Section 3. Concluding remarks are given in Section 4.

2 Competitive Ratio of D-SPT Algorithm

In order to present our analysis method more clearly, we first use it to verify the competitive ratio of D-SPT algorithm which was proposed and proven to be 2-competitive by Hoogeveen [5].

First we introduce an important lemma without proof since it can be derived from basic mathematics. The lemma will be repeatedly used in the competitive analysis.

Lemma 1. *Let $f(x)$ and $g(x)$ be two positive functions defined in the interval $[u, v]$, moreover $f(x)$ is convex and $g(x)$ is concave. Then $f(x)/g(x)$ reaches its maximum at one endpoint of the interval, i.e.,*$\frac{f(x)}{g(x)} \leq \max\left\{\frac{f(u)}{g(u)}, \frac{f(v)}{g(v)}\right\} \quad \forall x \in [u, v]$.

Next we will develop an upper bound on the competitive ratio of D-SPT Algorithm. Throughout the analysis the performance ratio is calculated with respected to the optimal preemptive schedule since its objective value is a lower bound of the optimal non-preemptive schedule and can be easily obtained by the shortest remain processing time(SRPT) rule [1]. For any instance I, denote the schedule obtained by D-SPT by $\sigma(I)$ and the optimal preemptive schedule by $\phi(I)$, and denote the objective values of these two schedules by $ALG(I)$ and $L(I)$, respectively.

In order to develop an upper bound, we begin with an arbitrary instance and modify the instance step by step, such that the modified instance has a simple structure and its performance ratio can be directly calculated or analyzed. Similar to the analysis in [5], we only need focus on such instances for which the schedules by D-SPT are composed of a single block. Denote any one of these instances by I_1. Denote the release time of the first released job by r_0. Denote the start time of the block by S. We further partition jobs into subblocks, such that the jobs within each subblock are ordered according to the SPT rule, and that the last job of a subblock is longer than the first job of the succeeding subblock if it exists [5].

We can keep the processing order unchanged and reduce the release dates of jobs released after S, such that each of these jobs is released either at S or at the staring time of the last job in a subblock plus an infinitely small positive value ε. The adjustment doesn't decrease the performance ratio since it keeps $ALG(I_1)$ unchanged and doesn't increase $L(I_1)$. Likewise, for these jobs released before S, we can reduce their release dates to r_0 since they don't start processed before the time S according to D-SPT rule. For some job released at S, say J_i , if $p_i \geq S$, we can also reduce its release time to r_0. Therefore, without loss of generality, all the jobs in I_1 can be classified into three sets in terms of their release dates, denoted by $Q_{S\uparrow}$, Q_S and $Q_{S\downarrow}$ respectively, i.e.

$$\begin{aligned} Q_{S\uparrow} &= \{J_i | r_i = r_0 \quad p_i \geq S\} \\ Q_S &= \{J_i | r_i = S \quad p_i < S\} \\ Q_{S\downarrow} &= \{J_i | r_i > S\} \end{aligned}$$

Denote these subblocks in $\sigma(I_1)$ by $B_1, B_2, \cdots, B_m$. Denote the last job of the jth subblock B_j by J_{B_j}, its processing time by p_{B_j}, and its starting time in $\sigma(I_1)$ by S_{B_j}. Then if $j < m$, the jth release date after S is $S_{B_j} + \varepsilon$. Denote the set of jobs released at $S_{B_j} + \varepsilon$ by $Q^j_{S\downarrow}$. Figure1 illustrates the structure of $\sigma(I_1)$.

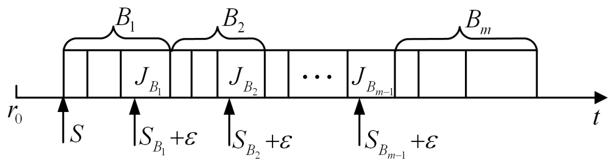

Figure 1: The schedule by D-SPT for I_1

Next we will develop two lemmas to show that the instance I_1 can be modified step by step such that in the new instance, Q_S and $Q_{S\downarrow}$ are both empty.

Lemma 2. *A new instance, denoted by I_2, can be constructed by modifying I_1, such that $Q_{S\downarrow}$ in I_2 is empty, moreover*

$$\frac{ALG(I_1)}{L(I_1)} \leq \max\left\{\frac{ALG(I_2)}{L(I_2)}, 2\right\} \qquad (1)$$

Proof. Consider the last release date, denote it by r_L for the sake of convenience. Suppose the job being processed at r_L in $\sigma(I_1)$ is J_k. After J_k is finished in $\sigma(I_1)$, all the remained jobs are processed according to SPT rule since there is no jobs released hereafter. Likewise, jobs are processed continuously according to SRPT rule from the time r_L in $\phi(I_1)$. Denote $Q^L_{S\downarrow}$ as the set of jobs released at r_L. Divide it into three sets,

$$\begin{aligned} Q^{L'}_{S\downarrow} &= \{J_i | J_i \in Q^L_{S\downarrow}, 0 < p_i < p_k\} \quad (2)\\ Q^{L''}_{S\downarrow} &= \{J_i | J_i \in Q^L_{S\downarrow}, p_i \geq p_k\} \quad (3)\\ Q^{L'''}_{S\downarrow} &= \{J_i | J_i \in Q^L_{S\downarrow}, p_i = 0\} \quad (4)\end{aligned}$$

The following adjustment procedure removes all the jobs in $Q^{L'}_{S\downarrow}$ into $Q^{L''}_{S\downarrow}$ or $Q^{L'''}_{S\downarrow}$.

Increase the processing time of all the jobs in $Q^{L'}_{S\downarrow}$ by δ, which is a parameter to be chosen hereafter. Let

$$\begin{aligned} \underline{\delta} &= -\min\{p_i | J_i \in Q^{L'}_{S\downarrow}\} \quad (5)\\ \bar{\delta} &= p_k - \max\{p_i | J_i \in Q^{L'}_{S\downarrow}\} \quad (6)\end{aligned}$$

Denote the intermediate instance after this adjustment by I'_1.

Jobs are continuously processed in $\sigma(I'_1)$ in the same order as in $\sigma(I_1)$ for any $\delta \in [\underline{\delta}, \bar{\delta}]$. Hence $ALG(I'_1)$ is a linear function with respected to δ. Consider how $\phi(I'_1)$ changes with respected to δ. If jobs keeps the same processing order as in $\phi(I_1)$, the objective value of the obtained preemptive schedule is also a linear function w.r.t δ. Since the optimal preemptive schedule $\phi(I'_1)$ is the one with the minimal objective value among all the feasible schedules, we can figure out that $L(I'_1)$ is a piecewise linear function with respected to δ, moreover its slope doesn't increase with δ increasing, i.e., $L(I'_1)$ is a concave function with respected to δ. According to Lemma1, we can obtain an instance with a worse performance ratio by choosing δ as either $\underline{\delta}$ or $\bar{\delta}$. Thus at least one job belonging to $Q^{L'}_{S\downarrow}$ whose processing time is adjusted to 0 or p_k according to (5) and (6).

Update $Q^{L'}_{S\downarrow}$, $Q^{L''}_{S\downarrow}$ and $Q^{L'''}_{S\downarrow}$ according to (2)(3)(4). We can repeat the above adjustment procedure. After finite steps, all the jobs in $Q^{L'}_{S\downarrow}$ can removed into $Q^{L''}_{S\downarrow}$ or $Q^{L'''}_{S\downarrow}$ since $Q^{L'}_{S\downarrow}$ is finite.

For all the jobs in $Q^{L'''}_{S\downarrow}$, they are processed continuously from r_L in $\phi(I_1)$ and processed immediately after J_k is finished in $\sigma(I_1)$. Suppose that there are q jobs in $Q^{L'''}_{S\downarrow}$. We can delete these q jobs and get an intermediate instance I'_1 which satisfies

$$\begin{aligned} \frac{ALG(I_1)}{L(I_1)} &= \frac{ALG(I'_1) + q(r_L + p_k)}{L(I'_1) + qr_L}\\ &\leq \max\left\{\frac{ALG(I'_1)}{L(I'_1)}, 2\right\} \quad (7)\end{aligned}$$

where the last relation comes from $p_k \leq r_L$. Furthermore the release dates of jobs in $Q^{L''}_{S\downarrow}$ can be reduced to the release date of J_k since they would not be selected before J_k is finished. Thus $Q^L_{S\downarrow}$ becomes empty. Therefore, we can begin from the new last release date again and repeatedly execute the above adjustment procedure for jobs in $Q_{S\downarrow}$. After finite loops, these jobs either have processing time 0 and are deleted, or are removed into Q_S and $Q_{S\uparrow}$. Moreover the relation (1) is always satisfied. □

Remark 3. *In the above proof, if there appears idle time between the execution of jobs in $\sigma(I'_1)$ after some adjustment due to the waiting strategy of D-SPT, we can obtain an smaller instance with a worse performance ratio. So for the sake of simplicity, we make an implicit assumption that jobs are always processed continuously in $\sigma(I'_1)$ during the adjustment.*

Lemma 4. *A new instance, denoted by I_3, can be constructed by modifying I_2, such that Q_S in I_3 is empty, moreover*

$$\frac{ALG(I_2)}{L(I_2)} \leq \frac{ALG(I_3)}{L(I_3)} \quad (8)$$

The proof is almost the same as that for Lemma 2 except that (2)(3)(4) are replaced by the following equations

$$\begin{aligned} Q'_S &= \{J_i | J_i \in Q_S, 0 < p_i < S\}\\ Q''_S &= \{J_i | J_i \in Q_S, p_i \geq S\}\\ Q'''_S &= \{J_i | J_i \in Q_S, p_i = 0\}\end{aligned}$$

and (5)(6) are replaced by

$$\begin{aligned} \underline{\delta} &= -\min\{p_i | J_i \in Q'_S\}\\ \bar{\delta} &= S - \max\{p_i | J_i \in Q'_S\}\end{aligned}$$

Another exception is that the upper bound becomes as follows when deleting jobs with the processing time of 0 in Q_S.

$$\frac{ALG(I_2)}{L(I_2)} = \frac{ALG(I'_2) + qS}{L(L'_2) + qS} \leq \frac{ALG(I'_2)}{L(I'_2)}$$

After the same adjustment in Lemma2, Q'_S becomes empty, jobs in Q''_S can be removed into $Q_{S\uparrow}$, and jobs in Q'''_S can be deleted. The detailed proof can be easily finished by readers.

Q_S and $Q_{S\downarrow}$ in I_3 are both empty. We can figure out that $\sigma(I_3)$ begins continuously processing all the jobs in $Q_{S\uparrow}$ from the time S according to SPT rule, and $\phi(I_3)$ processes jobs from the time r_0 in the same order as in $\sigma(I_3)$. Thus each job starts latter in $\sigma(I_3)$ by S than in $\phi(I_3)$. For the job J_i, denote its completion time in $\sigma(I_3)$ and $\phi(I_3)$ by C_i and C'_i. We have

$$\begin{aligned} \frac{ALG(I_3)}{L(I_3)} &= \frac{\sum\limits_{J_i \in Q_{S\uparrow}} C_i}{\sum\limits_{J_i \in Q_{S\uparrow}} C'_i} \leq \max_{J_i \in Q_{S\uparrow}} \left\{\frac{C'_i + S}{C'_i}\right\}\\ &\leq \min_{J_i \in Q_{S\uparrow}} \left\{1 + \frac{S}{p_i}\right\} \leq 2 \quad (9)\end{aligned}$$

where the last relation comes from (1). Combining Lemma 2, 4 with (9), we can immediately obtain the following lemma.

Lemma 5. *The D-SPT online algorithm in [5] is* 2*-competitive for* $1|r_j|\sum C_j$.

3 The αD-SPT online algorithm and its competitive analysis

In this section, we generalize the problem $1|r_j|\sum C_j$ by assuming that the ratio of the longest processing time among the jobs in any instance to the shortest is not greater than a constant, say γ. We denote the modified problem by $1|r_j, \frac{p_{max}}{p_{min}} \leq \gamma|\sum C_i$. Kaminsky and Simchi-Levi [6] make the same assumption and prove that the shortest processing time among available jobs algorithm(SPTA) is asymptotically optimal. This assumption is also adopted by He and Dósa [4] for parallel machine scheduling problems. For the modified problem, we will construct an online algorithm and employ the same method as in Section 2 to analysis its competitive performance.

The key point to develop a good online algorithm for the problem considered here is to determine whether to insert appropriate waiting time or to immediately schedule a job when the machine is idle and some jobs are available. D-SPT algorithm [5] makes this decision by comparing the processing time of jobs with the current time, which guarantees each job starts processed at the time no earlier than its processing time. The resulted additional waiting time for some jobs is necessary in order to avoid the performance ratio becoming too bad when there are jobs with extremely short processing time arriving immediately after starting to process a job. Under our assumption that the longest processing time among the jobs in any instance is not longer than γ times the shortest one, intuitively, less waiting time is needed in order to avoid the performance deterioration in the case mentioned above since processing time can only jump up and down over a limited proportional range. A ready-made example is that no-wait SPT strategy is optimal when all the jobs have the same processing time.

In view of the above consideration, we directly extend D-SPT algorithm [5] by introducing a well-designed parameter and construct an online algorithm, called by α-delayed SPT algorithm (αD-SPT). Denote the current decision time by t. αD-SPT algorithm can be described as follows. Whenever the machine is idle and some jobs are available, choose a job with the shortest processing time in all the arrived and unscheduled jobs, say J_i , if $p_i \leq \alpha t$, schedule J_i; otherwise, wait until the above inequality is satisfied or a new job arrives, where α is an algorithm parameter to be designed.

We use the same analysis method based on instance reduction as in Section 2 to develop an upper bound on the competitive ratio of αD-SPT Algorithm. Throughout the analysis, we employ the same notations as in Section 2 in condition that no confusion is induced. Similarly, we begin with an arbitrary instance I_1 for which the constructed schedule by αD-SPT is composed of a block. Denote the shortest and longest processing time among jobs in I_1 by p_{min} and p_{max}. We modify jobs in $Q_{S\downarrow}$ and Q_S in the inverse order of release data. Similar results as Lemma 2 and 4 can be reduced.

Lemma 6. *A new instance, denoted by I_2, can be constructed by modifying I_1, such that $Q_{S\downarrow}$ in I_2 either is empty, or only includes jobs with the shortest processing time p_{min}, moreover*

$$\frac{ALG(I_1)}{L(I_1)} \leq \max\left\{\frac{ALG(I_2)}{L(I_2)}, 1+\frac{1}{\alpha}\right\} \tag{10}$$

Lemma 7. *A new instance, denoted by I_3, can be constructed by modifying I_2, such that Q_S in I_3 either is empty, or only includes jobs with processing time p_{min}, moreover*

$$\frac{ALG(I_2)}{L(I_2)} \leq \max\left\{\frac{ALG(I_3)}{L(I_3)}, 1+\frac{1}{\alpha}\right\} \tag{11}$$

The proofs are similar to the one for Lemma 2. The essential idea is still to transform the instance along the direction of its performance ratio increasing. Intuitively, for the αD-SPT online algorithm, the worst case should occur in two possible cases. One is that there are several extremely small jobs released immediately after αD-SPT chooses to process a job. Another case is that there is no any jobs released when αD-SPT keeps the machine idle for a period of time when some jobs are available. The intuition is just validated by the above two lemmas, where I_3 is corresponding to the first case and $1+\frac{1}{\alpha}$ is the upper bound in the later case. The detailed proofs are omitted due to the space limitation.

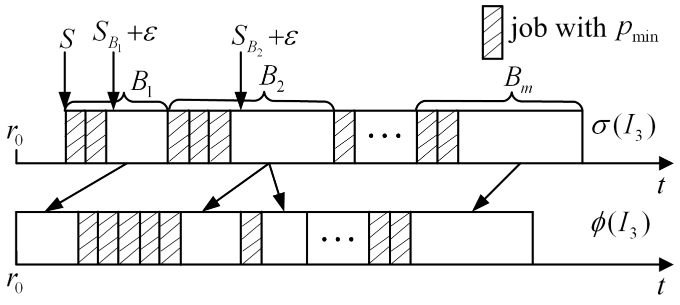

Figure 2: The schedule by αD-SPT and the optimal preemptive schedule for I_3

Q_S and $Q_{S\downarrow}$ in I_3 either are empty, or only include jobs with processing time p_{min}. We can figure out that $\sigma(I_3)$

begins with all the jobs in Q_S at the time S, then alternately processes jobs in $Q_{S\uparrow}$ and jobs in $Q_{S\downarrow}$. In $\phi(I_3)$ jobs are processed continuously from the time r_0, moreover these jobs in $Q_{S\uparrow}$ keep the same processing order as in $\sigma(I_3)$ except that some jobs are preempted by some jobs in Q_S or $Q_{S\downarrow}$. Both $\sigma(I_3)$ and $\phi(I_3)$ are illustrated in Figure2. We can analyze the performance ratio of I_3 by directly comparing jobs' completion time in $\sigma(I_3)$ and $\phi(I_3)$.

Lemma 8.

$$\frac{ALG(I_3)}{L(I_3)} \le \max\left\{1+\frac{1}{\alpha}, 1+\frac{\gamma-1}{\frac{\gamma}{\alpha}+2}\right\}$$

Proof. For the job J_i, denote its completion time in $\sigma(I_3)$ and $\phi(I_3)$ by C_i and C_i' respectively. Denote the set of all the jobs in I_3 by Q, and denote one of its subsets by Q'. We have

$$\frac{ALG(I_3)}{L(I_3)} \le \max_{Q'\subset Q}\left\{\frac{\sum_{J_i\in Q'} C_i}{\sum_{J_i\in Q'} C_i'}\right\} \quad (12)$$

Next we divide Q into different subsets and compare their completion time.

Denote the first processed job in $\phi(I_3)$ by J_1, which obviously belongs to $Q_{S\uparrow}$. Suppose there are q_1 jobs in Q_S. If J_1 is not preempted in $\phi(I_3)$ by these q_1 jobs, we have

$$C_1 = C_1' + S + q_1 p_{min}$$

Consider J_1 and these q_1 jobs together as a subset Q', we can obtain

$$\begin{aligned}
&\frac{C_1+\sum_{J_i\in Q_S} C_i}{C_1'+\sum_{J_i\in Q_S} C_i'}\\
=\ &\frac{C_1'+S+q_1p_{min}+q_1S+\frac{q_1(q_1+1)}{2}p_{min}}{C_1'+q_1p_1+\frac{q_1(q_1+1)}{2}p_{min}}\\
\le\ & 1+\frac{S+q_1p_{min}+q_1(S-p_1)}{p_1+q_1p_1+\frac{q_1(q_1+1)}{2}p_{min}}\\
\le\ & 1+\frac{S+q_1S}{p_1+q_1p_1}\\
\le\ & 1+\frac{1}{\alpha} \qquad (13)
\end{aligned}$$

where the last relation comes from $p_1 \ge \alpha S$ since J_1 is not processed before S in $\sigma(I_3)$.

If J_1 is preempted by the q_1 jobs and other k^1 jobs which belong to $Q^1_{S\downarrow}$, then we have

$$\begin{aligned}
C_1 &= C_1' + S - k^1 p_{min}\\
C_i &= C_i' \qquad \forall J_i \in Q_S\\
C_i &= C_i' + p_1 \qquad \forall J_i \in Q^1_{S\downarrow}
\end{aligned}$$

Likewise, consider J_1, Q_S and $Q^1_{S\downarrow}$ together as a subset Q', we can obtain

$$\begin{aligned}
&\frac{C_1+\sum_{J_i\in Q_S} C_i+\sum_{J_i\in Q^1_{S\downarrow}} C_i}{C_1'+\sum_{J_i\in Q_S} C_i'+\sum_{J_i\in Q^1_{S\downarrow}} C_i'}\\
=\ &\frac{C_1'+S-k^1p_{min}+\sum\limits_{J_i\in Q_S} C_i'+\sum\limits_{J_i\in Q^1_{S\downarrow}}(C_i'+p_1)}{C_1'+\sum\limits_{J_i\in Q_S} C_i'+\sum\limits_{J_i\in Q^1_{S\downarrow}} C_i'}\\
\le\ & 1+\frac{S-k^1p_{min}+k^1p_1}{C_1'+\sum\limits_{J_i\in Q^1_{S\downarrow}} C_i'}\\
\le\ & 1+\frac{S+k^1(p_1-p_{min})}{p_1+(q_1+k^1)p_{mim}+k^1S_{B_1}+\frac{k^1(k^1+1)}{2}p_{min}}\\
\le\ & \max\left\{1+\frac{S}{p_1}, 1+\frac{p_1-p_{min}}{S_{B_1}+2p_{min}}\right\}\\
\le\ & \max\left\{1+\frac{1}{\alpha}, 1+\frac{\gamma-1}{\frac{\gamma}{\alpha}+2}\right\} \qquad (14)
\end{aligned}$$

where the last relation comes from $p_1 \ge \alpha S$, $p_1 \le \gamma p_{min}$ and $p_1 \le \alpha S_{B_1}$ due to αD-SPT rule.

For the remained jobs except J_1 in $Q_{S\uparrow}$, we divide them into two subsets denoted by $Q'_{S\uparrow}$ and $Q''_{S\uparrow}$ such that, in $\phi(I_3)$, no jobs in $Q'_{S\uparrow}$ is preempted and each job in $Q''_{S\uparrow}$ is preempted. We can figure out that $Q''_{S\uparrow}$ is composed of the last job of some subblocks in $\sigma(I_3)$. Denote by A the index set such that jobs in $Q^j_{S\downarrow}$ preempt the job J_{B_j} in $\phi(I_3)$ for $j \in A$. Then $Q''_{S\uparrow} = \{J_{B_j} | j \in A\}$. Denote by $Q'_{S\downarrow}$ the set of all the remained jobs in $Q_{S\downarrow}$ which don't preempt any job in $\phi(I_3)$. Denote the number of jobs in $Q^j_{S\downarrow} (1 \le j < m)$ by k^j. We have

$$\begin{aligned}
C_i &= C_i' + S \qquad \forall J_i \in Q'_{S\uparrow}\\
C_{B_j} &= C'_{B_j} + S - k^j p_{min} \quad J_{B_j} \in Q''_{S\uparrow} \quad \forall j \in A\\
C_i &= C_i' + p_{B_j} \qquad \forall J_i \in Q^j_{S\downarrow} \quad \forall j \in A\\
C_i &= C_i' + S \qquad \forall J_i \in Q'_{S\downarrow}
\end{aligned}$$

So we have for any job J_i in $Q'_{S\uparrow}$ or $Q'_{S\downarrow}$,

$$\frac{C_i}{C_i'} = \frac{C_i'+S}{C_i'} \le 1+\frac{S}{p_1} \le 1+\frac{1}{\alpha} \quad (15)$$

For $j \in A$, consider the job J_{B_j} in $Q''_{S\uparrow}$ and all the jobs in $Q^j_{S\downarrow}$ together as a subset Q', we can obtain similar to the transformation in (14)

$$\begin{aligned}
&\frac{C_{B_j}+\sum_{J_i\in Q^j_{S\downarrow}} C_i}{C'_{B_j}+\sum_{J_i\in Q^j_{S\downarrow}} C_i'}\\
\le\ & \max\left\{1+\frac{S}{p_1}, 1+\frac{p_{B_j}-p_{min}}{S_{B_j}+2p_{min}}\right\}\\
\le\ & \max\left\{1+\frac{1}{\alpha}, 1+\frac{\gamma-1}{\frac{\gamma}{\alpha}+2}\right\} \qquad (16)
\end{aligned}$$

Each job in Q is involved in one of the subsets above mentioned and considered in one of (13)(14)(15)(16). Combining that with (12), we can immediately finish the proof. $\square$

Theorem 9. *For the problem* $1|r_j, \frac{p_{max}}{p_{min}} \leq \gamma|\sum C_j$*, the* α*D-SPT online algorithm with* $\alpha = \frac{1+\sqrt{1+\gamma(\gamma-1)}}{\gamma-1}$ *has the competitive ratio of* $1 + \frac{1}{\alpha}$*.*

Proof. From Lemma 6,7 and 8, we can obtain the competitive ratio ρ of αD-SPT

$$\rho \leq \max_{I_1} \frac{ALG(I_1)}{L(I_1)} \leq \max\left\{1 + \frac{1}{\alpha}, 1 + \frac{\gamma - 1}{\frac{\gamma}{\alpha} + 2}\right\}$$

Let $1 + \frac{1}{\alpha} = 1 + \frac{\gamma-1}{\frac{\gamma}{\alpha}+2}$, i.e., $\alpha = \frac{1+\sqrt{1+\gamma(\gamma-1)}}{\gamma-1}$, we have

$$\rho \leq 1 + \frac{1}{\alpha} \tag{17}$$

The lower bound can be obtain by constructing the following instance. Consider an instance with only one job released at the time 0 with processing time 1, αD-SPT schedules the job from the time $\frac{1}{\alpha}$, and the optimal objective value is obviously 1, so we have

$$\rho \geq 1 + \frac{1}{\alpha} \tag{18}$$

$\square$

Remark 10. *The result covers the D-SPT online algorithm since* α*D-SPT has the competitive ratio of* 2 *when* γ *tends to infinity. At the same time the algorithm degenerates into a no-wait SPT strategy when* γ *is equal to* 1*, which is obviously the optimal algorithm for the corresponding problem* $1|r_j, p_j = p|\sum C_j$*.*

4 Concluding remarks

In this work, we develop a novel analysis technique to derive the competitive ratio of the online algorithm based on the idea of instance transformation. In order to derive an upper bound on the performance ratio among all the instances, the analysis method exploits the possible structure of the worst case instance with respect to the given online algorithm. It begins with an arbitrary instance, and transforms the instance along the direction of its performance ratio increasing, such that the modified instance shows a more special structure of which we can take advantage to analyze its performance ratio. We present the method by verifying the competitive ratio of D-SPT algorithm proposed by Hoogeveen [5] for the single machine scheduling problem with the objective of minimizing total completion time. Then we generalize the problem by assuming that the ratio of the longest processing time to the shortest in any instance is not greater than a constant γ. We design an online algorithm called by αD-SPT and apply our analysis method to derive its competitive ratio is $1 + \frac{1}{\alpha}$ where α is equal to $\frac{1+\sqrt{1+\gamma(\gamma-1)}}{\gamma-1}$. Further work is needed to extend the analysis method to other online problems.

References

[1] K.R. Baker. *Introduction to Sequencing and Scheduling.* John Wiley & Sons, 1974.

[2] A. Fiat and G.J. Woeginger. *Online algorithms: the state of the art.* Springer, 1998.

[3] 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(3):513–544, 1997.

[4] Y. He and G. Dósa. Semi-online scheduling jobs with tightly-grouped processing times on three identical machines. *Discrete Applied Mathematics*, 150(1-3):140–159, 2005.

[5] J. A. Hoogeveen and A. P. A. Vestjens. Optimal on-line algorithms for single-machine scheduling. *Lecture Notes in Computer Science*, 1084:404–414, 1996.

[6] P. Kaminsky and D. Simchi-Levi. Asymptotic analysis of an on-line algorithm for the single machine completion time problem with release dates. *Operations Research Letters*, 29(3):141–148, 2001.

[7] E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors, *Handbooks in Operations research and Management Science*, volume 4, pages 445–522. North-Holland, 1993.

[8] X. Lu, RA Sitters, and L. Stougie. A class of on-line scheduling algorithms to minimize total completion time. *Operations Research Letters*, 31(3):232–236, 2003.

[9] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82(1):199–223, 1998.