This paper addresses a single machine scheduling problem in which the actual job processing times are determined by resource allocation function, its position in a sequence and a rate-modifying activity simultaneously. We discuss two objective functions with two resource allocation functions under the consideration of a rate-modifying activity. We show that the problems are solvable in $O(n^4)$ time for a linear resource allocation function and are solvable in $O(n^2 \log n)$ time for a convex resource allocation function.

1. Introduction

In realistic scheduling system, job processing times are usually affected by many practical settings. Learning effect of workers, different amount of resources allocated to jobs and when to schedule the rate-modifying activity may change the production rate due to which job processing times are variable.

Research involving human activities in production environment has received much attention in recent years and learning effect in the context of scheduling is one of the most important issues. The workers or processors obtain experience leading to the improvement of efficiency because of repeating similar or identical tasks. Such phenomenon is called learning effect. Biskup [1] was the first to discuss scheduling problems in a learning environment. He proposed an actual processing time formulation based on job scheduled positions which reflects the learning phenomenon and showed that the scheduling problems of minimizing the deviation from a common due date and minimizing the sum of flow times remain polynomially solvable. The well-known learning model can be expressed as follows: the actual processing time of job $j$ if it is scheduled in position $r$ in the sequence is $p_{jr} = p_j r^a$, where $p_j$ is the normal job processing time of job $j$ and $a$ is the negative learning index. Mosheiov [2] proposed polynomial-time solutions still exist for the makespan minimization problem, multi-criteria single-machine problems and the minimum flow-time problem on parallel identical machines with similar learning effect setting to Biskup [1]. Lee and Wu [3] studied two-machine flowshop with a learning effect problem for the objective of minimizing...
total completion time. Bachman and Janiak [4] considered scheduling jobs with position-dependent processing times in which they proved the makespan minimization problem is strong NP-hard for two different models of learning effect. They proposed Earliest Ready Date algorithms and showed that the makespan minimization problem with job ready times and maximum lateness minimization problems are equivalent. Cheng et al. [5] discussed some scheduling problems with learning effects in which the actual processing time of a job is dependent on the total normal processing times of the jobs already processed and of the jobs scheduled position. Koulamas [6] showed that the makespan minimization problem with job-dependent learning effects by Mosheiov and Sidney [7] can be solved in O(nlogn) time under some respective assumptions. Other recent related studies are Biskub [8], Wang [9], Eren [10], Toksar and Güner [11], Cheng et al. [12], Janiak and Rudek [13], Wang et al. [14], and so on.

In scheduling problems, the schedulers usually allocate finite amount of resource to a job to control its actual processing time. Many researchers focus on these kind of problems which are called scheduling problems with controllable processing problem since Vickson [15] initiated this field. Two different resource allocation functions were usually utilized in previous research. One is a linear function of the amount of resource associated to each job and the actual processing time under this setting can be defined as: \( p(i) = b_j u_j \), \( 0 \leq u_j \leq \frac{b_j}{p_j} \), where \( j = 1, 2, \ldots, n \). The other is a convex function of the amount of resource allocated to job \( j \). \( p(i) = b_j u_j \), where \( b_j > 0 \) is the compression rate of job \( j \). \( u_j \) is the amount of resource allocated to job \( j \). \( \tau_j \) is the maximal amount of resource that can be allocated to job \( j \). Janiak and Kovalyov [16] considered single machine scheduling problem in which each job has a deadline and a controllable processing time based on linear resource allocation function. Two cases based on whether the resource is continuously divisible or discrete are considered. Hoogeveen and Woeginger [17] studied sequencing problems with controllable processing time and showed several polynomial time results for the weighted total job completion time criterion and an NP-hardness result for the total weighted job completion time criterion. The other is a convex function of the amount of resource allocated to each job and the controllable job processing time can be written as: \( p(i) = \left( \frac{b_j}{p_j} \right)^k \), for \( j = 1, 2, \ldots, n \), where \( k > 0 \) is a constant. Monma et al. [18] were among the pioneers that utilized this convex function in resource allocation problem. Kaspi and Shabtay [19] studied a scheduling problem with convex resource allocation function and job release dates for minimizing the makespan. They provided two polynomial time algorithms for two different cases of release dates. Shabtay and Kaspi [20] studied a single scheduling to minimize the total weighted flow time with convex resource function. They proposed an exact dynamic programming algorithm for small or medium size problem and heuristic algorithms for large-scale problems. Shabtay and Steiner [21], Zhu et al. [22], Wang et al. [23] and Koulamas et al. [24] analyzed scheduling problems with these two resource allocation functions in their work. Zhu et al. [25] investigated two single-machine scheduling problems with limited resource and deteriorating jobs. They presented polynomial solutions for two objectives under different limits. For details on scheduling problems with controllable processing times, see the most recent survey by Shabtay and Steiner [26].

Different from the common assumption of classic maintenance in scheduling, a rate-modifying activity improves the production rate of a machine by changing the processing times of jobs following the activity. Lee and Leon [27] first investigated scheduling with the rate-modifying activity based on the practical phenomenon in electronic industry. However, relatively limited literature has involved this field although it is very important in practical industry. Lee and Lin [28] discussed scheduling problems with maintenance and repair rate-modifying activities. In their work, they assumed that machine breakdown is random, studied two types of processing cases, and provided some interesting results for several expected objective functions. He et al. [29] studied scheduling problem under consideration of a restricted rate-modifying activity. They analyzed the computational complexity and proposed pseudo-polynomial time optimal or fully polynomial time approximation algorithm for two objective functions. Mosheiov and Oron [30] and Gordon and Tarasevich [31] discussed scheduling problems with a rate-modifying activity in the context of a common due-date. Zhao et al. [32] investigated two parallel machines scheduling with rate-modifying activities and proposed efficient algorithms for two objective functions. Mosheiov and Sarig [33], Yang et al. [34] and Zhao and Tang [35] considered a rate-modifying activity in scheduling with a common due-window under different environment settings. Ji and Cheng [36] studied a scheduling problem with multiple rate-modifying activities. They allowed each machine to have multiple different rate-modifying activities and also introduced job-dependent learning effect into the problem. They provided polynomial solutions for the objective to minimize the total completion time. Lodree and Geiger [37] integrated time-dependent processing times and rate-modifying activity into the scheduling problem together. They pointed out that the specific position of the rate-modifying activity was in the middle of the task sequence.

However, to the best of our knowledge, all the existing literature has studied the settings mentioned above independently except that Wang et al. [23] first combined the effects of learning and resource allocation together. In this paper, we extend their model to include an additional rate-modifying activity which makes the problem under study more reasonable and realistic. The rest of this paper is organized as follows. In Section 2 the problem formulation is presented. Some preliminary results for further analysis are provided in Section 3. Our optimal analysis for both objective functions with learning effect and rate-modifying activity is presented in Section 4. The last section concludes this paper.

2. Problem formulation

The problem under the consideration of rate-modifying activity, learning effect, and resource allocation concurrently is described as follows. There are given a set \( J = \{J_1, J_2, \ldots, J_n \} \) of independent jobs to be processed on a single machine. Each
job is non-preemptive and available for processing at time 0. Associated with every job there is a normal processing time $p_j$. For any sequence $\pi = (J_1, J_2, \ldots, J_{|\pi|}), J_j$ denotes the job scheduled in position $r$, where $r = 1, 2, \ldots, n$. In addition, there is a rate-modifying activity the duration of which is $t$ on the single machine and no jobs are processed during the carrying out of the rate-modifying activity. Suppose the rate-modifying activity is in position $i$, if it is scheduled just after the completion of job in position $i$. The processing times of jobs scheduled after the rate-modifying activity will be changed. In this paper we discuss two different resource allocation functions including the linear and the convex function. For the linear one, we assume the actual processing time of job $j$ is $p_j^a = p_j(r)^a - b_j u_j$ if it is scheduled in position $r$ before the rate-modifying activity, otherwise, $p_j^a = \alpha_j p_j(r)^a - b_j u_j$, $\alpha_j$ is the modifying rate, where $0 < \alpha_j \leq 1$. $a$ is a learning index, $u_j$ denotes the amount of resource that can be allocated to job $j$, and $b_j$ is the compression rate of job $j$. For a job $j$, $u_j$ satisfies $0 \leq u_j \leq \bar{u}_j < \alpha_j \frac{p_j(r)^a}{a}$. For the convex case, we assume the actual processing time of job $j$ is $p_j^a = \left(\frac{p_j(r)^a}{a}\right)^k$ if it is scheduled in position $r$ before the rate-modifying activity, otherwise, $p_j^a = \left(\frac{\alpha_j p_j(r)^a}{a}\right)^k$. For any sequence $\pi$, $C_j = C_j(\pi)$ denotes the completion time of job $j$. As in Wang et al. [23], two cost functions discussed in this paper are $f(\pi, u) = \beta_1 C_{\max} + \beta_2 TC + \beta_3 TADC + \beta_4 \sum_{j=1}^{n} C_j u_j$ and $g(\pi, u) = \beta_1 C_{\max} + \beta_2 TW + \beta_3 TADW + \beta_4 \sum_{j=1}^{n} C_j u_j$, where $C_{\max} = \max\{C_j | j = 1, 2, \ldots, n\}$, $TC = \sum_{j=1}^{n} C_j$, $TW = \sum_{j=1}^{n} W_j$, $TADC = \sum_{j=1}^{n} \sum_{j'=1}^{n} C_{j'} - C_j$, and $TADW = \sum_{j=1}^{n} \sum_{j'=1}^{n} W_{j'} - W_j$ denote the makespan, the total completion times, total waiting times, the total absolute differences in completion times, and the total absolute differences in waiting times. $W_j = C_j - p_j^a$ denotes the waiting time of job $j$. The definitions of TADC and TADW refer to Bagchi [38]. $\beta_1$, $\beta_2$, $\beta_3$ and $\beta_4$ are positive parameters decided by the decision-makers. $C_{\max}$ is the unit time cost incurred by the resource allocation to job $j$.

Following the three-field notation of Graham et al. [39], we denote our problems as $1|\text{RALE}, \text{RM}|\beta_1 C_{\max} + \beta_2 \text{TC} + \beta_3 \text{TADC} + \beta_4 \sum_{j=1}^{n} C_j u_j$ and $1|\text{RALE}, \text{RM}|\beta_1 C_{\max} + \beta_2 \text{TW} + \beta_3 \text{TADW} + \beta_4 \sum_{j=1}^{n} C_j u_j$, where RALE means “resource allocation and learning effect” and RM means “rate-modifying activity”.

3. Preliminary results

In this section, we show some preliminary works for our further analysis of this problem. Based on the above notations and allowing to perform a rate-modifying activity in position $i$, the completion time of each job $j$, for $j = 1, 2, \ldots, n$ can be presented in the following:

For the linear case:
\[
C_{ji} = C_{j-1} + p_j(j)^a - b_j u_j, \quad j = 1, 2, \ldots, i,
\]
\[
C_{ji} = C_{j-1} + t + (\alpha_j p_j(j)^a - b_j u_j), \quad j = i + 1,
\]
\[
C_{ji} = C_{j-1} + (\alpha_j p_j(j)^a - b_j u_j), \quad j = i + 2, \ldots, n.
\]

For the convex case:
\[
C_{ji} = C_{j-1} + p_j(j)^a - b_j u_j, \quad j = 1, 2, \ldots, i,
\]
\[
C_{ji} = C_{j-1} + t + (\alpha_j p_j(j)^a - b_j u_j), \quad j = i + 1,
\]
\[
C_{ji} = C_{j-1} + (\alpha_j p_j(j)^a - b_j u_j), \quad j = i + 2, \ldots, n.
\]

where $C_{0i} = 0$.

The total completion time, the makespan, the total absolute differences in completion times, the total waiting time, and the total absolute differences in waiting times can be expressed as:

For the linear case:
\[
C_{\max} = \sum_{j=1}^{i_j} (p_j(j)^a - b_j u_j) + t + \sum_{j=i_{j+1}}^{n} (\alpha_j p_j(j)^a - b_j u_j),
\]
\[
TC = \sum_{j=1}^{n} C_j = \sum_{j=1}^{i_j} (n - j + 1) (p_j(j)^a - b_j u_j) + \sum_{j=i_{j+1}}^{n} (n - j + 1) (\alpha_j p_j(j)^a - b_j u_j) + (n - i_t) t,
\]
For the convex case:

\[ C_{\text{max}} = \sum_{j=1}^{i} (p_{j}(j)^{a}) \frac{k}{u_{j}} + t + \sum_{j=i+1}^{n} \left( \frac{x_{j}p_{j}(j)^{a}}{u_{j}} \right)^{k} \]

\[ T_{C} = \sum_{i=1}^{n} C_{i} = \sum_{j=1}^{n} (n-j+1) \left( \frac{p_{j}(j)^{a}}{u_{j}} \right)^{k} + \sum_{j=i+1}^{n} (n-j+1) \left( \frac{x_{j}p_{j}(j)^{a}}{u_{j}} \right)^{k} + (n-i)t \]

\[ T_{\text{ADC}} = \sum_{j=1}^{i} (j-1)(n-j+1)(p_{j}(j)^{a} - b_{j}u_{j}) + \sum_{j=i+1}^{n} (j-1)(n-j+1)(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) + i_{1}(n-i_{1})t \]

\[ T_{W} = \sum_{j=1}^{i} (n-j)(p_{j}(j)^{a} - b_{j}u_{j}) + \sum_{j=i+1}^{n} (n-j)(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) + (n-i_1)t \]

\[ T_{\text{ADW}} = \sum_{j=1}^{i} j(n-j)(p_{j}(j)^{a} - b_{j}u_{j}) + \sum_{j=i+1}^{n} j(n-j)(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) + i_{1}(n-i_{1})t \]

Besides, a useful lemma that can be applied in Section 4 to solve the problem is as follows.

**Lemma 1.** There are two vectors \( X \) and \( Y \), the elements of which are \( x_{i} \) and \( y_{i} \), respectively for \( i = 1, 2, \ldots, n \). If \( x_{i} \) and \( y_{i} \) are sorted in opposite order, \( \sum_{i=1}^{n} x_{i}y_{i} \) is the minimal value.

**Proof.** See the proof in page 261 by Hardy et al. [40]. \( \square \)

### 4. Optimal analysis for single-machine scheduling

In this section, we present optimal analysis for single machine scheduling problem with resource consumption functions and performing a rate-modifying activity. We discuss two total cost functions for each kind of resource consumption function. As in Section 2, we denote the problems under them as \( |R|AL|C_{\text{max}} + \beta_{2} T_{C} + \beta_{3} T_{\text{ADC}} + \beta_{4} \sum_{j=1}^{n} G_{j} u_{j} \) and \( |R|AL|C_{\text{max}} + \beta_{2} T_{W} + \beta_{3} T_{\text{ADW}} + \beta_{4} \sum_{j=1}^{n} G_{j} u_{j} \), respectively. The decisions to be made for each case of above problems include three parts: the resource assignment, the job sequencing and the position schedule of the rate-modifying activity.

#### 4.1. Linear resource consumption function case

In this subsection, we provide the optimal solutions for the problems under study with the linear resource consumption function. For the \( |R|AL|C_{\text{max}} + \beta_{2} T_{C} + \beta_{3} T_{\text{ADC}} + \beta_{4} \sum_{j=1}^{n} G_{j} u_{j} \) problem, considering the preliminary results, its cost function can be represented as:

\[
 f(\pi, u) = \beta_{1} C_{\text{max}} + \beta_{2} T_{C} + \beta_{3} T_{\text{ADC}} + \beta_{4} \sum_{j=1}^{n} G_{j} u_{j} = \beta_{1} \left( \sum_{j=1}^{i_{1}} (p_{j}(j)^{a} - b_{j}u_{j}) \right) + t + \sum_{j=i_{1}+1}^{n} \left( \frac{x_{j}p_{j}(j)^{a}}{u_{j}} \right) + \beta_{2} \left( \sum_{j=1}^{i_{1}} (n-j+1)(p_{j}(j)^{a} - b_{j}u_{j}) + \sum_{j=i_{1}+1}^{n} (n-j+1)(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) + (n-i_{1})t \right) + \beta_{3} \left( \sum_{j=1}^{i_{1}} (j-1)(n-j+1)(p_{j}(j)^{a} - b_{j}u_{j}) + \sum_{j=i_{1}+1}^{n} (j-1)(n-j+1)(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) + i_{1}(n-i_{1})t \right) + \beta_{4} \left( \sum_{j=1}^{i_{1}} (\beta_{1} + (n-j+1)\beta_{2} + (j-1)(n-j+1)\beta_{3})(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) \right) + \sum_{j=i_{1}+1}^{n} (\beta_{1} + (n-j+1)\beta_{2} + (j-1)(n-j+1)\beta_{3})(x_{j}p_{j}(j)^{a} - b_{j}u_{j}) + (\beta_{1} + \beta_{2}(n-i_{1}) + \beta_{3} i_{1}(n-i_{1})t + \sum_{j=1}^{n} G_{j} u_{j}) \right) \]

Let \( w_j = \beta_1 + (n - j + 1)\beta_2 + (j - 1)(n - j + 1)\beta_3 \),

\[
f(\pi, u) = \sum_{j=1}^{i_1} w_j p_j (j)^a + \sum_{j=i_1+1}^{n} w_j z_j p_j (j)^a + \sum_{j=1}^{n} (\beta_4 G_j - w_j b_j) u_{j} + (\beta_1 + \beta_2 (n - i_1) + \beta_3 i_1 (n - i_1)) t.
\] (1)

We discuss the determination of the resource allocation of each job \( j \) first. From above analysis, for given position of \( i_1 \) and job sequence, the amount \( (u_j) \) of resource allocated to each job \( j \) depends on the coefficient \( \beta_4 G_j - w_j b_j \). If \( \beta_4 G_j - w_j b_j \) is negative, considering the objective minimizing the cost function, the optimal amount of resource allocated should be \( \bar{u}_j \), the upper bound of the amount of the resource. Similarly, if the coefficient \( \beta_4 G_j - w_j b_j \) is positive, the optimal amount of resource allocated to job \( j \) should be 0. If \( \beta_4 G_j - w_j b_j \) is equal to 0, any value between 0 and \( \bar{u}_j \) does not affect the cost function. So for any job \( j \) of a given sequence, the optimal resource allocation is expressed in the following.

\[
u_j = \begin{cases} 
\bar{u}_j, & \text{if } \beta_4 G_j - w_j b_j < 0, \\
\bar{u}_j & \text{if } \beta_4 G_j - w_j b_j = 0, \\
0 & \text{if } \beta_4 G_j - w_j b_j > 0.
\end{cases}
\] (2)

\( u_j \) denotes the optimal amount of the resource allocated to job \( j \), for \( j = 1, 2, \ldots, n \).

Then we show the job sequence decision method. Set \( x_{jr} = 1 \) if job \( j \) is scheduled in position \( r \), otherwise \( x_{jr} = 0 \), where \( r = 1, 2, \ldots, n \).

Let

\[
B_{jr} = \begin{cases} 
w_j p_j (j)^a, & \text{if } \beta_4 G_j - w_j b_j \geq 0, r = 1, \ldots, i_1, \\
w_j p_j (j)^a + (\beta_4 G_j - w_j b_j) \bar{u}_j, & \text{if } \beta_4 G_j - w_j b_j < 0, r = 1, \ldots, i_1, \\
w_j z_j p_j (j)^a, & \text{if } \beta_4 G_j - w_j b_j \geq 0, r = i_1 + 1, \ldots, n, \\
w_j z_j p_j (j)^a + (\beta_4 G_j - w_j b_j) \bar{u}_j, & \text{if } \beta_4 G_j - w_j b_j < 0, r = i_1 + 1, \ldots, n.
\end{cases}
\] (5)

We can formulate the problem as the following binary integer programming problem:

\[
\min \sum_{j=1}^{n} \sum_{r=1}^{n} B_{jr} x_{jr} + (\beta_1 + \beta_2 (n - i_1) + \beta_3 i_1 (n - i_1)) t
\]

subject to

\[
\sum_{j=1}^{n} x_{jr} = 1, \quad j = 1, 2, \ldots, n,
\]

\[
\sum_{j=1}^{n} x_{jr} = 1, \quad r = 1, 2, \ldots, n,
\]

\[
x_{jr} \leq 1 \quad \text{or} \quad 0, \quad j = 1, 2, \ldots, n, \quad r = 1, 2, \ldots, n.
\]

The first set of constraints guarantees each job is scheduled once and the second set of constraints guarantees each position is taken by only one job. The third set of constraints means \( x_{jr} \) is a binary variable.

For given position \( i_1 \), the above problem is equivalent to minimize the following assignment problem.

\[
\text{(AP)} \min \sum_{j=1}^{n} \sum_{r=1}^{n} B_{jr} x_{jr}
\]

subject to

\[
\sum_{j=1}^{n} x_{jr} = 1, \quad j = 1, 2, \ldots, n,
\]

\[
\sum_{j=1}^{n} x_{jr} = 1, \quad r = 1, 2, \ldots, n,
\]

\[
x_{jr} \leq 1 \quad \text{or} \quad 0, \quad j = 1, 2, \ldots, n, \quad r = 1, 2, \ldots, n.
\]
From above analysis, we propose the following polynomial time algorithm to solve optimally the $1|\text{RALE, RM}|\beta_1C_{\max} + \beta_2TC + \beta_3TAD + \beta_4\sum_{j=1}^{n}G_ju_j$ problem.

**Algorithm 1.**

1. Set $i_1 = 1$.
2. Calculate the weight $B_p$ with (5)–(8).
3. Solve the assignment problem (AP) to obtain the local optimal sequence ($\pi'$) and the total cost.
4. If $i_1 \leq n$, then go to Step 2. Otherwise go to Step 5.
5. The global optimal sequence is the one with the minimum total cost, we denote the optimal sequence as $\pi^*$ and the optimal position of rate-modifying activity as $i_1$.
6. Calculate the optimal resources by (2)–(4) and calculate the actual processing time $p^*_j$ to obtain the objective value.

As we all know that this classical assignment problem can be solved with $O(n^3)$ (see [41,42], other related studies adopted this result include [1,36,43], etc.). Furthermore, to obtain the global schedule, since the position of rate-modifying activity is variable and may be 1, 2, ..., $n$, the complexity of the studied problem under linear resource function is $O(n^4)$ and the following theorem holds.

**Theorem 1.** The $1|\text{RALE, RM}|\beta_1C_{\max} + \beta_2TC + \beta_3TAD + \beta_4\sum_{j=1}^{n}G_ju_j$ problem under linear resource function can be solved in $O(n^4)$ time.

Then we analyze the $1|\text{RALE, RM}|\beta_1C_{\max} + \beta_2TW + \beta_3TADW + \beta_4\sum_{j=1}^{n}G_ju_j$ problem. The cost function can be represented as:

$$g(\pi, u) = \beta_1 C_{\max} + \beta_2 TW + \beta_3 TADW + \beta_4 \sum_{j=1}^{n} G_j u_j$$

Let $\varphi_j = \beta_1 + (n-j)\beta_2 + j(n-j)\beta_3$.

$$g(\pi, u) = \sum_{j=1}^{n_i} \varphi_j p_j(j)^a + \sum_{j=i+1}^{n} \varphi_j x_j p_j(j)^a + \sum_{j=1}^{n} (\beta_4 G_j - \varphi_j b_j) u_j + (\beta_1 + \beta_2(n-i_1) + \beta_3 i_1(n-i_1)) t. \quad (9)$$

Similar to the above analysis, the optimal resource allocation can be obtained from the following formulation.

$$u^*_j = \begin{cases} 
\hat{u}_j, & \text{if } \beta_4 G_j - \varphi_j b_j < 0, \\
[0, u_j], & \text{if } \beta_4 G_j - \varphi_j b_j = 0, \\
0, & \text{if } \beta_4 G_j - \varphi_j b_j > 0. 
\end{cases}$$

Set $x_r = 1$ if job $j$ is scheduled in position $r$, otherwise $x_r = 0$, where $r = 1, 2, \ldots, n$.

$$H_r = \begin{cases} 
\varphi_j p_r(r)^a, & \text{if } \beta_4 G_{j_r} - \varphi_j b_{j_r} \geq 0, \quad r = 1, \ldots, i_1, \\
\varphi_j p_r(r)^a + (\beta_4 G_{j_r} - \varphi_j b_{j_r}) \hat{u}_r, & \text{if } \beta_4 G_{j_r} - \varphi_j b_{j_r} < 0, \quad r = 1, \ldots, i_1, \\
\varphi_j x_j p_r(r)^a, & \text{if } \beta_4 G_{j_r} - \varphi_j b_{j_r} \geq 0, \quad r = i_1 + 1, \ldots, n, \\
\varphi_j x_j p_r(r)^a + (\beta_4 G_{j_r} - \varphi_j b_{j_r}) \hat{u}_r, & \text{if } \beta_4 G_{j_r} - \varphi_j b_{j_r} < 0, \quad r = i_1 + 1, \ldots, n. 
\end{cases}$$
We can formulate the problem as the following binary integer programming problem:

\[
\min \sum_{j=1}^{n} \sum_{r=1}^{n} H_{j} x_{jr} + (\beta_1 + \beta_2 (n - i_1) + \beta_3 i_1 (n - i_1)) t
\]

subject to

\[
\sum_{r=1}^{n} x_{jr} = 1, \quad j = 1, 2, \ldots, n,
\]

\[
\sum_{j=1}^{n} x_{jr} = 1, \quad r = 1, 2, \ldots, n,
\]

\[
x_{jr} = 1 \quad \text{or} \quad 0, \quad j = 1, 2, \ldots, n, \quad r = 1, 2, \ldots, n.
\]

For given \(i_1\), the above problem can be transferred to minimize the following assignment problem.

\[
\min \sum_{j=1}^{n} \sum_{r=1}^{n} H_{j} x_{jr}
\]

subject to

\[
\sum_{r=1}^{n} x_{jr} = 1, \quad j = 1, 2, \ldots, n,
\]

\[
\sum_{j=1}^{n} x_{jr} = 1, \quad r = 1, 2, \ldots, n,
\]

\[
x_{jr} = 1 \quad \text{or} \quad 0, \quad j = 1, 2, \ldots, n, \quad r = 1, 2, \ldots, n.
\]

The algorithm for the \(1|\text{RALE, RM}|\beta_1 C_{\text{max}} + \beta_2 T \| + \beta_3 \text{TADW} + \beta_4 \sum_{j=1}^{n} G_{j} u_{j}\) problem is similar to the Algorithm 1, and its complexity is \(O(n^4)\).

**Theorem 2.** The \(1|\text{RALE, RM}|\beta_1 C_{\text{max}} + \beta_2 T \| + \beta_3 \text{TADW} + \beta_4 \sum_{j=1}^{n} G_{j} u_{j}\) problem under linear resource function can be solved in \(O(n^4)\) time.

**Special case:**

Now we discuss the following special case: we assume \(\beta_1 = \beta_2 = \beta_4 = G_j = a = b_j = 0\). As above analysis, its cost function can be represented as

\[
f(\pi, u) = \sum_{j=1}^{i_1} w_j p_{j}^a + \sum_{j=i_1+1}^{n} w_j x_{j} p_{j}^a + \sum_{j=1}^{n} (\beta_4 G_{j} - w_j b_{j}) u_{j} + (\beta_1 + \beta_2 (n - i_1) + \beta_3 i_1 (n - i_1)) t,
\]

where \(w_j = \beta_1 + (n - j + 1)\beta_2 + (j - 1)(n - j + 1)\beta_3\).

Considering the assumption that \(\beta_1 = \beta_2 = \beta_4 = G_j = a = b_j = 0\),

\[
w_j = (n - j + 1)\beta_2, \quad \text{and} \quad f(\pi, u) = \beta_2 \sum_{j=1}^{i_1} (n - j + 1) p_{j} + \beta_2 \sum_{j=i_1+1}^{n} (n - j + 1) x_{j} p_{j} + \beta_2 (n - i_1)
\]

\[= \beta_2 \left( \sum_{j=1}^{i_1} (n - j + 1) p_{j} + \sum_{j=i_1+1}^{n} (n - j + 1) x_{j} p_{j} + (n - i_1) \right).
\]

It is easy find that this special case is just equivalent to one of Lee and Leon [27]'s case of \(1/rm/ \Sigma C_j\) (please refer to page 122–123 in their paper for details), the complexity is also \(O(n^4)\).

**4.2. Convex resource consumption function case**

In this part, we provide the optimal solutions for the problems under study with the convex resource consumption function. For the \(1|\text{RALE, RM}|\beta_1 C_{\text{max}} + \beta_2 T\| + \beta_3 \text{TADC} + \beta_4 \sum_{j=1}^{n} G_{j} u_{j}\) problem, the:

\[
f(\pi, u) = \beta_1 C_{\text{max}} + \beta_2 T\| + \beta_3 \text{TADC} + \beta_4 \sum_{j=1}^{n} G_{j} u_{j}
\]

\[= \sum_{j=1}^{i_1} w_j \left( \frac{p_{j}^a}{u_{j}} \right)^k + \sum_{j=i_1+1}^{n} w_j \left( \frac{x_{j} p_{j}^a}{u_{j}} \right)^k + \sum_{j=1}^{n} (\beta_4 G_{j} - w_j b_{j}) u_{j} + (\beta_1 + \beta_2 (n - i_1) + \beta_3 i_1 (n - i_1)) t,
\]

where \(w_j = \beta_1 + (n - j + 1)\beta_2 + (j - 1)(n - j + 1)\beta_3\).
For the $1|\text{RALE, RM}|\beta_1C_{\text{max}} + \beta_2TW + \beta_3TADW + \beta_4 \sum_{j=1}^{n} G_j u_j$ problem, the cost function is expressed as:

$$ g(\pi, u) = \beta_1C_{\text{max}} + \beta_2TW + \beta_3TADW + \beta_4 \sum_{j=1}^{n} G_j u_j $$

$$ = \sum_{j=1}^{i} \varphi_j \left( \frac{p_j(j)^{\alpha}}{u_j} \right)^k + \sum_{j=i+1}^{n} \varphi_j \left( \frac{x_j p_j(j)^{\alpha}}{u_j} \right)^k + \sum_{j=1}^{n} (\beta_4 G_j) u_j + (\beta_1 + \beta_2(n - i_1) + \beta_3 i_1(n - i_1)) t, $$

where $\varphi_j = \beta_1 + (n - j)\beta_2 + j(n - j)\beta_3$.

Following the analysis process of the linear case, we calculate the optimal resource allocation which is a function of a sequence of jobs first.

**Lemma 2.** For any specified sequence $\pi$, there exists an optimal resource allocation for the cost function $f(\pi, u) = \beta_1C_{\text{max}} + \beta_2TC + \beta_3TADC + \beta_4 \sum_{j=1}^{n} G_j u_j$, which is:

$$ u^*_j = \begin{cases} \left( \frac{k w_j (p_j(j)^{\alpha})^{1/k}}{\beta_4 G_j} \right)^{1/k}, & j = 1, 2, \ldots, i_1, \\ \left( \frac{k w_j (p_j(j)^{\alpha})^{1/k}}{\beta_4 G_j} \right)^{1/k} \left( \frac{1}{u_j} \right)^{k-1} x_j^{1/k}, & j = i_1 + 1, i_1 + 2, \ldots, n. \end{cases} $$

For the cost function $g(\pi, u) = \beta_1C_{\text{max}} + \beta_2TW + \beta_3TADW + \beta_4 \sum_{j=1}^{n} G_j u_j$, which is:

$$ u^*_j = \begin{cases} \left( \frac{k w_j (p_j(j)^{\alpha})^{1/k}}{\beta_4 G_j} \right)^{1/k}, & j = 1, 2, \ldots, i_1, \\ \left( \frac{k w_j (p_j(j)^{\alpha})^{1/k}}{\beta_4 G_j} \right)^{1/k} \left( \frac{1}{u_j} \right)^{k-1} x_j^{1/k}, & j = i_1 + 1, i_1 + 2, \ldots, n. \end{cases} $$

**Proof.** For the first cost function $f(\pi, u)$,

When $j = 1, 2, \ldots, i_1$, \( \frac{\partial f(\pi, u)}{\partial u_j} = \frac{\partial}{\partial u_j} \left( w_j \left( \frac{p_j(j)^{\alpha}}{u_j} \right)^k + \beta_4 G_j u_j \right) = \beta_4 G_j u_j - k w_j (p_j(j)^{\alpha})^{1/k} \left( \frac{1}{u_j} \right)^{k-1}, \)

Let \( \frac{\partial f(\pi, u)}{\partial u_j} = 0 \), we obtain $u^*_j = \left( \frac{k w_j (p_j(j)^{\alpha})^{1/k}}{\beta_4 G_j} \right)^{1/k}$.

When $j = i_1 + 1, i_1 + 2, \ldots, n$, \( \frac{\partial f(\pi, u)}{\partial u_j} = \frac{\partial}{\partial u_j} \left( w_j \left( \frac{x_j p_j(j)^{\alpha}}{u_j} \right)^k + \beta_4 G_j u_j \right) = \beta_4 G_j u_j - k w_j (x_j p_j(j)^{\alpha})^{1/k} \left( \frac{1}{u_j} \right)^{k-1}, \)

Let \( \frac{\partial f(\pi, u)}{\partial u_j} = 0 \), we obtain $u^*_j = \left( \frac{k w_j (p_j(j)^{\alpha})^{1/k}}{\beta_4 G_j} \right)^{1/k} x_j^{1/k}$.

The proof for the second cost function $g(\pi, u)$ is similar. \( \square \)

Substituting the optimal $u^*(\pi)$ to both of the cost functions, we get the following new expression:

$$ \hat{f}(\pi, u^*(\pi)) = \left( k^{1/k} + k^{1/k} \right) (\beta_4^{1/k}) \sum_{j=1}^{n} \rho_j \mu_j + (\beta_1 + \beta_2(n - i_1) + \beta_3 i_1(n - i_1)) t, $$

where,

$$ \rho_j = \begin{cases} (G_j p_j(j)^{\alpha})^{1/k}, & j = 1, 2, \ldots, i_1, \\ (x_j G_j p_j(j)^{\alpha})^{1/k}, & j = i_1 + 1, i_1 + 2, \ldots, n. \end{cases} $$

$$ \mu_j = \begin{cases} (w_j(j)^{\alpha})^{1/k}, & \text{for the first cost function.} \\ (\varphi(j)^{\alpha})^{1/k}, & \text{for the second cost function.} \end{cases} $$

For any given position of rate-modifying activity $i_1$, to minimize the problem expressed as in (21) is equivalent to minimizing the matching problem stated in Lemma 1. So based on the consideration of the rate-modifying activity and Lemma 1, we propose Algorithm 2 for The $1|\text{RALE, RM}|\beta_1C_{\text{max}} + \beta_2TC + \beta_3TADC + \beta_4 \sum_{j=1}^{n} G_j u_j$ problem.
Algorithm 2.

Step 1: Set $i_1 = 1$.
Step 2: Calculate $p_j$ and $\mu_j$ with (22)–(24).
Step 3: Obtain the local optimal job sequence with Lemma 1 and the total cost.
Step 4: $i_1 = i_1 + 1$. If $i_1 \leq n$, then go to Step 2. Otherwise go to step 5.
Step 5: The global optimal sequence is the one with the minimum total cost, we denote the optimal sequence as $\pi^*$ and the optimal position of rate-modifying activity as $i^*$.
Step 6: Calculate the optimal resources by (17) and (18) calculate the actual processing time to obtain the objective value.

Theorem 3. The 1|RALE, RME|$\beta_1 C_{\max} + \beta_2 TC + \beta_3 TADC + \beta_4 \sum_{j=1}^{n} C_j u_j$ problem under convex resource function can be solved in $O(n^2 \log n)$ time.

Proof. For given position of rate-modifying activity $i_1$, the problem with function $f(\pi, u(\pi^*))$ can be solved within $O(n \log n)$ by Lemma 2. In addition, the position $i_1$ may be $1, 2, \ldots, n$, so the overall complexity of Algorithm 2 for 1|RALE, RME|$\beta_1 C_{\max} + \beta_2 TC + \beta_3 TADC + \beta_4 \sum_{j=1}^{n} C_j u_j$ is $O(n^2 \log n)$. □

Theorem 4. The 1|RALE, RME|$\beta_1 C_{\max} + \beta_2 TW + \beta_3 TADW + \beta_4 \sum_{j=1}^{n} C_j u_j$ problem under convex resource function can be solved in $O(n^2 \log n)$ time.

Proof. The proof and the algorithm are similar to Theorem 3 and algorithm 2, respectively. □

5. Conclusions

The resource allocation, learning effect and the option for performing a rate-modifying activity have been investigated independently in recent years. The single machine scheduling problem with learning effect and resource allocation has been proved that it can be solved in $O(n^3)$ and $O(n \log n)$ for different resource allocation functions in the literature. In this paper we extend the setting by allowing to perform a rate-modifying activity. We prove that the problem can be solved in $O(n^4)$ and $O(n \log n)$ for different resource allocation function respectively. In future work, we will incorporate more realistic settings such as multiple rate-modifying activities, due-window assignment, and so on into the scheduling system.

Acknowledgment

This research was partially supported by the National Science Foundation of China under Grants 71201085, and 71101106.

References