Two-Agent Single Machine Scheduling with Deteriorating Jobs and Rejection

Dawei Li, Ganggang Li, and Fangzheng Cheng

1School of Statistics and Mathematics, Shanghai Lixin University of Accounting and Finance, Shanghai 201620, China
2School of Information Management, Jiangxi University of Finance and Economics, Nanchang 330013, China
3School of Economics and Management, Tongji University, Shanghai 200092, China

Correspondence should be addressed to Dawei Li; ldwfengyun@163.com

Received 12 February 2022; Revised 26 August 2022; Accepted 11 November 2022; Published 28 November 2022

1. Introduction

In the real life, several departments may share an operating room in a hospital, some operations from these departments will be done in this operating room. For a patient, the later perform an operation for him, the harder to cure his disease, which can modeled as job deterioration. Besides, restricted by medical condition and medical level, a patient in serious condition may be transferred to more advanced hospitals for more systematic and professional treatment, which can be modeled as job rejection. If we regard the operating room as machine and each department as an agent. The goal is to minimize the objective function of one agent, subject to an upper bound on the objective functions of the other agents. Motivated by this background, we study the multiagent scheduling with deteriorating jobs and rejection.

Scheduling with deteriorating jobs was first introduced by Gupta and Gupta [1] and Browne and Yechiali [2]. Since then, scheduling with deteriorating jobs has received extensive attention from researchers. The book [3] systematically studied the time-dependent scheduling from different perspectives. Liu et al. [4] considered problem Qm|pj = aj + bjt|Cmax and gave a fully polynomial-time approximation scheme (FPTAS). Wang et al. [5] gave a branch-and-price algorithm for the single machine scheduling problem with deteriorating jobs and flexible periodic maintenance to minimize the make span. Gawiejnowicz [6] gave a comprehensive review of the research on time-dependent scheduling in the past decades. For more papers on scheduling with deteriorating jobs, the reader can refer to [7–10] and so on.

In many practical cases, to obtain the maximum profits under the limited resources and increasing market competition, the manufacturer has to reject or outsource some jobs. The notion of scheduling with rejection was first introduced by Bartal et al. [11], they studied the problem Pm|rej|Cmax + \sum_{j \in R} e_j. They proposed an FPTAS to fixed m and (2 − 1/m)-approximation algorithm for arbitrary m. Since then, researchers have done a lot of research on the scheduling with rejection. Shabtay et al. [12] provided a comprehensive survey for most existing offline scheduling problems with rejection. Recently published papers from the area include [13–16].
In recent decades, multiagent scheduling has been widely concerned by scholars. Baker and Smith [17] and Agnetis et al. [18] are the pioneers of multiagent scheduling problem. Perez-Gonzalez and Framinan [19] and Agnetis et al. [20] provided reviews of this area, respectively. Other research in this field includes [21–24]. Although researchers conducted much research in the aspect of multiagent scheduling, few researchers consider multiagent scheduling and job rejection simultaneously, even with deteriorating jobs. Feng et al. [25] studied the two-agent single machine scheduling with rejection. They gave a 2-approximation algorithm and FPTASs. Li and Lu [26] extended the problem in [25] to the case of parallel machines. They presented dynamic programming algorithms and FPTASs. Oron [27] studied two single machine scheduling problems with two competing agents and rejection. Since multiagent scheduling, deterioration, and job rejection all can reflect many real-life situations, it is reasonable and meaningful to consider these cases simultaneously.

The remainder of this paper is organized as follows: in Section 2, we give the specific definition of our problem and the required notations. In Section 3, we consider the problem 1|\(rej, p_j^X = b_j^X (a + bt), C_{max} \sum_j e_j^X \leq Q\)|. \(J^A\), \(J^B\), and \(J^C\) are given a dynamic programming algorithm and an FPTAS. In Section 4, we present a dynamic programming algorithm and an FPTAS for problem 1|\(rej, p_j^X = b_j^X (a + bt), \sum_{t_j^A} C_j^A + \sum_{t_j^B} e_j^A \leq Q\)|. \(C_j^A\) and \(C_j^B\) are respectively. Finally, we conclude the paper and put forward some future research issues.

## 2. Problem Formulation and Notation

The problem can be formally described as follows: there are two competing agents \(A\) and \(B\) to be processed on a single machine, each of them has a set of nonpreemptive jobs \(J^A = \{J_{i_1}^A, J_{i_2}^A, \ldots, J_{i_n}^A\}\) and \(J^B = \{J_{j_1}^B, J_{j_2}^B, \ldots, J_{j_m}^B\}\). The jobs belonging to \(J^A\) and \(J^B\) are referred to as A-jobs and B-jobs, respectively. All jobs are available at time \(t_0\), where \(t_0 > 0\). The actual processing time \(p_j^X\) of job \(J_j^X\) is a linear increasing function of its starting time, given by \(p_j^X = b_j^X (a + bt)\). \(X \in \{A, B\}\), where \(b_j^X\) is the normal processing time of job \(J_j^X\), \(a \geq 0, b \geq 0\), and \(t\) denotes the starting time of \(J_j^X\). A job \(J_j^X\) is either rejected with a rejection penalty \(e_j^X\) has to be paid, or accepted and processed on the machine. For convenience, we define \(TP = t_0 \sum_{t_{i_j}^A} (1 + b_{i_j}^A) \sum_{t_{j_k}^B} (1 + b_{j_k}^B)\) and \(TD = (t_0 + a/b) \sum_{t_{i_j}^A} (1 + b_{i_j}^A) \sum_{t_{j_k}^B} (1 + b_{j_k}^B) - a/b\).

Under schedule \(\sigma\), \(C_j^X(\sigma)\) is the completion time of job \(J_j^X\) in schedule \(\sigma\). When there is no ambiguity, we abbreviate \(C_j^X(\sigma)\) to \(C_j^X\). Let \(J^X\) and \(J^e\) be the set of accepted X-jobs and rejected X-jobs, respectively. The objective is to minimize the sum of completion time of the accepted A-jobs and the total rejection penalty of the rejected A-jobs subject to an upper bound on the sum of the given objective function \(f_j^B\) of the accepted B-jobs and the total rejection penalty of the rejected B-jobs, where \(f_j^B \in [C_{max}^B + \sum_{t_j^B} e_j^B\leq Q]\). According to the three-field notation of Agnetis et al. [18], the problems can be defined as 1|\(rej, p_j^X = b_j^X (a + bt), C_{max} + \sum_{t_j^A} e_j^A \leq Q\)|.

### 3. Problem 1|\(rej, p_j^X = b_j^X (a + bt), C_{max} + \sum_{t_j^A} e_j^A \leq Q\)|

#### 3.1. Dynamic Programming Algorithm

In this section, we will give a dynamic programming algorithm for 1|\(rej, p_j^X = b_j^X (a + bt), C_{max} + \sum_{t_j^A} e_j^A \leq Q\)|. \(C_j^A\) and \(e_j^A\) are respectively. Feng et al. [25] proved 1|\(rej, C_{max} \leq Q\)|. 

**Lemma 1.** For 1|\(p_j = b_j (a + bt)C_{max}\), the maximum completion time of the jobs is \(C_{max}(\mathcal{J}) = (t_0 + a/b) \sum_{t_j^A} (1 + b_j^1)\) and is independent of the order of the job [3].

**Lemma 2.** Problem 1|\(p_j = b_j (a + bt)C_{max}\), can be solved by scheduling jobs in nondecreasing order of \(b_j^1 + b_j^2\) ratios [3].

According to Lemma 1 and 2, we can get the following lemma.

**Lemma 3.** For 1|\(rej, p_j^X = b_j^X (a + bt), C_{max} + \sum_{t_j^A} e_j^A \leq Q\)|, \(C_j^A\) and \(e_j^A\) are respectively. There exists an optimal schedule such that the accepted A-jobs are processed in non-decreasing order of \(b_j^1 + b_j^2\) and the accepted B-jobs are processed consecutively in arbitrary order.

Based on Lemma 3, we renumber the jobs such that \(b_j^1 \leq b_j^2 \leq \cdots \leq b_{j_n}^2\) and \(b_j^1 \leq b_j^2 \leq \cdots \leq b_{j_n}^1\), and let \(f_{i_j}(t, t_B, E_B)\) denote the optimal objective function value satisfying the following conditions: (1) the jobs in consideration are \(J_{i_1}^A, J_{i_2}^A, \ldots, J_{i_n}^A\) and \(J_{j_1}^B, J_{j_2}^B, \ldots, J_{j_m}^B\); (2) the completion time of the last job is \(t\); (3) the completion time of the last B-job is no more than \(t_0\); (4) the total rejection penalty of the rejected B-jobs among \(J_{j_1}^B, J_{j_2}^B, \ldots, J_{j_m}^B\) is \(E_B\).

The initial conditions are as follows:

\[
f_{0,0}(t, t_B, E_B) = \begin{cases} 
0, & \text{if } t = t_B = t_0, E_B = 0; \\
+\infty, & \text{otherwise.} 
\end{cases}
\]
Now we consider any optimal schedule for jobs $J^A_1$, $J^A_2, \ldots, J^A_n$, and $J^B_1, J^B_2, \ldots, J^B_n$, in which meets the above conditions. In any such schedule, either $J^A_i$ is rejected, or $J^B_i$ is rejected, or the last scheduled job is one of $J^A_i$ and $J^B_i$.

Case 1. Job $J^A_i$ is rejected. We have $f_{i,j}(t, t_B, E_B) = f_{i-1,j}(t, t_B, E_B) + e^A_i$.

Case 2. The last scheduled job is $J^A_i$. In this case, for the accepted job among $J^A_1, J^A_2, \ldots, J^A_{i-1}$ and $J^A_i, J^A_{i+1}, \ldots, J^A_n$, the completion time of the last job must be $t - ab^A_i/1 + bb^A_i$, the completion time of the last B-job is more than $t_B$, and the total rejection penalty of the rejected B-jobs is $E_B$. Then, we have $f_{i,j}(t, t_B, E_B) = f_{i-1,j}(t - ab^A_i/1 + bb^A_i, t_B, E_B) + t$.

Case 3. Job $J^B_i$ is rejected. At this point, in the corresponding optimal schedule for $J^A_1, J^A_2, \ldots, J^A_i$ and $J^B_1, J^B_2, \ldots, J^B_{i-1}$, the completion time of the last job is $t$

\[
\begin{align*}
 f_{i,j}(t, t_B, E_B) &= \min \left\{ f_{i-1,j}(t, t_B, E_B) + e^A_i, \\
 & \quad \quad f_{i-1,j}\left(\frac{t - ab^A_i}{1 + bb^A_i}, t_B, E_B\right) + t, \\
 & \quad \quad f_{i-1,j}(t, t_B, E_B - e^B_i), \quad \text{if } t_B + E_B \leq Q; \\
 & \quad \quad f_{i-1,j}\left(\frac{t - ab^B_i - ab^B_i}{1 + bb^B_i}, t_B, E_B\right), \quad \text{if } t_B + E_B > Q. \tag{2}
\end{align*}
\]

The optimal value is given by $\min\{f_{n,n}(t, t_B, E_B); t_0 \leq t \leq Q, t_B + E_B \leq Q\}$, and the corresponding optimal schedule can be found by backtracking.

**Theorem 1.** The problem \(1|\text{rej}, p^X_j = b^X_j(a + bt), C_{\text{max}} + \sum_{j \in \mathcal{A}} e^B_j \leq Q| \sum_{j \in \mathcal{A}} C^A_j + \sum_{j \in \mathcal{B}} e^A_j \) can be solved in $O(n_1n_BQ^2FP)$ time.

**Proof.** The recursion function can have up to $O(n_1n_BQ^2FP)$ states, and each iteration costs a constant time. Hence, the total running time is bounded by $O(n_1n_BQ^2FP)$.

\[3.2. \text{Fully Polynomial-Time Approximation Scheme.}\] In this subsection, we can give an FPTAS for problem \(1|\text{rej}, p^X_j = b^X_jt, C_{\text{max}} + \sum_{j \in \mathcal{A}} e^B_j \leq Q| \sum_{j \in \mathcal{A}} C^A_j + \sum_{j \in \mathcal{B}} e^A_j \) such that for any $\epsilon > 0$, the algorithm can find a solution $\sigma$ in polynomial time in the input size and $1/\epsilon$ which satisfies the following:

\[
\begin{align*}
 (1) & \sum_{j \in \mathcal{A}} C^A_j(\sigma) + \sum_{j \in \mathcal{B}} e^A_j(\sigma) \leq (1 + \epsilon)(\sum_{j \in \mathcal{A}} C^A_j(\pi) + \sum_{j \in \mathcal{B}} e^A_j(\pi)), \\
 (2) & C_{\text{max}}^*(\sigma) + \sum_{j \in \mathcal{B}} e^B_j(\sigma) \leq (1 + \epsilon)Q.
\end{align*}
\]

Here $\pi$ is an optimal solution.

Now, we will present an FPTAS by considering the modified deteriorating rates and the inflated rejection penalty with $a = 0$ and $b = 1$. The definition of the modified deteriorating rates involves a geometric rounding technique developed by Sengupta [28]. And the rounding technique is stated as follows:

For any $\epsilon > 0$ and $x \geq 1$, if $(1 + \epsilon)^{x-1} < x < (1 + \epsilon)^x$, then we define $[x]_\epsilon = (1 + \epsilon)^{x-1}$ and $\lceil x \rceil_\epsilon = (1 + \epsilon)^x - 1$. If $x$ is an exact power of $(1 + \epsilon)$, then $\lceil x \rceil_\epsilon = \lceil x \rceil_\epsilon = x$. Note that for any $x \geq 1$, we have $x \geq 1/(1 + \epsilon)$x.

For any $\epsilon > 0$, let $\epsilon' = \epsilon/2n$, the modified deteriorating rate is defined as $b^X_j = [1 + b^X_j]_\epsilon \epsilon'/\epsilon$. Let $L^X_j$ be the exponent of $1 + b^X_j$, i.e., $1 + b^X_j = (1 + \epsilon')^{L^X_j}$. Then $L^X_j = \log(1 + b^X_j) / \log(1 + \epsilon') = O(n\log(1 + b^X_j)/\epsilon)$.
For any $\epsilon > 0$, let $e'' = \epsilon / 2n_0$ and $\mathcal{R}_B = \left\{ j^B_{i_1}, j^B_{i_2}, \ldots, j^B_{i_k} \right\}$ be the set of the rejected $B$-jobs, where $i_1 < i_2 < \cdots < i_k$. The inflated rejection penalty of $\mathcal{R}_B$ is defined as follows:

$$f_\epsilon' (\mathcal{R}_B) = \left\{ \begin{array}{ll} e^B_j + f_\epsilon' (\mathcal{R}_B - \left\{ j^B_{i_1} \right\}) \epsilon'' & , \quad q \geq 1; \\
0, & \mathcal{R}_B = \emptyset \end{array} \right.$$  

(3)

**Lemma 4.** For any $0 < \epsilon \leq 1$ and any integer $n \geq 1$, $(1 + x/n)^n \leq 1 + 2x$ holds [29].

**Lemma 5.** For any $\mathcal{R}_B \subseteq \mathcal{J}^B$ and $e'' > 0$, $f_\epsilon' (\mathcal{R}_B) \leq (1 + e'') | \mathcal{R}_B | \sum\limits_{j^B \in \mathcal{R}_B} e^B_j$ [28].

\[
\sum\limits_{j^B \in \mathcal{R}_B} C_j^A + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B = \sum\limits_{i = 1}^{\mathcal{J}^A_A} t_0 \prod\limits_{j = 1}^{i} (1 + b_j^A) \prod\limits_{j^B \in \mathcal{R}_B} (1 + b_j^B) + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B \\
\leq \sum\limits_{i = 1}^{\mathcal{J}^A_A} t_0 (1 + e')^{i+|\mathcal{J}^A_A|} \prod\limits_{j = 1}^{i} (1 + b_j^A) \prod\limits_{j^B \in \mathcal{R}_B} (1 + b_j^B) + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B \\
\leq (1 + e')^n \sum\limits_{i = 1}^{\mathcal{J}^A_A} t_0 \prod\limits_{j = 1}^{i} (1 + b_j^A) \prod\limits_{j^B \in \mathcal{R}_B} (1 + b_j^B) + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B \\
\leq (1 + e')(\sum\limits_{j^B \in \mathcal{R}_B} C_j^A + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B),
\]

(4)

We then propose an exact dynamic programming algorithm for problem 1|$\mathcal{J}^A_1, j^A_2, \ldots, j^A_n$, and $\mathcal{J}^B_1, j^B_2, \ldots, j^B_n$; (2) the completion time of the last job is at least $t_0$; (3) the completion time of the last job is no more than $t_0$; and (4) the total rejection penalty of the rejected $B$-jobs is $\gamma_B$.

**Lemma 6.** For any $0 < \epsilon \leq 2$, the optimal objective function value for $1|p^A_j = b^A_j t, C_{\text{max}} = \sum\limits_{j^B \in \mathcal{R}_B} e_j^B \leq (1 + \epsilon)Q| \sum\limits_{j^B \in \mathcal{R}_B} C_j^A + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B$ with the modified deteriorating rates $b''_j$ is at most of $(1 + \epsilon)$ times the optimal objective function value for $1|p^A_j = b^A_j t, C_{\text{max}} = \sum\limits_{j^B \in \mathcal{R}_B} e_j^B \leq Q| \sum\limits_{j^B \in \mathcal{R}_B} C_j^A + \sum\limits_{j^B \in \mathcal{R}_B} e_j^B$.

**Proof.** Assume that jobs in set $\mathcal{J}^A_A$ are scheduled in the order of $[1], [2], \ldots, [|\mathcal{J}^A_A|]$, let $\mathcal{J}^B_B$ be the set of accepted $B$-jobs which precede $j^A_1$ and $\mathcal{J}^B_B$ be the set of accepted $A$-jobs before the last $B$-job. The objective function with the modified deteriorating rates $b''_j$ is as follows:

$$
\Phi_A = \sum\limits_{j^A \in \mathcal{J}^A_A} C_j^A + \sum\limits_{j^B \in \mathcal{J}^B_B} e_j^B + \sum\limits_{j^B \in \mathcal{J}^B_B} b''_j t_j
$$

We consider the optimal schedule for jobs $\mathcal{J}^A_A, \mathcal{J}^A_2, \ldots, \mathcal{J}^A_n$, and $\mathcal{J}^B_B, \mathcal{J}^B_2, \ldots, \mathcal{J}^B_n$ which meets the above conditions. In any such schedule, either $j^A_1$ is rejected, $j^B_1$ is rejected, or the last scheduled job is one of $j^A_1$ and $j^B_1$.

**Case 1.** Job $j^A_1$ is rejected. Then, we have $f_{i,j} (l, l_B, h) = f_{i-1,j} (l, l_B, h) + e_i^A$.

$$f_{0,0} (l, l_B, h) = \begin{cases} 0, & \text{if } l = l_B = h = -1; \\
+\infty, & \text{otherwise} \end{cases}
$$

(6)
where \( \log \epsilon / (\cdot) \) each iteration cost a constant time. Hence, the total running each iteration costs \( T_{\epsilon} \).

**Lemma 7.**

Theorem 2.

The recursive function has at most \( Bn_{\epsilon} \) elements. Let \( \text{list} = \{ \cdot \} \) be the optimal value is given by \( \min \{ f_{i,j-1}(l, i_{B}, h) \} \). The optimal objective function for each job is \( \min \{ f_{i,j-1}(l, i_{B}, h) \} \), if \( t_{\epsilon}r_{i} + \epsilon h \leq (1 + \epsilon)Q \); otherwise, it is \( \min \{ f_{i,j-1}(l, i_{B}, h) \} \).

**Theorem 2.**

**Problem 1** \( 1|\text{rej}, p_{j}^{X} = b_{j}^{X} | \sum_{j \in \mathcal{X}} C_{j}^{X} + f_{\epsilon} (\mathcal{R}_{B}) \leq (1 + \epsilon)Q | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \text{ can be solved in } O(n_{\epsilon}^{3/2}e^{\log^{3}(Q \log Q)TP}) \text{ time.} \)

**Proof.**

The recursive function has at most \( O(n_{\epsilon}^{3/2}e^{\log^{3}(Q \log Q)TP}) \) states, if the \( B \)-jobs are rejected, each iteration costs \( O(H) \) time; for the other three cases, each iteration cost a constant time. Hence, the total running time is bounded by \( O(n_{\epsilon}^{3/2}e^{\log^{3}(Q \log Q)TP}) \).

**4. Problem 1** \( 1|\text{rej}, p_{j}^{X} = b_{j}^{X} | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \leq Q | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \text{ is at least } \text{NP-hard.} \)

**4.1 Dynamic Programming Algorithm.**

Agnetsis et al. [18] showed that problem 1 \( 1|\text{Cmax} | \sum_{j \in \mathcal{X}} C_{j}^{X} \) a special case of our problem when \( a = 1, b = 0 \) and the rejection penalty is large enough, i.e., it is optimal to accept all the jobs, is \( \text{NP-hard.} \)

Hence, our problem 1 \( 1|\text{rej}, p_{j}^{X} = b_{j}^{X} | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \leq Q | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \text{ is at least } \text{NP-hard.} \)

**Lemma 7.** There exists an optimal schedule for 1 \( 1|\text{rej}, p_{j}^{X} = b_{j}^{X} | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \leq Q | \sum_{j \in \mathcal{X}} C_{j}^{X} + \sum_{j \in \mathcal{X}} e_{j}^{X} \) such that the accepted \( A \)-jobs and accepted \( B \)-jobs are processed in nondecreasing order of \( b_{j}^{A}/1 + b_{j}^{B}/t \), \( X \in \{ A, B \} \).

Based on Lemma 7, we renumber the jobs such that \( b_{j}^{A}/1 + b_{j}^{B}/1 + b_{j}^{B}/b_{j}^{B} \leq \cdots \leq b_{j}^{A}/1 + b_{j}^{B} \text{ and } b_{j}^{A}/1 + b_{j}^{B} \leq b_{j}^{A}/1 + b_{j}^{B} \leq \cdots \leq b_{j}^{A}/1 + b_{j}^{B} \).

Let \( f(i, j, t, C_{B}, E_{B}) \) be the optimal objective function when the jobs in consideration are \( f_{i}^{A}, f_{j}^{A}, \ldots, f_{j}^{A} \) and \( f_{j}^{B}, f_{j}^{B}, \ldots, f_{j}^{B} \), where \( t \) is the completion time of the last job, \( C_{B} \) is the total completion time of the accepted \( A \)-jobs and \( E_{B} \) is the total rejection penalty of the rejected \( B \)-jobs. Let \( \Gamma_{ij} \) include all the possible quadruplets solutions \( (f(i, j, t, C_{B}, E_{B}), t, C_{B}, E_{B}) \). Then, we can give the following forward dynamic programming algorithm. First, we initialize \( \Gamma_{00} = \{ (0, 0, 0, 0, 0, 0, 0, 0) \} \). For each quadruplet \( (f(i, j, t, C_{B}, E_{B}), t, C_{B}, E_{B}) \in \Gamma_{ij} \), we construct either two quadruplets in \( \Gamma_{(i+1)j} \) by adding job \( f_{j+1}^{A} \) to this state or two quadruplets in \( \Gamma_{ij+(j+1)} \) by adding job \( f_{j+1}^{B} \) to this state. More specifically, for any quadruplet \( (f(i, j, t, C_{B}, E_{B}), t, C_{B}, E_{B}) \in \Gamma_{ij} \) we do the following:

Case 1: Job \( f_{j+1}^{A} \) is rejected. At this point, we have \( (g(i + 1, j + 1, t, C_{B}, E_{B}), t, C_{B}, E_{B}) \) in \( \Gamma_{(i+1)j} \), where \( g(i + 1, j + 1, t, C_{B}, E_{B}) = f(i, j + 1, t, C_{B}, E_{B}) + e_{j+1}^{A} \).

Case 2: The last scheduled job is \( f_{j+1}^{A} \). At this point, we have \( (g(i + 1, j + 1, t', C_{B}, E_{B}), t, C_{B}, E_{B}) \) in \( \Gamma_{(i+1)j} \), where \( g(i + 1, j + 1, t', C_{B}, E_{B}) = f(i, j + 1, t, C_{B}, E_{B}) + t' \) and \( t' = (t + a/b)(1 + b_{j+1} - a/b) \), where \( t' \) is the completion time of \( f_{j+1}^{A} \).

Case 3: Job \( f_{j+1}^{B} \) is rejected. If \( C_{B} + E_{B} + t_{j+1} \leq Q \), we include the quadruplet \( (g(i, j + 1, t, C_{B}, E_{B} + e_{j+1}^{B}), t, C_{B}, E_{B} + e_{j+1}^{B}) \) in \( \Gamma_{ij+(j+1)} \), where \( g(i, j + 1, t, C_{B}, E_{B} + e_{j+1}^{B}) = f(i, j, t, C_{B}, E_{B}) \).
Proof. We prove the lemma by induction on \(i + j\). Obviously, lemma holds for \(i + j = 1\). Now let’s assume that the lemma holds for \(i + j = 1\), i.e.,

(1) For any eliminated quadruplet \(f(i - 1, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{(i+j)}\), there exists a quadruplet \(f(i - 1, j, t, C_B, E_B) \in \Gamma_{(i+j)}\) such that \(f(i - 1, j, t, C_B, E_B) \leq f(i, j, t, C_B, E_B, t, C_B, E_B)\) and \(E_B, E_B \leq E_B\).

(2) For any eliminated quadruplet \(f(i - 1, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{(i+j)}\), there exists a quadruplet \(f(i - 1, j, t, C_B, E_B) \in \Gamma_{(i+j)}\) such that \(f(i - 1, j, t, C_B, E_B) \leq f(i, j, t, C_B, E_B)\) and \(E_B, E_B \leq E_B\).

We need to show that the lemma holds also for \(i + j\). That is to say, for any eliminated quadruplet \(f(i, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\), there exists a quadruplet \(f(i, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\) such that \(f(i, j, t, C_B, E_B) \leq f(i, j, t, C_B, E_B)\) and \(E_B, E_B \leq E_B\).

Consider an arbitrary quadruplet \(f(i, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\), we first consider Case 1 that \(f^j\) is rejected. While implementing Algorithm 2, the quadruplet \(f(i, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\) is constructed from the quadruplet \(g(i - 1, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\) where \(g(i - 1, j, t, C_B, E_B) = f(i, j, t, C_B, E_B)\) and \(E_B, E_B \leq E_B\). According to the induction hypothesis, there exists a quadruplet \(g(i - 1, j, t, C_B, E_B) \leq g(i, j, t, C_B, E_B)\) (which might also be the quadruplet \(g(i, j, t, C_B, E_B, t, C_B, E_B)\)) with \(g(i - 1, j, t, C_B, E_B) \leq g(i, j, t, C_B, E_B)\). It directly follows that before the elimination in \(i+j\), there exists a quadruplet \(g(i, j, t, C_B, E_B, t, t, C_B, E_B) \in \Gamma_{i+j}\) with the following:

\[
\begin{align*}
\text{a)} & \quad g(i, j, t, C_B, E_B) \leq \delta g(i, j, t, C_B, E_B) = g(i, j, t, C_B, E_B) + \delta e^{i-j} t, & \quad & \text{b)} \quad t' \leq t', \\
\text{c)} & \quad C_B' \leq C_B + \delta t', & \quad & \text{d)} \quad E_B' \leq E_B, \quad E_B' \leq E_B.
\end{align*}
\]

Let us now consider Case 2, where the last scheduled job is \(f^j\), the quadruplet \(f(i, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\) is constructed from the quadruplet \(g(i - 1, j, t, C_B, E_B) \in \Gamma_{i+j}\) where \(g(i - 1, j, t, C_B, E_B) = f(i, j, t, C_B, E_B)\) and \(E_B, E_B \leq E_B\). According to the induction hypothesis, there exists a quadruplet \(g(i - 1, j, t, C_B, E_B) \leq g(i, j, t, C_B, E_B)\) (which might also be the quadruplet \(g(i - 1, j, t, C_B, E_B, t, C_B, E_B)\)) with \(g(i - 1, j, t, C_B, E_B) \leq g(i, j, t, C_B, E_B)\). It directly follows that before the elimination in \(i+j\), there exists a quadruplet \(g(i, j, t, C_B, E_B, t, C_B, E_B) \in \Gamma_{i+j}\) with the following:

\[
\begin{align*}
\text{a)} & \quad g(i, j, t, C_B, E_B) \leq \delta g(i, j, t, C_B, E_B) = g(i, j, t, C_B, E_B) + \delta e^{i-j} t, & \quad & \text{b)} \quad t' \leq t', \\
\text{c)} & \quad C_B' \leq C_B + \delta t', & \quad & \text{d)} \quad E_B' \leq E_B, \quad E_B' \leq E_B.
\end{align*}
\]
Step 1: [Preprocessing]: Renumber the jobs such that \( b_i^1/1 + bb_i^4 \leq b_t^1/1 + bb_t^4 \leq \ldots \leq b_{n_a}^1/1 + bb_{n_a}^4 \) and \( b_i^{b_i^4}/1 + bb_t^{b_t^4} \leq b_t^{b_t^4}/1 + bb_t^{b_t^4} \leq \ldots \leq b_{n_a}^{b_i^4}/1 + bb_{n_a}^{b_i^4} \).

Step 2: [Initialization]: Set \( \Gamma_{00} = \{ f(0,0,t_0,0,0),t_0,0,0 \} = \{ (0,t_0,0,0) \} \).

Step 3: [Generation]:

For \( i = 0 \) to \( n_A \) do

For \( j = 0 \) to \( n_B \) do

Out of each quadruplet \( (f(i,j,t,C_B,E_B),t,C_B,E_B) \in \Gamma_{ij} \) do

Construct the quadruplet \( (g(i+1+1,t,C_B,E_B),t,C_B,E_B) \), where \( g(i+1+1,t,C_B,E_B) = f(i+1,t,C_B,E_B) + e_{j+1}^1 \)

and the quadruplet \( (g(i+1,j+t',C_B,E_B),t',C_B,E_B) \), where \( g(i+1,j+t',C_B,E_B) = f(i+1,j,t,C_B,E_B) + t' \), and \( t' = (t + a/b)(1 + b_i^1) - a/b \).

If \( C_B + E_B + e_{j+1}^1 \geq Q \):

Construct in \( \Gamma_{ij+1} \) the quadruplet \( (g(i,j+1,t,C_B,E_B + e_{j+1}^1),t,C_B,E_B + e_{j+1}^1) \), where \( g(i,j+1,t,C_B,E_B + e_{j+1}^1) = f(i,j,t,C_B,E_B) \); Endif

If \( C_B + (t + a/b)(1 + b_i^1) - a/b + E_B \leq Q \):

Construct in \( \Gamma_{ij+1} \) the quadruplet \( (g(i,j+1,t'+1,C_B,E_B),t',C_B,E_B), \) where \( g(i,j+1,t'+1,C_B,E_B) = f(i,j+1,t,C_B,E_B) + t' = (t+a)(1+b_i^1) - a/b \), \( C_B + t' \) and \( E_B = E_B \); Endif

[Elimination]:

(1) Among the \( (i+1,j,t,C_B,E_B) \) quadruplets \( (g(i+1+1,j,t,C_B,E_B),t,C_B,E_B), (g(i+1,j,t,C_B,E_B),t,C_B,E_B), \ldots, (g(i+1,j,t,t+1,C_B,E_B),t,t,C_B,E_B) \) in \( \Gamma_{ij} \) with the same \( t, C_B, \) and \( E_B \) values, we keep in \( \Gamma_{i+1,j} \) only the quadruplet \( (f(i+1+1,j,t,C_B,E_B),t,C_B,E_B) \), where

\[
f(i+1+1,j,t,C_B,E_B) = \min_{i=1,\ldots,n_A} g(i+1+1,j,t,C_B,E_B);\]

(2) Among the \( (i+1,j+1,t,C_B,E_B) \) quadruplets \( (g(i+1+1,j+1,t,C_B,E_B),t,C_B,E_B), (g(i+1,j+1,t,C_B,E_B),t,C_B,E_B), \ldots, (g(i+1,j+1,t+1,C_B,E_B),t+1,C_B,E_B) \) in \( \Gamma_{ij} \) with the same \( t, C_B, \) and \( E_B \) values, we keep in \( \Gamma_{i+1,j+1} \) only the quadruplet \( (f(i+1+1,j+1,t,C_B,E_B),t,C_B,E_B) \), where

\[
f(i+1+1,j+1,t,C_B,E_B) = \min_{i=1,\ldots,n_A} g(i+1+1,j+1,t,C_B,E_B);\]

Endfor

Endfor

Step 4: [Result]: The optimal solution value is given by \( f(n_A,n_B,t,C_B,E_B) \) with the minimum \( f(n_A,n_B,t,C_B,E_B) \) value among all quadruplets in \( \Gamma_{n_A,n_B} \), the optimal solution can be found by backtracking.

**Algorithm 1**: Dynamic programming algorithm.

---

Step 1: [Preprocessing]: Renumber the jobs such that \( b_i^1/1 + bb_i^4 \leq b_t^1/1 + bb_t^4 \leq \ldots \leq b_{n_a}^1/1 + bb_{n_a}^4 \) and \( b_i^{b_i^4}/1 + bb_t^{b_t^4} \leq b_t^{b_t^4}/1 + bb_t^{b_t^4} \leq \ldots \leq b_{n_a}^{b_i^4}/1 + bb_{n_a}^{b_i^4} \).

Step 2: [Initialization]: Set \( \Gamma_{00} = \{ f(0,0,t_0,0,0),t_0,0,0 \} = \{ (0,t_0,0,0) \} \).

Step 3: [Generation]:

For \( i = 0 \) to \( n_A \) do

For \( j = 0 \) to \( n_B \) do

[State Generation]:

Generate \( \Gamma_{i+1,j} \) or \( \Gamma_{i,j+1} \) from \( \Gamma_{ij} \) the same as that in Algorithm 1 except for replacing \( Q \) by \( \overline{Q} = (1+\varepsilon)Q \).

Labeling:

For each quadruplet \( (f(i,j,t,C_B,E_B),t,C_B,E_B) \in \Gamma_{ij} \), attach the label \( \Delta(f(i,j,t,C_B,E_B)) \), \( \Delta(t), \) and \( \Delta(C_B) \) to it, where the function \( \Delta \) is defined as \( \Delta(x) = k \) in which \( x \) satisfies \( \delta \leq x \leq \delta + 1 \) and \( \delta = 1 + \varepsilon/2(1+\varepsilon)n \); Endfor

[Elimination]:

(1) For any two quadruplets \( (f(i+1,j,t,C_B,E_B),t,C_B,E_B) \) and \( (f(i+1,j,t',C_B,E_B),t',C_B,E_B) \) in \( \Gamma_{i+1,j} \) with the same label and \( E_B \leq E_B \) eliminate the latter quadruplet from \( \Gamma_{i+1,j} \);

(2) For any two quadruplets \( (f(i+1,j,t,C_B,E_B),t,C_B,E_B) \) and \( (f(i+1,j,t',C_B,E_B),t',C_B,E_B) \) in \( \Gamma_{i,j+1} \) with the same label and \( E_B \leq E_B \) eliminate the latter quadruplet from \( \Gamma_{i,j+1} \);

Endfor

Endfor

Step 4: [Result]: The approximate solution value is given by \( f(n_A,n_B,t,C_B,E_B) \) with the minimum \( f(n_A,n_B,t,C_B,E_B) \) value among all the quadruplets in \( \Gamma_{n_A,n_B} \).

**Algorithm 2**: Fully polynomial-time approximation scheme.
According to the induction hypothesis, there exists a quadruplet \((g(i, j, t', C_B, E_B), t', C_B, E_B)\) from the quadruplet \((g(i, j, t', C_B, E_B), t', C_B, E_B)\) implies that there exists a quadruplet 

\[
8 \text{ Mathematical Problems in Engineering}
\]

\[
\text{Algorithm 2 finds in O}(n \log n + \log Q) = O(n \log n) \text{ time a quadruplet } (f(i, j, t, C_B, E_B), t, C_B, E_B) \text{ such that } f(i, j, t, C_B, E_B) \leq (1 + e) t' \text{ and } C_B + E_B \leq (1 + e) Q.
\]

Then, we discuss the time complexity of Algorithm 2. Step 1 takes \(O(n \log n + n \log n)\) time. Step 2 takes \(O(1)\) time. Note that there are at most \(Q\) possible values for \(i\), \(j\), \(t\), \(C_B\) and \(E_B\). Therefore, the number of possible values of \(\Delta(t)\) is \(Q\). Similarly, the number of possible values \(\Delta(C_B)\) is \(Q\). Hence, the total number of different boxes at the beginning of each iteration equals the number of different quadruplets at the beginning of each iteration is at most \(O(n \log n)\). Because after the elimination process, each possible label can retain at most one quadruplet. In addition, for each quadruplet in \(\Gamma_{ij}\), we can construct at most two quadruplets in \(\Gamma_{ij}^{(1)}\) or two quadruplets in \(\Gamma_{ij}^{(1)}\). So, Step 3 takes \(O(n \log n)\) time to construct \(\Gamma_{ij}^{(1)}\) and \(\Gamma_{ij}^{(1)}\). At the same time, this is the time required for the elimination process. In Step 4, the quadruplet \((f(i, j, t, C_B, E_B), t, C_B, E_B)\) with minimum \(f(i, j, t, C_B, E_B)\) in \(\Gamma_{ij}^{(1)}\) can be found in \(O(n \log n)\) time. Because Step 3 requires at most \(O(n \log n)\) iterations, the overall complexity of Algorithm 2 is \(O(n \log n)\).

5. Conclusions

This paper investigated several two-agent scheduling problems with deteriorating jobs and rejection. Two agents share a single machine, and each agent has its own optimization criteria. We proposed a dynamic programming algorithm for \(1|\text{re}j, p_{j}^x = b_j^x (a + bt), C_{\text{max}}^x + \sum_{j \in \mathbb{A}_n} p_{j}^x \)}
\[ e^B_j \leq Q! \sum_{i \in A_R} C_i^j + \sum_{i \in A_R} e_i^j + 1 | re_j, p_j^Y = b_j^Y (a + bt), \sum_{i \in A_R} C_i^j + \sum_{i \in A_R} e_i^j \leq Q! \sum_{i \in A_R} C_i^j + \sum_{i \in A_R} e_i^j, \]

respectively. For the former problem, we give an FPTAS when \( a = 0, b = 1 \). For the latter problem, we also give an FPTAS.

For future research, we can analyze scheduling problems with more than two agents. In addition, other machine environments, such as parallel machines or flow shop settings, are also worth studying.

**Data Availability**

No data were used to support the findings of the study.

**Conflicts of Interest**

The authors declare that they have no conflicts of interest.

**Acknowledgments**

The research was supported by the National Natural Science Foundation of China under Grant nos. 11901255 and 71901163.

**References**


