Appl. Math. J. Chinese Univ.
2020, 35(2): 141-156

Parallel-batch scheduling with deterioration and rejection on a single machine

LI Da-wei LU Xi-wen*

Abstract. The single machine parallel-batch scheduling with deteriorating jobs and rejection is considered in this paper. A job is either rejected, in which a rejection penalty should be paid, or accepted and processed on the machine. Each job's processing time is an increasing linear function of its starting time. The machine can process any number of jobs simultaneously as a batch. The processing time of a batch is equal to the largest processing time of the jobs in the batch. The objectives are to minimize the makespan and the total weighted completion time, respectively, under the condition that the total rejection penalty cannot exceed a given upper bound Q. We show that both problems are NP-complete and present dynamic programming algorithms and fully polynomial time approximation schemes ($FPTASs$) for the considered problems.

§1 Introduction

In traditional scheduling problems, it is often assumed that each machine can process at most one job at a time, the processing time of a given job is fixed and all the jobs have to be processed on the machine. However, some jobs may be processed together as a batch in the real world. The motivation for batching jobs is a gain in efficiency: it may be cheaper or faster to process jobs in a batch than to process them individually. Examples can be found in chemical, food and mineral processing, wafer fabrication process, etc. Furthermore, the actual processing time of a job grows when it is scheduled for processing later since the machine efficiency deteriorates over time due to machine usage and aging, which more accurately reflects real-life production. Examples can be found in steel production, fire fighting, financial problems, military problems, etc, where any delay in processing a task increases its completion time. Furthermore, in many practical cases, the scheduler has to reject or outsource some jobs with the larger processing times that

Received: 2018-07-11. Revised: 2020-01-15.
MR Subject Classification: 90B35, 90C39.
Keywords: parallel-batch scheduling, rejection, deterioration, $FPTAS$, NP-complete.
Digital Object Identifier(DOI): https://doi.org/10.1007/s11766-020-3624-2.
Supported by the National Natural Science Foundation of China (11871213, 71431004).
*Corresponding author.

bring the relatively smaller profits. Examples can be found in make-to-order production systems with limited production capacity and tight delivery requirements.

Parallel-batch scheduling problems have been extensively investigated for many years. The bounded version is motivated by the burn-in operations in semiconductor manufacturing. The original model of the bounded parallel-batch scheduling problem was introduced by Lee et al. in [8]. Brucker et al. [2] provided a systematic complexity analysis of parallel-batch scheduling problems on a single machine with respect to various regular objective functions. Lu et al. [11] considered the bound single machine parallel-batch scheduling problem with release dates and rejection to minimize the sum of makespan of the accepted jobs and the total rejection penalty. They gave a 2-approximation algorithm and a polynomial-time approximation algorithm for the general problem. Liu et al. [10] proposed two heuristics for the problem of scheduling jobs with release dates on parallel-batch processing machines to minimize the makespan and analyzed their worst-case performance ratios. For more papers on this topic, the reader can refer to Zhang et al. [23], Gao and Yuan [6] and so on.

Scheduling problems with time-dependent processing times have received more attention in recent years. Cheng et al. [3] presented a survey on scheduling problems with time-dependent processing times. Cheng and Sun [4] considered several single machine scheduling problems in which the processing time of a job is a linear function of its starting time and jobs can be rejected by paying penalties. The objectives are to minimize the makespan, the total weighted completion time and the maximum lateness/tardiness plus the total penalty of the rejected jobs. They showed that all these problems are NP-hard and designed algorithms based on dynamic programming. More works on scheduling problems with deteriorating jobs can be seen in Pei et al. [14], Yin et al. [21] and so on.

Scheduling problems with rejection are quite interesting both on practical and theoretical research. Researchers have paid a great deal of attention to them over the last decade. Bartal et al. [1] first introduced the notion of rejection. They considered the problem of minimizing the sum of makespan and total rejection penalty on identical parallel machines. Zhang et al. [22] investigated the single machine scheduling problem with release dates and rejection. The objective is to minimize the sum of makespan of the accepted jobs and the total rejection penalty of the rejected jobs. Lu et al. [12] considered the single-machine scheduling problem with rejection. The objective is to minimize the sum of the makespan of the accepted jobs and the total rejection penalty of the rejected jobs. They presented a polynomial-time algorithm for the off-line problem. Furthermore, they gave two on-line algorithms with the best-possible competitive ratio for the on-line problem, respectively. Shabtay et al. [17] provided a comprehensive survey for most existing off-line scheduling problems with rejection. For more off-line models and results on this topic, the reader can refer to Shabtay and Oran [18], Thevenin et al. [19], Zhang et al. [24] and so on.

Research on classical batch scheduling which is relevant to our model has been conducted by different researchers. Qi et al. [15] considered the unbounded parallel-batch scheduling problem with deteriorating jobs on a single machine. They gave polynomial time algorithms

for minimizing maximum cost, the number of tardy jobs, and the total weighted completion time and proved the NP-hardness for minimizing the weighted number of tardy jobs. Li et al. [9] studied the problem of scheduling n deteriorating jobs with release dates on a single batching machine to minimize the makespan. Both the bounded and unbounded issues have been investigated. For the unbounded model, they gave a dynamic programming algorithm in $O(n \log n)$ time and for the bounded model they showed that the problem is binary NP-hard even if there are only two distinct release dates. Several algorithms were proposed for both unbounded and bounded models. He et al. [5] considered the single machine parallel-batch scheduling with rejection. Two bi-criteria problems were considered and several algorithms were proposed for both unbounded and bounded models. Kong et al. [7] studied the bounded parallel-batch scheduling problem considering job rejection, deteriorating jobs, setup time, and non-identical job sizes. The objective is to minimize the sum of the makespan of the accepted jobs and the total rejection penalty of the rejected jobs. When the jobs have the identical size, They gave an dynamic programming algorithms. When the jobs have non-identical size, they presented a hybrid algorithm to obtain satisfactory solutions within reasonable time. Zou and Miao [25] studied parallel-batch scheduling of deteriorating jobs with release dates and rejection on a single machine, the objective is to minimize the sum of makespan of the accepted jobs and total rejection penalty of the rejected jobs. They gave two dynamic programming algorithms and an $FPTAS$. In this paper, we mainly study two constrained versions:

1. Minimize the makespan under the condition that the total rejection cost no more than a given upper bound Q.

2. Minimize the total weighted completion time under the condition that the total rejection cost does not exceed a given upper bound Q.

The remainder of this paper is organized as follows. In Section 2, we give a formal description of the considered problems and the relevant notations. In Section 3, for the problem to minimize the makespan subject to an upper bound on the total rejection penalty, we first show that it is NP-complete, then give a dynamic programming algorithm and a fully polynomial time approximation scheme ($FPTAS$). In Section 4, for problem to minimize the total weighted completion time subject to an upper bound on the total rejection penalty, we also first show that it is NP-complete, and then give a dynamic programming algorithm and an $FPTAS$. In Section 5, we summarize the results and give some topics for future research.

§2 Problem Formulation and Notation

The scheduling problem considered in this paper can be described as follows. Given a set of independent and non-preemptively deteriorating jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$. Job J_j is to be scheduled and processed on a single batching machine or rejected by paying a rejection penalty. Each job J_j $(j = 1, 2, \cdots, n)$ is associated with a deteriorating rate b_j, a weight w_j, a rejection penalty e_j and a release date r_j. The processing time p_j of job J_j is an increasing linear function of its starting time, given by $p_j = b_j t$, where $t \geq r_j$ is the starting time of job J_j. The batching

machine can process up to b jobs simultaneously. The processing time of a batch is equal to the largest processing time of the jobs in the batch. Whenever a set of jobs are processed on the machine in batch mode, the jobs in the batch have to start at the same time and can only be taken off the machine when the longest job in the set has completed its processing. This type of batching is referred to as parallel-batch or p-batch. Let A and R be the set of accepted jobs and the set of rejected jobs, respectively. A schedule σ can be simply denoted by a sequence of batches $\sigma = (B_1, B_2, \cdots, B_h)$, where each batch B_k $(k = 1, 2, \cdots, h)$ is a set of jobs. Let $S(B_k, \sigma)$, $C(B_k, \sigma)$, $b(B_k) = \max\{b_j : J_j \in B_k\}$, $r(B_k) = \max\{r_j : J_j \in B_k\}$ and $p(B_k) = \max\{p_j : J_j \in B_k\}$ be the starting time, the completion time, the deteriorating rate, the release date and the processing time of a batch B_k in a schedule σ, respectively. Note that the completion time of job $J_j \in B_k$ in σ is $C_j(\sigma) = C(B_k, \sigma)$. Unless ambiguity would result, we simplify $S(B_k, \sigma)$, $C(B_k, \sigma)$ and $C_j(\sigma)$ to $S(B_k)$, $C(B_k)$ and C_j, respectively.

In this paper, we study the unbounded parallel-batch scheduling with deteriorating jobs and rejection. The considered problems can be denoted as $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \le Q|F$, where $F \in \{C_{\max}, \sum_{J_j \in A} w_j C_j\}$. We study the computational complexities of these problems and present dynamic programming algorithms and $FPTASs$ for the considered problems.

§3 Scheduling to Minimize the Makespan

3.1 NP-completeness

For $1|p-batch, p_j = b_j t, r_j, b = \infty|C_{\max}$, Li et al. [9] showed that there exists an optimal batch processing order $\sigma = (B_1, B_2, \cdots, B_h)$ such that if two jobs J_i and J_j belong to distinct batches with $J_i \in B_x, J_j \in B_y$, and $x < y$, then $b_i > b_j$. Thus we can get the following lemma.

Lemma 3.1. *For the problem* $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \le Q|C_{\max}$, *there exists an optimal schedule* $\pi = (B_1^*, B_2^*, \cdots, B_h^*)$ *with* $\min\{b_k : J_k \in B_i^*\} \ge \max\{b_k : J_k \in B_j^*\}$ *for any* $i < j$.

In this subsection we will show that $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \le Q|C_{\max}$ is computationally intractable by performing a reduction from the following strongly NP-complete Product Partition problem (Ng et al. [13]).

Product Partition: Given a set of t positive integers $a_1, a_2, \cdots, a_t$, is there a subset $X \subseteq M := \{1, 2, \cdots, t\}$ such that $\prod_{i \in X} a_i = \prod_{i \in M \setminus X} a_i$?

Theorem 3.1. $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \le Q|C_{\max}$ *is* NP*-complete.*

Proof. The decision version of the scheduling problem is clearly NP. Given an arbitrary instance I of Product Partition, in the instance I we can omit $j \in M$ with $a_j = 1$ since it will not affect the product of any subset. Thus we can assume that $a_j \ge 2$ for every $j \in M$. In addition, we can assume that $\prod_{i=1}^{t} a_i = B^2$ and B is an integer. Otherwise it can immediately be answered that there is no solution to the instance. We construct an instance I' of the scheduling problem as follows:

- $n = 2t$ jobs.
- For each i with $1 \le i \le t$, we define two jobs J_{2i-1} and J_{2i} with

$$\begin{array}{lll} r_{2i-1} = B^{(i-1)(2t-i+2)}, & b_{2i-1} = B^{2(t-i+1)} - 1, & e_{2i-1} = \ln B + 1, \\ r_{2i} = B^{(i-1)(2t-i+2)}, & b_{2i} = B^{2(t-i+1)} a_i - 1, & e_{2i} = \ln a_i. \end{array}$$

- The upper bound is defined by $Q = \ln B$.
- The threshold value is defined by $Y = B^{t(t+1)+1}$.
- The decision asks whether there is a schedule π such that $C_{\max} \le Y$ under the constraint that $\sum_{J_j \in R} e_j \le Q$.

It can be observed that the above construction can be done in polynomial time. Assume first that there exists a subset X such that $\prod_{i \in X} a_i = \prod_{i \in M \setminus X} a_i$ for the instance I. We can construct a schedule for instance I' such that $C_{\max} \le Y$ and $\sum_{J_j \in R} e_j \le Q$ by the following way: If $i \in X$, we assign jobs J_{2i-1} and J_{2i} as a batch B_i. If $i \notin X$, i.e., $i \in M \setminus X$, we assign job J_{2i-1} as a batch B_i and reject job J_{2i} for $i = 1, 2, \cdots, t$. Then process the batches in the order of $B_1, B_2, \cdots, B_t$. It is not hard to verify that $C_{\max} = \max\limits_{1 \le i \le t} \{r(B_i) \prod_{j=i}^{t} (1 + b(B_j))\} = B^{t(t+1)+1} = Y$ and $\sum_{J_j \in R} e_j = \sum_{i \notin X} e_i = \ln(\prod_{i \notin X} a_i) = \ln B = Q$.

Conversely, suppose that there exists a schedule π for instance I' such that $C_{\max} \le Y = B^{t(t+1)+1}$ and $\sum_{J_j \in R} e_j \le Q = \ln B$. Based on Lemma 3.1, we can assume that schedule π satisfies Lemma 3.1. We will show that there is a subset X such that $\prod_{i \in X} a_i = \prod_{i \in M \setminus X} a_i$ for instance I. Denote by A and R the sets of accepted jobs and rejected jobs, respectively. We have the following claims.

Claim 1: Job $J_{2i-1} \in A$ for each $1 \le i \le t$.

In fact, if there exists some job $J_{2i-1} \in R$ with $1 \le i \le t$, then we have $\sum_{J_j \in R} e_j \ge e_{2i-1} = \ln B + 1 > Q$. A contradiction yields. Thus, we have $J_{2i-1} \in A$ for each $1 \le i \le t$.

Claim 2: For each pair i and j with $1 \le i < j \le t$, J_{2i-1} and J_{2j-1} cannot be contained in the same batch.

If Claim 2 is wrong, we can assume that J_{2i-1} and J_{2j-1} is the last pair of jobs containing in a same batch for $1 \le i < j$, i.e., jobs $J_{2j+1}, J_{2j+3}, \cdots, J_{2t-1}$ are contained in different batches. Then we have

$$\begin{aligned} C_{\max} &\ge \max\{r_{2j-1} + p_{2i-1}, r_{2j+1}\} + \textstyle\sum_{i=j+1}^{t} p_{2i-1} \\ &= \max\{B^{(j-1)(2t-j+2)} B^{2(t-i+1)}, B^{j(2t-j+1)}\} B^{(t-j)(t-j+1)} \\ &\ge B^{(j-1)(2t-j+2)+2(t-i+1)+(t-j)(t-j+1)} \\ &\ge B^{(j-1)(t+1)+(t-j+1)(t-1)+2(t-j+1)+2} \\ &= B^{t(t+1)+2} > Y. \end{aligned} \tag{1}$$

A contradiction yields. The penultimate inequality can be justified by $i < j, i.e., i \le j - 1$.

Claim 3: The number of batches in the schedule π is exactly t.

Let the number of batches in the schedule π be h. By Claim 2, we can know that $h \ge t$. If $h > t$, there exists at least a job $J_l \in A$ for $2(i-1) < l < 2i+1$ such that $J_l \in B_{t+1}$. Then we

have $C_{\max} \geq \prod_{k=1}^{h}(1+b(B_k)) \geq \prod_{k=1}^{t}(1+b_{2k-1})(1+b_l) > B^{t(t+1)}B^{2(t-i+1)} > B^{t(t+1)+2} > Y$. A contradiction yields.

Claim 4: If $J_{2i} \in A$, then we have $J_{2i} \in B_i$.

Assuming to the contrary that if there exists a job $J_{2i} \in A$ such that $J_{2i} \notin B_i$. According to Claims 1-3, we can know that $J_{2i-1} \in B_i$ for $1 \leq i \leq t$. If $J_{2i} \in A$, combining with Lemma 3.1, we can get that $J_{2i} \in B_k$ for $1 \leq k \leq i$.

If $i > k$, we have

$$\begin{aligned}
C_{\max} &\geq \max\{r_{2i} + p_{2k-1}, r_{2k+1}\} + \sum_{i=k+1}^{t} p_{2i-1} \\
&\geq \max\{B^{(i-1)(2t-i+2)}B^{2(t-k+1)}, B^{k(2t-k+1)}\} \prod_{i=k+1}^{t}(1+b_{2i-1}) \\
&\geq B^{(i-1)(2t-i+2)}B^{2(t-k+1)}B^{(t-k)(t-k+1)} \\
&\geq B^{k(2t-k+1)+2(t-k+1)+(t-k)(t-k+1)} \\
&= B^{t^2+3t-2k+2} \\
&\geq B^{t(t+1)+4} \\
&> Y
\end{aligned}$$

a contradiction.

By Claims 2 and 4, we have $B_i = \{J_{2i-1}, J_{2i}\}$ if $J_{2i} \in A$, and $B_i = \{J_{2i-1}\}$ if $J_{2i} \in R$. Let $X = \{i : J_{2i} \in A\}$. We are ready to show that X is a solution of instance I.

Since $\sum_{J_j \in R} e_j \leq Q = \ln B$, we have $\prod_{i \notin X} a_i \leq B$. If $\prod_{i \notin X} a_i < B$ then we have $C_{\max} = \max_{1 \leq i \leq t}\{r(B_i)\prod_{j=i}^{t}(1+b(B_j))\} = B^{t(t+1)}\prod_{i \in X} a_i > B^{t(t+1)+1} = Y$. A contradiction yields. Hence, we have $\prod_{i \in X} a_i = B$, and X is the required subset of instance I. Hence, Theorem 3.1 is true. □

3.2 Dynamic Programming Algorithm

Based on the above Lemma 3.1, we can only consider the schedules in which the accepted jobs are processed in non-increasing order of the deteriorating rates. Assume first that jobs have been indexed such that $b_1 \geq b_2 \geq \cdots \geq b_n$, then we present a dynamic programming algorithm for $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$.

Let function $A(j,k,t)$ be the minimum total rejection penalty when the jobs in consideration are $J_1, J_2, \cdots, J_j$, where k is the smallest index of the jobs in the last batch, t is the makespan of the accepted jobs among $J_1, J_2, \cdots, J_j$ and job J_j is accepted.

Let function $R(j,k,t)$ be the minimum total rejection penalty when the jobs in consideration are $J_1, J_2, \cdots, J_j$, where k is the smallest index of the jobs in the last batch, t is the makespan of the accepted jobs among $J_1, J_2, \cdots, J_j$ and job J_j is rejected.

In what follows we describe the recursion formulas for $A(j,k,t)$ and $R(j,k,t)$, we distinguish the following four cases for J_{j-1} and J_j.

Case 1. J_{j-1} and J_j are rejected. In the corresponding schedule for $J_1, J_2, \cdots, J_{j-1}$, the smallest index of the jobs in the last batch is k and the makespan of the accepted jobs is t. Since job J_j is rejected, we have $R(j,k,t) = R(j-1,k,t) + e_j$.

Case 2. J_{j-1} is accepted and J_j is rejected. In the corresponding schedule for $J_1, J_2, \cdots, J_{j-1}$, the smallest index of the jobs in the last batch is k and the makespan of the accepted jobs is t. Since job J_j is rejected, we have $R(j,k,t) = A(j-1,k,t) + e_j$.

Case 3. Job J_j is accepted and processed in the current batch, i.e., $k < j$. In the corresponding schedule for $J_1, J_2, \cdots, J_{j-1}$, if the completion time of the last batch is t', $r_k(1+b_k) \leq t' \leq t$, then the starting time of the last batch is $\frac{t'}{1+b_k}$. Since J_j is accepted and processed with the jobs in the last batch, the smallest index of the jobs in the last batch is still k and the starting time of the last batch is $\max\{\frac{t'}{1+b_k}, r_j\}$. Then we have that

$$A(j,k,t) = \min \begin{cases} \min\{A(j-1,k,t') : r_k(1+b_k) \leq t' \leq t\}, \\ \min\{R(j-1,k,t') : r_k(1+b_k) \leq t' \leq t\}, \end{cases}$$

where $t = \max\{\frac{t'}{1+b_k}, r_j\}(1+b_k) = \max\{t', r_j(1+b_k)\}$.

Case 4. Job J_j is accepted and processed in a new batch, i.e., $k = j$. In the corresponding schedule for $J_1, J_2, \cdots, J_{j-1}$, if the smallest index of the jobs in the last batch B_q is k', and the completion time of batch B_q is t''. The release date of job J_j must be larger than the starting time of batch B_q; otherwise, job J_j can be included in batch B_q. Therefore, we have $r_j > \frac{t''}{1+b_{k'}}$, or $t'' < r_j(1+b_{k'})$. Since job J_j is accepted and processed in a new batch, the starting time of the last batch is $\max\{t'', r_j\}$ and the smallest index of the jobs in the last batch is j. Then we have

$$A(j,k,t) = \min \begin{cases} \min\{A(j-1,k',t'') : 1 \leq k' \leq j-1, r_{k'}(1+b_{k'}) \leq t'' \leq r_j(1+b_{k'})\}, \\ \min\{R(j-1,k',t'') : 0 \leq k' \leq j-2, r_{k'}(1+b_{k'}) \leq t'' \leq r_j(1+b_{k'})\}, \end{cases}$$

where $t = \max\{t'', r_j\}(1+b_j)$.

Combining the above four cases, we design the following dynamic programming algorithm $DP1$.

Dynamic programming algorithm $DP1$

The initial conditions:

$$A(1,k,t) = \begin{cases} 0, & if\ k = 1, t = r_1(1+b_1); \\ +\infty, & otherwise. \end{cases}$$

$$R(1,k,t) = \begin{cases} e_1, & if\ k = 0, t = 0; \\ +\infty, & otherwise. \end{cases}$$

The recursive function:

If $k < j$, then

$$A(j,k,t)=\min\begin{cases}\min\{A(j-1,k,t'):r_k(1+b_k)\le t'\le t\},\\ \min\{R(j-1,k,t'):r_k(1+b_k)\le t'\le t\},\end{cases}$$

where $t=\max\{\frac{t'}{1+b_k},r_j\}(1+b_k)=\max\{t',r_j(1+b_k)\}$.

If $k=j$, then

$$A(j,k,t)=\min\begin{cases}\min\{A(j-1,k',t''):1\le k'\le j-1,r_{k'}(1+b_{k'})\le t''\le r_j(1+b_{k'})\},\\ \min\{R(j-1,k',t''):0\le k'\le j-2,r_{k'}(1+b_{k'})\le t''\le r_j(1+b_{k'})\},\end{cases}$$

where $t=\max\{t'',r_j\}(1+b_j)$.

Furthermore,

$R(j,k,t)=\min\{R(j-1,k,t)+e_j,A(j-1,k,t)+e_j\}$

The optimal value is given by

$\min\{t:0\le k\le n,0\le t\le r_{\max}\prod_{j=1}^{n}(1+b_j),\min\{A(n,k,t),R(n,k,t)\}\le Q\}$.

Theorem 3.2. *Algorithm $DP1$ solves problem $1|rej,p-batch,p_j=b_jt,r_j,b=\infty,\sum_{J_j\in R}e_j\le Q|C_{\max}$ in $O((nr_{\max}\prod_{j=1}^{n}(1+b_j))^2)$ time.*

Proof. The correctness of algorithm $DP1$ is guaranteed by the above discussion. The recursive function has at most $O(n^2r_{\max}\prod_{j=1}^{n}(1+b_j))$ states. If the job is accepted and processed in the current batch, i.e., $j>k$, each iteration costs $O(r_{\max}\prod_{j=1}^{n}(1+b_j))$ time. If the job is accepted and processed in a new batch, i.e., $k=j$, each iteration costs $O(nr_{\max}\prod_{j=1}^{n}(1+b_j))$ time; otherwise, each iteration costs a constant time. Hence, the total running time is bounded by $O((nr_{\max}\prod_{j=1}^{n}(1+b_j))^2)$. □

3.3 Fully Polynomial Time Approximation Scheme

An algorithm A is a $(1+\rho)$-approximation algorithm for a minimization problem if it produces a solution that is at most of $(1+\rho)$ times the optimal solution. A family of algorithms $\{A_\epsilon:\epsilon>0\}$ is called a fully polynomial time approximation scheme ($FPTAS$), if for each $\epsilon>0$, the algorithm A_ϵ is a $(1+\epsilon)$-approximation algorithm running in polynomial time in the input size and $1/\epsilon$.

An $FPTAS$ can be designed by rounding the input data of the instance or iteratively thinning out the state space of the dynamic program. A general technique for getting an $FPTAS$ from a dynamic program was developed by Woeginger in [20]. In this subsection, we present an $FPTAS$ by considering the modified deteriorating rates and the modified release dates of the scheduled jobs which involves a geometric rounding technique developed by Sengupta in [16]. And the rounding technique is stated as follows:

For any $\epsilon'>0$ and $x\ge1$, if $(1+\epsilon')^{k-1}<x<(1+\epsilon')^k$, then we define $\lceil x\rceil_{\epsilon'}=(1+\epsilon')^k$, $\lfloor x\rfloor_{\epsilon'}=(1+\epsilon')^{k-1}$. If x is an exact power of $(1+\epsilon')$, then $\lceil x\rceil_{\epsilon'}=\lfloor x\rfloor_{\epsilon'}=x$. Note that $\lceil x\rceil_{\epsilon'}\le(1+\epsilon')x$ for any $x\ge1$.

For any $\epsilon>0$, let $\epsilon'=\frac{\epsilon}{2(n+1)}$, we define the modified deteriorating rates and the modified release dates as $b'_j=\lceil1+b_j\rceil_{\epsilon'}-1$, $r'_j=\lceil r_j\rceil_{\epsilon'}$. Let L_j and R_j denote the exponent of $1+b'_j$ and r'_j, respectively, i.e., $1+b'_j=(1+\epsilon')^{L_j}$, $r'_j=(1+\epsilon')^{R_j}$, then $L_j=\frac{\log\lceil1+b_j\rceil_{\epsilon'}}{\log(1+\epsilon')}=O(\frac{n\log(1+b_j)}{\epsilon})$, $R_j=\frac{\log\lceil r_j\rceil_{\epsilon'}}{\log(1+\epsilon')}=O(\frac{n\log r_j}{\epsilon})$.

Lemma 3.2. *For any $0 < \epsilon \leq 2$, the optimal objective function value for problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$ with the modified deteriorating rates b'_j and the modified release dates r'_j is at most of $(1+\epsilon)$ times the optimal objective function value for problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$.*

Proof. Assume that $\sigma = (B_1, B_2, \cdots, B_h)$ is an optimal batch processing order of the accepted jobs for problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$. Correspondingly, let $\sigma' = (B'_1, B'_2, \cdots, B'_h)$ be a batch processing order of the accepted jobs for problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$ with the modified deteriorating rates b'_j and the modified release dates r'_j, where the jobs in B_i and B'_i have the same index. Then we have that

$$\begin{aligned} C'_{\max} &= \max_{1\leq i\leq m}\{r'(B'_i)\prod_{j=i}^{h}(1+b'(B'_j))\} \\ &\leq (1+\epsilon')\max_{1\leq i\leq m}\{r(B_i)\prod_{j=i}^{h}(1+b'(B'_j))\} \\ &\leq (1+\epsilon')^{n+1}\max_{1\leq i\leq m}\{r(B_i)\prod_{j=i}^{h}(1+b(B_j))\} \\ &\leq (1+\epsilon)C^*_{\max}, \end{aligned} \tag{2}$$

where $C'_{\max}$ is the objective function value of schedule σ' and $C^*_{\max}$ is the objective function value of schedule σ.

Consequently, $C'^*_{\max} \leq C'_{\max} \leq (1+\epsilon)C^*_{\max}$, where $C'^*_{\max}$ is the optimal value of the problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$ with the modified deteriorating rates b'_j and the modified release dates r'_j. □

Based on algorithm $DP1$, we can design a dynamic program for problem $1|rej, p-batch, p'_j = b'_j t, r'_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$. Since the deteriorating rate and the release date for any jobs are of the form $(1+\epsilon')^h$, so wan can get that the makespan is the form of $(1+\epsilon')^l$ and store the exponent l instead of the actual value t in the state of algorithm $DP1$. It's easily know that the maximum makespan C_{MAX} can be obtained if all the jobs are accepted and scheduled consecutively each job as a batch after all the jobs arrive, i.e., $C_{MAX} = r'_{\max}\prod_{j=1}^{n}(1+b'_j) = (1+\epsilon')^{R_{\max}+\sum_{j=1}^{n} L_j}$. Thus, it is easy to see that l range from -1 to L, where L is the smallest integer such that $(1+\epsilon')^L \geq (1+\epsilon')^{R_{\max}+\sum_{j=1}^{n} L_j}$. Hence, $L = O(\frac{n\log(\prod_{j=1}^{n}(1+b_j)r_{\max})}{\epsilon})$.

According to the above analysis and algorithm $DP1$, we can get the following conclusion.

Theorem 3.3. *The algorithm above can solve the problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$ with the modified deteriorating rates and the modified release dates in $O(n^2L^2) = O(\frac{n^4}{\epsilon^2}\log^2(\prod_{j=1}^{n}(1+b_j)r_{\max}))$ time.*

Proof. The correctness of algorithm is guaranteed by the above discussion. Based on the above analysis and Theorem 3.2, the algorithm can be run in $O(n^2L^2)$ time, where $L = O(\frac{n\log(\prod_{j=1}^{n}(1+b_j)r_{\max})}{\epsilon})$. □

Combining Lemma 3.2 and Theorem 3.3, we can easily get the following conclusion.

Theorem 3.4. *The algorithm above is an $FPTAS$ for problem $1|rej, p-batch, p_j = b_j t, r_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$.*

Proof. Let π'^* be an optimal schedule for problem $1|rej, p-batch, p_j = b'_j t, r'_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$. Replace b'_j and r'_j with b_j and r_j, $j = 1, 2, \cdots, n$ respectively, we can get a new schedule π. Based on Lemma 3.2, we have $C_{\max} \leq C'^*_{\max} \leq (1+\epsilon)C^*_{\max}$, where $C'^*_{\max}$ is the value of the optimal schedule π'^* for problem $1|rej, p-batch, p_j = b'_j t, r'_j, b = \infty, \sum_{J_j \in R} e_j \leq Q|C_{\max}$, $C_{\max}$ is the corresponding objective function value of the origin problem under the schedule π and $C^*_{\max}$ is the optimal value of the origin problem. Combining with Theorem 3.3, we know the algorithm is an $FPTAS$. □

§4 Scheduling to Minimize the Total Weighted Completion Time

4.1 NP-completeness

When rejection is not allowed, Qi et al. [15] showed that the non-decreasing order of b_j is optimal for problem $1|p-batch, p_j = b_j t, r_j = t_0, b = \infty| \sum w_j C_j$, which leads to the following lemma.

Lemma 4.1. *For problem* $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q| \sum_{J_j \in A} w_j C_j$, *there exists an optimal schedule* $\pi = (B^*_1, B^*_2, \cdots, B^*_h)$ *with* $\max\{b_k : J_k \in B^*_i\} < \min\{b_k : J_k \in B^*_j\}$ *for any* $i < j$.

Now we will show that the problem $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q| \sum_{J_j \in A} w_j C_j$ is NP-complete by reducing the Product Partition problem to our problem in polynomial time. Without loss of generality, we can assume that the parameter $t > 4$ which is defined in Product Partition problem.

Theorem 4.1. $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q| \sum_{J_j \in A} w_j C_j$ *is NP-complete.*

Proof. We show the result by reducing the Product Partition problem, which is strongly NP-complete [13], to our problem in polynomial time.

The decision version of the scheduling problem is clearly NP. Given an arbitrary instance I of Product Partition, we construct an instance I' of the scheduling problem as follows:

- $n = 2t + 1$ jobs, $t_0 = 1$.
- For each i with $1 \leq i \leq t$, we define two jobs J_{2i-1} and J_{2i} with
  $b_{2i-1} = B^{4i} - 1$, $w_{2i-1} = B^{2t(t+1)-2i(i+1)}$, $e_{2i-1} = \ln B + 1$.
  $b_{2i} = B^{4i} a_i - 1$, $w_{2i} = B^{2t(t+1)-2i(i+1)}$, $e_{2i} = \ln a_i$.
- For $j = 2t + 1$, we only have one job J_{2t+1} with
  $b_{2t+1} = B^{4(t+1)} - 1$, $w_{2t+1} = B^{-2-4t}$, $e_{2t+1} = \ln B + 1$.
- The upper bound is defined by $Q = \ln B$.
- The threshold value is defined by $Y = (2t + B^2) B^{2t(t+1)+1}$.

- The decision asks whether there is a schedule π such that $\sum_{J_j\in A} w_jC_j \le Y$ under the constraint that $\sum_{J_j\in R} e_j \le Q$.

It can be observed that the above construction can be done in polynomial time. Assume first that there exists a subset X such that $\prod_{i\in X} a_i = \prod_{i\in M\setminus X} a_i$ for instance I. We can construct a schedule for instance I' such that $\sum_{J_j\in A} w_jC_j \le Y$ and $\sum_{J_j\in R} e_j \le Q$ by the following way: If $i \in X$, we assign jobs J_{2i-1} and J_{2i} as a batch B_i. If $i \notin X$, i.e., $i \in M\setminus X$, we assign job J_{2i-1} as a batch B_i and reject job J_{2i} for $i = 1, 2, \cdots, t$. We also assign the job J_{2t+1} as a batch B_{t+1}. Then process the batches in the order of $B_1, B_2, \cdots, B_{t+1}$. It is not hard to verify that

$$\begin{aligned}\sum_{J_j\in A} w_jC_j &= \sum_{i=1}^{t+1}\sum_{J_j\in B_i} w_j\prod_{k=1}^{i}(1+b(B_k))\\ &\le 2B^{2t(t+1)}\sum_{j\in X}\Big(\prod_{i\in X, i\le j} a_i\Big) + B^{2t(t+1)}\sum_{j\notin X}\Big(\prod_{i\in X\cup\{0\}, i<j} a_i\Big)\\ &\quad + B^{2t(t+1)+3}\\ &\le (2t+B^2)B^{2t(t+1)+1} = Y\end{aligned} \tag{3}$$

where $a_0 = 1$, and $\sum_{J_j\in R} e_j = \sum_{i\notin X} e_i = \ln(\prod_{i\notin X} a_i) = \ln B = Q$.

Conversely, suppose that there exists a schedule π for instance I' such that $\sum_{J_j\in A} w_jC_j \le Y$ and $\sum_{J_j\in R} e_j \le Q$. Based on Lemma 4.1, we can assume that schedule π satisfies Lemma 4.1. We will show that there is a subset X such that $\prod_{i\in X} a_i = \prod_{i\in M\setminus X} a_i$ for instance I. Denote by A and R the sets of accepted jobs and rejected jobs, respectively. We have the following claims.

Claim 5: Job $J_{2i-1} \in A$ for each $1 \le i \le t+1$.

In fact, if there exists some job $J_{2i-1} \in R$ with $1 \le i \le t+1$, then we have $\sum_{J_j\in R} e_j \ge e_{2i-1} = \ln B + 1 > Q$. A contradiction yields. Thus, we have $J_{2i-1} \in A$ for each $1 \le i \le t+1$.

Claim 6: For each pair i and j with $1 \le i < j \le t+1$, J_{2i-1} and J_{2j-1} cannot be contained in the same batch.

If Claim 6 is wrong, we can assume that job J_{2i-1} and J_{2j-1} is the first pair of jobs containing in a same batch, $1 \le i < j$, i.e., jobs $J_1, J_3, \cdots, J_{2i-3}$ are contained in different batches. Then we have $\sum_{J_j\in A} w_jC_j \ge w_{2i-1}C_{2i-1} > Y$. A contradiction yields.

Claim 7: The number of batches in the schedule π is exactly $t+1$.

Let the number of batches in the schedule π be h. By Claim 6, we can know that $h \ge t+1$. If $h > t+1$, by Lemma 4.1, we can get that $J_{2t+1} \in B_h$. Then we have $\sum_{J_j\in A} w_jC_j \ge w_{2t+1}C_{2t+1} \ge B^{-2-4t}\prod_{i=1}^{t+1}(1+b_{2i-1})(1+b_k) \ge B^{2t(t+1)+6} > Y$. A contradiction yields.

Claim 8: If $J_{2i} \in A$, then we have $J_{2i} \in B_i$.

Assume to the contrary that there exists a job $J_{2i} \in A$ such that $J_{2i} \notin B_i$. According to Claims 5-7, we can know that $J_{2i-1} \in B_i$ for $1 \le i \le t+1$. If $J_{2i} \in A$, combining with Lemma 4.1, we can get that $J_{2i} \in B_k$ for $i \le k \le t+1$.

If $i < k$, we have

$$\sum_{J_j\in A} w_jC_j > w_{2i}C_{2i} \ge B^{2t(t+1)-2i(i+1)}B^{2k(k+1)} \ge B^{2t(t+1)+4(i+1)} > Y$$

a contradiction.

By Claims 6 and 8, we have $B_i = \{J_{2i-1}, J_{2i}\}$ if $J_{2i} \in A$, and $B_i = \{J_{2i-1}\}$ if $J_{2i} \in R$. Let $X = \{i : J_{2i} \in A\}$. We are ready to show that X is a solution of the instance I.

Since $\sum_{J_j \in R} e_j \leq Q = \ln B$, we have $\prod_{i \notin X} a_i \leq B$. If $\prod_{i \notin X} a_i < B$ then we have $\sum_{J_j \in A} w_j C_j > w_{2t+1} C_{2t+1} \geq B^{2t(t+1)+1}(B^2 + B) > B^{2t(t+1)+1}(B^2 + 2t) = Y$. A contradiction yields. The last inequality can be justified by $t > 4$. Hence we have $\prod_{i \in X} a_i = B$, and X is the required subset of instance I. Hence, Theorem 4.1 is true. □

4.2 Dynamic Programming Algorithm

In this subsection, we will give a dynamic programming algorithm for $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q| \sum_{J_j \in A} w_j C_j$. Based on the above Lemma 4.1, we sort the jobs such that $b_1 \leq b_2 \leq \cdots \leq b_n$.

Let function $A(j, k, W)$ be the minimum total rejection penalty when the jobs in consideration are $J_j, J_{j+1}, \cdots, J_n$, where k is the largest index of jobs in the first batch, W is the objective function value and job J_j is accepted.

Let function $R(j, k, W)$ be the minimum total rejection penalty when the jobs in consideration are $J_j, J_{j+1}, \cdots, J_n$, where k is the largest index of jobs in the first batch, W is the objective function value and job J_j is rejected.

In what follows we describe the recursion formulas for $A(j, k, W)$ and $R(j, k, W)$, we distinguish the following four cases for J_{j+1} and J_j.

Case 1. If J_{j+1} and J_j are rejected, we have $R(j, k, W) = R(j + 1, k, W) + e_j$.

Case 2. If J_{j+1} is accepted and J_j is rejected, we have $R(j, k, W) = A(j + 1, k, W) + e_j$.

Case 3. Job J_j is accepted and processed in the current batch, i.e., $k > j$. In the corresponding schedule for $J_{j+1}, J_{j+2}, \cdots, J_n$, the largest index of jobs in the first batch is k and the objective function value is $W - w_j t_0(1 + b_k)$. The contribution of J_j to the objective function value is $w_j t_0(1 + b_k)$, so we have

$$A(j, k, W) = \min\{A(j + 1, k, W - w_j t_0(1 + b_k)), R(j + 1, k, W - w_j t_0(1 + b_k))\}.$$

Case 4. Job J_j is accepted and processed in a new batch, i.e., $k = j$. In the corresponding schedule for $J_{j+1}, J_{j+2}, \cdots, J_n$, if the largest index of jobs in the first batch is k' , after J_j is added to this schedule, the completion time of the accepted jobs among $J_{j+1}, J_{j+2}, \cdots, J_n$ becomes $(1 + b_j)$ times their original value and the contribution of J_j to the objective function is $w_j t_0(1 + b_j)$. Then we have

$$A(j, k, W) = \min \left\{ \begin{array}{l} \min\limits_{j+1 \leq k' \leq n} \{A(j + 1, k', \frac{W - w_j t_0(1+b_j)}{1+b_j})\} \\ \min\limits_{j+2 \leq k' \leq n+1} \{R(j + 1, k', \frac{W - w_j t_0(1+b_j)}{1+b_j})\} \end{array} \right. .$$

Combining the above four cases, we design the following dynamic programming algorithm $DP2$.

Dynamic programming algorithm $DP2$

The initial conditions:

$$A(n,k,W)=\begin{cases}0, & if\ k=n, W=w_nt_0(1+b_n);\\ +\infty, & otherwise.\end{cases}$$

$$R(n,k,W)=\begin{cases}e_n, & if\ k=n+1, W=0;\\ +\infty, & otherwise.\end{cases}$$

The recursive function:

If $k=j$, then we have

$$A(j,k,W)=\min\begin{cases}\min\limits_{j+1\le k'\le n}\{A(j+1,k',\frac{W-w_jt_0(1+b_j)}{1+b_j})\};\\ \min\limits_{j+2\le k'\le n+1}\{R(j+1,k',\frac{W-w_jt_0(1+b_j)}{1+b_j})\}.\end{cases}$$

If $k>j$, then we have

$A(j,k,W)=\min\{A(j+1,k,W-w_jt_0(1+b_k)),R(j+1,k,W-w_jt_0(1+b_k))\}$.

Furthermore,

$R(j,k,W)=\min\{A(j+1,k,W)+e_j,R(j+1,k,W)+e_j\}$.

The optimal value is given by

$\min\{W:1\le k\le n+1, W\le nw_{\max}t_0\prod_{j=1}^{n}(1+b_j),\min\{A(1,k,W),R(1,k,W)\}\le Q\}$.

Theorem 4.2. *Algorithm $DP2$ solves problem $1|rej,p-batch,p_j=b_jt,r_j=t_0,b=\infty,\sum_{J_j\in R}e_j\le Q|\sum_{J_j\in A}w_jC_j$ in $O(n^3w_{\max}\prod_{j=1}^{n}(1+b_j))$ time.*

Proof. The correctness of the algorithm $DP2$ is guaranteed by the above discussion. The recursive function has at most $O(n^3w_{\max}\prod_{j=1}^{n}(1+b_j))$ states. If the job is accepted and processed in a new batch, i.e., $k=j$, each iteration costs $O(n)$ time; otherwise, each iteration costs a constant time. Hence, the total running time is bounded by $O(n^3w_{\max}\prod_{j=1}^{n}(1+b_j))$. □

4.3 Fully Polynomial Time Approximation Scheme

In this subsection, we present an $FPTAS$ for $1|rej,p-batch,p_j=b_jt,r_j=t_0,b=\infty,\sum_{J_j\in R}e_j\le Q|\sum_{J_j\in A}w_jC_j$ by trimming the state space, which means that for any state (j,k,W), we need to stretch w_j such that W is in the form of $t_0(1+\epsilon')^l$ for $l\ge -1$, and for any $\epsilon>0$, $\epsilon'=\frac{\epsilon}{2n}$. For example, for a schedule $\sigma=(J_{i_1},J_{i_2},\cdots,J_{i_k})$, if $\sum_{h=1}^{j-1}w'_{i_h}C_{i_h}+w_{i_j}C_{i_j}\in(t_0(1+\epsilon')^{a-1},t_0(1+\epsilon')^a]$, then we stretch w_{i_j} such that $\sum_{h=1}^{j-1}w'_{i_h}C_{i_h}+w'_{i_j}C_{i_j}=t_0(1+\epsilon')^a\le(1+\epsilon')(\sum_{h=1}^{j-1}w'_{i_h}C_{i_h}+w_{i_j}C_{i_j})$ for $j=1,2,\cdots,k$. We call such schedules ϵ'-aligned schedules.

Similarly with the method in Section 3.3, we can present a dynamic programming algorithm for $1|rej,p-batch,p_j=b_jt,r_j=t_0,b=\infty,\sum_{J_j\in R}e_j\le Q|\sum_{J_j\in A}w_jC_j$ in any ϵ'-aligned schedules by algorithm $DP2$. For convenience, we sort the jobs such that $b_1\le b_2\le\cdots\le b_n$, and define

$$\phi_i=\begin{cases}(1+\epsilon')^i, & if\ i\ne -1;\\ 0, & if\ i=-1.\end{cases}\tag{4}$$

Since the objective value is form of $t_0(1+\epsilon')^l$ in any ϵ'-aligned schedules, we can store the exponent l instead of the actual value W in the state of algorithm $DP2$. It's easily know that the maximum makespan C_{MAX} can be obtained if all the jobs are accepted and scheduled

consecutively each job as a batch, i.e., $C_{MAX} = t_0 \prod_{j=1}^{n}(1+b_j)$. Then, the objective function value is $\sum_{j=1}^{n} w'_j C_j \leq \sum_{j=1}^{n} w'_j C_{MAX} \leq (1+\epsilon')^n \sum_{j=1}^{n} w_j C_{MAX} \leq (1+\epsilon')^n n w_{max} C_{MAX}$. Thus, it is easy to see that l range from -1 to L, where L is the smallest integer such that $(1+\epsilon')^L \geq n w_{\max}(1+\epsilon')^n \prod_{j=1}^{n}(1+b_j)$. Hence, $L = O(\frac{n \log(n w_{\max} \prod_{j=1}^{n}(1+b_j))}{\epsilon})$.

According to the above analysis and algorithm $DP2$, we can get the following conclusion

Theorem 4.3. *The algorithm above yields an $FPTAS$ for $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q | \sum_{J_j \in A} w_j C_j$, which runs in $O(\frac{n^3}{\epsilon} \log(n w_{\max} \prod_{j=1}^{n}(1+b_j))$.*

Proof. Assume that $\sigma = (B_1, B_2, \cdots, B_h)$ is a batch processing order of the accepted jobs for $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q | \sum_{J_j \in A} w_j C_j$. Let W'_j be the total weighted completion time of the accepted jobs among $J_j, J_{j+1}, \cdots, J_n$ for $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q | \sum_{J_j \in A} w_j C_j$ in schedule σ when we restrict our attention to ϵ'-aligned schedule only and similarly for $1|rej, p-batch, p_j = b_j t, r_j = t_0, b = \infty, \sum_{J_j \in R} e_j \leq Q | \sum_{J_j \in A} w_j C_j$ with the original value in schedule σ we use W_j.

We can prove that $W'_j \leq (1+\epsilon')^{n-j+1} W_j$ in an ϵ'-aligned schedule by induction on j when we considered the jobs among $J_j, J_{j+1}, \cdots, J_n$. When $j = n$, if J_n is rejected, the conclusion is obviously true; if J_n is accepted, we can easily get that $W'_n = w'_n C_n \leq (1+\epsilon') w_n C_n = (1+\epsilon') W_n$. Let us now assume that the conclusion holds for j; that is, the objective function value $W'_j \leq (1+\epsilon')^{n-j+1} W_j$ in an ϵ'-aligned schedule. We need to show the conclusion also holds for $j-1$; that is, the objective function value $W'_{j-1} \leq (1+\epsilon')^{n-j+2} W_{j-1}$ in an ϵ'-aligned schedule. If job J_{j-1} is rejected, the conclusion is obviously true; if job J_{j-1} is accepted, there exists a positive integer l such that $w_{j-1} C_{j-1} + W'_j \leq w_{j-1} C_{j-1} + (1+\epsilon')^{n-j+1} W_j \in (t_0 \phi_{l-1}, t_0 \phi_l]$. Then we can get that the objective function value is

$$\begin{aligned} W'_{j-1} = t_0 \phi_l &\leq (1+\epsilon')(w_{j-1} C_{j-1} + (1+\epsilon')^{n-j+1} W_j) \\ &\leq (1+\epsilon')^{n-j+2}(w_{j-1} C_{j-1} + W_j) \\ &= (1+\epsilon')^{n-j+2} W_{j-1} \end{aligned}$$

in an ϵ'-aligned schedule. Hence, by induction hypothesis, we can get that $W'_1 \leq (1+\epsilon')^n W_1 \leq (1+\epsilon) W_1$.

Combining the above analysis and Theorem 4.2, the total running time of the algorithm is bounded by $O(n^2 L) = O(\frac{n^3}{\epsilon} \log(n w_{\max} \prod_{j=1}^{n}(1+b_j))$. □

§5 Conclusions

In this paper, we study the unbounded parallel-batch scheduling problems with deteriorating jobs and rejection on a single machine. The objectives are to minimize the makespan and the total weighted completion time, respectively, subject to an upper bound on the total rejection penalty. In these problems jobs are processed in batches and the actual processing time of a job is a linear function of its starting time. We show that both problems are NP-complete and present dynamic programming algorithms and $FPTASs$ for the considered problems. Some similar algorithms can be obtained for the problems to minimize the total rejection penalty

subject to an upper bound on the makespan or the total weighted completion time. For the problem to minimize the maximum tardiness subject to an upper bound on the total rejection penalty, we can also prove it is NP-complete by the Product Partition problem. Due to the space constraints, we don't give the algorithms and the proof of NP-complete for these problems in this paper.

For future research, it is interesting to focus on scheduling problems with jobs of more general deteriorating types. The problems with the bounded model is another worthy topic for future research. Furthermore, it would be interesting to extend our models to different machine environments, e.g., parallel machines or the flow shop settings.

References

[1] Y Bartal, S Leonardi, A Spaccamela, J Sgall, L Stougie. *Multi-processor scheduling with rejection*, Siam Journal on Discrete Mathematics, 2000, 13(1): 64-78.

[2] P Brucker, A Gladky, H Hoogeveen, M Y Kovalyov, C N Potts, T Tautenhahn, S van de Velde. *Scheduling a batching machine*, Journal of Scheduling, 1998, 1(1): 31-54.

[3] T C E Cheng, Q Ding, B M T Lin. *A concise survey of scheduling with time-dependent processing times*, European Journal of Operational Research, 2004, 152(1): 1-13.

[4] Y S Cheng, S J Sun. *Scheduling linear deteriorating jobs with rejection on a single machine*, European Journal of Operational Research, 2009, 194(1): 18-27.

[5] C He, Y T Leung, K Lee, M L Pinedo. *Scheduling a single machine with parallel batching to minimize makespan and total rejection cost*, Discrete Applied Mathematics, 2016, 204(C): 150-163.

[6] Y Gao, J J Yuan. *Unbounded parallel-batch scheduling under agreeable release and processing to minimize total weighted number of tardy jobs*, Journal of Combinatorial Optimization, 2019, 38(3): 698-711.

[7] M Kong, X B Liu, J Pei, Z P Zhou, P M Pardalos. *Parallel-batching scheduling of deteriorating jobs with non-identical sizes and rejection on a single machine*, Optimization Letters, 2019, 1-15.

[8] C Y Lee, R Uzsoy, L A Martin-Vega. *Efficient algorithms for scheduling semi-conductor burn-in operations*, Operations Research, 1992, 40(4): 764-775.

[9] S S Li, C T Ng, T C E Cheng, J J Yuan. *Parallel-batch scheduling of deteriorating jobs with release dates to minimize the makespan*, European Journal of Operational Research, 2011, 210(3): 482-488.

[10] L L Liu, C T Ng, T C E Cheng. *Scheduling jobs with release dates on parallel batch processing machines to minimize the makespan*, Optimization Letters, 2014, 8(1): 307-318.

[11] L F Lu, T C E Cheng, J J Yuan, L Q Zhang. *Bounded single-machine parallel-batch scheduling with release dates and rejection*, Computers and Operations Research, 2009, 36(10): 2748-2751.

[12] L F Lu, C T Ng, L Q Zhang. *Optimal algorithms for single-machine scheduling with rejection to minimize the makespan*, International Journal of Production Economics, 2011, 130(2): 153-158.

[13] C T Ng, M S Barketau, T C E Cheng, M Y Kovalyov. *Product Partition and related problems of scheduling and systems reliability: Computational complexity and approximation*, European Journal of Operational Research, 2010, 207(2): 601-604.

[14] J Pei, X B Liu, P M Pardalos, W J Fan, S L Yang. *Scheduling deteriorating jobs on a single serial-batching machine with multiple job types and sequence-dependent setup times*, Annals of Operations Research, 2017, 249(1-2): 175-195.

[15] X L Qi, S G Zhou, J J Yuan. *Single machine parallel-batch scheduling with deteriorating jobs*, Theoretical Computer Science, 2009, 410(8): 830-836.

[16] S Sengupta. *Algorithms and approximation schemes for minimum lateness/tardiness scheduling with rejection*, Lecture Notes in Computer Science, 2003, 2748: 79-90.

[17] D Shabtay, N Gaspar, M Kaspi. *A survey on off-line scheduling with rejection*, Journal of Scheduling, 2013, 16(1): 3-28.

[18] D Shabtay, D Oran. *Proportionate flow-shop scheduling with rejection*, Journal of the Operational Research Society, 2016, 67(5): 752-769.

[19] S Thevenin, N Zufferey, M Widmer. *Metaheuristics for a scheduling problem with rejection and tardiness penalties*, Journal of Scheduling, 2015, 18(1): 89-105.

[20] G J Woeginger. *When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)*, Informs Journal on Computing, 2000, 12(1): 57-74.

[21] Y Q Yin, Y Wang, T C E Cheng, W Q Liu, J H Li. *Parallel-machine scheduling of deteriorating jobs with potential machine disruptions*, Omega, 2017, 69: 17-28.

[22] L Q Zhang, L F Lu, J J Yuan. *Single machine scheduling with release dates and rejection*, European Journal of Operational Research, 2009, 198(2): 975-978.

[23] L Q Zhang, L F Lu, C T Ng. *The unbounded parallel batch scheduling with rejection*, Journal of the Operational Research Society, 2012, 63(3): 293-298.

[24] X Z Zhang, D C Xu, D L Du, C C Wu. *Approximation algorithms for precedence constrained identical machine scheduling with rejection*, Journal of Combinatorial Optimization, 2018, 35(1): 318-330.

[25] J Zou, C X Miao. *Parallel batch scheduling of deteriorating jobs with release dates and rejection*, Scientific World Journal, 2014, 2014: 1-7.

School of Science, East China University of Science and Technology, Shanghai 200237, China.
Email: xwlu@ecust.edu.cn