Online scheduling with linear deteriorating jobs to minimize the total weighted completion time

Ran Ma\textsuperscript{a,c}, Jiping Tao\textsuperscript{b}, Jinjiang Yuan\textsuperscript{c,*}

\textsuperscript{a} School of Mathematics and Information Science, Henan Polytechnic University, Jiaozuo, Henan 454000, PR China
\textsuperscript{b} Department of Automation, Xiamen University, Xiamen, Fujian 361005, PR China
\textsuperscript{c} School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan 450001, PR China

A R T I C L E   I N F O

Keywords:
Scheduling
Online algorithms
Total weighted completion time
Linear deterioration

A B S T R A C T

In this paper, we study the online scheduling of linear deteriorating jobs on a single machine to minimize the total weighted completion time. In the problem, a set of \( n \) independent linear deteriorating jobs arriving online over time has to be scheduled on a single machine, where the information of each job including its deterioration rate and weight is unknown in advance. Linear deterioration means that the processing time \( p_j \) of a job \( J_j \) is a linear function of its starting time \( s_j \). In this paper, we assume that \( p_j = \alpha_j(A + B s_j) \), where \( A \) and \( B \) are nonnegative with \( A + B > 0 \) and \( \alpha_j \geq 0 \) is the deterioration rate of \( J_j \). The goal is to minimize the total weighted completion time, i.e., \( \sum w_j C_j \). For this problem, we provide a best possible online algorithm with a competitive ratio of \( 1 + \lambda(A) + \alpha_{\text{max}} B \), where \( \alpha_{\text{max}} = \max_{1 \leq j \leq n} \alpha_j \) and \( \lambda(A) = 0 \) or \( \lambda(A) = 1 \) depending on whether \( A = 0 \) or \( A > 0 \).

1. Introduction

Traditional scheduling assumes that the processing time of a job is fixed. However, in the real world, there are numerous situations in which the processing time increases (deteriorates) as the starting time increases. For example, to schedule maintenance or cleaning, a delay often requires additional effort to accomplish the task. Other examples can be found in fire fighting, steel production, and financial management [13,21]. Scheduling of deteriorating jobs was first introduced by Browne and Yechiali [3] and Gupta and Gupta [10], independently. Following their research, this topic has received more and more attention. Related research can be found in [1,4–7,15,23,29–32] among many others.

Online scheduling has been a hot research topic in the last few decades. In contrast to the off-line version, an online algorithm must produce a sequence of decisions based on past events without any information about the unreleased jobs. The lack of knowledge of the future does not generally guarantee the optimality of the schedule generated by an online algorithm. Thus a natural issue is how to evaluate different online algorithms for a same scheduling problem. A widely used criterion to evaluate an online algorithm is its competitive ratio. For a minimization problem, the competitive ratio \( \rho_A \) of an online algorithm \( A \) is defined to be

\[
\rho_A = \sup \{ A(I)/\text{OPT}(I) : I \text{ is an instance with } \text{OPT}(I) > 0 \}.
\]
Here, for an instance $I$, $\mathcal{A}(I)$ is used to denote the objective value of the schedule generated by the online algorithm $\mathcal{A}$, and $OPT(I)$ is the objective value of an (off-line) optimal schedule. The closer the competitive ratio approaches 1, the better the online algorithm we have. An online algorithm $\mathcal{A}$ is called best possible if no online algorithm has a competitive ratio less than $\rho_\mathcal{A}$.

**Problem formulation:** In this paper, we focus on the online over time scheduling of linear deteriorating jobs to minimize the total weighted completion time on a single machine. Here **linear deterioration** means that the processing time of job $J_j$ is given by $p_j = \alpha_j(A + B_j)$, where $A \geq 0, B \geq 0$, at least one of $A$ and $B$ is nonzero, $\alpha_j \geq 0$ is called the deterioration rate of job $J_j$, and $s_j \geq 0$ is the starting time of job $J_j$. We use $r_j$ to denote the release date of a job $J_j$. In the problem, the characteristics of a job $J_j$, including its deterioration rate $\alpha_j$ and weight $w_j$, become known at its release date $r_j$. Then the released jobs will be processed on a single machine. Letting $C_j$ be the completion time of job $J_j$ in a given schedule, we are interested in minimizing the **total weighted completion time:** $\sum w_j C_j$. In the case that $A = 0$, for each job $J_j$, we have $p_j = \alpha_j (A + B_j) = \alpha_j B_j$, so in order to avoid the trivial situation in which all jobs start and complete at time 0, we assume that all jobs are released at least at a given time $t_0 > 0$. If $A > 0$, we just define $t_0 = 0$. In the scheduling notation of Graham et al. [9], the online scheduling problem studied in this paper can be denoted by $1|\text{online}$, $r_j \geq t_0$, $p_j = \alpha_j (A + B_j) | \sum w_j C_j$.

This problem has two interesting special versions. In the first special version, we have $B = 0$ and $A > 0$, and so, by scaling, we may assume that $p_j = \alpha_j$. Then the problem degenerates to the classical online scheduling problem $1|\text{online}$, $r_j \geq t_0$, $p_j = \alpha_j s_j | \sum w_j C_j$.

In the second special version, we have $A = 0$ and $B > 0$, and so, by scaling, we may assume that $p_j = \alpha_j s_j$ and call it **simple linear deterioration**. Then the problem is denoted by $1|\text{online}$, $r_j \geq t_0$, $p_j = \alpha_j s_j | \sum w_j C_j$.

**Most related work:** There has been an enormous amount of work on scheduling of deteriorating jobs. Until recently most research has focused on the off-line setting. We do not intend to do a complete review of results in the area and only restrict our attention to some most relevant work on online scheduling problems and scheduling problems of deteriorating jobs directly related to the matter of this paper.

For problem $1|\text{online}, r_j \geq t_0, \sum w_j C_j$, Vestjens [28] showed that no online algorithm has a competitive ratio of less than 2. Vestjens [28] and Philips et al. [22] presented distinct online algorithms with a best possible competitive ratio of 2. In the online algorithm Delayed SPT (DSPT) proposed by Vestjens [28], whenever the machine is available at the present time $t$, an available job $J_j$ with the smallest processing time is scheduled if $p_j \leq t$. The technique “quasi-schedule” was introduced in their competitive ratio analysis. The same idea of shifted release times was used by Stougie (cited in Vestjens [28]) who obtained a third algorithm with a best possible competitive ratio of 2. Following their research, the technique “quasi-schedule” in [28] was also applied in [16–21]. Ma and Yuan [19] extended the technique “instance reduction” in their research for the online scheduling with job rejection to minimize the total weighted completion time plus reduction” in their research for the online scheduling with job rejection to minimize the total weighted completion time plus

For problem $1|\text{online}, r_j \geq t_0, \sum w_j C_j$, Anderson and Potts [2] presented a best possible deterministic online algorithm, called Delayed SWPT (DSWPT), with a competitive ratio of 2 under the assumption that all release dates and processing times are integers. The main technique used in the competitive ratio analysis in Anderson and Potts [2] is to create a “doubled problem” and an “extended problem”.

For the off-line version that all jobs are released at time $t_0$ and under simple linear deterioration of processing times, Mosheiov [21] showed that the schedule in which the jobs are arranged in the non-decreasing order of the growth rates $\alpha_j$ (Smallest Growth Rate or SGR for short) can minimize the total completion time. Moreover, Mosheiov [21] also showed that the schedule in which the jobs are arranged in the non-decreasing order of the ratios $\alpha_j/(1 + \alpha_j w_j)$ can minimize the total weighted completion time. Using the idea of SGR, for problem $1|\text{online}, r_j \geq t_0, p_j = \alpha_j s_j| \sum w_j C_j$, Liu et al. [16] first introduced an optimal scheduling rule for the corresponding preemptive-resumption model. They further showed that $1 + \alpha_{\text{max}}$ is a lower bound of competitive ratio for the non-preemptive problem, where $\alpha_{\text{max}} = \max_j \alpha_j$ is the maximum deterioration rate of all jobs, and presented an online algorithm named D-SGR (Delayed Smallest Growth Rate). By using the technique “quasi-schedule”, Liu et al. [16] showed that D-SGR is best possible for problem $1|\text{online}, r_j \geq t_0, p_j = \alpha_j s_j| \sum C_j$ with a competitive ratio of $1 + \alpha_{\text{max}}$. For problem $1|\text{online}, r_j \geq t_0, p_j = \alpha_j s_j| \sum C_j$, where $\beta > 0$ is the general index of completion times, Yu and Wong [34] showed that no deterministic online algorithm can be better than $(1 + \alpha_{\text{max}})^{\beta}$-competitive. Also using the technique “quasi-schedule”, they gave a best possible online algorithm for $1|\text{online}, r_j \geq t_0, p_j = \alpha_j s_j| \sum C_j$ with a competitive ratio of $(1 + \alpha_{\text{max}})^{\beta}$.

By Kononov [12], the off-line scheduling problem $1|p_j = \alpha_j (A + B_j) | \sum w_j C_j$ is solvable in $O(n \log n)$ time by scheduling jobs in the non-decreasing order of $\alpha_j/(1 + \alpha_j B_j)$ values. To the best of our knowledge, there are no results for online scheduling with linear deteriorating jobs to minimize the total weighted completion time plus an auxiliary parameter.
the rejection cost. Furthermore, by using the framework established in Ma and Yuan [19] and the technique “instance reduction” in Tao et al. [24], Ma and Yuan [20] studied the online tradeoff scheduling on a single machine to minimize makespan and total weighted completion time.

In this paper, we focus on the online scheduling problem \( \text{online} \). \( r_j \geq t_0 \). \( p_j = \alpha_j(A + B_j) \sum w_j C_j \). To simplify the presentation, we denote \( t_j = \alpha_j/(1 + \alpha_j B w_j) \) and call ratio \( t_j \) the weighted growth rate of \( J_j \). Then in the Smallest Weighted Growth Rate rule (SWGR rule for short), the jobs are sequenced in the non-decreasing order of the ratios \( t_j \). For the online scheduling problem \( \text{online} \), \( r_j \geq t_0 \). \( p_j = \alpha_j(A + B_j) \sum w_j C_j \). We first show that every online algorithm has a competitive ratio of at least \( 1 + \lambda(A) + \alpha \max B \), where \( \alpha \max = \max_{1 < i} \alpha_j \) and \( \lambda(A) = 0 \) if \( A = 0 \) and \( \lambda(A) = 1 \) if \( A > 0 \). Based on the SWGR rule, we propose an online algorithm, called Delayed-SWGR (DSWGR for short), for problem \( \text{online} \). \( r_j \geq t_0 \). \( p_j = \alpha_j(A + B_j) \sum w_j C_j \). Again, by using the framework established in Ma and Yuan [19] and the technique “instance reduction” in Tao et al. [24], we show that DSWGR is an online algorithm with a competitive ratio of \( 1 + \lambda(A) + \alpha \max B \), which matches the lower bound we provide. This means that DSWGR is a best possible online algorithm for problem \( \text{online} \).

**Organization of the paper.** This paper is organized as follows. In Section 2, we present some notations, some useful lemmas, and the general lower bound of our problem. The online algorithm DSWGR is provided in Section 3, followed by the framework of analysis. In Section 4, we provide the rigorous analysis. Finally, some conclusions are provided in Section 5.

2. Preliminaries

The following notations will be used throughout this paper:

- \( J_j \): the job of index \( j \), \( j = 1, 2, \ldots, n \).
- \( r_j \): the release date of job \( J_j \).
- \( t_0 \): the lower bound of the earliest release date of all the jobs. When \( A = 0 \), we assume that \( t_0 > 0 \), and when \( A > 0 \), we define \( t_0 = 0 \). Throughout this paper, when we mention a time \( t \), it means that \( t \) is a number in the interval \([t_0, +\infty)\).
- \( w_j \): the weight of job \( J_j \).
- \( \alpha_j \): the deterioration rate of job \( J_j \).
- \( \alpha \max = \max \{\alpha_j : J_j \in \mathcal{I}\} \): the maximum deterioration rate of all jobs.
- \( \rho(A, B) = 1 + \lambda(A) + \alpha \max B \): the competitive ratio of our online algorithm, where \( \lambda(A) = 0 \) if \( A = 0 \) and \( \lambda(A) = 1 \) if \( A > 0 \).
- \( t_j = \alpha_j/(1 + \alpha_j B w_j) \): the weighted growth rate of job \( J_j \).
- \( s_j(x) \): the starting time of job \( J_j \) in a schedule \( X \).
- \( p_j(x) \): the processing time of job \( J_j \) in a schedule \( X \).
- \( C_j(x) \): the completion time of job \( J_j \) in a schedule \( X \).
- \( \sigma(I) \): the schedule constructed by our online algorithm DSWGR for instance \( I \).
- \( \text{DSWGR}(I) \): the objective value of \( \sigma(I) \).
- \( \pi(I) \): an optimal off-line schedule for instance \( I \).
- \( \text{OPT}(I) \): the objective value of \( \pi(I) \).

The following lemmas, which will be repeatedly used in our competitive analysis, are provided without proof, since they can be derived from the basic mathematics and have been frequently used in the literature such as [19,20,24,26].

**Lemma 2.1.** Let \( f(x) \) and \( g(x) \) be two positive-valued functions defined in the interval \([u, v]\), where \( f(x) \) is convex and \( g(x) \) is concave. Then \( \frac{f(x)}{g(x)} \) reaches its maximum value at one endpoint of the interval, i.e., \( \frac{f(x)}{g(x)} \leq \max \left\{ \frac{f(u)}{g(u)}, \frac{f(v)}{g(v)} \right\} \) for any \( x \in [u, v] \).

**Lemma 2.2.** Let \( I \) be a job instance and let \( r_j \) be the release date of \( J_j \) in \( I \). Let \( I' \) be a job instance obtained from \( I \) by modifying the release date of each job \( J_j \) as \( r_j' \leq r_j \). Then \( \text{OPT}(I') \leq \text{OPT}(I) \).

**Lemma 2.3.** Let \( I' \) and \( I'' \) be a partition of the job instance \( I \), i.e., \( I = I' \cup I'' \) and \( I' \cap I'' = \emptyset \). Then \( \text{OPT}(I) \geq \text{OPT}(I') + \text{OPT}(I'') \).

For a job instance \( I \) and a schedule \( \sigma \) of \( I \), we call \( (I, \sigma) \) an instance-schedule pair. We use \( Z(I, \sigma) = \sum_{j \in I} Z_j(I, \sigma) = \sum_{j \in I} w_j C_j(\sigma) \) to denote the objective value of \( \sigma \) on instance \( I \), where \( Z_j(I, \sigma) = w_j C_j(\sigma) \) is the contribution of job \( J_j \) to the objective value. \( s_{\min}(I, \sigma) \) and \( C_{\max}(I, \sigma) \) are used to denote the starting time of the first job in \( \sigma \) and the completion time of the last job in \( \sigma \) on instance \( I \). In our discussion, we frequently generate a new instance \( I' \) from a given instance \( I \) by modifying the weights of the jobs in \( I \). Then we use \( w_j(I) \) to denote the weight of job \( J_j \) in instance \( I \). The notation \( r_j(I) \) can be understood similarly. Moreover, when the instance \( I \) must be specified in schedule \( \sigma \), we use \( s_j(I, \sigma) \) and \( C_j(I, \sigma) \) to denote \( s_j(\sigma) \) and \( C_j(\sigma) \), respectively.

Let \( I \) be a job instance. For each \( T \subseteq I \) and each positive number \( \delta \), we use \( I^{T, \delta} \) to denote the job instance obtained from \( I \) such that the weight \( w_j(I) \) of each job \( J_j \in T \) is modified as \( w_j(I^{T, \delta}) = \delta \cdot w_j(I) \). Since each job \( J_j \in I^{T, \delta} \) still has release date \( r_j \) and deterioration rate \( \alpha_j \), a (feasible) schedule of \( I \) is also a (feasible) schedule of \( I^{T, \delta} \), and vice versa.

Although the problem of online scheduling with job rejection on a single machine to minimize the total weighted completion time plus the rejection cost) studied in Ma and Yuan [19] is different from the problem in this paper, the proofs of the following three lemmas (Lemmas 2.4, 2.5, and 2.6) are similar to the proofs of three corresponding lemmas in Ma and Yuan [19]. We present the proofs of the following three lemmas in the appendix for completion.
Lemma 2.4. Let \( I \) be a job instance and let \( T \subseteq I \). Let \( \delta \) be a variable with \( \delta \in (u, v) \) for positive numbers \( u \) and \( v \) with \( u < v \). Then \( \text{OPT}(f^{T, \delta}) \) is a concave function in \( \delta \in (u, v) \).

Proof. See the proof in Appendix A. \( \square \)

Let \( (I, \sigma) \) be an instance-schedule pair and let \( T \) be a proper subset of \( I \). We denote by \( \sigma|_T \) the subschedule of the jobs restricted to \( T \) in schedule \( \sigma \), i.e., the subschedule obtained from \( \sigma \) by removing jobs in \( I - T \) from schedule \( \sigma \) and keeping the jobs in \( T \) unchanged (including their starting times and completion times). Then we have the following lemmas.

Lemma 2.5. Suppose that \( (I, \sigma) \) be an instance-schedule pair such that \( Z(I, \sigma) > \rho(A, B) \cdot \text{OPT}(I) \). Suppose that \( T \) is a proper subset of \( I \) such that \( Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T) \). Let \( \delta \) be an arbitrary positive number with \( \delta \leq \frac{\rho(A, B)}{\rho(A, B) - \sigma(T)} \). Then \( (f^{T, \delta}, \sigma) \) is an instance-schedule pair such that \( Z((f^{T, \delta}), \sigma)/\text{OPT}(f^{T, \delta}) \geq Z(I, \sigma)/\text{OPT}(I) \). Consequently, \( Z((f^{T, \delta}), \sigma) > \rho(A, B) \cdot \text{OPT}(f^{T, \delta}) \).

Proof. See the proof in Appendix B. \( \square \)

Lemma 2.6. Suppose that \( (I, \sigma) \) be an instance-schedule pair such that \( Z(I, \sigma) > \rho(A, B) \cdot \text{OPT}(I) \). Suppose that \( T \) is a proper subset of \( I \) such that \( Z(I, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(I|_T) \). Let \( \delta \) be an arbitrary number with \( \delta \geq 1 \). Then \( (f^{I(T, \delta)}, \sigma) \) is an instance-schedule pair such that \( Z((f^{I(T, \delta)}, \sigma)/\text{OPT}(I(T, \delta)) \geq Z(I, \sigma)/\text{OPT}(I) \). Consequently, \( Z((f^{I(T, \delta)}, \sigma) > \rho(A, B) \cdot \text{OPT}(f^{I(T, \delta)}) \).

Proof. See the proof in Appendix C. \( \square \)

Let \( I \) be a job instance and let \( \sigma \) be a schedule of \( I \). We call \( (I, \sigma) \) a regular instance-schedule pair if \( I \) and \( \sigma \) satisfy the following three conditions:

(i) All jobs in \( I \) have a common release date \( r \geq t_0 \);
(ii) All jobs in \( I \) have a common weighted growth rate \( \tau \), i.e., \( \tau_j = \tau \) for all jobs \( j \in I \);
(iii) The jobs in \( I \) are scheduled consecutively in the time interval \( [s_{\min}(I, \sigma), C_{\max}(I, \sigma)] \) such that \( s_{\min}(I, \sigma) \leq r(1 + \alpha_{\max}B) + \alpha_{\max}A \).

The above three conditions may guarantee the desired inequality \( Z(I, \sigma) \leq \rho(A, B) \cdot \text{OPT}(I) \) in Lemma 2.7. Especially, the condition \( s_{\min}(I, \sigma) \leq r(1 + \alpha_{\max}B) + \alpha_{\max}A \) is necessary. In fact, if \( s_{\min}(I, \sigma) > r(1 + \alpha_{\max}B) + \alpha_{\max}A \), the desired inequality will be violated even when \( |I| = 1 \).

Lemma 2.7. Let \( (I, \sigma) \) be a regular instance-schedule pair. Then \( Z(I, \sigma) \leq \rho(A, B) \cdot \text{OPT}(I) \).

Proof. Suppose that \( |I| = n \) and the jobs in \( I \) are processed in the order \( (J_1, J_2, \ldots, J_n) \) in \( \sigma \). Let \( \sigma' \) be the schedule of \( I \) obtained by shifting \( \sigma \) left such that \( s_{\min}(I, \sigma') = r \). Since the jobs in \( I \) have a common release date \( r \geq t_0 \) and a common weighted growth rate \( \tau \), the smallest weighted growth rate (SWGR) rule in Mosheiov [21] and Kononov [12] implies that \( \sigma' \) is an optimal schedule of \( I \). Then \( \text{OPT}(I) = Z(I, \sigma') \).

If \( A = 0 \), then for each \( \pi \in \{\sigma, \sigma'\} \) and each \( j = 1, 2, \ldots, n \), we have

\[
    p_j(\pi) = \alpha_j B s_j(\pi) \quad \text{and} \quad C_j(\pi) = s_j(\pi)(1 + \alpha_j B).
\]

From (1), we can inductively show that

\[
    C_j(\sigma) = s_{\min}(I, \sigma)(1 + \alpha_1 B)(1 + \alpha_2 B) \cdots (1 + \alpha_j B)
\]

and

\[
    C_j(\sigma') = s_{\min}(I, \sigma')(1 + \alpha_1 B)(1 + \alpha_2 B) \cdots (1 + \alpha_j B)
\]

for \( j = 1, 2, \ldots, n \). From (2) and (3), we have \( C_j(\sigma)/C_j(\sigma') = s_{\min}(I, \sigma)/s_{\min}(I, \sigma') \) which is independent of the choice of \( J_\tau \). Then we have

\[
    \frac{Z(I, \sigma)}{Z(I, \sigma')} = \frac{w_1 C_1(\sigma) + \cdots + w_n C_n(\sigma)}{w_1 C_1(\sigma') + \cdots + w_n C_n(\sigma')} = \frac{\min(I, \sigma)}{\min(I, \sigma')}
\]

Since \( A = 0 \), from the third condition in the definition of a regular instance-schedule pair, we have \( s_{\min}(I, \sigma') \leq r(1 + \alpha_{\max}B) + \alpha_{\max}A = r \cdot \rho(A, B) \). Since \( s_{\min}(I, \sigma') = r \), we have

\[
    s_{\min}(I, \sigma)/s_{\min}(I, \sigma') \leq \rho(A, B).
\]

From (4) and (5), together with the fact \( \text{OPT}(I) = Z(I, \sigma') \), we conclude that \( Z(I, \sigma) \leq (1 + \alpha_{\max}B) \cdot Z(I, \sigma') = \rho(A, B) \cdot \text{OPT}(I) \), as required.

Suppose in the following that \( A > 0 \). Then \( \rho(A, B) = 2 + \alpha_{\max}B \). Since the jobs in \( I \) have a common weighted growth rate \( \tau \), the processing order of the jobs in \( \sigma \) and \( \sigma' \) does not affect the objective value of the schedules. Then we may assume that \( \alpha_1 = \alpha_{\max} \). By induction on \( j = 1, 2, \ldots, n \), we will show in the following that

\[
    C_j(\sigma)/C_j(\sigma') \leq \rho(A, B).
\]

Note that, for each \( \pi \in \{\sigma, \sigma'\} \) and each \( j = 1, 2, \ldots, n \), we have

\[
    p_j(\pi) = \alpha_j A + \alpha_j B s_j(\pi) \quad \text{and} \quad C_j(\pi) = \alpha_j A + s_j(\pi)(1 + \alpha_j B).
\]
For $j = 1$, from (7), we have

$$C_1(\sigma) = \alpha_1 A + s_1(\sigma)(1 + \alpha_1 B)$$

$$= \alpha_1 A + s_{\min}(I, \sigma)(1 + \alpha_1 B) \text{ (since } s_1(\sigma) = s_{\min}(I, \sigma))$$

$$= \alpha_{\max} A + s_{\min}(I, \sigma)(1 + \alpha_{\max} B) \text{ (since } \alpha_1 = \alpha_{\max})$$

$$\leq \alpha_{\max} A + (r_1(1 + \alpha_{\max} B) + \alpha_{\max} A)(1 + \alpha_{\max} B),$$

where the last inequality follows from the fact $s_{\min}(I, \sigma) \leq r_1 + \alpha_{\max} B + \alpha_{\max} A$ which is guaranteed by the third condition in the definition of a regular instance-schedule pair. Moreover, from (7), together with the facts that $s_1(\sigma') = r$ and $\alpha_1 = \alpha_{\max}$, we have

$$C_1(\sigma') = \alpha_{\max} A + r(1 + \alpha_{\max} B).$$

From (8) and (9), we have $C_1(\sigma)/C_1(\sigma') \leq 1 + \alpha_{\max} B + \alpha_{\max} A/(\alpha_{\max} A + r(1 + \alpha_{\max} B)) \leq 2 + \alpha_{\max} B = \rho(A, B)$. It follows that (6) holds for $j = 1$.

Inductively, suppose that (6) holds for some $j$ with $1 \leq j \leq n - 1$. Then $C_j(\sigma)/C_j(\sigma') \leq \rho(A, B)$. Note that $s_{j+1}(\pi) = C_j(\pi)$ for $\pi \in \{\sigma, \sigma'\}$. From (7), we have $C_{j+1}(\sigma) = \alpha_{j+1} A + C_j(\sigma)(1 + \alpha_{j+1} B)$ and $C_{j+1}(\sigma') = \alpha_{j+1} A + C_j(\sigma')(1 + \alpha_{j+1} B)$. Then we have $C_{j+1}(\sigma)/C_{j+1}(\sigma') < C_j(\sigma)/C_j(\sigma') \leq \rho(A, B)$. It follows that (6) holds for each $j = 1, 2, \ldots, n$.

The above discussion implies that $C_j(\sigma)/C_j(\sigma') \leq \rho(A, B)$ for $j = 1, 2, \ldots, n$. It follows from the fact $\OPT(I) = Z(I, \sigma')$

$$Z(I, \sigma)/\OPT(I) = Z(I, \sigma')/\OPT(I) = w_1 C_1(\sigma) + \ldots + w_n C_n(\sigma)/w_1 C_1(\sigma') + \ldots + w_n C_n(\sigma') \leq \rho(A, B).$$

The lemma follows. \hfill \Box

**Lemma 2.8.** For problem 1|online, $r_j \geq t_0$, $p_j = \alpha_j(A + B s_j) \sum w_j C_j$, every online algorithm has a competitive ratio of at least $\rho(A, B) = 1 + \lambda(A) + \alpha_{\max} B$. The result is still valid when $w_j = 1$ for all jobs.

**Proof.** Assume that $w_j = 1$ for all jobs. When $A = 0$, the result holds trivially since Liu et al. [16] showed that, for problem 1|online, $r_j \geq t_0$, $p_j = \alpha_j s_j \sum C_j$, every online algorithm has a competitive ratio of at least $1 + \alpha_{\max}$. We may assume in the following that $A > 0$. Then $\rho(A, B) = 2 + \alpha_{\max} B$.

Now we prove the result by using the standard adversary method: The adversary provides an instance in which the jobs are released over time, and the prover schedules the released jobs. At time $t_1 = 0$, the adversary releases a job $J_0$ with $\alpha_0 = \alpha_{\max}$. The adversary executes algorithm $H$ and begins to process the job $J_0$ at some time $t$. If $t \geq \alpha_{\max} A$, then no jobs are released later, and so, $t = [J_0]$ is the job instance. We have $H(t) = t + \alpha_{\max} A + B t \geq \alpha_{\max} A(2 + \alpha_{\max} B)$ and $\OPT(t) = \alpha_{\max} A$. Thus, $H(t)/\OPT(t) \geq 2 + \alpha_{\max} B = \rho(A, B)$.

Suppose in the following that $t < \alpha_{\max} A$. Then just after job $J_0$ starts at time $t$, the adversary releases a set of $n$ jobs $J_1, J_2, \ldots, J_n$ with $\alpha_j = 0$ for $1 \leq j \leq n$ at time $t$ and no other jobs are released later. We have $t = [J_0, J_1, \ldots, J_n]$ and $H(t) \geq (t + \alpha_{\max} A + B t)(n + 1)$. In the off-line setting, the $n + 1$ jobs can all start at time $t$. Then we have $\OPT(t) \leq (n + 1) t + \alpha_{\max} A$. It follows that

$$\lim_{n \to +\infty} \frac{H(t)}{\OPT(t)} \geq 1 + \alpha_{\max} B + \frac{\alpha_{\max} A}{t} > 2 + \alpha_{\max} B.$$

The result follows. \hfill \Box

3. Algorithm and the framework of analysis

Let $I$ be a job instance whose jobs are released online over time. At time $t$, the online algorithm can only know the information of the jobs in $I$ released by time $t$. A job in $I$ is said to be available at time $t$ if the job has been released by time $t$ and has not been scheduled before time $t$. Let $U(t)$ be the set of all available jobs at time $t$.

Motivated by the idea of SWGR rule for the off-line problem 1|\ $p_j = \alpha_j(A + s_j) \sum w_j C_j$ in Mosheiov [21] and Kononov [12], we propose an online algorithm, called DSWGR, for the corresponding online version. A time $t$ is called a decision time in the online algorithm if either $t$ is the release date of some job, or $t$ is the completion time of some job, or $t = t_0(1 + \alpha B) + \alpha \lambda A$ for the candidate job $J_j$ at time $t$.

**DSWGR.** At the present time $t$, when $t$ is a decision time, $U(t) \neq \emptyset$, and the machine is idle, we choose one job $J_j \in U(t)$ with the smallest weighted growth rate $r_j$ as the candidate job at time $t$. If $t \geq t_0(1 + \alpha B) + \alpha \lambda A$, then start to process the candidate job $J_j$ at time $t$ immediately; otherwise, we do nothing until the next decision time.

Note that the inequality $t \geq t_0(1 + \alpha B) + \alpha \lambda A$ in the above algorithm is necessary for guarantee a competitive ratio of $\rho(A, B)$ for an online algorithm $H$. A modification of the proof of Lemma 2.8 can show its necessity: The first job $J_1$ with $\alpha_1 = \alpha_{\max}$ is released at time $r_1$, where $t_0 \leq r_1 < t_0(1 + \alpha_{\max} B) + \alpha \lambda A$. If $H$ begins to process $J_1$ at a time $t < t_0(1 + \alpha_{\max} B) + \alpha \lambda A$, then a large group of jobs with $\alpha_j = 0$ arrives at time $t$ and no other jobs are released later. Through a simple calculation, we can conclude that $H(t)/\OPT(t) > \rho(A, B)$. This proves the necessity of the inequality.
There are at most $O(n)$ decision times in algorithm DSWGR and at each decision time the candidate job can be determined in $O(n)$ time. Hence, the online algorithm DSWGR runs in polynomial $O(n^2)$ time. A more refined analysis can show that DSWGR runs in $O(n \log n)$ time. But in DSWGR whence a candidate $J_j$ is chosen at time $t$ and $t < t_0(1 + \alpha B) + \alpha A$, we have nothing to do until the next decision time. Such a rigidity causes a difficulty for the direct analysis of the competitive ratio. To avoid this rigidity, we propose the following generalized algorithm, called Flexible DSWGR (shortly FDSWGR).

**FDSWGR.** At the present time $t$, when $U(t) \neq \emptyset$ and the machine is idle, we choose one job $J_j \in U(t)$ with the smallest weighted growth rate $\tau_j$ as the candidate job at time $t$. If $t \geq t_0(1 + \alpha B) + \alpha A$, then start to process $J_j$ at time $t$ immediately; otherwise, wait until the next time.

We list three conditions for processing a job $J_j$ at time $t$ in FDSWGR, which may lead to a better understanding for the algorithm.

**Availability Condition.** $U(t) \neq \emptyset$ and the machine is idle at time $t$.

**Candidate Condition.** $J_j \in U(t)$ has the smallest weighted growth rate $\tau_j$ in $U(t)$.

**Processing Condition.** $t \geq t_0(1 + \alpha B) + \alpha A$.

At each time $t$ with $U(t) \neq \emptyset$ and the machine being idle, FDSWGR needs to choose a candidate job at time $t$. Such times $t$ may be regarded as the decision times in algorithm FDSWGR. Therefore, FDSWGR may have infinitely many decision points. As a result, the running time of FDSWGR is not polynomial. Due to the flexibility of FDSWGR, for a given job instance, FDSWGR may generate different schedules, each one of them is called a possible schedule generated by FDSWGR. But, the flexibility of FDSWGR enables us to use the technique of “instance-reduction” freely in the analysis of the competitive ratio of FDSWGR. We finally show that, for a given job instance, each possible schedule generated by FDSWGR has an objective value at most $\rho(A, B)$ times the objective value of an optimal (off-line) schedule. Note that the schedule generated by DSWGR is also a possible schedule generated by FDSWGR. Then DSWGR is also an online algorithm of competitive ratio $\rho(A, B)$.

We concentrate our attention on the analysis of FDSWGR in the sequel. For a job instance $I$ and a schedule $\sigma$ obtained by FDSWGR on $I$, we call $(I, \sigma)$ an online instance-schedule pair. A time $t$ is called a machine-available time in an online instance-schedule pair $(I, \sigma)$ if either $t = s_j(I, \sigma)$ for some job $J_j \in I$ or the machine is idle at time $t$ in $(I, \sigma)$.

The following two lemmas are implied in the implementation of FDSWGR and can be easily verified from the above three conditions.

**Lemma 3.1.** For each job $J_j$ in an online instance-schedule pair $(I, \sigma)$, we have $s_j(I, \sigma) \geq t_0(1 + \alpha B) + \alpha A$.

**Lemma 3.2.** At each time $t$ in the online algorithm, the candidate job $J_j$ (if any) selected by FDSWGR has the smallest weighted growth rate $\tau_j$ in $U(t)$.

**Lemma 3.3.** Let $I$ be a job instance and let $\sigma$ be a feasible schedule of $(I, \sigma)$ is an online instance-schedule pair if and only if the following two conditions are satisfied at every time $t$:

1. **If** $t = s_j(I, \sigma)$ for some job $J_j \in I$, **then** $t \geq t_0(1 + \alpha B) + \alpha A$, and $J_j$ has the smallest weighted growth rate in $U(t)$.
2. **If** $t$ is an idle time in $(I, \sigma)$ with $U(t) \neq \emptyset$ and no jobs starts at time $t$ in $(I, \sigma)$, **then** there is a job $J_j \in U(t)$ such that $J_j$ has the smallest weighted growth rate in $U(t)$ and $t_0(1 + \alpha B) + \alpha A > t$.

**Proof.** Suppose first that $(I, \sigma)$ is an online instance-schedule pair. Then condition (i) follows from the candidate condition and the processing condition. To show the validity of condition (ii), let $J_j$ be the candidate job at time $t$. Then $J_j$ has the smallest weighted growth rate in $U(t)$. Since $J_j$ is not processed at time $t$, the only possibility is that $J_j$ does not satisfy the processing condition at time $t$, and therefore, $t_0(1 + \alpha B) + \alpha A > t$. Condition (ii) follows.

Conversely, suppose that $(I, \sigma)$ is an instance-schedule pair such that the above two conditions are satisfied. A possible implementation of the online algorithm FDSWGR on instance $I$ can be stated in the following way:

At each time $t \geq t_0$ with $U(t) \neq \emptyset$ and the machine being idle, do the following:

1. **If** $t = s_j(I, \sigma)$ for some job $J_j \in I$, **choose** $J_j$ as the candidate job at time $t$. In this case, condition (i) guarantees that FDSWGR begins to process job $J_j$ at time $t$.
2. **If** $t$ is an idle time in $(I, \sigma)$ with $U(t) \neq \emptyset$ and no jobs starts at time $t$ in $(I, \sigma)$, **choose** the job $J_j \in U(t)$ described in condition (ii) as the candidate job at time $t$. In this case, since $t_0(1 + \alpha B) + \alpha A > t$, the processing condition is not satisfied by the candidate job $J_j$ at time $t$. Then FDSWGR processes no jobs at time $t$.

The above implementation of FDSWGR schedules the jobs in $I$ in the same way as that in $I$. It follows that $(I, \sigma)$ is an online instance-schedule pair. The lemma follows. \qed

In the following, we will show that, for each online instance-schedule pair $(I, \sigma)$, we have $Z(I, \sigma) \leq \rho(A, B) \cdot \text{OPT}(I)$. As a result, FDSWGR has a competitive ratio of $\rho(A, B)$. By combining the lower bound given in Lemma 2.8, we can conclude that FDSWGR is a best possible online algorithm for problem 1 online. $r_j \geq t_0$, $p_j = \alpha_j(A + s_j B) \sum \omega_j C_j$.

An instance $I$ is said to be a counterexample if $Z(I, \sigma) > \rho(A, B) \cdot \text{OPT}(I)$ for a certain possible schedule $\sigma$ obtained by FDSWGR on instance $I$. In this case, we also say that $(I, \sigma)$ is a violated pair. Furthermore, $I$ is called a smallest counterexample if $I$ is a
Table 1
The definitions of $\Omega_0 = \Omega$, $\Omega_1$, $\Omega_2$ and $\Omega_3$.

| $\Omega_0$ | the set of smallest violated pairs $(I, \sigma)$; |
| $\Omega_1$ | the set of instance-schedule pairs $(I, \sigma) \in \Omega$ in which the jobs in each block are scheduled in the SWGR order; |
| $\Omega_2$ | the set of instance-schedule pairs $(I, \sigma) \in \Omega_1$ in which all jobs have a common weighted growth rate; |
| $\Omega_3$ | the set of regular instance-schedule pairs $(I, \sigma) \in \Omega_2$. |

Fig. 1. Processing blocks in FDSWGR schedules for $\Omega$, $\Omega_1$, $\Omega_2$ and $\Omega_3$.

counterexample such that $|I|$ (the number of jobs in $I$) is as small as possible. If $I$ is a smallest counterexample and $(I, \sigma)$ is a violated pair, we also say that $(I, \sigma)$ is a smallest violated pair. We define $\Omega$ to be the set of all smallest violated pairs. A block in $(I, \sigma)$ is defined to be a maximal subset $B$ of $I$ so that the jobs in $B$ are consecutively scheduled in $\sigma$.

Suppose to the contrary that FDSWGR has a competitive ratio greater than $\rho(A, B)$. Then $\Omega$ is not empty. We will make reductions step by step under the assumption that FDSWGR results a possible schedule with a competitive ratio greater than $\rho(A, B)$. The reduction procedure can be simply described as

$$(I, \sigma) \in \Omega \Rightarrow (I, \sigma) \in \Omega_1 \Rightarrow (I, \sigma) \in \Omega_2 \Rightarrow (I, \sigma) \in \Omega_3,$$

where each $\Omega_i$ has a more special structure than $\Omega_{i-1}$ with $\Omega_0 = \Omega$. To make it clearer to the readers, we list the definitions of $\Omega_0 = \Omega$, $\Omega_1$, $\Omega_2$ and $\Omega_3$ in the following Table 1.

The following Fig. 1 can easily show us the procedure of reductions. The reduction ends up with a regular instance-schedule pair in $\Omega_3$. Recall that in Lemma 2.7, for a regular instance-schedule pair $(I, \sigma)$, we have $Z(I, \sigma) \leq \rho(A, B) \cdot \text{OPT}(I)$, which contradicts the assumption that FDSWGR has a competitive ratio greater than $\rho(A, B)$. Thus, FDSWGR is an online algorithm with a competitive ratio of $\rho(A, B)$.

4. Rigorous analysis

We first establish some lemmas to reveal the properties of the violated pairs $(I, \sigma) \in \Omega$ in the following. The following inequality will be repeatedly used in our discussion:

$$Z(I, \sigma) > \rho(A, B) \cdot \text{OPT}(I) \text{ for all } (I, \sigma) \in \Omega. \quad (10)$$

**Lemma 4.1.** Let $(I, \sigma) \in \Omega$. Then $\tau_j > 0$ for every job $J_j \in I$.

**Proof.** It is sufficient to show that $\alpha_j > 0$ for every job $J_j \in I$. Suppose to the contrary that there is a job $J_j \in I$ such that $\alpha_j = 0$. Then $p_j(\sigma) = 0$ and $s_j(\sigma) = C_j(\sigma)$. We claim that

$$s_j(\sigma) \leq \rho(A, B) \cdot \tau_j. \quad (11)$$
If \( s_j(\sigma) = r_j \), the claim in (11) holds trivially since \( \rho(A, B) > 1 \). Thus we may assume that \( s_j(\sigma) > r_j \). From the implementation of FDSWGR, there is a job \( J_i \in I \) with \( \alpha_i > 0 \) satisfying
\[
s_i(\sigma) < r_j
\] (12)
and
\[
s_j(\sigma) = C_j(\sigma) = \alpha_i A + s_i(\sigma) \cdot (1 + \alpha_i B).
\] (13)

From **Lemma 3.1**, we have
\[
s_i(\sigma) \geq t_0(1 + \alpha_i B) + \alpha_i A.
\] (14)

Since \( \lambda(A) = 0 \) if \( A = 0 \) and \( \lambda(A) = 1 \) if \( A > 0 \), from (14), we have
\[
s_i(\sigma) \lambda(A) \geq \alpha_i A.
\] (15)

Combining (12), (13) and (15), we have
\[
s_j(\sigma) = \alpha_i A + s_j(\sigma) \cdot (1 + \alpha_i B) \text{ (from (13))}
\leq s_i(\sigma) \cdot (1 + \lambda(\sigma) A + \alpha_i B)
\leq s_i(\sigma) \cdot (1 + \lambda(A) + \alpha \lambda B) \text{ (from (15))}
= \rho(A, B) \cdot s_i(\sigma)
< \rho(A, B) \cdot r_j. \text{ (from (12))}
\]

The claim in (11) follows.

From **Lemma 3.3**, we can observe that \( (I \setminus \{J_j\}, \sigma |_{I \setminus \{J_j\}}) \) is also an online instance-schedule pair such that
\[
Z(I \setminus \{J_j\}, \sigma |_{I \setminus \{J_j\}}) = Z(I, \sigma) - w_jr_j.
\] (16)

From **Lemma 2.3**, we have
\[
\text{OPT}(I \setminus \{J_j\}) \leq \text{OPT}(I) - w_jr_j.
\] (17)

Combining (10), (11), (16) and (17), we have
\[
Z(I \setminus \{J_j\}, \sigma |_{I \setminus \{J_j\}}) = Z(I, \sigma) - w_jr_j \text{ (from (16))}
\geq Z(I, \sigma) - \rho(A, B) \cdot w_jr_j \text{ (from (11))}
> \rho(A, B) \cdot \text{OPT}(I) - \rho(A, B) \cdot w_jr_j \text{ (from (10))}
= \rho(A, B) \cdot \text{OPT}(I \setminus \{J_j\}). \text{ (from (17))}
\]

This contradicts the fact that \((I, \sigma)\) is a smallest violated pair. Hence, \( \alpha_j > 0 \), and so, \( \tau_j > 0 \) for every job \( J_j \in I \). The lemma follows. \( \square \)

**Lemma 4.2.** Let \((I, \sigma) \in \Omega\). Then there is no partition \((I_1, I_2)\) of \( I \) such that \( Z(I_1, \sigma |_{I_1}) \leq \rho(A, B) \cdot \text{OPT}(I_1) \) and \( Z(I_2, \sigma |_{I_2}) \leq \rho(A, B) \cdot \text{OPT}(I_2) \).

**Proof.** Suppose to the contrary that there is a partition \((I_1, I_2)\) of \( I \) such that \( Z(I_1, \sigma |_{I_1}) \leq \rho(A, B) \cdot \text{OPT}(I_1) \) and \( Z(I_2, \sigma |_{I_2}) \leq \rho(A, B) \cdot \text{OPT}(I_2) \). Then
\[
Z(I_1, \sigma_1) + Z(I_2, \sigma_2) = Z(I, \sigma),
\] (18)

From **Lemma 2.3**, we have
\[
\text{OPT}(I) \geq \text{OPT}(I_1) + \text{OPT}(I_2).
\] (19)

Combining (18), (19) and the contradicting assumption, we have \( Z(I, \sigma) \leq Z(I_1, \sigma_1) + Z(I_2, \sigma_2) \leq \rho(A, B) \cdot \text{OPT}(I_1) + \rho(A, B) \cdot \text{OPT}(I_2) \leq \rho(A, B) \cdot \text{OPT}(I) \). This contradicts the assumption that \((I, \sigma)\) is a violated pair. The lemma follows. \( \square \)

**Lemma 4.3.** Let \((I, \sigma) \in \Omega\). Suppose that \( J_i \) and \( J_j \) are two jobs in \( I \) such that \( s_j(I, \sigma) > s_i(I, \sigma) \). Then either \( \tau_j(I) \geq \tau_i(I) \) or \( \tau_j(I) < \tau_i(I) \) and \( r_j(I) > s_i(I, \sigma) \).

**Proof.** Assume otherwise. Then \( \tau_j(I) < \tau_i(I) \) and \( r_j(I) \leq s_i(I, \sigma) \). This means that both \( J_i \) and \( J_j \) are available at time \( s_i(I, \sigma) \), i.e., \( J_i, J_j \in U(s_i(I, \sigma)) \). Since \( J_j \) starts at time \( s_j(I, \sigma) \) in \((I, \sigma)\), \( J_j \) must be the candidate job at time \( s_j(I, \sigma) \). From **Lemma 3.2** (which is just the candidate condition), \( J_j \) is a job in \( U(s_j(I, \sigma)) \) with the smallest weighted growth rate. This violates the fact \( \tau_j(I) < \tau_i(I) \). The lemma follows. \( \square \)

Consider a violated pair \((I, \sigma)\). Recall that a block in \((I, \sigma)\) is a maximal subset \( B \) of \( I \) so that the jobs in \( B \) are consecutively scheduled in \( \sigma \). The last job in \( B \) is called the end-job of \( B \) in \((I, \sigma)\). For each job \( J_k \in I \), we use \( B_k^{(k)} \) to denote the block in \((I, \sigma)\) including \( J_k \).
Let \( J_k \in I \) which is not the end-job of \( B^{(k)} \). We use \( J_{k^+} \) to denote the successor job of \( J_k \) in \( I, \sigma \), \( i.e., J_{k^+} \in B^{(k)} \) and \( s_{k^+} (I, \sigma) = C_k (I, \sigma) \). Then \( J_k \) and \( J_{k^+} \) are two consecutive jobs in \( B^{(k)} \). We call \((J_k, J_{k^+})\) an SWGR-reverse pair in \((I, \sigma)\) if \( \tau_k (I) > \tau_{k^+} (I) \). In the case that there is no other SWGR-reverse pair \((J_j, J_{j^+})\) in \((I, \sigma)\) with \( s_j (I, \sigma) > s_k (I, \sigma) \), we also call \((J_k, J_{k^+})\) the last SWGR-reverse pair in \((I, \sigma)\).

For an SWGR-reverse pair \((J_k, J_{k^+})\) in \((I, \sigma)\), the following notations are used in our deduction:

- \( f^{(k)} \) is the maximal subset of \( B^{(k)} \) such that
  1. \( J_k \in f^{(k)} \) is the first starting jobs of \( f^{(k)} \) in \((I, \sigma)\).
  2. the jobs in \( f^{(k)} \) are consecutively scheduled in \( \sigma \), and
  3. \( \tau_j (I) = \tau_k (I) \) for all jobs \( J_j \in f^{(k)} \).

- \( s_{p_k} (I, \sigma) \) and \( C_{p_k} (I, \sigma) \) are the first starting time and the last completion time of the jobs in \( f^{(k)} \), respectively, in \((I, \sigma)\). Then \( s_{p_k} (I, \sigma) = s_{k^+} (I, \sigma) \).

- \( Q^{(k,1)} (I, \sigma) = \{ J_j \in B^{(k)} : s_j (I, \sigma) \geq C_{p_k} (I, \sigma) \} \).

- \( Q^{(k,2)} (I, \sigma) = \{ J_j \in I : \tau_j (I) < C_{p_k} (I, \sigma), s_j (I, \sigma) > C_{p_k} (I, \sigma) \} \).

- For \( i = 1, 2 \), \( \tau^{(k,i)} (I, \sigma) = \min \{ \tau_j (I) : J_j \in Q^{(k,i)} (I, \sigma) \} \).

- \( Q^{(k,1)} (I, \sigma) \) is called the SWGR-reverse index of the SWGR-reverse pair \((J_k, J_{k^+})\) in \((I, \sigma)\).

The above notations \( J_k, J_{k^+}, f^{(k)}(I, \sigma), Q^{(k,1)} (I, \sigma), \) and \( Q^{(k,2)} (I, \sigma) \) are described in Fig. 2 for a better understanding.

**Lemma 4.4.** For each SWGR-reverse pair \((J_k, J_{k^+})\) (if any) in a violated pair \((I, \sigma) \in \Omega \), we have

1. \( \tau_j (I) > s_k (I, \sigma) \) for each \( J_j \in f^{(k)} \), and
2. \( Z(f^{(k)}), \sigma) \geq \rho(A, B) \cdot \text{OPT}(f^{(k)}) \).

**Proof.** Let \((I, \sigma) \in \Omega \) and let \((J_k, J_{k^+})\) be an SWGR-reverse pair in \((I, \sigma)\). Then \( J_k \) is the successor job of \( J_k \) in \((I, \sigma)\) and \( \tau_k (I) = \tau_{k^+} (I) \) for all \( J_j \in f^{(k)} \). From **Lemma 4.3**, the relation \( \tau_{k^+} (I) = \tau_k (I) \) implies that \( \tau_j (I) > s_k (I, \sigma) \) for each \( J_j \in f^{(k)} \). This proves statement (i).

To show statement (ii), let \( I' \) be the job instance obtained from \( f^{(k)} \) by resetting \( \tau_j (I') = \tau_{\min} (f^{(k)}) \) for each \( J_j \in f^{(k)} \), where \( \tau_{\min} (f^{(k)}) = \min \{ \tau_j (I) : J_j \in f^{(k)} \} \). From **Lemma 2.2**, we have

\[
\text{OPT}(I') \leq \text{OPT}(f^{(k)}).
\]

We claim that \((I', \sigma')\) is a regular instance-schedule pair, where \( \sigma' = \sigma|_{I'} \). Recall that \( s_{p_k} (I, \sigma) \) and \( C_{p_k} (I, \sigma) \) are the first starting time and the last completion time of the jobs in \( f^{(k)} \), respectively, in \((I, \sigma)\).

In fact, from the definitions of \( f^{(k)} \) and \( I' \), all jobs in \( I' \) have a common release date \( \tau_{\min} (f^{(k)}) \) and a common weighted growth rate \( \tau_{k^+} (I) \). The jobs in \( I' \) are consecutively scheduled in the interval \([s_{p_k} (I, \sigma), C_{p_k} (I, \sigma)]\) in \( \sigma' \). Moreover, from statement (i) of this lemma, we have \( \tau_{\min} (f^{(k)}) \geq s_k (I, \sigma) \geq t_0 (1 + \alpha_B) + \alpha A \), where the second inequality follows from the processing condition of FDSWGR. Then \( s_{p_k} (I, \sigma) = C_k (I, \sigma) = \alpha A + s_k (I, \sigma) (1 + \alpha_B) < \tau_{\min} (f^{(k)}) (1 + \alpha_{\text{max}} B) + \alpha_{\text{max}} A \). Thus, from the definition of regularity, we conclude that \((I', \sigma')\) is a regular instance-schedule pair. The claim follows.

From the above claim and by using **Lemma 2.7**, we have

\[
Z(I', \sigma') \leq \rho(A, B) \cdot \text{OPT}(I').
\]

Note that \( Z(f^{(k)}, \sigma|_{f^{(k)}}) = Z(I', \sigma') \). From (20) and (21), we conclude that \( Z(f^{(k)}, \sigma|_{f^{(k)}}) \leq \rho(A, B) \cdot \text{OPT}(f^{(k)}) \). The lemma follows. □

**Lemma 4.5.** Let \((J_k, J_{k^+})\) be an SWGR-reverse pair (if any) in a violated pair \((I, \sigma) \in \Omega \). Then \( Q^{(k,1)} (I, \sigma) \neq \emptyset \) and \( \tau^{(k,2)} (I, \sigma) \geq \tau^{(k,1)} (I, \sigma) \).

**Proof.** Recall that \( Q^{(k,1)} (I, \sigma) \) is the set of jobs \( J_j \in B^{(k)} \) with \( s_j (I, \sigma) \geq C_{p_k} (I, \sigma) \), and \( Q^{(k,2)} (I, \sigma) \) is the set of jobs \( J_j \in I \) with \( r_j (I) < C_{p_k} (I, \sigma) \) and \( s_j (I, \sigma) > C_{p_k} (I, \sigma) \). We first show that \( Q^{(k,1)} (I, \sigma) \neq \emptyset \).
Suppose to the contrary that $Q^{(k,1)}(I, \sigma) = \emptyset$. Set $T = I \setminus I^{(k)}$. We claim that

$$Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T). \quad (22)$$

In fact, if $Q^{(k,2)}(I, \sigma) = \emptyset$, we can observe that, for each $J_j \in T$, either $C_{\text{w}}(I_j, \sigma) \leq C_k(I, \sigma)$ or $\tau_j(I) \geq C_{\text{w}}(I, \sigma) = C_{\text{w}}^{(k)}(I, \sigma)$. Then $(T, \sigma|_T)$ is clearly an online instance-schedule pair. Since $(I, \sigma) \in \Omega$ is a smallest violated pair and $|T| < |I|$, we have $Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T)$, as required in (22).

Suppose in the following that $Q^{(k,2)}(I, \sigma) \neq \emptyset$. Then there is some job $J_j \in I$ so that $\tau_j(I) < C_{\text{w}}^{(k)}(I, \sigma)$ and $s_j(I, \sigma) > C_{\text{w}}^{(k)}(I, \sigma)$. This means that at least one job is available at time $C_{\text{w}}^{(k)}(I, \sigma)$ in $(I, \sigma)$. Let $J_k$ be the candidate job at time $C_{\text{w}}^{(k)}(I, \sigma)$ in $(I, \sigma)$. Then the definition of a candidate job implies that $J_k$ has the smallest weighted growth rate in $U(C_{\text{w}}^{(k)}(I, \sigma))$. Since no job starts at time $C_{\text{w}}^{(k)}(I, \sigma)$ in $(I, \sigma)$, the only possibility is that the processing condition $t \geq t_0(1 + \alpha_k B) + \alpha_k A$ is not satisfied by $J_j = J_k$ at time $C_{\text{w}}^{(k)}(I, \sigma)$ in $(I, \sigma)$. Then we have

$$t < t_0(1 + \alpha_k B) + \alpha_k A \text{ for every } t \leq C_{\text{w}}^{(k)}(I, \sigma). \quad (23)$$

Now we define $T'$ to be a new instance obtained from $T$ by resetting the release date of $J_k$ as $r_{\text{release}}(T') = \min\{r_j(I), C_k(I, \sigma)\}$. Since $r_j(T') \leq r_j(I) = r_j(T)$ for each $J_j \in T$, from Lemma 2.2, we have

$$\text{OPT}(T') \leq \text{OPT}(T). \quad (24)$$

A possible implementation of the online algorithm FDSWGR for instance $T'$ can be stated as follows:

- In the time interval $[t_0, C_k(I, \sigma)]$, we run the algorithm as in $(I, \sigma)$.
- In the time interval $[C_k(I, \sigma), +\infty)$, we choose $J_k$ as the candidate job at each time. Then the fact in (23) guarantees that no jobs are processed in $[Q_k(I, \sigma), C_{\text{w}}^{(k)}(I, \sigma)]$.
- In the time interval $[C_{\text{w}}^{(k)}(I, \sigma), +\infty)$, we still run the algorithm as in $(I, \sigma)$.

The above possible implementation of FDSWGR on instance $T'$ implies that $(T', \sigma|_{T'})$ is an online instance-schedule pair. Since $(I, \sigma)$ is a smallest violated pair and $|T'| = |T| < |I|$, we have $Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T')$. Note that $Z(T, \sigma|_T) = Z(T', \sigma|_{T'})$. From (24), we conclude that $Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T)$. This proves the claim in (22).

From (22) and Lemma 4.4(iii), we can observe that $(T, I^{(k)})$ forms a partition of $I$ such that $Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T)$ and $Z(I^{(k)}), \sigma) \leq \rho(A, B) \cdot \text{OPT}(I^{(k)})$. This contradicts Lemma 4.2. It follows that $Q^{(k,1)}(I, \sigma) \neq \emptyset$.

Now, if $Q^{(k,2)}(I, \sigma) = \emptyset$ then $\tau^{(k,2)}(I, \sigma) = +\infty$. If $Q^{(k,2)}(I, \sigma) \neq \emptyset$ then all jobs in $Q^{(k,2)}(I, \sigma)$ are released before the starting time of the first job in $Q^{(k,1)}(I, \sigma)$ in $(I, \sigma)$. From Lemma 3.2, we get that $\tau^{(k,2)}(I, \sigma) \geq \tau^{(k,1)}(I, \sigma)$ The lemma follows. \quad \square

**Lemma 4.6.** There is a violated pair $(I, \sigma) \in \Omega$ such that the jobs in each block are scheduled in the SWGR order in $(I, \sigma)$.

**Proof.** Suppose to the contrary that every violated pair in $\Omega$ has at least one block in which the jobs are not scheduled in the SWGR order. Then every violated pair in $\Omega$ has at least one SWGR-reverse pair. Let $(I, \sigma)$ be a violated pair in $\Omega$ such that the following two conditions are satisfied:

(i) the number of SWGR-reverse pairs in $(I, \sigma)$ is as small as possible, and

(ii) subject to (i), the SWGR-reverse index of the last SWGR-reverse pair in $(I, \sigma)$ is as small as possible.

Let $(J_k, J_{k+1})$ be the last SWGR-reverse pair in $(I, \sigma)$. Then $\tau_k(I) = \tau_{k+1}(I) < \tau_{k+1}(I)$ for all jobs $J_j \in I^{(k)}$. From Lemma 4.1, we have $\min\{\tau_k(I), \tau^{(k,1)}(I, \sigma)\} > 0$. Set

$$\delta = \tau_k(I) / \min\{\tau_k(I), \tau^{(k,1)}(I, \sigma)\}. \quad (25)$$

Then $0 < \delta < 1$. Note that $I^{(k)} : \delta$ is the job instance obtained from $I$ such that the weight $w_j(I)$ of each job $J_j \in I^{(k)}$ is modified as $w_j(I^{(k)} : \delta) = \delta \cdot w_j(I)$. Moreover, since $\tau_j(I) = \tau_{k+1}(I)$ for each $J_j \in I^{(k)}$, the weighted growth rate of job each job $J_j \in I^{(k)}$ in the instance $I^{(k)} : \delta$ is given by

$$\tau_j(I^{(k)} : \delta) = \alpha_j / ((1 + \alpha_k B) \delta \cdot w_j(I))$$

$$= \tau_j(I)/\delta = \tau_{k+1}(I)/\delta$$

$$\geq \min\{\tau_k(I), \tau^{(k,1)}(I, \sigma)\}, \quad (26)$$

where the last equality follows from (25). Combining (26) and the fact $\tau^{(k,2)}(I, \sigma) \geq \tau^{(k,1)}(I, \sigma)$ in Lemma 4.5, we conclude the following claim.

**Claim 1.** Each job $J_j \in I^{(k)}$ has the minimum weighted growth rate at time $s_j(I, \sigma) = s_j(I^{(k)} : \delta, \sigma)$ in $(I^{(k)} : \delta, \sigma)$.

From Lemma 4.4(i), we have $r_j(I) > s_j(I, \sigma)$ for each $J_j \in I^{(k)}$. By putting Claim 1 in consideration, we can observe that for each job $J_j \in I^{(k)} : \delta$, $J_j$ is a job with the smallest weighted growth rate in $U(s_j(I^{(k)} : \delta, \sigma))$. Thus condition (i) in Lemma 3.3 holds for the time $t = s_j(I^{(k)} : \delta, \sigma)$. Moreover, since $\tau_j(I) < \tau_j(I^{(k)} : \delta)$ for $J_j \in I^{(k)}$ and there are no idle times directly before every job $J_j \in I^{(k)}$ in $(I^{(k)} : \delta, \sigma)$, condition (ii) in Lemma 3.3 holds trivially. Thus, from Lemma 3.3, we conclude the following claim.
Claim 2. \((f^{(k)}, \delta), \sigma)\) is an online instance-schedule pair.

From Lemmas 2.5 and 4.4, we have \(Z(f^{(k)}, \delta), \sigma) > \rho(A, B) \cdot \text{OPT}(f^{(k)}, \delta)\). Hence, from Claim 2, we have

Claim 3. \((f^{(k)}, \delta), \sigma)\) is also a violated pair in \(\Omega\).

If \(\delta = \tau_k(I) / \tau_k(I)\), from (26), we have

\[ \tau_k(I) / \tau_k(I) = \tau_k(I) / \tau_k(I) = \tau_k(I) / \tau_k(I) \leq \tau^{(k,1)}(I, \sigma) \]

for all \(I \in \mathcal{I}\). This implies that \((J_k, J_{k'}I)\) is no longer an SWGR-reverse pair in \((f^{(k)}, \delta), \sigma)\) and no new SWGR-reverse pair occurs in \((f^{(k)}, \delta), \sigma)\). From Claim 3, we conclude that \((f^{(k)}, \delta), \sigma)\) is a violated pair in \(\Omega\) such that the number of SWGR-reverse pairs in \((f^{(k)}, \delta), \sigma)\) is less than that in \((I, \sigma)\). This contradicts the assumption that \((I, \sigma)\) is chosen under condition (i).

If \(\delta = \tau_k(I) / \tau^{(k,1)}(I, \sigma)\), from (26), we have

\[ \tau_k(I) / \tau^{(k,1)}(I, \sigma) = \tau_k(I) / \tau^{(k,1)}(I, \sigma) = \tau^{(k,1)}(I, \sigma) \]

for all \(I \in \mathcal{I}\). Then the number of SWGR-reverse pairs in \((f^{(k)}, \delta), \sigma)\) is the same as that in \((I, \sigma)\) and \((J_k, J_{k'}I)\) is also the last SWGR-reverse pair in \((f^{(k)}, \delta), \sigma)\). But then, at least the first jobs in \(Q^{(k,1)}(I, \sigma)\) are moved to \(f^{(k)}\) in \((f^{(k)}, \delta), \sigma)\), and so, \(|Q^{(k,1)}(f^{(k)}, \delta), \sigma)| < |Q^{(k,1)}(I, \sigma)|.\) This contradicts the assumption that \((I, \sigma)\) is chosen under condition (ii). The lemma follows. □

Based on Lemma 4.6, as in Table 1, we define \(\Omega_1\) to be the set of violated pairs \((I, \sigma) \in \Omega\) such that the jobs in each block in \((I, \sigma)\) are scheduled in the SWGR order in \(\sigma\). Then \(\Omega_1\) is not empty.

Lemma 4.7. There is a violated pair \((I, \sigma) \in \Omega_1\) in which all jobs have a common weighted growth rate.

Proof. Let \((I, \sigma) \in \Omega_1\) such that the number of distinct weighted growth rates of the jobs in \(I\) is as small as possible. From the definition of \(\Omega_1\), the jobs in each block of \((I, \sigma)\) are scheduled in the SWGR order in \((I, \sigma)\). If possible, suppose that the jobs in \(I\) have at least two distinct weighted growth rates. Let \(\tau_{\max}(I) = \max\{\tau(I) : J \in I\}\) and set \(T = \{J \in I : \tau(I) = \tau_{\max}(I)\}\). Since every job in \(\Lambda\) has a weighted growth rate less than \(\tau_{\max}(I)\), from Lemma 4.3, we have the following claim.

Claim 1. For each job \(J \in T\), there is no job \(J \in \Lambda\) with \(r(I) \leq s(I, \sigma)\) and \(s(I, \sigma) > t(I, \sigma)\).

Claim 1 implies that, if \(J \in T\) and \(J \in T\) such that \(s(I, \sigma) > s(I, \sigma)\), then \(s(I, \sigma) < t(I, \sigma)\). Let \(T^*\) be the set of jobs \(J \in T\) such that there is some job \(J \in T\) with \(s(I, \sigma) < r(I, \sigma)\). Then \(J \in T^*\) is scheduled in the SWGR order in \(\sigma\). From Claim 1, we have the following claim.

Claim 2. Each job in \(T^*\) is an end-job of a block in \((I, \sigma)\).

For each job \(J \in T^*\), we define \(J_f\) to be the candidate job at time \(C(J, \sigma)\). From Claim 2, FDSWGR does not schedule \(J_f\) starting at time \(C(J, \sigma)\). Then the only possibility is that the processing condition is not satisfied by \(J_f\) at time \(C(J, \sigma)\), and so,

\[ t_0(1 + \alpha B) + \alpha A > C(J, \sigma) \]

for each \(J \in T^*\). Let \(T^*\) be the instance obtained from \(\Lambda\) by resetting \(r(I') = s(I, \sigma) < t(I', \sigma)\) for each job \(J \in T^*\).

Claim 3. \((I', \sigma_{|I'})\) is an online instance-schedule pair.

To prove Claim 3, we partition the time space \([t_0, +\infty)\) into three subsets \(L_1, L_2, L_3, L_1\) consists the time points \(t\) at which some job in \(\Lambda\) is processed in \((I, \sigma)\). \(L_2 = \sum_{J \in \tau_{\max}(I)} s(I, \sigma)\). \(C(J, \sigma)\), and \(L_3 = [t_0, +\infty) \setminus (L_1 \cup L_2)\). The following three facts for \((I', \sigma_{|I'})\) can be observed.

• At each time \(t \in L_1\), there are no available jobs in \(I'\), i.e., \(U(t) = \emptyset\) for instance \(I'\). This fact follows from Claim 1 and the definition of \(T^*\). Then the two conditions for \((I', \sigma_{|I'})\) in Lemma 3.3 are not related to \(L_1\).

• Each time \(t \in L_2\) is an idle time in \((I', \sigma_{|I'})\). Then condition (i) for \((I', \sigma_{|I'})\) in Lemma 3.3 is not related to \(L_2\). Now let \(t \in L_2\) and let \(J \in T^*\) such that \(t \in [s(I, \sigma), C(J, \sigma))\). Then \(r(I') = s(I, \sigma)\) \(\leq t\). Since \(J_f\) is the candidate job at time \(C(J, \sigma)\), \((I', \sigma_{|I'})\) has the smallest weighted growth rate at time \(t\) in \([s(I, \sigma), C(J, \sigma)]\), and so, at time \(t\). From (27), we further have

\[ t_0(1 + \alpha B) + \alpha A > t. \]

Then condition (ii) for \((I', \sigma_{|I'})\) in Lemma 3.3 is satisfied at time \(t\).

• For each time \(t \in L_3\), the two conditions for \((I', \sigma_{|I'})\) in Lemma 3.3 are naturally satisfied at time \(t\). This fact follows from the observation that, at each time \(t \in L_3\), the two conditions in Lemma 3.3 for \((I, \sigma_{|I'})\) are not related to \(T^*\). Since \(T^*\) consists of all jobs with the maximum weighted growth rate.

From Lemma 3.3 and the above discussions, we conclude that \((I', \sigma_{|I'})\) is an online instance-schedule pair. Claim 3 follows.

Since \(|I| = |I'|\) and \(I\) is a smallest counterexample, we have \(Z(I', \sigma_{|I'}) \leq \rho(A, B) \cdot \text{OPT}(I')\). Since \(\text{OPT}(I') \leq \text{OPT}(I\setminus T)\) from Lemma 2.2) and \(Z(I\setminus T, \sigma_{|I'}) = Z'(I', \sigma_{|I'})\), we have

\[ Z(I\setminus T, \sigma_{|I'}) \leq \rho(A, B) \cdot \text{OPT}(I\setminus T). \]

(28)

Let \(\tau^* = \max\{\tau(I) : J \in I\}\) be the second maximum weighted growth rate of the jobs in \(I\). Set \(\delta = \tau_{\max}/\tau^*.\) Then \(\delta > 1\). Since \((I, \sigma)\) is a violated pair, from (28) and Lemma 2.6, we have

\[ Z(I^{'}, \delta), \sigma) > \rho(A, B) \cdot \text{OPT}(I^{'}, \delta)). \]

(29)
Note that each job in $T$ has a weighted growth rate $\tau^*$ in instance $I(T, 3)$ and $\tau^*$ is the maximum weighted growth rate of the jobs in $I(T, 3)$. Then the two conditions in Lemma 3.3 are satisfied at any time $t \geq t_0$ for the instance-schedule pair $(I(T, 3), \sigma)$. From Lemma 3.3, $(I(T, 3), \sigma)$ is an online instance-schedule pair. From (29), we conclude that $(I(T, 3), \sigma) \in \Omega_1$. But then, the number of weighted growth rates of the jobs in $I(T, 3)$ is less than that in $I$. This contradicts the assumption that $(I, \sigma) \in \Omega_1$ such that the number of distinct weighted growth rates of the jobs in $I$ is as small as possible. Hence, $(I, \sigma)$ is a violated pair in $\Omega_1$ in which all jobs have a common weighted growth rate. The lemma follows. □

Based on Lemma 4.7, as in Table 1, we define $\Omega_2$ to be the set of violated pairs $(I, \sigma) \in \Omega_1$ such that the jobs in $I$ have a common weighted growth rate. Then $\Omega_2$ is not empty.

**Lemma 4.8.** There is a regular instance-schedule pair in $\Omega_2$.

**Proof.** Let $(I, \sigma)$ be a violated pair in $\Omega_2$. From the definition of $\Omega_2$, the jobs in $I$ have a common weighted growth rate, say $\tau$. We use $s^*$ to denote the starting time of the last block in $(I, \sigma)$. Let $I_1 = \{j \in I : s_j(I, \sigma) < s^*\}$ and $I_2 = \{j \in I : s_j(I, \sigma) \geq s^*\}$. Note that $I_2 \neq \emptyset$. We distinguish the following two cases.

**Case 1:** $\tau(I_1) + \alpha A < s^*$ for all jobs $J_j \in I_2$. Then the processing condition is satisfied by all jobs in $I_2$ before time $s^*$ in $(I, \sigma)$. Since no jobs in $I_2$ are processed before time $s^*$ in $(I, \sigma)$ and all jobs have a common weighted growth rate, the only possibility is that the availability condition is not satisfied by each job in $I_2$ before time $s^*$. This implies that $r_j(I) \geq s^*$ for all jobs $J_j \in I_2$. We claim that $I_1 = \emptyset$.

In fact, if $I_1 \neq \emptyset$, then both $(I_1, \sigma|_{I_1})$ and $(I_2, \sigma|_{I_2})$ are online instance-schedule pairs. From the minimality of $I$, we have $Z(I_1, \sigma|_{I_1}) \leq \rho(A, B) \cdot \text{OPT}(I_1)$ and $Z(I_2, \sigma|_{I_2}) \leq \rho(A, B) \cdot \text{OPT}(I_2)$. This contradicts Lemma 4.2. The claim follows.

The above claim implies that $I_2 = I$. Let $I'$ be the instance schedule obtained from $I$ by resetting $r_j(I') = s^*$ for all jobs $J_j \in I$. Then $(I', \sigma)$ is clearly a regular instance-schedule pair. From Lemma 3.3, it also can be observed that $(I', \sigma)$ is an online instance-schedule pair. From the facts that $Z(I', \sigma) = Z(I, \sigma)$, $\text{OPT}(I') \leq \text{OPT}(I)$ and $Z(I, \sigma)/\text{OPT}(I) > \rho(A, B)$, we conclude that $Z(I', \sigma)/\text{OPT}(I') > \rho(A, B)$. It follows that $(I', \sigma)$ is a regular instance-schedule pair in $\Omega_2$, as required.

**Case 2:** There is a job $J_j \in I_2$ such that $t_0(1 + \alpha A) + \alpha A \geq s^*$. Let $I''$ be the job instance obtained from $I$ by resetting $r_j(I'') = t_0$ for all jobs $J_j \in I$. Then an online schedule $\sigma''$ of $I''$ can be generated by DSWGR by the following way:

- At each time $t \in [0, s^*)$, DSWGR assigns $J_j$ as the candidate job. Since $t < s^* \leq t_0(1 + \alpha A) + \alpha A$, $J_j$ does not satisfy the processing condition at time $t$, and so, no job is processed in the time interval $[t_0, s^*)$.
- In the time interval $[s^*, C_2(I, \sigma)]$, the jobs in $I_2$ are scheduled in the same way as that in $\sigma$.
- For each available job $J_j$ with $s_j(I, \sigma) < s^*$, from time $C_2(I, \sigma)$, we have no restriction on the implementation of DSWGR and just schedule these jobs one by one at each time when the machine is available.

We can observe that there is only one block in $(I'', \sigma'')$. For each $J_j \in I$ with $s_j(I, \sigma) \geq s^*$, we have $s_j(I'', \sigma'') = s_j(I, \sigma)$, and for each $J_j \in I$ with $s_j(I, \sigma) < s^*$, we have $s_j(I'', \sigma'') \geq s_j(I, \sigma)$. Hence, we have $Z(I'', \sigma'') \geq Z(I, \sigma)$. Since $	ext{OPT}(I'') \leq \text{OPT}(I)$ and $Z(I, \sigma) > \rho(A, B) \cdot \text{OPT}(I)$, we conclude that $(I'', \sigma'')$ is a violated pair in $\Omega_2$. Note that the jobs in $I''$ have a common release date $t_0$, $(I'', \sigma'')$ has just one block, and $t_0(1 + \alpha_{\text{max}}) + \alpha_{\text{max}} A \geq t_0(1 + \alpha A) + \alpha A \geq s^*$. Then $(I'', \sigma'')$ is a regular instance-schedule pair. The lemma follows. □

The result in Lemma 4.8 tells us that there is a regular instance-schedule pair $(I, \sigma)$ in $\Omega_2$. As in Table 1, we define $\Omega_3$ to be the set of regular instance-schedule pairs $(I, \sigma) \in \Omega_2$, then $\Omega_3$ is not empty. But from Lemma 2.7, we have $Z(I, \sigma) \leq \rho(A, B) \cdot \text{OPT}(I)$ for any regular instance-schedule pair $(I, \sigma)$, which contradicts the fact that $(I, \sigma)$ is a violated pair. Then we have the following result.

**Lemma 4.9.** For problem 1|online, $r_j \geq t_0$, $p_j = \alpha_j(A + s_j B) \sum w_j C_j$, DSWGR is a best possible online algorithm with a competitive ratio of $\rho(A, B)$.

Since the schedule generated by DSWGR is also a possible schedule generated by DSWGR, from Lemma 4.9, we have the following final result.

**Theorem 4.10.** For problem 1|online, $r_j \geq t_0$, $p_j = \alpha_j(A + s_j B) \sum w_j C_j$, DSWGR is a best possible online algorithm with a competitive ratio of $\rho(A, B)$.

5. Conclusions

Deteriorating job scheduling has caught much interest and attention in the literature due to its various applications, and many authors consider the off-line scene where all the information of each job is known at the beginning. In this work, we visit for the first time the problem 1|online, $r_j \geq t_0$, $p_j = \alpha_j(A + s_j B) \sum w_j C_j$. By a novel analysis technique, our algorithm DSWGR is a best possible online algorithm with a competitive ratio of $\rho(A, B)$. Furthermore, the analysis technique can also be applied not only in online scheduling but also in other online combinatorial optimization problems. However, there is no any result for the online scheduling with deteriorating job on parallel machines. The problem is also very interesting and it will be worth the effort.
Appendix A. Proof of Lemma 2.4

Let \( \Pi \) be the set of all feasible schedules of instance \( I \). Then \( \Pi \) is also the set of all feasible schedules of instance \( I^{T, \delta} \). For each \( \sigma \in \Pi \), we have \( Z(I^{T, \delta}, \sigma) = Z(I, \sigma) + (\delta - 1) \cdot \sum_{j \in T} Z_j(I, \sigma) \). This implies that \( Z(I^{T, \delta}, \sigma) \) is a linear function, and so, a concave function in \( \delta \in [u, v] \). Since \( \text{OPT}(I^{T, \delta}) = \min \{ Z(I^{T, \delta}, \sigma) : \sigma \in \Pi \} \), we conclude that \( \text{OPT}(I^{T, \delta}) \) is a concave function in \( \delta \in [u, v] \). The lemma follows.

Appendix B. Proof of Lemma 2.5

Suppose to the contrary that \( Z(I^{T, \delta}, \sigma) / \text{OPT}(I^{T, \delta}) < Z(I, \sigma) / \text{OPT}(I) \). Let \( M \) be an arbitrary positive number with \( M > 1 \). For each \( \delta \in [\delta, M] \), we have \( Z(I^{T, \delta}, \sigma) = Z(I, \sigma) + (\delta - 1) \cdot \sum_{j \in T} Z_j(I, \sigma) \). Thus, \( Z(I^{T, \delta}, \sigma) \) is a monotonically increasing linear function in \( \delta \in [\delta, M] \). From Lemma 2.4, \( \text{OPT}(I^{T, \delta}) \) is a concave function in \( \delta \in [\delta, M] \). By Lemma 2.1, we have

\[
\frac{Z(I^{T, \delta}, \sigma)}{\text{OPT}(I^{T, \delta})} \leq \max \left\{ \frac{Z(I^{T, \delta}, \sigma)}{\text{OPT}(I^{T, \delta})}, \frac{Z(I^{T, M}, \sigma)}{\text{OPT}(I^{T, M})} \right\},
\]

for all \( \delta \in [\delta, M] \). Note that \( 1 \in [\delta, M] \). From (30), we have

\[
\frac{Z(I, \sigma)}{\text{OPT}(I)} \leq \max \left\{ \frac{Z(I^{T, \delta}, \sigma)}{\text{OPT}(I^{T, \delta})}, \frac{Z(I^{T, M}, \sigma)}{\text{OPT}(I^{T, M})} \right\}.
\]

Together with the assumption that \( Z(I^{T, \delta}, \sigma) / \text{OPT}(I^{T, \delta}) < Z(I, \sigma) / \text{OPT}(I) \), we have

\[
\frac{Z(I^{T, M}, \sigma)}{\text{OPT}(I^{T, M})} \geq \frac{Z(I, \sigma)}{\text{OPT}(I)},
\]

for every \( M > 1 \).

Consider a feasible schedule of instance \( I^{T, M} \) in which the jobs in \( T^{T, M} \) are scheduled optimally and all the jobs in \( I^{T, M} \setminus T^{T, M} \) are scheduled after the jobs in \( T^{T, M} \) and are processed as early as possible. Then we have \( \text{OPT}(I^{T, M}) \leq \text{OPT}(I^{T, T}) + \sum_{j \in I^{T, M} \setminus T^{T, M}} w_j C^* \leq \text{OPT}(T^{T, M}) + \sum_{j \in T} w_j C^* = M \cdot \text{OPT}(T) + \sum_{j \in T} w_j C^* \), where \( C^* \) is an upper bound of the completion times of all jobs. Since \( T^{T, M} \) is a proper subset of \( I^{T, M} \), we also have \( M \cdot \text{OPT}(T) = \text{OPT}(I^{T, M}) \leq \text{OPT}(I^{T, M}) \). Then we have

\[
\text{OPT}(T) \leq \frac{\text{OPT}(I^{T, M})}{M} \leq \text{OPT}(T) + \frac{\sum_{j \in T} w_j C^*}{M},
\]

and hence,

\[
\lim_{M \to +\infty} \frac{\text{OPT}(I^{T, M})}{M} = \text{OPT}(T).
\]

Note that \( Z(I^{T, M}, \sigma) \leq M \cdot \text{OPT}(T) \),\( Z(I, \sigma) \), and \( \text{OPT}(I^{T, M}) \geq M \cdot \text{OPT}(T) \). Recall that \( T \) is a proper subset of \( I \) such that \( Z(T, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(T) \). Then we have

\[
\lim_{M \to +\infty} \frac{Z(I^{T, M}, \sigma)}{\text{OPT}(I^{T, M})} \leq \frac{Z(T, \sigma|_T)}{\text{OPT}(T)} \leq \rho(A, B).
\]

From (31) and (32), we conclude that \( Z(I, \sigma) / \text{OPT}(I) \leq \rho(A, B) \). This contradicts the assumption that \( Z(I, \sigma) / \text{OPT}(I) > \rho(A, B) \). The lemma follows.

Appendix C. Proof of Lemma 2.6

Let \( \overline{T} = I \setminus T \). Then \( Z(\overline{T}, \sigma|_T) \leq \rho(A, B) \cdot \text{OPT}(\overline{T}) \). Let \( \delta = 1 / \overline{\delta} \). Then \( 0 < \delta \leq 1 \). From Lemma 2.5, we have \( Z(I^{T, \delta}, \sigma)/\text{OPT}(I^{T, \delta}) \geq Z(I, \sigma) / \text{OPT}(I) \). Since \( Z(I^{T, \delta}, \sigma) = \delta \cdot Z(I^{T, \delta}, \sigma) \) and \( \text{OPT}(I^{T, \delta}) = \delta \cdot \text{OPT}(I^{T, \delta}) \), we also have \( Z(I^{T, \delta}, \sigma) / \text{OPT}(I^{T, \delta}) \geq Z(I, \sigma) / \text{OPT}(I) \). The lemma follows.
References

[19] R. Ma, J.J. Yuan. Online scheduling to minimize the total weighted completion time plus the rejection cost. Submitted for publication.