Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 135610, 10 pages
http://dx.doi.org/10.1155/2013/135610

Research Article

Single Machine Problem with Multi-Rate-Modifying Activities under a Time-Dependent Deterioration

M. Huang,[1] Huaping Wu,[1] Vincent Cho,[2] W. H. Ip,[3] Xingwei Wang,[1] and C. K. Ng[3]

[1] *College of Information Science and Engineering, Northeastern University, and State Key Laboratory of Synthetical Automation for Process Industries (Northeastern University) Shenyang, Wenhua Road, Heping District, No. 11, Lane 3, Shenyang, Liaoning 110819, China*
[2] *Department of Management and Marketing, The Hong Kong Polytechnic University, Hong Kong*
[3] *Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong*

Correspondence should be addressed to Huaping Wu; wuhuaping2010@126.com

Received 13 February 2013; Revised 26 April 2013; Accepted 12 May 2013

Academic Editor: Ferenc Hartung

The single machine scheduling problem with multi-rate-modifying activities under a time-dependent deterioration to minimize makespan is studied. After examining the characteristics of the problem, a number of properties and a lower bound are proposed. A branch and bound algorithm and a heuristic algorithm are used in the solution, and two special cases are also examined. The computational experiments show that, for the situation with a rate-modifying activity, the proposed branch and bound algorithm can solve situations with 50 jobs within a reasonable time, and the heuristic algorithm can obtain the near-optimal solution with an error percentage less than 0.053 in a very short time. In situations with multi-rate-modifying activities, the proposed branch and bound algorithm can solve the case with 15 jobs within a reasonable time, and the heuristic algorithm can obtain the near-optimal with an error percentage less than 0.070 in a very short time. The branch and bound algorithm and the heuristic algorithm are both shown to be efficient and effective.

1. Introduction

In the classical deterministic scheduling problems, job processing times are supposed to be constant; however, it is not the case for all industrial processes, for example, in cleaning assignments, fire fighting, steel production, and so on. A job that is processed later takes more time than when it is processed earlier, and the phenomenon is known as scheduling with deterioration jobs [1].

J. N. D. Gupta and S. K. Gupta [1] and Browne and Yechiali [2] first proposed the job deterioration scheduling problem, and since then, it has been extensively studied. For instance, Wang et al. present a single machine scheduling problem with deteriorating jobs, where the jobs are subject to several constraints. They proved that minimizing the makespan and the total weighted completion time can be determined in polynomial time [3]. Cheng and Sun considered the problem with a linear deteriorating function [4] and showed that several related problems are NP-hard and used dynamic programming for solution [4]. Yan et al. studied a single machine scheduling problem with the effects of deteriorating and learning based on group consumption, in which the actual processing time of a job is a function of the starting time and position in the group [5]. Wang and Cheng addressed the machine scheduling problem with deterioration and learning effects simultaneously and gave polynomial solutions for single machine problems [6]. Ng et al. also studied the problem of scheduling n deteriorating jobs with release dates on a single machine [7]. These studies all focused on linear deteriorating jobs. A few papers refer to nonlinear deterioration jobs. Kuo and Yang introduced a single machine with a time-dependent learning effect based on the notion that the more the one practices, the better the one learns [8]. In this regard, jobs processed later need less time than that for normal processing due to the learning effect. They defined a time-dependent learning effect as

follows. Let p_{ir} be the actual processing time of J_i ($i = 1, 2, \ldots, n$) if it is scheduled in position r in a sequence. $a_{[r]}$ is the normal processing time of a job if scheduled in the rth position of a sequence. a_i is the normal (sequence-independent) processing time of job J_i. Namely,

$$p_{ir} = \left(1 + a_{[1]} + a_{[2]} + \cdots + a_{[r-1]}\right)^b a_i = \left(1 + \sum_{s=1}^{r-1} a_{[s]}\right)^b a_i, \quad (1)$$

where $b \leq 0$ and b is a constant learning index. According to the time-dependent learning effect introduced by Kuo and Yang [8], Wang et al. considered the single machine scheduling problem with a time-dependent deterioration [9]. They defined the actual processing times as follows:

$$p_{ir} = \left(1 + a_{[1]} + a_{[2]} + \cdots + a_{[r-1]}\right)^b a_i = \left(1 + \sum_{s=1}^{r-1} a_{[s]}\right)^b a_i, \quad (2)$$

where $b \geq 0$ is a constant deterioration index. They showed that a single machine problem can be solved polynomially under the proposed model. Bank et al. addressed two machine scheduling problems with deterioration effects in which the actual job processing time was a function of its starting time [10]. Sun et al. considered flow shop scheduling problems with deteriorating jobs in which the actual job processing time was defined as a function of its start time [11].

On the other hand, if the machine is stopped for maintenance, it can change from a subnormal production state to a normal one [12, 13]. Therefore, a scheduling model in a realistic environment should consider machine maintenance [12]. For example, in electronic assembly lines, Lee and Leon first considered the single machine scheduling problem with a rate-modifying activity (RMA) [13]. They solved problems with a number of objective functions by polynomial and pseudopolynomial algorithms. Later, Lodree et al. introduced human characteristics into scheduling models [14]. Motivated by the rate-modifying activity, Lodree and Christopher integrated a rate-modifying activity into machine environments [15] and assumed that the processing time of each job is 1 and followed a variation of simple linear deterioration.

The concept of deterioration effects and maintenance activities has been mentioned in a few publications in the literature. However, there are no research results on scheduling models with time-dependent deterioration jobs in which the normal processing job times are arbitrary and multi-RMAs. Hence, the main contribution of this paper was two aspects: one is for jobs with arbitrary normal processing time and nonlinear deterioration, and the other is for multi-RMAs.

The remaining sections of this paper are organized as follows. In Section 2, the problem is formulated. In Section 3, the branch and bound and the heuristic algorithm are proposed for solving single machine makespan minimization problem with an RMA, under nonlinear time-dependent deterioration, and a special case of this problem is given. In Section 4, similar ideas are used to solve the problem of single machine makespan minimization with multi-RMAs under a nonlinear time-dependent deterioration, and a special case is also given. The numerical experiments are described in Section 5, followed by the conclusions in the last section.

2. Problem Formulation and Notation

This paper studies the single machine scheduling problem with an RMA or multi-RMAs under a time-dependent deterioration to minimize the makespan. More formally, this problem can be described as follows.

Assume that there are n independent jobs $J = \{1, 2, \ldots, n\}$ to be processed nonpreemptively on a single machine which is available at time 0. The release times of all jobs are 0. The normal processing time of each job i ($i \in J$) is a_i. If it is scheduled in position r ($1 \leq r \leq n$) in a given sequence, then the normal processing time can be denoted as $a_{[r]}$. The deterioration rate b ($b > 0$) is a constant. The actual processing time of job i is p_{ir} if it is scheduled in position r ($1 \leq r \leq n$) in a given sequence. p_{ir} is a function of the normal processing times of all jobs before it; that is, $p_{ir} = (1 + a_{[1]} + a_{[2]} + \cdots + a_{[r-1]})^b a_i$. To decrease the deterioration effect, an RMA with duration t (i.e., maintenance time) needs to be considered and inserted in a certain position k_m ($m = 1, 2, \ldots, M, 1 \leq M \leq n, 1 \leq k_m \leq n$) in a sequence; namely, the RMA is assigned before the k_mth job (such as in Figure 1), where M denotes the number of RMAs inserted in the sequence. After the RMA, the machine can be fully restored [14], and the actual processing time of the first job after it is the normal processing time of the job. When multi-RMAs are inserted in a sequence, the actual processing time of a job i can be denoted as follows:

$$p_{ir} = \begin{cases} \left[1 + a_{[1]} + a_{[2]} + \cdots + a_{[r-1]}\right]^b a_i & r = 1, 2, \ldots, k_1 - 1 \\ \left[1 + a_{[k_1]} + a_{[k_1+1]} + \cdots + a_{[r-k_1]}\right]^b a_i & r = k_1, \ldots, k_2 - 1 \\ \vdots & \\ \left[1 + a_{[k_m]} + a_{[k_m+1]} + \cdots + a_{[r-k_m]}\right]^b a_i & r = k_m, \ldots, k_{m+1} \\ \vdots & \\ \left[1 + a_{[k_M]} + a_{[k_M+1]} + \cdots + a_{[r-k_M]}\right]^b a_i & r = k_M, \ldots, n. \end{cases} \quad (3)$$

The objective is to jointly find the number of RMAs and positions of each RMA and an optimal schedule S^* to minimize the makespan $C_{\max}$. Specifically, when $M = 1$, the objective is to jointly find a position k for inserting an RMA and the optimal schedule S^* to minimize the makespan $C_{\max}$.

In this study, we consider the problem of minimizing the makespan with an RMA or multi-RMAs on a single machine under a time-dependent deterioration. We denote them as $1|p_{ir}, \text{rm}|C_{\max}$ and $1|p_{ir}, \text{mrm}|C_{\max}$, respectively, by using the three-field notation scheme $\alpha|\beta|\gamma$ introduced by

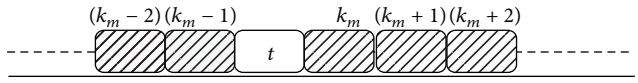

Figure 1: The position of maintenance times.

Graham et al. [16], where rm represents inserting an RMA and mrm denotes multi-RMAs for modifying the processing rate of the machine.

3. The Problem of $1|p_{ir}, \mathrm{rm}|C_{\max}$

Here, the single machine problem with assigning an RMA in a sequence under a time-dependent deterioration is considered. Firstly, several preliminaries are proposed in Section 3.1, followed by the branch and bound algorithm and heuristic algorithm in Sections 3.2 and 3.3, respectively. Finally, a special case is described.

3.1. Preliminaries. In this subsection, several properties and a lower bound are proposed for solving the problem of $1|p_{ir}, \mathrm{rm}|C_{\max}$.

For convenience, assume that $S = \{\pi, \mathrm{rm}, \pi'\}$ is a full schedule, in which π and π' are partial sequences, and set $a_{\max} = \max_{i=1}^{n}\{a_i\}$ and $a_{\min} = \max_{i=1}^{n}\{a_i\}$. Based on these values, the following properties and lemmas are proposed.

Property 1. It is never optimal to schedule an RMA in the first sequence position $k = 1$ (similar to Lodree et al. [14]).

Property 2. If an RMA is assigned in a given position k and the elements in π and π' are known, then there is an optimal schedule obtained by sequencing jobs in a nondecreasing order of a_i in π and π', respectively.

Proof. If an RMA is assigned in a given position k, the elements in π and π' are known, and the release times of all jobs are 0. Hence, minimizing makespan of the schedule S is equal to minimizing the makespan of π and π', respectively.

Firstly, we prove how to order jobs in π so as to minimize the makespan of π. Assume that $\pi = \{Q, i, j, Q'\}$ with job i in the rth position and job j in the $(r+1)$th position. The completion time of the last job in Q is t_Q, and the sum of the normal processing times of jobs in Q is P, and $0 < a_i < a_j$. Also $\overline{\pi} = \{Q, j, i, Q'\}$ is obtained from exchanging the position i and j in π.

According to the above statement, the completion time of job j in π is as follows:

$$\begin{aligned} C_j(\pi) &= t_Q + a_i(1+P)^b + a_j\left(1+P+a_{[r]}\right)^b \\ &= t_Q + a_i(1+P)^b + a_j\left(1+P+a_i\right)^b. \end{aligned} \tag{4}$$

The completion time of job i in $\overline{\pi}$ is:

$$\begin{aligned} C_i(\overline{\pi}) &= t_Q + a_j(1+P)^b + a_i\left(1+P+a_{[r]}\right)^b \\ &= t_Q + a_j(1+P)^b + a_i\left(1+P+a_j\right)^b. \end{aligned} \tag{5}$$

Set $\delta = 1 + P$, $\lambda = a_i/a_j$, and $\mu = a_j/\delta$, then

$$\begin{aligned} &C_j(\pi) - C_i(\overline{\pi}) \\ &\quad = \left(a_i - a_j\right)(1+P)^b + a_j\left(1+P+a_i\right)^b - a_i\left(1+P+a_j\right)^b \\ &\quad = \left(a_i - a_j\right)\delta^b + a_j\left(\delta + a_i\right)^b - a_i\left(\delta + a_j\right)^b. \end{aligned} \tag{6}$$

Since $\delta = 1 + P \neq 0$, the two sides of (6) are divided by δ^b, then

$$\begin{aligned} &\frac{C_j(\pi) - C_i(\overline{\pi})}{\delta^b} \\ &\quad = \left(a_i - a_j\right) + a_j\left(1 + \frac{a_i}{\delta}\right)^b - a_i\left(1 + \frac{a_j}{\delta}\right)^b \\ &\quad = \left(\lambda a_j - a_j\right) + a_j\left(1 + \frac{\lambda a_j}{\delta}\right)^b - \lambda a_j\left(1 + \frac{a_j}{\delta}\right)^b \\ &\quad = a_j(\lambda - 1) + a_j(1+\lambda\mu)^b - \lambda a_j(1+\mu)^b \\ &\quad = a_j\left[\lambda - 1 + (1+\lambda\mu)^b - \lambda(1+\mu)^b\right] \\ &\quad = a_j\left[\lambda\left(1 - (1+\mu)^b\right) - \left(1 - (1+\lambda\mu)^b\right)\right]. \end{aligned} \tag{7}$$

Set $f(\lambda) = \lambda(1 - (1+\mu)^b) - (1 - (1+\lambda\mu)^b)$, and the first derivative of $f(\lambda)$ is

$$f'(\lambda) = 1 - (1+\mu)^b - b\mu(1+\lambda\mu)^{b-1}. \tag{8}$$

Since $0 < a_i < a_j$ and $0 < \lambda < 1$, clearly $f'(\lambda) < 0$; that is, $f(\lambda)$ is a decreasing function for $\lambda \in (0, 1)$, then $f(\lambda) < f(0) = 0$.

Equation (7) can be expressed as

$$\frac{C_j(\pi) - C_i(\overline{\pi})}{\delta^b} = a_j f(\lambda) < 0. \tag{9}$$

If the two sides of (9) are multiplied by δ^b, then

$$C_j(\pi) - C_i(\overline{\pi}) < 0. \tag{10}$$

That is, $C_j(\pi) < C_i(\overline{\pi})$, so there is an optimal schedule obtained by sequencing jobs in nondecreasing order of a_i in π.

Similarly, we also can prove that there is an optimal schedule obtained by sequencing jobs in nondecreasing order of a_i in π'.

Therefore, if an RMA is assigned in a given position k, and the elements in π and π' are known, there is an optimal schedule obtained by sequencing jobs in nondecreasing order of a_i in π and π', respectively. □

Property 3. For a given schedule $S = \{\pi, \mathrm{rm}, \pi'\}$, and a new schedule $S' = \{\pi', \mathrm{rm}, \pi\}$ is obtained by changing π with π', $C_{\max}(S) = C_{\max}(S')$.

Proof. Since the job processing times after an RMA are not dependent on that of jobs before the RMA and the makespan is equal to the sum of the completion time of the last job in π and that of the last job in π', therefore the makespan has nothing to do with the order of π and π'. □

According to the above properties, the following two lemmas can be obtained.

Lemma 1. *If an RMA is scheduled in the second position (the last position) in a full schedule S and the normal processing time of the first job (the last job) in π (π') is not equal to $a_{\max}$, then there is an optimal solution S^* with which the normal processing time of the last job in π' (π) is $a_{\max}$.*

Lemma 2. *If an RMA is scheduled in the second position (the last position) in a full schedule S and the normal processing time of the first job (the last job) in π (π') is not equal to $a_{\min}$, then there is an optimal solution S^* with which the normal processing time of the first job in π' (π) is $a_{\min}$.*

The two lemmas can be easily determined from the above two properties; hence, their proofs are not included in the paper.

In the following, the lower bound is presented according to the completion time.

For the schedule S, when $r + 1 < k$, the completion time of the $(r + 1)$th job is

$$C_{[r+1]}(S) = C_{[r]}(S) + a_{[r+1]}\left(1 + a_{[1]} + a_{[2]} + \cdots + a_{[r]}\right)^b. \tag{11}$$

According to a similar deduction, when $r + l < k$, the completion time of the $(r + l)$th job is

$$\begin{aligned} C_{[r+l]}(S) &= C_{[r+l-1]}(S) + a_{[r+l]}\left(1 + a_{[1]} + a_{[2]} + \cdots + a_{[r+l-1]}\right)^b \\ &= C_{[r]}(S) + \sum_{i=1}^{l} a_{[r+i]}\left(1 + \sum_{j=1}^{r+i-1} a_{[j]}\right)^b \\ &\geq C_{[r]}(S) + \sum_{i=1}^{l} a_{[r+i]}. \end{aligned} \tag{12}$$

When $r+l = k$, the completion time of the kth job is $C_{[k]}(S) \geq C_{[r]}(S) + \sum_{i=1}^{k-r-1} a_{[r+i]} + t + a_{[k]}$.

When $r + l > k$, the completion time of the $(r + l)$th job is $C_{[r+l]}(S) \geq C_{[r]}(S) + \sum_{i=1}^{l} a_{[r+i]} + t$.

Similarly, when $r + l = n$, the completion time of the nth job is $C_{[n]}(S) \geq C_{[r]}(S) + \sum_{i=1}^{n-r} a_{[r+i]} + t$. That is, the makespan of schedule S is

$$C_{\max}(S) \geq C_{[r]}(S) + \sum_{i=1}^{n-r} a_{[r+i]} + t. \tag{13}$$

Therefore, the lower bound is

$$\mathrm{LB} = C_{[r]}(S) + \sum_{i=1}^{n-r} a_{[r+i]} + t. \tag{14}$$

3.2. The Branch and Bound Algorithm. The branch and bound algorithm mainly uses a backtracking method, which incorporates a system with jumping characteristics. The former adapts the depth first search strategy to start from a root node to the whole solution space. When the algorithm searches any node in the solution space tree, it needs to judge whether the subtree of the node as a root contains solutions of the problem or not. If not, it will jump over all the subtrees of the node as a root and then backtrack to its father node step by step. Otherwise, it continues to search for its subtrees. If a whole sequence is obtained and its objective value is less than the current one, then it will replace the current one. These reflect the method using jumping characteristics. Moreover, since the backtracking method only records a current sequence and its lower bound, it makes the storage space become small, to a great extent. In this paper, the depth first search and the lower bound are adopted in the branch and bound procedure, respectively. Dominance properties and the lower bound are used for eliminating a node which does not satisfy the solutions of the problem. The primary procedure of the branch and bound algorithm is described as follows.

Step 1 (the position of the RMA). Set the initial position of the RMA $k = 2$.

Step 2 (initial solution). Obtain an initial solution according to the short processing time rule.

Step 3 (branching). Search the whole solution space tree according to the depth first search strategy.

Step 4 (eliminating). Apply the properties and lemmas in Section 3.1 to eliminate the dominant partial sequences.

Step 5 (calculating). Calculate the lower bound for the partial sequences. If it is less than the current optimal solution, continue to search in its branches. When a whole sequence is obtained, replace the current optimal sequence with it. Otherwise, eliminate it, and go to Step 6.

Step 6 (backtracking). Backtrack to the father node of the current node and continue to search for other branches.

Step 7 (stopping). Repeat Steps 3 to 5 until no more nodes can be searched, then set $k = k + 1$, and go to Step 3.

Obtain the makespan C^*_{BR} of an optimal solution from the above steps. Then, in the case without scheduling an RMA, compute the makespan C^*_{NR} of a sequence by ordering the normal job processing time using the short processing time (SPT) rule, compare C^*_{NR} with C^*_R, and select the smallest value between of them.

3.3. A Heuristic Algorithm. Since the branch and bound algorithm takes a long time for a large size and cannot be accepted, a heuristic algorithm is proposed for obtaining the near-optimal solution to a problem. To understand easily, firstly, we give a heuristic algorithm for the problem $1|p_{ir}, \mathrm{rm}|C_{\max}$.

Based on the above Property 2 and Lemma 1, a heuristic algorithm is proposed. The main idea is to order the normal job processing time according to the SPT rule. Then, for each job with a current maximum normal processing time in set A, we try to determine where it is scheduled, in π or π'. The details of the heuristic algorithm are as follows. Assume that $S^* = \{\pi, \mathrm{rm}, \pi'\}$ and π and π' are empty.

Step 1. Obtain a sequence $A = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\}$ by ordering the normal processing time of jobs using the SPT rule.

Step 2. Select the job with the largest normal processing time $a_{\max} = \max_{a_i \in A}\{a_i\}$ added to π and eliminate $a_{\max}$ from set A.

Step 3. Select the job with the largest normal processing time $a_{\max} = \max_{a_i \in A}\{a_i\}$ adding to π' and eliminate $a_{\max}$ from the set A.

Step 4. Select the job with the largest normal processing time $a_{\max} = \max_{a_i \in A}\{a_i\}$ inserted before all jobs in π, eliminate $a_{\max}$ from the set A, and calculate the makespan $C'(S^*)$. Then, eliminate $a_{\max}$ from π, insert it before all jobs in π', and calculate the makespan $C''(S^*)$.

Step 5. Compare $C'(S^*)$ and $C''(S^*)$, and if $C'(S^*) < C''(S^*)$, eliminate $a_{\max}$ from π' and insert it before all jobs in π. Otherwise, go to Step 6.

Step 6. Repeat Steps 4 to 5 until $A = \phi$. At this time, the algorithm stops.

Obtain the makespan C^*_{HR} of an optimal solution from the above steps. Then, in the case without scheduling an RMA, compute the makespan C^*_{NR} of a sequence by ordering the normal job processing time using the short processing time (SPT) rule, compare C^*_{NR} with C^*_{HR}, and select the smallest one of them.

It is easy to see the normal job processing time ordered by the SPT rule in Step 1. The job with the largest normal processing time is assigned in π in Step 2. The job with the second largest normal processing time is assigned in π' in Step 3. In Steps 4 and 5, the job with the current largest normal processing time in set A is assigned in π or π' according to Property 2 and Lemma 1. The stopping condition of the algorithm is given in Step 6.

In order to better understand the details of the heuristic algorithm, an example is given.

Example 3. Consider that $n = 5$, $a_1 = 2$, $a_2 = 5$, $a_3 = 3$, $a_4 = 6$, and $a_5 = 1$. The deterioration rate is $b = 2$, and the duration of the RMA is $t = 2$.

For solving, we have the following.

(1) According to the SPT rule, obtain a sequence $A = \{5, 1, 3, 2, 4\}$, $\pi = \Phi$, and $\pi' = \Phi$. Go to Step 2.

(2) Job 4 with the largest normal processing time $a_{\max} = 6$, add it to π and eliminate it from the set A, then $\pi = \{4\}$, $A = \{5, 1, 3, 2\}$ and $\pi' = \Phi$. Go to Step 3.

(3) Job 2 with the largest normal processing time $a_{\max} = 5$; add it to π' and eliminate it from the set A; then $\pi = \{4\}$, $A = \{5, 1, 3\}$, and $\pi' = \{2\}$. Go to Step 4.

(4) Job 3 with the largest normal processing time $a_{\max} = 3$. Firstly, insert job 3 before all jobs in π; then $\pi = \{3, 4\}$, $A = \{5, 1\}$, and $\pi' = \{2\}$. Calculate the makespan $C' = 3 + 6(1 + 3)^2 + 2 + 5 = 106$. Secondly, eliminate job 3 from π, insert job 3 before all jobs in π', then $\pi = \{4\}$, $A = \{5, 1\}$ and $\pi' = \{3, 2\}$. Calculate the makespan $C'' = 6 + 2 + 3 + 5(1 + 3)^2 = 91$. Go to Step 5.

(5) Since $C' > C''$, then $\pi = \{4\}$, $A = \{5, 1\}$, and $\pi' = \{3, 2\}$. Go to Step 6.

(6) Since $A \neq \Phi$, go to Step 4.

(7) Job 1 with the largest normal processing time $a_{\max} = 2$. Firstly, insert job 1 before all jobs in π, then $\pi = \{1, 4\}$, $A = \{5\}$, and $\pi' = \{3, 2\}$. Calculate the makespan $C' = 141$. Secondly, eliminate the job 1 from π, insert job 1 before all jobs in π', then $\pi = \{4\}$, $A = \{5\}$, and $\pi' = \{1, 3, 2\}$. Calculate the makespan $C'' = 217$. Go to Step 5.

(8) Since $C' < C''$, eliminate job 1 from π', and insert it before all jobs in π again; then $\pi = \{1, 4\}$, $A = \{5\}$, and $\pi' = \{3, 2\}$. Go to Step 6.

(9) Since $A \neq \Phi$, go to Step 4.

(10) Job 5 is the only one in A. Firstly, insert job 5 before all jobs in π, then $\pi = \{5, 1, 4\}$, $A = \Phi$, and $\pi' = \{3, 2\}$. Calculate the makespan $C' = 190$. Secondly, eliminate job 5 from π, insert job 5 before all jobs in π', then $\pi = \{1, 4\}$, $A = \Phi$, and $\pi' = \{5, 3, 2\}$. Calculate the makespan $C'' = 196$. Go to Step 5.

(11) Since $C' < C''$, eliminate job 5 from π', and insert it before all jobs in π again, then $\pi = \{5, 1, 4\}$, $A = \Phi$ and $\pi' = \{3, 2\}$. Go to Step 6.

(12) Since $A = \Phi$, the algorithm stops, $C^*_{\mathrm{HR}} = C' = 190$. Compute the makespan $C^*_{\mathrm{NR}} = 1166$ of a sequence by ordering the normal job processing time using the short processing time (SPT) rule in the case without scheduling an RMA. Clearly, an optimal schedule is obtained from the heuristic algorithm.

3.4. The Special Case $1|p_{ir}, \mathrm{rm}, \mathrm{a_i} = \mathrm{a}|\mathrm{C_{max}}$. This subsection considers the special case $1|p_{ir}, \mathrm{rm}, a_i = a|C_{\max}$, where the normal processing time for all jobs is a.

In a given sequence, while an RMA is inserted in position k, the makespan can be expressed as follows:

$$C_{\max}(k) = a + a(1 + a)^b + \cdots + a(1 + (k - 2)a)^b + t + a + a(1 + a)^b + \cdots + a(1 + (n - k)a)^b. \quad (15)$$

Clearly, the makespan is related to the sequence position k. The problem can be solved by determining the value k to minimize (15). To determine the value k, we propose the following properties.

Property 4. (a) For an odd n and $1 < k \le n$, it has $C_{\max}((n+1)/2) = C_{\max}((n+1)/2+1)$; and if $k < (n+1)/2$, then $C_{\max}(k) > C_{\max}(k+1)$; if $k > (n+1)/2$, then $C_{\max}(k) < C_{\max}(k+1)$.

(b) For an even n and $1 < k \le n$, if $k < n/2$, then $C_{\max}(k) > C_{\max}(k+1)$; if $k > n/2+1$, then $C_{\max}(k) < C_{\max}(k+1)$.

Proof. (a) When the RMA is scheduled in the sequence position k and $k+1$, the makespan can be expressed as follows:

$$\begin{aligned} &C_{\max}(k) \\ &\quad = p + p(1+p)^b + \cdots + p(1+(k-2)p)^b \\ &\qquad + t + p + p(1+p)^b + \cdots + p(1+(n-k)p)^b, \\ &C_{\max}(k+1) \\ &\quad = p + p(1+p)^b + \cdots + p(1+(k-2)p)^b \\ &\qquad + p(1+(k-1)p)^b + t + p + p(1+p)^b + \cdots \\ &\qquad + p(1+(n-k-1)p)^b, \\ &C_{\max}(k) - C_{\max}(k+1) \\ &\quad = p\left[(1+(n-k)p)^b - (1+(k-1)p)^b\right], \\ &C_{\max}\left(\frac{n+1}{2}\right) - C_{\max}\left(\frac{n+1}{2}+1\right) = 0. \end{aligned} \tag{16}$$

That is, $C_{\max}((n+1)/2) = C_{\max}((n+1)/2+1)$.

If $k < (n+1)/2$ and $p > 0$, we have $(1+(n-k)p)^b > (1+(k-1)p)^b$, $C_{\max}(k) > C_{\max}(k+1)$.

If $k > (n+1)/2$ and $p > 0$, we have $(1+(n-k)p)^b < (1+(k-1)p)^b$, $C_{\max}(k) < C_{\max}(k+1)$.

The proof for (b) is analogous. □

Theorem 4. *The optimal policy for scheduling an RMA of length t under a time-dependent deterioration with $b \ge 0$ for all jobs is as follows. If n is an odd integer and $t < p\sum_{i=0}^{(n-1)/2}[(1+((n+1)/2-1+i)p)^b - (1+ip)^b]$, assign the RMA to sequence position $k^* = (n+1)/2$ or $k^* = (n+1)/2+1$. If n is an even integer and $t < p\sum_{i=0}^{n/2-1}[(1+(n/2+i)p)^b - (1+ip)^b]$, assign the RMA to sequence position $k^* = n/2+1$. Otherwise, do not schedule the RMA.*

Proof. (For an odd n). Based on Property 4, we have

$$\begin{aligned} &C_{\max}\left(\frac{n+1}{2}\right) < C_{\max}\left(\frac{n+1}{2}-1\right) < \cdots < C_{\max}(2), \\ &C_{\max}\left(\frac{n+1}{2}+1\right) < C_{\max}\left(\frac{n+1}{2}+2\right) < \cdots < C_{\max}(n). \end{aligned} \tag{17}$$

Since $C_{\max}((n+1)/2) = C_{\max}((n+1)/2+1)$, (17) imply that the minimum makespan occurs when $k^* = (n+1)/2$ or $k^* = (n+1)/2+1$.

Let $C_{\max}(k > n)$ represent the makespan without scheduling an RMA. Thus the RMA is scheduled only if $C_{\max}(k^*) < C_{\max}(k > n)$; that is,

$$\begin{aligned} &C_{\max}\left(\frac{n+1}{2}\right) \\ &\quad = p + p(1+p)^b + \cdots + p\left(1+\left(\frac{n+1}{2}-2\right)p\right)^b \\ &\qquad + t + p + p(1+p)^b + \cdots + p\left(1+\left(n-\frac{n+1}{2}\right)p\right)^b, \\ &C_{\max}(k > n) = p + p(1+p)^b + \cdots + p(1+(n-1)p)^b. \end{aligned} \tag{18}$$

Since $C_{\max}(k^*) < C_{\max}(k > n)$, then $t < p\sum_{i=0}^{(n-1)/2}[(1+((n+1)/2-1+i)p)^b - (1+ip)^b]$.

Again, an analogous proof holds if n is an even integer. This concludes the proof. □

4. The Problem of $1|p_{ir}, \mathrm{mrm}|C_{\max}$

In this section, the single machine problem assigning multi-RMAs in a sequence under time-dependent deterioration is considered. Firstly, several preliminaries are proposed in Section 4.1, followed by the branch and bound algorithm and a heuristic algorithm in Sections 4.2 and 4.3, respectively. Finally, a special case is shown.

4.1. Preliminaries. In this subsection, several properties and a lower bound are proposed for solving the problem of $1|p_{ir}, \mathrm{mrm}|C_{\max}$.

Property 5. If M RMAs are assigned in given positions, jobs are divided into $M+1$ groups, and the elements in each group are known, then there is an optimal schedule that can be obtained by sequencing jobs in nondecreasing order of a_i in each group.

The proof of Property 5 is similar to that of Property 2.

Property 6. For a given schedule $S = \{\pi_1, \mathrm{rm}_{k_1}, \pi_2, \mathrm{rm}_{k_2}, \ldots, \pi_m, \mathrm{rm}_{k_m}, \ldots, \mathrm{rm}_{k_M}, \pi_{M+1}\}$, the makespan remains equivalent by arbitrarily exchanging two groups.

The proof of Property 6 is similar to Property 3.

4.2. The Branch and Bound Algorithm. In solving the problem $1|p_{ir}, \mathrm{mrm}|C_{\max}$, the branch and bound algorithm needs to add an outside cycle and change properties. The primary procedure is described as follows.

Step 1. Set the number of RMAs $M = 1$.

Step 2 (the position of the RMA). Set the initial position of the RMA for $k = 2$.

Step 3 (initial solution). Obtain the initial solution according to the short processing time rule.

Step 4 (branching). Search the whole solution space tree according to the depth first search strategy.

Step 5 (eliminating). Apply properties in Section 4.1 to eliminate the dominant partial sequences.

Step 6 (calculating). Calculate the lower bound for the partial sequences. If it is less than the current optimal solution, continue to search in its branch. When a whole sequence is obtained, replace the current optimal sequence with it. Otherwise, eliminate it, and go to Step 7.

Step 7 (backtracking). Backtrack to the father node of the current node and continue to search other branches.

Step 8 (stopping). Repeat Steps 4 to 6 until no more nodes can be searched, and then set $k = k + 1$, and go to Step 4. Repeat the above steps until $k > n$, then set $M = M + 1$, and go to Step 4. Repeat the above steps until $M > n$. At this time, the algorithm stops.

Similarly, obtain the makespan C^*_{BR} of an optimal solution from the above steps. Then, in the case without scheduling an RMA, compute the makespan C^*_{NR} of a sequence by ordering the normal job processing time using the short processing time (SPT) rule, and compare C^*_{NR} with C^*_R, selecting the smallest one of them.

4.3. A Heuristic Algorithm. For the problem $1|p_{ir}, \mathrm{mrm}|C_{\max}$, the above heuristic algorithm is still efficient. It only needs to add M ($1 \le M \le n$) cycles, then assign jobs to m ($m = 1, 2, \ldots M$) groups according to the heuristic algorithm.

4.4. The Special Case $1|p_{ir}, \mathrm{mrm}, \mathrm{a_i} = \mathrm{a}|\mathrm{C_{max}}$. This subsection considers the special case $1|p_{ir}, \mathrm{mrm}, a_i = a|C_{\max}$, where the normal processing time for all jobs is a, and mrm denotes the multi-rate-modifying activities inserted in the sequence.

Property 7. If the number of multi-rate-modifying activities m is given, and jobs are divided into $m + 1$ groups, then there exists an optimal schedule in which each group includes $\lfloor n/m \rfloor$ or $\lfloor n/m \rfloor + 1$ jobs.

Proof. Assume that one of the groups includes the number of jobs greater than $\lfloor n/m \rfloor + 1$, then jobs are moved in other groups to decrease the makespan until each group includes $\lfloor n/m \rfloor$ or $\lfloor n/m \rfloor + 1$ jobs. □

Property 8. The time complexity for the problem $1|p_{ir}, \mathrm{mrm}, a_i = a|C_{\max}$ is $O(n^2)$.

Proof. For finding the optimal schedule with the determinate number m, we need to check the value of m from 1 to n, and calculate the makespan corresponding with each m. Then, we compare them and select the value of m corresponding to the minimal makespan. Moreover, the time complexity in calculating the makespan is $O(n)$. Hence, the time complexity of the problem $1|p_{ir}, \mathrm{mrm}, a_i = a|C_{\max}$ is $O(n^2)$. □

5. Numerical Experiments

In this section, the numerical experiment designs are as follows. The normal processing times of all jobs are generated from a uniform distribution over the integers between 1 and 100. The deterioration rate b takes the values of 0.05, 0.07, and 0.09. For the single machine scheduling problem with a rate-modifying activity, the size of job n takes the values of 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. For the single machine scheduling problem with multi-rate-modifying activities, the size of job n takes the values of 5, 7, 9, 11, 13, and 15, with the duration of the RMA $t = 30$. There are 50 $n - b$ combinations. Based on these, the CPU time and the solution performance of the branch and bound (B & B) and the heuristic algorithm (HA) are tested.

Two algorithms are used on the same personal computer with an Intel (R) Core (TM) 2 processor. The results of $1|p_{ir}, \mathrm{rm}|C_{\max}$ and $1|p_{ir}, \mathrm{mrm}|C_{\max}$ are recorded in Tables 1 and 2, respectively. In Table 1, it shows the optimal position of the RMA k from the heuristic algorithm and the error percentage of the heuristic algorithm relative to the optimal solution obtained from the branch and bound algorithm; that is, the error percentage is var = $(H - H^*)/H^* \times 100\%$, where H is a solution from the heuristic algorithm and H^* is the optimal solution from the branch and bound. It also gives the CPU time of the branch and bound, the optimal position of the RMA k, and the optimal solutions from the branch and bound. Since the CPU time of the heuristic algorithm for all sizes of jobs is less than 1 s, it is omitted. Difference between Tables 1 and 2 records the number of RMAs and their positions. Here, "—" denotes no position.

Observations from Table 1 are as follows.

(1) The branch and bound algorithm can obtain the optimal solution when the job size is less than or equal to 50, and the running time gradually increases with increase of the job size.

(2) For the heuristic algorithm, the maximum error percentage of the heuristic algorithm is no more than 0.053. For certain sizes of jobs, the error percentage of the heuristic algorithm slowly increases with deterioration rate increase. Moreover, according to Table 1, the mean error percentage var related to the job size for the heuristic algorithm is given in Figure 2. From Figure 2, it is seen that the mean error percentage decreases with increase of the job size, and the maximum mean error percentage is only 0.042.

Observations of the results from Table 2 are as follows.

(1) The branch and bound algorithm can obtain the optimal solution when the job size is less than or equal to 15, and the running time suddenly increases when the job size is 15.

(2) For the heuristic algorithm, the maximum error percentage of the heuristic algorithm is not more than 0.070. According to Table 2, the mean error percentage var related to the job size for the heuristic algorithm is shown in Figure 3. From Figure 3, it is seen that the mean error percentage still decreases

Table 1: Comparison of results from B & B; HA for $1|p_{ir}, \text{rm}|C_{\max}$.

		Branch and bound			Heuristic algorithm				Branch and bound			Heuristic algorithm	
n	b	Optimal solution	k	CPU time (s)	Var	k	n	b	Optimal solution	k	CPU time (s)	Var	k
	0.05	231.26	3	0	0.039	2		0.05	1737.96	15	66.782	0.023	28
5	0.07	319.34	4	0	0.037	2	30	0.07	2103.11	13	67.438	0.028	4
	0.09	483.31	3	0	0.051	4		0.09	3080.54	16	66.313	0.015	21
	0.05	739.34	6	0.200	0.029	2		0.05	2177.56	19	154.984	0.016	31
10	0.07	653.29	5	0.201	0.053	2	35	0.07	2652.92	22	153.266	0.014	28
	0.09	666.42	6	0.182	0.052	3		0.09	2790.96	18	153.484	0.020	8
	0.05	801.51	9	1.656	0.025	12		0.05	2362.75	26	318.391	0.018	37
15	0.07	918.13	10	1.656	0.035	14	40	0.07	2929.35	23	317.781	0.015	10
	0.09	916.67	5	1.657	0.041	14		0.09	2532.46	22	322.297	0.034	37
	0.05	1102.83	14	7.625	0.035	2		0.05	3074.90	27	611.469	0.015	7
20	0.07	1044.45	13	7.594	0.030	18	45	0.07	3082.73	24	610.953	0.023	40
	0.09	1381.71	11	7.594	0.036	4		0.09	3568.99	21	610.282	0.031	41
	0.05	1693.42	14	25.016	0.021	4		0.05	3337.76	25	1092.14	0.025	48
25	0.07	1530.00	17	25.063	0.020	21	50	0.07	3758.05	23	1091.81	0.020	8
	0.09	2169.22	17	25.047	0.034	4		0.09	4285.62	28	1089.73	0.015	13

Table 2: Comparison of results from B & B; HA for $1|p_{ir}, \text{mrm}|C_{\max}$.

n	b	Branch and bound				Heuristic algorithm		
		Optimal solution	M	Positions	CPU time (s)	Var	M	Positions
	0.05	253.11	1	2	0.031	0.011	0	—
5	0.07	255.50	1	3	0.031	0.055	0	—
	0.09	366.12	2	4, 5	0.031	0.058	1	5
	0.05	390.36	0	—	0.328	0.012	0	—
7	0.07	567.46	2	4, 5	0.344	0.037	3	5, 6, 7
	0.09	344.59	1	5	0.328	0.070	1	7
	0.05	637.21	1	7	2.672	0.025	0	—
9	0.07	451.64	1	3	2.172	0.042	1	9
	0.09	660.17	4	5, 6, 7, 8	2.719	0.049	4	6, 7, 8, 9
	0.05	559.14	1	5	26.563	0.031	0	—
11	0.07	858.35	3	6, 7, 9	31.500	0.032	5	7, 8, 9, 10, 11
	0.09	853.83	6	2, 4, 5, 6, 8, 9	31.078	0.025	6	6, 7, 8, 9, 10, 11
	0.05	826.54	1	11	486.547	0.034	1	13
13	0.07	826.22	1	10	359.484	0.048	3	11, 12, 13
	0.09	1051.81	8	2, 5, 6, 7, 8, 9, 10, 11	509.407	0.024	7	7, 8, 9, 10, 11, 12, 13
	0.05	867.21	1	9	6810.547	0.035	2	14, 15
15	0.07	1062.18	1	7	5164.200	0.013	0	—
	0.09	1117.52	6	4, 6, 8, 10, 13, 14	9923.344	0.017	7	9, 10, 11, 12, 13, 14, 15

with job size increase, and the maximum mean error percentage is only 0.041.

Based on the above analysis, the branch and bound algorithm can solve problems with RMA or multi-rate-modifying activity within a reasonable time. The heuristic algorithm can obtain near-optimal solutions for the problem in a very short time. Therefore, the two algorithms proposed in this paper are very efficient and effective for the single machine scheduling problem with rate-modifying activity under a time-dependent deterioration. At the same time, they can be also used as a reference for other problems with the rate-modifying activity.

6. Conclusion

This paper integrates a time-dependent deterioration considered as a nonlinear function multi-rate-modifying activities

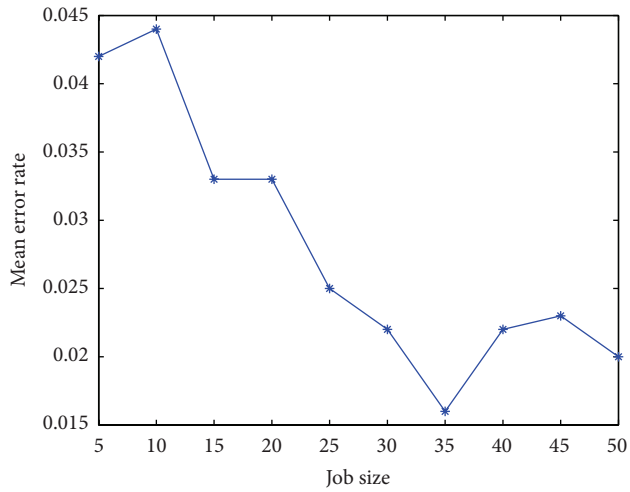

Figure 2: The mean error rate of the HA for $1|p_{ir}, \mathrm{rm}|C_{\max}$.

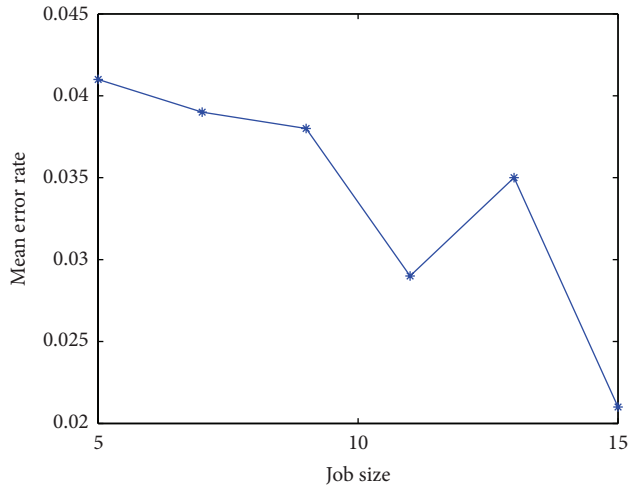

Figure 3: The mean error rate of the HA for $1|p_{ir}, \mathrm{mrm}|C_{\max}$.

into the single machine scheduling problem to minimize makespan. A branch and bound algorithm and a heuristic algorithm are proposed to solve such problems. At the same time, for special cases, the propositions, theorems, correlated proofs on the optimal policy of scheduling the RMA for minimal makespan are derived. Finally, the results of numerical experiments indicate that the branch and bound algorithm and the heuristic algorithm are efficient. In future, the research will be extended to single machine scheduling problems with release dates and due dates, which are more general cases in actuality.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 71071028, no. 71021061, no. 70931001, and No. 61070162, the National Science Foundation for Distinguished Young Scholars of China under Grant no. 61225012; the Specialized Research Fund of the Doctoral Program of Higher Education for the Priority Development Areas under Grant no. 20120042130003; Specialized Research Fund for the Doctoral Program of Higher Education under Grants no. 20110042110024 and no. 20100042110025; the Specialized Development Fund for the Internet of Things from the ministry of industry and information technology of the P.R. China; the Fundamental Research Funds for the Central Universities under Grants no. N110204003 and no. N120104001. This work was also supported by Department of ISE of The Hong Kong Polytechnic University (H-ZJE5).

References

[1] J. N. D. Gupta and S. K. Gupta, "Single facility scheduling with nonlinear processing times," *Computers and Industrial Engineering*, vol. 14, no. 4, pp. 387–393, 1988.

[2] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.

[3] J. B. Wang, C. T. Ng, and T. C. E. Cheng, "Single-machine scheduling with deteriorating jobs under a series-parallel graph constraint," *Computers & Operations Research*, vol. 35, no. 8, pp. 2684–2693, 2008.

[4] Y. Cheng and S. Sun, "Scheduling linear deteriorating jobs with rejection on a single machine," *European Journal of Operational Research*, vol. 194, no. 1, pp. 18–27, 2009.

[5] Y. Yan, D. Z. Wang, D.-W. Wang, W. H. Ip, and H.-F. Wang, "Single machine group scheduling problems with the effects of deterioration and learning," *Acta Automatica Sinica*, vol. 35, no. 10, pp. 1290–1295, 2009.

[6] J. B. Wang and T. C. E. Cheng, "Scheduling problems with the effects of deterioration and learning," *Asia-Pacific Journal of Operational Research*, vol. 24, no. 2, pp. 245–261, 2007.

[7] C. T. Ng, S. Li, T. C. E. Cheng, and J. Yuan, "Preemptive scheduling with simple linear deterioration on a single machine," *Theoretical Computer Science*, vol. 411, no. 40–42, pp. 3578–3586, 2010.

[8] W. H. Kuo and D. L. Yang, "Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect," *European Journal of Operational Research*, vol. 174, no. 2, pp. 1184–1190, 2006.

[9] J. B. Wang, L. Y. Wang, D. Wang, and X. Y. Wang, "Single-machine scheduling with a time-dependent deterioration," *International Journal of Advanced Manufacturing Technology*, vol. 43, no. 7-8, pp. 805–809, 2009.

[10] M. Bank, S. M. T. Fatemi Ghomi, F. Jolai, and J. Behnamian, "Two-machine flow shop total tardiness scheduling problem with deteriorating jobs," *Applied Mathematical Modelling*, vol. 36, no. 11, pp. 5418–5426, 2012.

[11] L. H. Sun, L. Y. Sun, M. Z. Wang, and J. B. Wang, "Flow shop makespan minimization scheduling with deteriorating jobs under dominating machine," *International Journal of Production Economics*, vol. 138, no. 1, pp. 195–200, 2012.

[12] C. Zhao and H. Tang, "Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan," *Applied Mathematical Modelling*, vol. 34, no. 3, pp. 837–841, 2010.

[13] C. Y. Lee and V. J. Leon, "Machine scheduling with a rate-modifying activity," *European Journal of Operational Research*, vol. 128, no. 1, pp. 119–128, 2001.

[14] E. J. Lodree, C. D. Geiger, and X. Jiang, "Taxonomy for integrating scheduling theory and human factors: review and research opportunities," *International Journal of Industrial Ergonomics*, vol. 39, no. 1, pp. 39–51, 2009.

[15] E. J. Lodree and D. G. Christopher, "A note on the optimal sequence position for a rate-modifying activity under simple linear deterioration," *European Journal of Operational Research*, vol. 201, no. 2, pp. 644–648, 2010.

[16] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.