SINGLE-MACHINE RESCHEDULING PROBLEMS WITH LEARNING EFFECT UNDER DISRUPTIONS

MINGBAO CHENG, SHUXIAN XIAO AND GUOSHENG LIU*

School of Management, Guangdong University of Technology
Guangzhou, Guangdong 510520, China

(Communicated by Frank Werner)

Abstract. Rescheduling in production planning means to schedule the sequenced jobs again together with a set of new arrived jobs so as to generate a new feasible schedule, which creates disruptions to any job between the original and adjusted position. In this paper, we study rescheduling problems with learning effect under disruption constraints to minimize several classical objectives, where learning effect means that the workers gain experience during the process of operation and make the actual processing time of jobs shorter than their normal processing time. The objectives are to find optimal sequences to minimize the makespan and the total completion time under a limit of the disruptions from the original schedule. For the considered objectives under a single disruption constraint or a disruption cost constraint, we propose polynomial-time algorithms and pseudo-polynomial time algorithms, respectively.

1. Introduction. Learning effect in manufacturing process means that many similar products or parts are produced continuously and repetitiously, which makes the manufacturer/worker to perform subsequent operating faster than before. The impact of learning on productivity in manufacturing was first observed by Wright [22], who used a learning curve to model the aircraft industry. Recently, Anzanello and Fogliatto [1] provided a good review of learning curve. The concept of learning effect was first introduced into machine scheduling problems by Biskup in 1999. For several scheduling problems with learning effect, Mosheiov [17] gave several examples to demonstrate that the optimal schedules are very different from the corresponding classical versions of them. Bachman and Janiak [4] considered some single-machine scheduling problems with a job’s processing time depended on the position of this job in a sequence. What’s more, Biskup [6] surveyed most of articles recently published about machine scheduling problems with learning effects. Recently, Wang et al. [21] considered single-machine scheduling problems with controllable processing time, truncated job-dependent learning and deterioration effects to minimize a cost function containing several classical objectives and total resource cost. There are still many interesting topics to be focused, readers can refer to Yin et al. [25], Vakedi-Nouri et al. [20] and Cheng et al. [8], etc.

2010 Mathematics Subject Classification. Primary: 90B35; Secondary: 90C26.

Key words and phrases. Single-machine, rescheduling, learning effect, disruption.

The research is supported partially by the National Natural Science Foundation Committee of China grant 71571050, 71502100, 71401044.

* Corresponding author.
In actual manufacturing environment, unforeseen disruptions often occur, such as the arrivals of new jobs, the change of order priority and the change of release dates, etc. Such uncertainties create a need for rescheduling jobs to meet the emergent request. Rescheduling means to schedule the jobs again together with a set of new jobs so as to minimize objectives. There are many researchers to study rescheduling problems. For example, Hall and Potts [11, 12] considered a rescheduling problem in a single-machine setting with newly arrived jobs to minimize the maximum lateness and the total completion time under a limit of disruptions. Azizoglu and Alagoz [3] considered the rescheduling problems with a period of unavailability on one machine due to disruption. They traded off between two conflicting criteria: total flow time and the number of disrupted jobs, and illustrated that all the efficient schedules with respect to the two criteria can be found in polynomial time. Yang [24] studied a single-machine rescheduling problem with the arrivals of new jobs and compressed processing time. Yuan and Mu [26] developed a single machine rescheduling problem with release dates to minimize makespan under the maximum sequence disruption. While, Zhao and Tang [28] studied two single-machine rescheduling problems with linear deteriorating jobs under a disruption to minimize the total completion time, which can be solved in polynomial time. Liu and Young [16] provided three approximation algorithms to study a single-machine rescheduling problem to minimize some cost objectives while the rescheduled jobs without excessively disrupting the original schedule.

In recent years, there are also many research study rescheduling problems in multiple-machine settings. Ozlen and Azizoglu [18] considered an unrelated parallel machines rescheduling problem with a disruption on one of the machines, they provided polynomial-time algorithms to minimize total disruption cost among the minimum total flow time. Chiu and Shih [9] considered an integrated model that analyzed both preventive maintenance and rush orders in a two-machine flow shop using two different rescheduling methods. Katragjini et al. [14] developed rescheduling algorithms to consider flow shop rescheduling layouts and generated three types of disruption that interrupted the original schedules simultaneously. Liu and Zhou [15] investigated an identical parallel-machine rescheduling problem, and proposed two polynomial time algorithms for lexicographically optimizing two conflicting rescheduling criteria: the total completion time and the disruption cost. Sun et al. [19] investigated a hybrid flow shop rescheduling problem to minimize the total weighted completion time, the total waiting time, and the difference in the number of operations processed on different machines for different stages in the original schedule and revised schedule. There are also other references to be focused on, such as, Filar et al. [10], Hoogeveen et al. [13], Zhang et al. [27] and Arnaout [2], etc.

The above references few attempted to take into account learning effect in rescheduling problems. However, there are many applications in production or service operation, i.e., emergent orders and emergency patients, etc, which should be dealt quickly so as to decrease loss. In this paper, we consider single-machine rescheduling problems with learning effect under disruption constraints to minimize the makespan and the total completion time of the new jobs being integrated into the schedule, respectively. For such problems, we propose polynomial-time algorithms to find the optimal job sequences, respectively.

The remainder of this paper is organized as follows. Notations and formulation of the considered problem are described in Section 2. Section 3 provides structural
results to be useful for the proposed problems, and studies several rescheduling models with a single disruption constraint. The rescheduling models with several practical measures of disruptions for minimizing the makespan and the total completion time will be considered in Section 4. Finally, concluding remarks and ideas for future research are given in Section 5.

2. Formulation and notations. A set of original jobs \( \mathcal{L}_O = \{J_1, \ldots, J_{n_O}\} \) is to be processed on a single-machine setting without being allowed preemption. We assume that the jobs in \( \mathcal{L}_O \) have been scheduled optimally to minimize some classical objective and \( \pi^* \) is an optimal schedule. Let \( \mathcal{L}_N = \{J_{n_O+1}, \ldots, J_{n_O+n_N}\} \) denote a set of new jobs which will be inserted to be processed along with jobs in \( \mathcal{L}_O \), write as \( \mathcal{L} = \mathcal{L}_O \cup \mathcal{L}_N \). Let \( n = n_O + n_N \). If Job \( J_j \in \mathcal{L} \) is processed at \( r \)th position in a sequence, its actual processing time is denoted as \( p_{jr} = p_j r^\alpha \), where \(-1 \leq \alpha < 0\) is the learning index of all jobs, and \( p_j \) denotes normal processing time. Suppose the new jobs arrive together at time zero after that the sequence of the jobs in \( \mathcal{L}_O \) has been determined without being processed. For any schedule \( \sigma \) of the jobs of \( \mathcal{L} \), we define the following notations:

- \( C_j(\sigma) \) denotes the complete time of job \( J_j \in \mathcal{L} \).
- \( J_i \prec J_j \) denotes the normal processing time of job \( J_i \) and \( J_j \) satisfies \( p_i \leq p_j \).
- \( C_{max}(\sigma) = \max\{C_j(\sigma)\} \) is the makespan of jobs in \( \mathcal{L} \) under schedule \( \sigma \).
- \( D_j(\pi^*, \sigma) = |r_2 - r_1| \) denotes the position disruption of job \( J_j \in \mathcal{L}_O \), i.e., if \( J_j \) is the \( r_1 \)th job to be processed in \( \pi^* \) and the \( r_2 \)th job to be processed in \( \sigma \), respectively.
- \( \Delta_j(\pi^*, \sigma) = |C_j(\sigma) - C_j(\pi^*)| \) denotes the time disruption of job \( J_j \in \mathcal{L}_O \).
- If there is no ambiguity, the above parameters are simplified to \( C_j, C_{max}, L_j, D_j(\pi^*) \) and \( \Delta_j(\pi^*) \), respectively.
- \( D_{max}(\pi^*) = \max\{D_j(\pi^*)\} \) denotes the maximum sequence disruption of the jobs in \( \mathcal{L}_O \).
- \( \Delta_{max}(\pi^*) = \max\{\Delta_j(\pi^*)\} \) denotes the maximum time disruption of the jobs in \( \mathcal{L}_O \).
- \( k \) denotes a known non-negative integer.

3. Rescheduling problem with a single disruption constraint. In this section, we study rescheduling problems with learning effect under a single disruption constraint. Before discussing the models, we give several lemmas which will be used in the subsequent sections.

**Lemma 3.1.** (Biskup, [5]) For problem \( 1|p_{jr} = p_j r^\alpha|\gamma \), where \( \gamma \in \{C_{max}, \sum C_j\} \), an optimal schedule can be obtained by sequencing the jobs in a non-decreasing order of \( p_j \) (SPT-rule).

From Hall and Potts [11], we know that if a rescheduling problem has the (SPT, SPT) property, then a polynomial-time algorithm can be designed. Therefore, if the objective is to minimize the makespan or the total completion time, we assume that the jobs in \( \mathcal{L}_O \) are indexed and sequenced in SPT-order in \( \pi^* \), i.e. \( p_1 \leq p_2 \leq \ldots \leq p_{n_O} \) for all objectives. In the following, we will show that the SPT-rule apply to the several considered rescheduling problems. Let \( \mathcal{L}_M = \{J_j \in \mathcal{L}_N|p_j < p_{n_O}\} \), obviously, we have the following lemma.
Lemma 3.2. For problems $1|p_{jr} = p_jr^\alpha, D_{\max}(\pi^*) \leq k|\gamma$ and $1|p_{jr} = p_jr^\alpha, \Delta_{\max}(\pi^*) \leq k|\gamma$, where $\gamma \in \{C_{\max}, \sum C_j\}$ under the job set $(\mathbb{L}_O, \mathbb{L}_N)$ can be polynomially reduced to the corresponding problem under the job set $(\mathbb{L}_O, \mathbb{L}_M)$.

Lemma 3.3. For Problems $1|p_{jr} = p_jr^\alpha, D_{\max}(\pi^*) \leq k|\gamma$ and $1|p_{jr} = p_jr^\alpha, \Delta_{\max}(\pi^*) \leq k|\gamma$, where $\gamma \in \{C_{\max}, \sum C_j\}$, all have an optimal schedule with no idle time between jobs, and

(i) a schedule for problem $1|p_{jr} = p_jr^\alpha, D_{\max}(\pi^*) \leq k|\gamma$ is feasible if and only if the number of jobs of $\mathbb{L}_N$ scheduled before the last job of $\mathbb{L}_O$ is no less than $k$;

(ii) a schedule for problem $1|p_{jr} = p_jr^\alpha, \Delta_{\max}(\pi^*) \leq k|\gamma$ is feasible if and only if the total actual processing time of jobs of $\mathbb{L}_O$ scheduled before the last job of $\mathbb{L}_O$ is no less than $k$.

Proof. The process of proof is similar to that of Lemma 1 in Hall and Potts [11], and omitted here. 

Lemma 3.4. For problems $1|p_{jr} = p_jr^\alpha, D_{\max}(\pi^*) \leq k|\gamma, 1|p_{jr} = p_jr^\alpha, \Delta_{\max}(\pi^*) \leq k|\gamma$, where $\gamma \in \{C_{\max}, \sum C_j\}$, there exists an optimal schedule in which the jobs of $\mathbb{J}_O$ are sequenced in SPT order, the jobs of $\mathbb{J}_N$ are sequenced in SPT order, and there is no idle time between jobs.

Proof. For jobs in $\mathbb{L}_O$, suppose there exists an optimal schedule $\sigma^*$ in which all jobs are sequenced in non-SPT order. Let $J_i$ be the job with the smallest index that appears relatively later to other jobs of $\mathbb{L}_O$ in $\sigma^*$ than in $\pi^*$, and $J_j(i < j)$ be the last job of $\mathbb{L}_O$ that precedes job $J_i$ in $\sigma^*$. Suppose jobs $J_1, J_2, ..., J_h$ are processed between $J_j$ and $J_i$. Let the starting time of job $J_j$ be $s$ and be scheduled in the $r$th position in $\sigma^*$. By interchanging the position of job $J_j$ and job $J_i$ to get a new schedule $\tilde{\sigma}$, and the starting time of job $J_i$ is $s$ which is scheduled in the $r$th position in $\tilde{\sigma}$, we have

\[ C_j(\sigma^*) = s + p_jr^\alpha, \]
\[ C_i(\sigma^*) = C_j(\sigma^*) + \tilde{p}_1(r + 1)^\alpha + ... + \tilde{p}_h(r + h)^\alpha + p_i(r + h + 1)^\alpha, \]

and

\[ C_i(\tilde{\sigma}) = s + p_ir^\alpha < C_j(\sigma^*) \]
\[ C_j(\tilde{\sigma}) = C_i(\tilde{\sigma}) + \tilde{p}_1(r + 1)^\alpha + ... + \tilde{p}_h(r + h)^\alpha + p_j(r + h + 1)^\alpha. \]

Since $i < j$, which implies $p_i < p_j$, then we have

\[ C_i(\sigma^*) - C_j(\tilde{\sigma}) = (p_j - p_i)(r^\alpha - (r + h + 1)^\alpha) > 0. \]

Therefore, jobs $\tilde{J}_1, \tilde{J}_2, ..., \tilde{J}_h$ are completed earlier in $\tilde{\sigma}$ than in $\sigma^*$, which means that the value of corresponding objective function does not increase after the interchange.

On one hand, since the job $J_i$ is processed at $i$th position in $\pi^*$ and job $J_j$ at $j$th position, we suppose that job $J_j$ is processed at $r_1$th position in $\tilde{\sigma}$. If $r_1 > j$, then $D_j(\pi^*, \tilde{\sigma}) = r_1 - j$ and $D_i(\pi^*, \sigma^*) = r_1 - i$, thus $D_j(\pi^*, \tilde{\sigma}) < D_i(\pi^*, \sigma^*)$; otherwise $D_j(\pi^*, \tilde{\sigma}) = j - r_1$ and $D_i(\pi^*, \sigma^*) = j - (r_1 - h) = j - r_1 + h$, thus $D_j(\pi^*, \tilde{\sigma}) < D_i(\pi^*, \sigma^*)$. Therefore, we have $D_{\max}(\pi^*, \tilde{\sigma}) \leq D_{\max}(\pi^*, \sigma^*)$.

On the other hand, if $C_j(\tilde{\sigma}) \geq C_j(\pi^*)$ (note that $C_i(\tilde{\sigma}) \geq C_i(\pi^*)$), we have $\Delta_j(\pi^*, \tilde{\sigma}) = C_j(\tilde{\sigma}) - C_j(\pi^*)$ and $\Delta_i(\pi^*, \tilde{\sigma}) = C_i(\tilde{\sigma}) - C_i(\pi^*)$, while $\Delta_j(\pi^*, \sigma^*) = C_j(\pi^*) - C_j(\pi^*) = C_i(\pi^*) - C_i(\pi^*) = C_j(\pi^*) - C_i(\pi^*)$ and $C_i(\pi^*) < C_j(\pi^*)$, then, we have $\Delta_j(\pi^*, \tilde{\sigma}) < \Delta_i(\pi^*, \sigma^*)$ and $\Delta_i(\pi^*, \tilde{\sigma}) < \Delta_i(\pi^*, \sigma^*)$. Therefore, we have $\Delta_{\max}(\pi^*, \tilde{\sigma}) \leq \Delta_{\max}(\pi^*, \sigma^*)$. 

If \( C_j(\tilde{\sigma}) < C_j(\pi^*) \), we have \( \Delta_j(\pi^*, \tilde{\sigma}) = C_j(\pi^*) - C_j(\tilde{\sigma}) \) and \( \Delta_j(\pi^*, \sigma^*), \) then \( \Delta_j(\pi^*, \tilde{\sigma}) < \Delta_j(\pi^*, \sigma^*), \) and \( \Delta_j(\pi^*, \tilde{\sigma}) = C_i(\tilde{\sigma}) - C_i(\pi^*) < \Delta_j(\pi^*, \sigma^*) = C_i(\sigma^*) - C_i(\pi^*). \) Therefore, we also have \( \Delta_{\text{max}}(\pi^*, \tilde{\sigma}) \leq \Delta_{\text{max}}(\pi^*, \sigma^*). \)

Based on the above analysis, \( \tilde{\sigma} \) is a feasible and optimal sequence.

A finite number of repetitions of this argument show that there exists an optimal schedule in which the jobs in \( L_0 \) are sequenced in SPT order as in \( \pi^* \), and a similar job insertion argument establishes that the jobs in \( L_N \) can also be sequenced in SPT order. Obviously, an optimal schedule should not be no-idle time in \( \pi^* \); otherwise, by removing this idle time maintains feasibility and decreases the value of objective function.

For problem 1 \( |p_jr = p_jr^{x}, D_{\text{max}}(\pi^*) \leq k|\gamma \), where \( \gamma \in \{C_{\text{max}}, \sum C_j\} \). From lemma 3, we know that there are at most \( k \) jobs in \( L_N \) to be sequenced before the last job in \( L_0 \). Thus, By referring to the (SPT, SPT) property and lemma 4, we propose the following algorithm.

**Algorithm I**

Step 0. Set the value of \( k (\geq 0) \).

Step 1. Index jobs of \( L_N \) in SPT-order: \( J_{n_0+1} \prec J_{n_0+2} \prec ... \prec J_{n_0+n_N} \).

Step 2. If \( p_{n_0+1} > p_{n_0} \), then goto Step 5; otherwise, choose \( m(\leq k) \) feasible jobs.

Step 3. Sequence jobs \( J_1, ..., J_{n_0}, J_{n_0+1}, ..., J_{n_0+m} \) in SPT-order in the first \( n_0 + m \) positions, and update the job set \( L_O \).

Step 4. Sequence jobs \( J_{n_0+m+1}, ..., J_{n_0+n_N} \) in SPT-order in the last \( n_N - m \) positions, and update the job set \( L_N \).

Step 5. Sequence the job sets of \( L_N \) following \( L_O \).

Obviously, from Step 1, we know that sequencing \( n_N \) jobs requires \( O(n_N \log n_N) \) time. In Steps 3 and 4, it requires \( O(n) \) time to merge the first \( m \) jobs of the SPT-ordered jobs of \( L_N \) with the jobs of \( L_O \) as sequenced in \( \pi \) and then place the last \( n_N - m \) jobs of the SPT-ordered jobs of \( L_N \) at the end of the schedule. Therefore, the complexity of Algorithm I is \( O(n + n_N \log n_N) \), and we have the following proposition.

**Proposition 1.** An optimal schedule for problem 1 \( |p_jr = p_jr^{x}, D_{\text{max}}(\pi^*) \leq k|\gamma \), where \( \gamma \in \{C_{\text{max}}, \sum C_j\} \), can be obtained in \( O(n + n_N \log n_N) \).

Similarly, we consider problem 1 \( |p_jr = p_jr^{x}, \sum D_j(\pi^*) \leq k|\sum C_j \). Lemma 1 and Lemma 4 show that a local optimal partial schedule can be found by merging several appropriate jobs of \( L_O \) and \( L_N \) in SPT-order, respectively. An optimal merging can be performed subject to the given constraint on the total sequence disruptions by the following dynamic programming algorithm.

**Algorithm DP**

Step 0. Set the value of \( k (\leq n_{ON}) \).

Step 1. Index jobs of \( L_N \) in SPT-order: \( J_{n_0+1} \prec J_{n_0+2} \prec ... \prec J_{n_0+n_N} \).

Step 2. Preprocessing compute \( \sum_{h=1}^{i} p_h h^{a} \) for \( i = 1, 2, ..., n_0 \) and \( \sum_{h=n_0+1}^{n_0+j} p_h h^{a} \) for \( j = 1, 2, ..., n_N \), respectively.
Step 3. Let \( f(i,j,\delta) = \text{minimum total completion time of a partial schedule for } J_1, J_2, \ldots, J_i \) and \( J_{nO+1}, \ldots, J_{nO+j} \), where the total sequence disruption is equal to \( \delta \). (Boundary condition \( f(0,0,0) = 0 \))

Step 4. Recurrence relation \( f(i,j,\delta) = \sum_{h=1}^i \alpha_h + \sum_{h=nO+1}^{nO+j} \alpha_h + \min \{ f(i-1, j, \delta - j), f(i, j-1, \delta) \} \)

Step 5. Optimal solution value \( \min_{0 \leq \delta \leq k} f(nO, nN, \delta) \).

Obviously, for the recurrence relation in Step 4, the term \( f(i-1, j, \delta - j) \) corresponds to the case where the partial schedule ends with job \( J_i \in \mathcal{L}_O \), and \( j \) jobs of \( \mathcal{L}_N \) appear before job \( J_i \) in such a partial schedule. The increase in the total sequence disruption is equal to \( j \). The term \( f(i, j-1, \delta) \) corresponds to the case where the partial schedule ends with job \( J_{nO+j} \in \mathcal{L}_N \). The number of the total sequence disruption is still equal to \( j \). Thus, we have

**Proposition 2.** An optimal schedule for problem 1\( |p_{jr} = p_j r^\alpha, \sum D_j(\pi^*) \leq k| \sum C_j \) can be found in \( O(n_O^2 n_N^2) \).

**Proof.** On the basis of lemma 3, we enumerate all possible ways of merging the SPT-ordered lists of jobs of \( \mathcal{L}_O \) and \( \mathcal{L}_N \), respectively. Algorithm DP does so by comparing the cost of all possible state transitions to find an optimal schedule. Since \( i \leq nO, j \leq nN \) and \( \delta \leq k \leq nO nN \), the values of state variables are at most \( O(n_O^2 n_N^2) \) time. We know that indexing the jobs of \( \mathcal{L}_N \) in SPT-order in Step 1 and evaluating the partial sums \( \sum_{h=1}^i \alpha_h \) for \( i = 1, 2, \ldots, nO \) and \( \sum_{h=nO+1}^{nO+j} \alpha_h \) for \( j = 1, 2, \ldots, nN \) in Step 2 requires \( O(n + nN \log nN) \) time. The recurrence relation in Step 4 requires constant time for each set of the values of the state variables. Therefore, the time complexity of Algorithm DP is \( O(n_O^2 n_N^2) \). \( \square \)

For problem 1\( |p_{jr} = p_j r^\alpha, \sum D_j(\pi^*) \leq k| \sum C_j \) we only modify the recurrence relation of the above algorithm into \( f(i, j, \delta) = \sum \alpha_j + \sum_{h=nO+1}^{nO+j} \alpha_j + \min \{ f(i-1, j, \delta - j), f(i, j-1, \delta) \} \). Similarly, we have the following proposition.

**Proposition 3.** An optimal schedule for problem 1\( |p_{jr} = p_j r^\alpha, \sum D_j(\pi^*) \leq k| \sum C_j \) can be found in \( O(n_O^2 n_N^2) \).

Similarly, we consider 1\( |p_{jr} = p_j r^\alpha, \sum D_j(\pi^*) \leq k| \sum C_j \), where \( \gamma \in \{ C_{max}, \sum C_j \} \).

The (SPT, SPT) property of Lemma 4 and Lemma 1 shows that an optimal schedule is found by merging the SPT-order lists of jobs of \( \mathcal{L}_O \). We know that \( C_j(\pi^*) \) can be obtained by sequencing the \( j \) jobs in \( \mathcal{L}_O \) in SPT-order. Because of the (SPT, SPT) property of Lemma 2, we have \( C_j(\sigma) \geq C_j(\pi^*) \) for \( J_j \in \mathcal{L}_O \). Thus, the constraint \( \Delta_{max}(\pi^*) \leq k \) reduces to \( C_j(\sigma) \leq C_j(\pi^*) + k \), equivalently, which means job \( J_j \in \mathcal{L}_O \) with a deadline \( \bar{d}_j = C_j(\pi^*) + k \) and \( \bar{d}_j = \infty \) for job \( J_j \in \mathcal{L}_N \), and it is obvious that \( \bar{d}_1 \leq \bar{d}_2 \leq \ldots \leq \bar{d}_{nO} \). Once the deadlines have been computed, the problem can be denoted as 1\( |p_{jr} = p_j r^\alpha, d_j| \gamma \), where \( \gamma \in \{ C_{max}, \sum C_j \} \). Since Smith (1956) solved the problem 1\( |d_j| \sum C_j \) by a backward procedure in which a feasible job with the largest due-date is assigned to the last unfilled position in a sequence, we propose an algorithm through modifying Smith’s rule to solve the problem.

**Algorithm MS**
Input: a job set $\mathcal{J} = \mathcal{J}_O + \mathcal{J}_N$
Output: an optimal sequence so as to minimize the total completion time (makespan)

Step 0. Set the value of $k, \zeta = \phi$ and compute $d_j = C_j(\pi^*) + k, j = 1, 2, ..., n_O$. 

Step 1. Index jobs of $\mathcal{J}_O$ in SPT-order: $J_1 < J_2 < ... < J_{n_O}$, and $\mathcal{J}_N$ in SPT-order: $J_{n_O+1} < J_{n_O+2} < ... < J_{n_O+n_N}$, then reorder the jobs of $\mathcal{J}$ in SPT-order and let $\zeta$ be the set of jobs sequenced after job $J_{n_O}$.

Step 2. If job $J_{n_O}$ is sequenced before job $J_{n_O+1}$, then terminate; otherwise, let job $J_{n_O+j}$ be the last job of $\mathcal{J}_N$ that sequenced before job $J_{n_O}$.

Step 3. Compute $C_j(\mathcal{J})$ for all jobs of $\mathcal{J}_O$, if $C_j(\mathcal{J}) \leq d_j$, goto Step 5; otherwise, remove job $J_{n_O+j}$ from $\mathcal{J}$.

Step 4. Let $\mathcal{J} = \mathcal{J} - \{J_{n_O+j}\}, \zeta = \zeta + \{J_{n_O+j}\}$ and $j = j - 1$, goto Step 2.

Step 5. Sequence the jobs of $\zeta$ in SPT-order after $\mathcal{J}$ and obtain the optimal schedule.

Proposition 4. An optimal schedule for problem $1|p_{jr} = p_jr^\alpha, \Delta_{\max}(\pi^*) \leq k|\gamma$, where $\gamma \in \{C_{\max}, \sum C_j\}$, can be found in $O(n + n_N \log n_N)$.

Proof. The jobs in $\mathcal{J}_N$ are indexed in SPT-order, so that the job to be scheduled in the last unfilled position in the sequence is either the last unscheduled job in $\pi^*$ or the last unscheduled job among the SPT-indexed in $\mathcal{J}_N$. Because the ordering of the jobs of $\mathcal{J}_N$ requires $O(n_N \log n_N)$ time and the schedule construction requires $O(n)$ time. From lemma 3, it only remains to enumerate all possible ways of merging the SPT-order lists of jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$. Algorithm MS does so by comparing the cost of all possible state transitions to find an optimal schedule, while the recurrence relation requires constant time for each set of the values of the state variables. Thus, the overall time complexity of Algorithm MS is $O(n + n_N \log n_N)$.

Now, we focus on problem $1|p_{jr} = p_jr^\alpha, \Delta_{\max}(\pi^*) \leq k|\sum C_j$. We know that the problem can be deduced to $1|\sum \Delta_j(\pi^*) \leq k|\sum C_j$ when $\alpha = 0$, and which is proved to be binary NP-hard (Hall and Potts, 2004). Therefore, problem $1|p_{jr} = p_jr^\alpha, \sum \Delta_j(\pi^*) \leq k|\sum C_j$ is at least binary NP-hard. Lemma 1 and lemma 4 show that local optimal partial schedule can be found by merging several appropriate jobs of $\mathcal{J}_O$ and $\mathcal{J}_N$ by SPT-order, respectively. For such problem, an optimal merging can be performed subject to the given constraint on the total sequence disruptions by the following dynamic programming algorithm.

Algorithm DP2

Step 0. Given $p_j$ ($j = 1, 2, ..., n$) and $k (\leq n_O P_N)$, where $P_N = \sum_{j=1}^{n_N} p_{n_O+j} r^\alpha$.

Step 1. Index jobs of $\mathcal{J}_O$ in SPT-order: $J_1 < J_2 < ... < J_{n_O}$, and $\mathcal{J}_N$ in SPT-order: $J_{n_O+1} < J_{n_O+2} < ... < J_{n_O+n_N}$.

Step 2. If $p_r < p_{n_O+i} \leq p_{r+1}$ ($i = 1, 2, ..., n_O, r = 1, 2, ..., n_O - 1$) and merge job $J_{n_O+i}$ in the $(r+1)$th position of $\mathcal{J}_O$. If $\sum_{j=r+1}^{n_O} |C_j(\mathcal{J}_O + \{J_{n_i}\}) - C_j(\pi^*)| \leq k$, then let $\mathcal{J}_O = \mathcal{J}_O + \{J_{n_i}\}$.

Step 3. Preprocessing compute $\sum_{l=1}^{i} p_l r^\alpha$ for $i = 1, 2, ..., n_O$ and $\sum_{l=n_O+1}^{n_O+j} p_l r^\alpha$ for $j = 1, 2, ..., n_N$, respectively.
Step 4. Let \( f(i, j, \tau) = \text{minimum total completion time of a partial schedule for jobs } J_1, J_2, ..., J_i \text{ and } J_{n_O+1}, J_{n_O+2}, ..., J_{n_O+j}, \text{ where the total time disruption is equal to } \tau. \) (boundary condition \( f(0, 0, 0) = 0 \))

Step 5. Recurrence relation \( f(i, j, \tau) = \sum_{l=1}^{i} p_l \tau^a + \sum_{l=n_O+1}^{n_O+j} p_l \tau^a + \min\{ f(i-1, j, \tau - \sum_{l=n_O+1}^{n_O+j} p_l \tau^a), f(i, j-1, \tau) \} \)

Step 6. Optimal solution value \( \min_{0 \leq \tau \leq k} f(n_O, n_N, \tau). \)

Obviously, for the recurrence relation in Step 5, the term \( f(i-1, j, \tau - \sum_{l=n_O+1}^{n_O+j} p_l \tau^a) \) corresponds to the case where the partial schedule ends with job \( J_i \in L_O \), and \( j \) jobs of \( L_N \) appear before job \( J_i \) in such a partial schedule. The increase in the total sequence disruption is equal to \( \tau \). The term \( f(i, j-1, \tau) \) corresponds to the case where the partial schedule ends with job \( J_{n_O+j} \in L_N \). The total sequence disruption is still equal to \( \tau \). Thus, we have

**Proposition 5.** An optimal schedule for problem 1 \(| p_{jr} = p_j r^a, \sum \Delta_j(\pi^*) \leq k | \sum C_j \) can be found in \( O(n_O n_N \min\{n_O P_N, n_N P_O\}) \).

**Proof.** Similar to Proposition 2, we enumerate all possible ways of merging the SPT-ordered lists of jobs of \( L_O \) and \( L_N \), respectively. Since \( i \leq n_O, j \leq n_N \) and \( \delta \leq k \leq n_O P_N \), the values of state variables are at most \( O(n_O^2 n_N P_N) \), where \( P_N = \sum_{j=1}^{n_N} p_{n_O+j} r^a \). We know that indexing the jobs of \( L_N \) in SPT-order in Step 1 and evaluating the partial sums \( \sum_{h=1}^{i} p_h \tau^a \) for \( i = 1, 2, ..., n_O \) and \( \sum_{h=n_O+1}^{n_O+j} p_h \tau^a \) for \( j = 1, 2, ..., n_N \) in Step 2 requires \( O(n + n_N \log n_N) \) time. The recurrence relation in Step 4 requires constant time for each set of the values of the state variables. Therefore, the time complexity of Algorithm DP2 is \( O(n_O^2 n_N P_N) \).

Similarly, by reversing the role of \( J_O \) and \( J_N \), we can regard the jobs of \( J_N \) having a time disruption. Let \( P_O = \sum_{j=1}^{n_O} p_j j^a, \) and set \( k \leq n_N P_O \). We can show that for any no-idle schedule where the (SPT, SPT) property holds. The total completion time is equal to the total time disruption of the jobs of \( J_O \) plus the total time disruption of jobs of \( J_N \), plus a constant. Thus, we can define an alternative algorithm which is similar to algorithm DP2 where \( i, j \) and the total time disruption of the jobs of \( J_N \) are the state variables and the constraint on the total time deviation of the jobs of \( J_O \) is implemented in Step 6. The time complexity of this alternative algorithm is \( O(n_O n_N^2 P_O) \). Therefore, integrating DP2 with the alternative algorithm, an optimal schedule of the problem can be obtained in \( O(n_O n_N \min\{n_O P_N, n_N P_O\}) \).

For problem 1 \(| p_{jr} = p_j r^a, \sum \Delta_j(\pi^*) \leq k | C_{max} \), we only modify the recurrence relation of the above algorithm into \( f(i, j, \delta) = p_i \tau^a + p_{n_O+j}(n_O+j)^a + \min\{ f(i-1, j, \delta - \sum_{l=n_O+1}^{n_O+j} p_l \tau^a), f(i, j-1, \delta) \} \). Similarly, we have the following proposition.

**Proposition 6.** An optimal schedule for problem 1 \(| p_{jr} = p_j r^a, \sum \Delta_j(\pi^*) \leq k | C_{max} \) can be found in \( O(n_O n_N \min\{n_O P_N, n_N P_O\}) \).
4. Rescheduling objective and disruption cost problem. In this section, we consider rescheduling problems to minimize the sum of several classical objectives and the costs of position disruption caused by inserting new jobs. For such problems, we try to find an appropriate partial sequence to insert the original job sequence so as to minimize the objectives. That is, if there exist $n_N$ new jobs to be inserted to the original job sequence, the position disruption reaches up to $n_N$ times at most. Thus, according to the definition of $D_{\text{max}}$, we use the approach which has solved problem $1|p_{jr} = p_j r^\alpha, D_{\text{max}}(\pi^*) \leq k|C_{\text{max}}$ $(k = 0, 1, 2, ..., n_N)$ to study problem $1|p_{jr} = p_j r^\alpha|C_{\text{max}} + \mu D_{\text{max}}(\pi^*)$. Let $\sigma_k$ denote an optimal schedule and compute $\min_{0 \leq k \leq n_N} \{C_{\text{max}}(\sigma_k) + \mu k\}$, we propose the following algorithm.

**Algorithm II**

Step 1. Index the jobs of $\mathcal{L}_N$ in SPT-order, i.e. $J_{n_O+1} \prec J_{n_O+2} \prec ... \prec J_{n_O+n_N}$.

Step 2. If job $p_{n_O} \leq p_{n_O+1}$, then terminate; otherwise, define $j$ such that number $n_O + j$ is the index of the last job of $J_N$ sequenced before job $J_{n_O}$.

Step 2.1 Let $r_j$ be the processed position of job $J_{n_O+j}$, and $x_j$ be the number of those jobs which sequenced after jobs $J_{n_O+j}$ up to and including $J_{n_O}$.

Step 2.2 Set $\sigma_j$ to be the schedule defined by the current job sequence, and compute $C_{\text{max}}(\sigma_j)$.

Step 3. Execute the following for $k = j, j-1, ..., 1$.

Step 3.1 Form schedule $\sigma_{j-1}$ from $\sigma_j$ by moving job $J_{n_O+k}$ immediately after job $J_{n_O}$.

Step 3.2 Let $A_j = (p_{n_O-x_j+1}, p_{n_O-x_j+2}, ..., p_{n_O}, p_{n_O+j})$.

$$H_{x_j+1} = \begin{pmatrix} 1 & -1 & 0 & ... & 0 \\ 0 & 1 & -1 & ... & 0 \\ ... & ... & ... & ... & 1 \end{pmatrix} \quad \text{and} \quad B_j = (r_j^\alpha, (r_j + 1)^\alpha, ..., (r_j + x_j)^\alpha)^T.$$

Compute $C_{\text{max}}(\sigma_j) = C_{\text{max}}(\sigma_{j-1}) + A_j H_{x_j+1} B_j$.

Step 4. Find the optimal schedule $\sigma^* = \sigma_k^*$ such that $k^* = \arg \min_{0 \leq k \leq j} \{C_{\text{max}}(\sigma_k) + \mu k\}$.

**Proposition 7.** For problem $1|p_{jr} = p_j r^\alpha|C_{\text{max}} + \mu D_{\text{max}}(\pi^*)$, an optimal schedule can be found Algorithm II in $O(n + n_N \log n_N)$.

**Proof.** Algorithm II computes $\min_{0 \leq k \leq n_N} \{C_{\text{max}}(\sigma_k) + \mu k\}$, where $\sigma_k$ is found by solving problem $1|p_{jr} = p_j r^\alpha, D_{\text{max}}(\pi^*) \leq k|C_{\text{max}}$, and it generates an optimal schedule since all the possible values of $k$ are considered. Step 1 requires $O(n_N \log n_N)$ time to construct a SPT schedule of all jobs of $\mathcal{L}_N$, and $O(n)$ time to construct a SPT schedule of all jobs, while Step 2 requires $O(n)$ time. Step 3 performs at most $j$ times since $j \leq n_N$, and each application requires constant time. Therefore, the computational complexity of Algorithm II is $O(n + n_N \log n_N)$. \(\square\)

The following numerical example will illustrate the working principle of the above algorithm.

**Example 1.** Consider a set of original scheduled jobs $\mathcal{L}_O = \{J_1, J_2, J_3, J_4, J_5, J_6\}$ with their normal processing time $p_1 = 1, p_2 = 3, p_3 = 4, p_4 = 7, p_5 = 8, p_6 = 12$, respectively. A set of newly arrived jobs $\mathcal{L}_N = \{J_a, J_b, J_c, J_d, J_e, J_f\}$ with their
normal processing time $p_a = 2, p_b = 5, p_c = 6, p_d = 10, p_e = 12, p_f = 13$. Suppose learning index $\alpha = -0.322$ and $\mu = 2.5$.

Obviously, if all the jobs are rescheduled by SPT rule, the objective value is 53.4803. By using above algorithm, there exist a finite number of new jobs (such as $j = 4$) to be inserted to the original sequence to construct a new sequence, i.e. $\sigma_j = (J_1, J_a, J_2, J_b, J_c, J_d, J_e, J_f)$, and according to Step 3.1, we have $\sigma_{j-1} = (J_1, J_a, J_2, J_b, J_c, J_d, J_e, J_f)$. Thus, we have $r_j = 9, x_j = 1$ and obtain $A_j = (p_{n_O}, p_{n_O+4}) = (p_{n_O}, p_d)$, $B_j = (r_j^\alpha, (r_j + x_j)^\alpha) = (9^\alpha, 10^\alpha)$. Therefore, $C_{max}(\sigma_j) = C_{max}(\sigma_{j-1}) + (p_{n_O}, p_d)H_2(9^\alpha, 10^\alpha)^T$. Repeating such computation, we have the optimal sequence $J_1 \prec J_a \prec J_2 \prec J_b \prec J_c \prec J_d \prec J_e \prec J_f$ from such $k$ sequences and the optimal cost is 51.032.

Secondly, we consider problem $1|p_{jr} = p_j r^\alpha| \sum C_j + \mu D_{max}(\pi^*)$. For such problem, we propose the following algorithm.

Algorithm III

Step 1. Index the jobs of $\mathcal{L}_N$ in SPT order.

Step 2. If job $p_{n_O} \leq p_{n_O+1}$, then terminate; otherwise, define $j$ such that $n_O + j$ is the index of the last job of $J_N$ that is sequenced before job $J_{n_O}$.

Step 2.1 Let $r_j$ be the processed position of job $J_{n_O+j}$, and $x_j$ be the number of those jobs which are sequenced after jobs $J_{n_O+j}$ up to and including $J_{n_O}$.

Step 2.2 Let $A_j = (n - n_O + x_j - j + 1, n - n_O + x_j - j, ..., n - n_O - j + 1)$, $B_j = (p_{n_O-x_j+1} - p_{n_O+j}, p_{n_O-x_j+2} - p_{n_O-x_j}, ..., p_{n_O+j} - p_{n_O})^T$ and $H_{x_j+1} = \begin{pmatrix} r_j^\alpha & 0 & \cdots & 0 \\ 0 & (r_j + 1)^\alpha & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (r_j + x_j)^\alpha \end{pmatrix}$.

Step 2.3 Set $\sigma_j$ to be the schedule defined by the current job sequence, and compute $\sum_{h \in \mathcal{L}} C_h(\sigma_j)$.

Step 3. Execute the following for $k = j, j-1, ..., 1$.

Step 3.1 Form schedule $\sigma_{k-1}$ from $\sigma_k$ by moving job $J_{n_O+k}$ immediately after job $J_{n_O}$.

Step 3.2 Compute $\sum_{h \in \mathcal{L}} C_h(\sigma_{k-1}) = \sum_{h \in \mathcal{L}} C_h(\sigma_k) + A_k H_{x_j+1} B_k$.

Step 4. Find the optimal schedule $\pi^* = \sigma_k$ such that $k^* = \arg \min_{0 \leq k \leq j} \{ \sum_{h \in \mathcal{L}} C_h(\sigma_k) + \mu k \}$.

Proposition 8. For problem $1|p_{jr} = p_j r^\alpha| \sum C_j + \mu D_{max}(\pi^*)$, an optimal schedule can be found by Algorithm III in $O(n + n N \log n N)$.

Proof. The proof is similar to that of Proposition 4. \qed

Example 2. Suppose $\mathcal{L}_O = \{J_1, J_2, J_3, J_4, J_5\}$ and $\mathcal{L}_N = \{J_a, J_b, J_c, J_d, J_e, J_f, J_g\}$ such that $\alpha = -0.322, \mu = 2.5, p_1 = 1, p_2 = 3, p_3 = 7, p_4 = 8, p_5 = 10, p_a = 2, p_b = 4, p_c = 5, p_d = 6, p_e = 9, p_f = 12, p_g = 13$.

If all the jobs are rescheduled by SPT rule, the objective value is 225.4965. Similar to do the procedure in example 1 and using above algorithm, we can obtain the optimal schedule by sequencing the jobs in the order as $J_1 \prec J_a \prec J_2 \prec J_b \prec J_c \prec J_d \prec J_3 \prec J_4 \prec J_5 \prec J_e \prec J_f \prec J_g$, and the optimal cost is 223.5387.
For problem 1\( p_{jr} = p_j r^\alpha | \gamma + \mu \Delta_{\text{max}}(\pi^*) \), where \( \gamma \in \{ C_{\text{max}}, \sum C_j \} \). For such problem, similar to the above proposed algorithm, we develop following algorithms by considering \( \sigma^* \) as an optimal schedule and the maximum time disruption to be \( \Delta_{\text{max}}(\pi^*) = |C_{n_O}(\sigma^*) - C_{n_O}(\pi^*)| \).

Algorithm IV

Step 1. Index the jobs of \( \mathcal{L}_N \) in SPT order.

Step 2. If job \( p_{n_O} \leq p_{n_O+1} \), then terminate; otherwise, define \( j \) such that \( n_O + j \)

is the index of the last job of \( \mathcal{J}_N \) that is sequenced before job \( J_{n_O} \).

Step 2.1 Let \( r_j \) be the processed position of job \( J_{n_O+j} \), and \( x_j \) be the number of those jobs which are sequenced after jobs \( J_{n_O+j} \) up to and including \( J_{n_O} \).

Step 2.2 Set \( \sigma_j \) to be the schedule defined by the current job sequence, and compute 

\( C_{n_O}(\sigma_j) \) and \( C_{\text{max}}(\sigma_j) \).

Step 3. Execute the following for \( k = j, j - 1, ..., 1 \).

Step 3.1 Form schedule \( \sigma_{j-1} \) from \( \sigma_j \) by moving job \( J_{n_O+k} \) immediately after job \( J_{n_O} \).

Step 3.2 Let \( A_j = (p_{n_O-x_j+1}, p_{n_O-x_j+2}, ..., p_{n_O}, p_{n_O+j}) \),

\[
H_{x_j+1} = \begin{pmatrix}
1 & -1 & 0 & \ldots & 0 \\
0 & 1 & -1 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & 0 & 0 & \ldots & 1
\end{pmatrix}
\]

and \( B_j = (r_j^\alpha, (r_j + 1)^\alpha, ..., (r_j + x_j)^\alpha)^T \).

Compute \( C_{\text{max}}(\sigma_j) = C_{\text{max}}(\sigma_{j-1}) + A_j H_{x_j+1} B_j \) and \( C_{n_O}(\sigma_{k-1}) = C_{n_O}(\sigma_k) + A_j H_{x_j+1} B_j - p_{n_O+j}(r_j + x_j)^\alpha \).

Step 4. Find schedule \( \sigma_k^* \) such that \( k^* = \arg \min_{0 \leq k \leq j} \{ C_{\text{max}}(\sigma_k) + \mu |C_{n_O}(\sigma_k) - C_{n_O}(\pi^*)| \} \).

**Proposition 9.** For problem 1\( p_{jr} = p_j r^\alpha | C_{\text{max}} + \mu \Delta_{\text{max}}(\pi^*) \), an optimal schedule can be found by Algorithm IV in \( O(n + n_N \log n_N) \).

**Proof.** The process of proof is similar to that of Proposition 4.

**Example 3.** Suppose \( \mathcal{L}_O = \{ J_1, J_2, J_3, J_4, J_5 \} \) and \( \mathcal{L}_N = \{ J_a, J_b, J_c, J_d, J_e, J_f, J_g \} \) such that \( \alpha = -0.322, \mu = 2.5, p_1 = 1, p_2 = 3, p_3 = 7, p_4 = 8, p_5 = 12, p_a = 2, p_b = 4, p_c = 5, p_d = 6, p_e = 10, p_f = 12, p_g = 13 \).

If all the jobs are rescheduled by SPT rule, the objective value is 47.0544. Similar to do the procedure in example 1 and using above algorithm, we can obtain the optimal schedule by sequencing the jobs in the order as \( J_1 < J_2 < J_3 < J_4 < J_5 < J_a < J_b < J_c < J_d < J_e < J_f < J_g \), and the optimal cost is 45.5074.

For problem 1\( p_{jr} = p_j r^\alpha | \sum C_j + \mu \Delta_{\text{max}}(\pi^*) \), we propose the following algorithm.

Algorithm V

Step 1. Index the jobs of \( \mathcal{L}_N \) in SPT order.

Step 2. If job \( p_{n_O} \leq p_{n_O+1} \), then terminate; otherwise, define \( j \) such that \( n_O + j \)

is the index of the last job of \( \mathcal{J}_N \) that is sequenced before job \( J_{n_O} \).

Step 2.1 Let \( r_j \) be the processed position of job \( J_{n_O+j} \), and \( x_j \) be the number of those jobs which are sequenced after jobs \( J_{n_O+j} \) up to and including \( J_{n_O} \).

Step 2.2 Set \( \sigma_j \) to be the schedule defined by the current job sequence, and compute 

\( C_{n_O}(\sigma_j) \) and \( C_{\text{max}}(\sigma_j) \).
Step 3. Execute the following for $k = j, j - 1, \ldots, 1$.
Step 3.1 Form schedule $\sigma_{j-1}$ from $\sigma_j$ by moving job $J_{nO+k}$ immediately after job $J_{nO}$.
Step 3.2 Let $A_j = (n - nO + x_j - j + 1, n - nO + x_j - j, \ldots, n - nO - j + 1)$, $B_j = (p_{nO-x_j+1} - p_{nO+j}, p_{nO-x_j+2} - p_{nO+j}, \ldots, p_{nO} - p_{nO})^T$, and $H_{x_j+1} = \begin{bmatrix} r_j^a & 0 & \cdots & 0 \\ 0 & (r_j + 1)^a & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (r_j + x_j)^a \end{bmatrix}$ and $B_j = (r_j^a,(r_j + 1)^a,\ldots,(r_j + x_j)^a)^T$.
Compute $C_{\max}(\sigma_j) = C_{\max}(\sigma_{j-1}) + A_jH_{x_j+1}B_j$ and $C_{\min}(\sigma_{k-1}) = C_{\min}(\sigma_k) + A_jH_{x_j+1}B_j + \mu C_{\min}(\sigma_k)^a$.

Step 4. Find schedule $\sigma_k$ such that $k^* = \arg \min_{0 \leq k \leq j} \{C_{\max}(\sigma_k) + \mu C_{\min}(\sigma_k) - C_{\min}(\pi^*)\}$.

Proposition 10. For problem 1, $\mathbb{P}_{xj} = p_{xj}^a\sum C_j + \mu \Delta_{\max}(\pi^*)$, an optimal schedule can be found by Algorithm V in $O(n + nN \log nN)$.

Proof. The process of proof is similar to that of Proposition 4. \qed

Example 4. Suppose $L_O = \{J_1, J_2, J_3, J_4, J_5\}$ and $L_N = \{J_a, J_b, J_c, J_d, J_e, J_f, J_g\}$ such that $\alpha = -0.322, \mu = 2.5, \tau_1 = 2, \tau_2 = 3, \tau_3 = 7, \tau_4 = 8, \tau_5 = 10, \tau_6 = 2, \tau_7 = 4, \tau_8 = 5, \tau_9 = 6, \tau_10 = 9, \tau_11 = 12, \tau_12 = 13$.

If all the jobs are rescheduled by SPT rule, the objective value is 216.5706. Similar to do the procedure in example 1 and using above algorithm, we can obtain the optimal schedule by sequencing the jobs in the order as $J_1 < J_a < J_2 < J_b < J_c < J_d < J_e < J_3 < J_4 < J_5 < J_6 < J_f < J_g$, and the optimal cost is 216.1061.

5. Conclusions. In this paper, we consider single-machine rescheduling problems with learning effect under disruption constraints. In the proposed scheduling models, learning effect means that the workers gain experience from producing jobs and make the actual processing time of jobs shorter than their normal processing time, while rescheduling means to schedule the sequenced jobs again together with a set of new arrived jobs so as to generate a new feasible schedule and create disruptions. The goal of optimizing system performance is addressed by minimizing the makespan or the total completion time of the new jobs being integrated into the schedule under the limited disruption constraints, respectively. For the considered problems, by using several proposed lemmas and classical scheduling rules, we provide polynomial-time algorithms of minimizing the makespan and the total completion time problems under several disruption constraints between the original and revised schedules, respectively. For some models, we also designed dynamic programming algorithms to analyze them. However, in this paper, we only study single-machine rescheduling problems with learning effect consideration for the determinate new arrival jobs, therefore, we will extend the models to dynamic new job arrivals and multiple machines in the future.

Acknowledgments. The authors are grateful to the Editor and the anonymous referees for their valuable comments and suggestions to significantly improve the quality of the paper.
REFERENCES

[1] M. J. Anzanello and F. S. Fogliatto, Learning curve models and applications: Literature review and research directions, *International Journal of Industrial Ergonomics*, 41 (2011), 573–583.

[2] J. P. Arnaout, Rescheduling of parallel machines with stochastic processing and setup times, *Journal of Manufacturing Systems*, 33 (2014), 376–384.

[3] M. Azizoglu and O. Alagoz, Parallel-machine rescheduling with machine disruptions, *IIE Transactions*, 37 (2007), 1113–1118.

[4] A. Bachman and A. Janiak, Scheduling job with position-dependent processing times, *Journal of the Operational Research Society*, 55 (2004), 257–264.

[5] D. Biskup, Single-machine scheduling with learning considerations, *European Journal of Operational Research*, 115 (1999), 173–178.

[6] D. Biskup, A state-of-the-art review on scheduling with learning effects, *European Journal of Operational Research*, 188 (2008), 315–329.

[7] M. B. Cheng and S. J. Sun, The single-machine scheduling problems with deteriorating jobs and learning effect, *Zhejiang Univ. Science A*, 7 (2006), 597–601.

[8] M. B. Cheng, P. R. Tadikamalla, J. Shang and B. X. Zhang, Single machine scheduling problems with exponentially time-dependent learning effects, *Journal of Manufacturing Systems*, 34 (2015), 60–65.

[9] Y. Chiu and C. J. Shih, Rescheduling strategies for integrating rush orders with preventive maintenance in a two-machine flow shop, *International Journal of Production Research*, 50 (2011), 5783–5794.

[10] J. A. Filar, P. Manyem, D. M. Panton and K. White, A model for adaptive rescheduling of flights in emergencies, *Journal of Industrial and Management Optimization*, 3 (2007), 335–356.

[11] N. G. Hall and C. N. Potts, Rescheduling for new orders, *Operations Research*, 52 (2004), 440–453.

[12] N. G. Hall and C. N. Potts, Rescheduling for Job Unavailability, *Operations Research*, 58 (2010), 746–755.

[13] H. Hoogeveen, C. Lent and V. Tkindtal, Rescheduling for new orders on a single machine with setup times, *European Journal of Operational Research*, 223 (2012), 40–46.

[14] K. Katragjini, E. Vallada and R. Ruiz, Flow shop rescheduling under different types of disruption, *International Journal of Production Research*, 51 (2013), 780–797.

[15] L. Liu and H. Zhou, On the identical parallel-machine rescheduling with job rework disruption, *Computers & Industrial Engineering*, 66 (2013), 186–198.

[16] Z. Liu and K. Ro Young, Rescheduling for machine disruption to minimize makespan and maximum lateness, *Journal of Scheduling*, 17 (2014), 339–352.

[17] G. Mosheiov, Scheduling problems with learning effect, *European Journal of Operational Research*, 132 (2001), 687–693.

[18] M. Ozlen and M. Azizoglu, Rescheduling unrelated parallel machines with total flow time and total disruption cost criteria, *Journal of the Operational Research Society*, 62 (2011), 152–164.

[19] L. L. Sun, F. J. Luan, Y. Ying and K. Mao, Rescheduling optimization of steelmaking-continuous casting process based on the Lagrangian heuristic algorithm, *Journal of Industrial and Management Optimization*, 13 (2017), 1431–1448.

[20] B. Vahedi-Nouri, P. Fattahi, M. Rohaninejad and R. Tavakkoli-Moghaddam, Minimizing the total completion time on a single machine with the learning effect and multiple availability constraints, *Applied Mathematical Modelling*, 37 (2013), 3126–3137.

[21] J. B. Wang, M. Q. Liu, N. Yin and P. Ji, Scheduling jobs with controllable processing time, truncated job-dependent learning and deterioration effects, *Journal of Industrial and Management Optimization*, 13 (2017), 1025–1039.

[22] T. P. Wright, Factors Affecting the Cost of Airplanes, *Journal of Aeronautical Sciences* 3 (1936), 122–128.

[23] S. D. Wu, R. H. Storer and P. C. Chang, A rescheduling procedure for manufacturing systems under random disruptions, in: G. Fandel, T. Gulledge, A. Jone (Eds.), *New Directions for Operations Research in Manufacturing*, Springer, Berlin, Germany, 1992, pp. 292–306.

[24] B. B. Yang, Single machine rescheduling with new jobs arrivals and processing time compression, *International Journal of Advanced Manufacture Technology*, 34 (2007), 378–384.
[25] Y. Q. Yin, D. H. Xu, K. B. Sun and H. X. Li, Some scheduling problems with general position-dependent and time-dependent learning effects, *Information Sciences*, 179 (2009), 2416–2425.

[26] J. J. Yuan and Y. Mu, Rescheduling with release dates to minimize makespan under a limit on the maximum sequence disruption, *European Journal of Operational Research*, 182 (2007), 936–944.

[27] L. Zhang, L. Gao and X. Li, A hybrid intelligent algorithm and rescheduling technique for job shop scheduling problems with disruptions, *International Journal of Advanced Manufacturing Technology*, 65 (2013), 1141–1156.

[28] C. Zhao and H. Tang, Rescheduling problems with deteriorating jobs under disruptions, *Applied Mathematical Modelling*, 34 (2010), 238–243.

Received March 2016; 1st revision March 2017; final revision August 2017.

E-mail address: chengmb99@hotmail.com
E-mail address: shallshould@outlook.com
E-mail address: gshliu000@163.com