Some Single-Machine Scheduling Problems with Learning Effects and Two Competing Agents

Hongjie Li, Zeyuan Li, and Yunqiang Yin

1 Mathematics Department, Zhoukou Normal University, Zhoukou 466001, China
2 College of Mechanical and Electrical, Harbin Institute of Technology, Harbin 150001, China
3 College of Sciences, East China Institute of Technology, Nanchang 330013, China
4 Faculty of Science, Kunming University of Science and Technology, Kunming 650093, China

Correspondence should be addressed to Yunqiang Yin; yunqiangyin@gmail.com

Received 29 April 2014; Accepted 10 May 2014; Published 25 May 2014

Academic Editor: Dar-Li Yang

Copyright © 2014 Hongjie Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This study considers a scheduling environment in which there are two agents and a set of jobs, each of which belongs to one of the two agents and its actual processing time is defined as a decreasing linear function of its starting time. Each of the two agents competes to process its respective jobs on a single machine and has its own scheduling objective to optimize. The objective is to assign the jobs so that the resulting schedule performs well with respect to the objectives of both agents. The objective functions addressed in this study include the maximum cost, the total weighted completion time, and the discounted total weighted completion time. We investigate three problems arising from different combinations of the objectives of the two agents. The computational complexity of the problems is discussed and solution algorithms where possible are presented.

1. Introduction

In traditional scheduling research, it is commonly assumed that the processing times of the jobs remain unchanged throughout the scheduling horizon. However, under certain circumstances, the job processing times may become short due to learning effects in the production environment. For example, Biskup [1] points out that the repeated processing of similar tasks will improve workers’ efficiency; that is, it takes workers shorter times to process setups, operate machines or software, or handle raw materials and components. In such an environment, a job scheduled later will consume less time than the same job when scheduled earlier. Jobs in such a setting are said to be under the “learning effect” in the literature.

Biskup [1] and Cheng and Wang [2] first introduce the idea of learning into the field of scheduling independently. Since then, a large body of literature on scheduling with learning effects has emerged. Examples of such studies are Mosheiov [3], Mosheiov and Sidney [4], Bachman and Janiak [5], Janiak and Rudek [6], Wang [7], and Yin et al. [8]. Biskup [9] provides a comprehensive review of research on scheduling with learning effects. For more recent studies in this line of research, the reader is referred to Jiang et al. [10,11], Yang [12], S.-J. Yang and D.-L. Yang [13], Wang et al. [14], Wu et al. [15], Xu et al. [16], and Yin et al. [8].

All the above papers consider the traditional case with a single agent. In recent years scheduling researchers have increasingly considered the setting of multiple competing agents, in which multiple agents need to process their own sets of jobs, competing for the use of a common resource and each agent has its own objective to optimize. However, there is little scheduling research in the multiagent setting in which the jobs are under learning effects. Liu et al. [17] study two models with two agents and position-dependent processing times. They assume that the actual processing time of job \(J_j\) is \(p_j + br\) in the aging-effect model, while the actual processing time of \(J_j\) is \(p_j - br\) in the learning-effect model, where \(r\) represents the processed position of \(J_j\) and \(b > 0\) denotes the aging or learning index. Ho et al. [18] define the actual processing time of job \(J_j\) as \(p_j = a_j(1 - kr)\) if it is processed at time \(t\), where \(a_j\) denotes the normal processing time of job \(J_j\).
and \( k \geq 0 \) represents a constant such that \( k(t_0 + \sum_{j=1}^{n} a_j - a_{\min}) < 1 \) with \( a_{\min} = \min_{j=1,2,...,n} \{ a_j \} \). Inspired by Ho et al. [18], Yin et al. [19] consider some two-agent scheduling problems under the learning effect model proposed in Ho et al. [18], in which the objective functions for agent \( A \) include the maximum earliness cost, the total earliness cost, and the total weighted earliness cost, and the objective function for agent \( B \) is always the same, that is, maximum earliness cost, and the objective is to minimize the objective function of agent \( A \) while keeping the objective function of agent \( B \) not greater than a given level. Similar models have been further studied by Wang and Xia [20], Wang [21], and so on. For the other related two-agent works without time-dependent processing times, the reader can refer to Baker and Smith [22], Agnetis et al. [23, 24], Cheng et al. [25, 26], Ng et al. [27], Mor and Mosheiov [28], Lee et al. [29], Leung et al. [30], Wan et al. [31], Yin et al. [19, 32], Yu et al. [33], and Zhao and Lu [34].

This study introduces a new scheduling model in which both the two-agent concept and the learning effects exist, simultaneously. We consider the following objective functions: the maximum cost, total completion time, total weighted completion time, and discounted total weighted completion time. The structural properties of optimal schedules are derived and polynomial time algorithms are developed for the problems where possible.

The remaining part of the study is organized as follows: Section 2 introduces the notation and terminology used throughout the paper. Sections 3–6 analyze the computational complexity and derive the optimal properties of the problems under study. The last section concludes the paper and suggests topics for future research.

2. Model Formulation

The problem investigated in this paper can be formally described as follows. Suppose that there are two agents \( A \) and \( B \), each of whom has a set of nonpreemptive jobs. The two agents compete to process their jobs on a common machine. Agent \( A \) has to process the job set \( J^A = \{ J^A_1, J^A_2, \ldots, J^A_{n_A} \} \), while agent \( B \) has to process the job set \( J^B = \{ J^B_1, J^B_2, \ldots, J^B_{n_B} \} \). All the jobs are available for processing at time \( t_0 \), where \( t_0 \geq 0 \). Let \( X \in \{ A, B \} \). The jobs belonging to agent \( X \) are called \( X \)-jobs. Associated with each job \( J^X_j \) (\( j \in \{ 1, 2, \ldots, n_X \} \)), there are normal processing time \( a_j^X \) and weight \( w_j^X \). Due to the learning effect, the actual processing time \( p_j^X \) of job \( J^X_j \) is defined as

\[
p_j^X = a_j^X (1 - kt), \quad j = 1, 2, \ldots, n_X,
\]

where \( t \geq t_0 \) denotes jobs’ starting time and \( k \geq 0 \) represents constant such that \( k(t_0 + \sum_{j=1}^{n} a_j - a_{\min}) < 1 \), with \( a_{\min} = \min_{j=1,2,...,n} \{ a_j^X \} \) (see Ho et al. [18] for details).

Given a feasible schedule \( S \) of the \( n = n_A + n_B \) jobs, we use \( C_{\max}^X (S) \) to denote the completion time of job \( J^X_j \) and omit the argument \( S \) whenever this does not cause confusion. The makespan of agent \( X \) is \( C_{\max}^X = \max_{j=1,2,\ldots,n_X} (C^X_j) \). For each job \( J^X_j \), let \( f_j^X(\cdot) \) be a nondecreasing function. In this case, the maximum cost is defined as \( f_{\max}^X = \max_{j=1,2,\ldots,n} (f_j^X(C^X_j)) \). The objective function of agent \( X \) considered in this paper includes the following: \( f_{\max}^X \) (maximum cost), \( \sum_{j} C_j^X \) (total completion time), \( \sum w_j^X C_j^X \) (total weighted completion time), and \( \sum w_j^X (1 - e^{-rc_j^X}) \) (discounted total weighted completion time).

Using the three-field notation scheme \( a(\beta|\gamma) \) introduced by Graham et al. [35], the problems considered in this paper are denoted as follows: \( 1|p_j^X = a_j^X (1 - kt) | f_{\max}^X(C^X_B) \leq U, 1|p_j^X = a_j^X (1 - kt) | \sum C_j^X : f_{\max}^B(C^B_X) \leq U, 1|p_j^X = a_j^X (1 - kt) | \sum w_j^A C_j^A : f_{\max}^B(C^B_X) \leq U, 1|p_j^X = a_j^X (1 - kt) | \sum w_j^B (1 - e^{-rc_j^B}) : f_{\max}^B(C^B_X) \leq U \).

Note that all the objective functions involved in the considered problems are regular; that is, they are nondecreasing in the job completion times. Hence there is no benefit in keeping the machine idle.

3. Problem 1 \( |p_j^X = a_j^X (1 - kt) | f_{\max}^A : f_{\max}^B \leq U \)

In this section we address the problem \( |p_j^X = a_j^X (1 - kt) | f_{\max}^A : f_{\max}^B \leq U \) and show that it can be solved optimally in polynomial time. We first develop some structural properties of optimal schedules for the problem which will be used in developing the algorithm.

Lemma 1 (see [19]). For problem 1 \( |p_j^X = a_j^X (1 - kt) | C_{\max} \), the makespan is equal to

\[
(t_0 - \frac{1}{k}) \prod_{j=1}^{n_A} \left( 1 - ka_j^X \right) + \frac{1}{k}
\]

(2)

In the sequel, we set \( u = (t_0 - (1/k)) \prod_{j=1}^{n_A} \left( 1 - ka_j^B \right) + (1/k) \).

Then the following results hold.

Proposition 2. For the problem 1 \( |p_j^X = a_j^X (1 - kt) | f_{\max}^B \leq U \), if there is a \( B \)-job \( J^B_k \) such that \( f_{\max}^B(u) \leq U \), then there exists an optimal schedule such that \( J^B_k \) is scheduled last and there is no optimal schedule where an \( A \)-job is scheduled last.

Proof. Assume that \( S \) is an optimal schedule where the \( B \)-job \( J^B_k \) is not scheduled in the last position. Let \( \pi \) denote the set of jobs scheduled prior to job \( J^B_k \). We construct from \( S \) a new schedule \( S' \) by moving job \( J^B_k \) to the last position and leaving the other jobs unchanged in \( S \). Then, the completion times of the jobs processed before job \( J^B_k \) in \( S \) are the same as that in \( S \) since there is no change for any job preceding \( J^B_k \) in \( S \). The jobs belonging to \( \pi \) are scheduled earlier, so their completion times are smaller in \( S \) by Lemma 1. It follows that \( f_{\max}^X(C^X_k(S')) \leq f_{\max}^X(C^X_k(S)) \) for any job \( J^X_k \) in \( \pi \), where \( X \in \{ A, B \} \). By the assumption that \( f_{\max}^B(u) \leq U \), job \( J^B_k \)
is feasible in $S'$, so schedule $S'$ is feasible and optimal, as required.

For each $B$-job $f_{jl}^B$, let us define a deadline $D_j^B$ such that $f_j^B(C_j^B) \leq L$ for $C_j^B = D_j^B$ and $f_j^B(C_j^B) > L$ for $C_j^B > D_j^B$ (if the inverse function $f_j^B(\cdot)$ is available, the deadlines can be evaluated in constant time; otherwise, this requires logarithmic time).

**Proposition 3.** For the problem $1 \mid p_j^X = a_j^X(1 - kt) \mid f_{\text{max}}^A : f_{\text{max}}^B \leq U$, there exists an optimal schedule where the $B$-jobs are scheduled according to the nondecreasing order of $D_j^B$.

**Proof.** Assume that $S$ is an optimal schedule where the $B$-jobs are not scheduled according to the nondecreasing order of $D_j^B$. Let $f_{jl}^B$ and $f_{jh}^B$ be the first pair of jobs such that $D_j^B > D_h^B$.

In this schedule, job $J_j$ is processed earlier; then a set of $A$-jobs, denoted as $\pi$, are consecutively processed and then job $J_h$. In addition, denote by $\pi'$ the set of jobs processed after job $J_h^B$. We construct from $S$ a new schedule $S'$ by extracting job $J_h^B$ and inserting it just after job $J_j^B$, leaving the other jobs unchanged in schedule $S$. Then the completion times of the jobs processed prior to job $J_j^B$ in $S'$ are the same as that in $S$. By Lemma 1, the completion time of job $J_h^B$ in $S$ equals that of job $J_h^B$ in $S'$; that is, $C_h^B(S') = C_h^B(S)$, so the completion times of the jobs belonging to $\pi'$ are identical in both $S$ and $S'$. Since $S$ is feasible, it follows that $C_h^A(S') = C_h^A(S) \leq D_h^B < D_h^B$, so job $J_h^B$ is feasible in $S'$. The $\pi$-jobs and job $J_h^B$ are scheduled earlier in $S'$, implying that their actual processing times are smaller in $S'$, so their completion times are earlier in $S'$, and thus they remain feasible. Therefore, schedule $S'$ is feasible and optimal. Thus, repeating this procedure for all the $B$-jobs not sequenced according to nondecreasing order of $D_j^B$ completes the proof.

**Proposition 4.** For the problem $1 \mid p_j^X = a_j^X(1 - kt) \mid f_{\text{max}}^A : f_{\text{max}}^B \leq U$, if $f_{jl}^B(u) > U$ for any $B$-job $J_j^B$, then there exists an optimal schedule where the $A$-job with the smallest cost $f_j^A(u)$ is processed in the last position.

**Proof.** Assume that $S$ is an optimal schedule where the $A$-job with the smallest cost $f_j^A$, that is, $f_j^A(u) = \min_{J_j} \{ f_j^A(u) \}$, is not processed in the last position. By the hypothesis, the last job in schedule $S$ is an $A$-job, say $J_l^A$. This means $f_j^A(u) \leq f_{jl}^A(u)$. In this schedule, job $J_l^A$ is scheduled earlier. Let $\pi$ denote the set of jobs scheduled after job $J_l^A$ and prior to job $J_j^B$. We construct from $S$ a new schedule $S'$ by extracting job $J_l^A$, reinserting it just after job $J_j^B$, and leaving the other jobs unchanged in schedule $S$. There is no change for any job preceding $J_l^A$ in $S$. We claim the following.

(1) Schedule $S'$ is feasible. First, the completion times of the jobs processed prior to job $J_l^A$ in $S'$ are the same as that in $S$. Since the jobs belonging to $\pi$ are scheduled earlier in $S'$, their actual processing times are smaller in $S'$, so their completion times are earlier in $S'$. It follows that $f_j^X(C_j^X(S')) \leq f_j^X(C_j^X(S))$ for any job $f_j^X$ in $\pi$, where $X \in \{ A, B \}$, as required.

(2) Schedule $S'$ is a better schedule than $S$. By Lemma 1, the completion time of the last job $f_{jl}^A$ in $S$ equals that of the last job $f_{jl}^A$ in $S'$; that is, $C_l^A(S) = C_l^A(S') = u$. Thus, to prove that $S'$ is better than $S$, it suffices to show that

$$
\max \{ f_j^A(C_j^A(S')), f_j^A(u) \} 
\leq \max \{ f_j^A(C_j^A(S')), f_j^A(u) \} .
$$

Since $f_j^A(\cdot)$ is a nondecreasing function of the completion time of job $f_{jl}^A$ and $C_j^A(S') < u$, we have $f_j^A(C_j^A(S')) \leq f_j^A(u)$. Thus, $\max\{ f_j^A(C_j^A(S')), f_j^A(u) \} \leq f_j^A(u)$, as required.

The result follows.

Summing up the above analysis, our algorithm for problem $1 \mid p_j^X = a_j^X(1 - kt) \mid f_{\text{max}}^A : f_{\text{max}}^B \leq U$ can be formally described as in Algorithm 1.

**Theorem 5.** Algorithm 1 solves problem $1 \mid p_j^X = a_j^X(1 - kt) \mid f_{\text{max}}^A : f_{\text{max}}^B \leq U$ in $O(n_A^2 + n_B \log n_B)$ time.

**Proof.** Step 1 requires a sorting operation of the $B$-jobs, which takes $O(n_B \log n_B)$ time. Step 2 takes $O(n_A)$ time since the calculation of the $f_j^A(\cdot)$ functions in Step 2 can be evaluated in constant time by the assumption. In Step 3 we calculate the $f_j^A(\cdot)$ value for all the remaining unscheduled $A$-jobs, which takes $O(n_A)$ time. Thus, after $n_A$ iterations, Step 3 can be executed in $O(n_A^2)$ time. Therefore, the overall time complexity of the algorithm is indeed $O(n_A^2 + n_B \log n_B)$.

4. Problem $1 \mid p_j^X = a_j^X(1 - kt) \mid \sum w_j^A C_j^A : f_{\text{max}}^B \leq U$.

Leung et al. [30] show that problem $1 \mid \sum w_j^A C_j^A : f_{\text{max}}^B \leq U$ is NP-hard in the strong sense. Since our problem $1 \mid p_j^X = a_j^X(1 - kt) \mid \sum w_j^A C_j^A : f_{\text{max}}^B \leq U$ is a generalization of the problem $1 \mid \sum w_j^A C_j^A : f_{\text{max}}^B \leq U$, then so is our problem. In what follows we show that the problem $1 \mid p_j^X = a_j^X(1 - kt) \mid \sum w_j^A C_j^A : f_{\text{max}}^B \leq U$ is polynomially solvable if the $A$-jobs have reversely agreeable weights; that is, $a_j^A \leq b_j^A$ implies $w_j^A \geq w_j^B$ for all jobs $J_j^A$ and $J_j^B$. It is clear that Propositions 2 and 3 still hold for this problem. We modify Proposition 4 as follows.

**Proposition 6.** For the problem $1 \mid p_j^X = a_j^X(1 - kt) \mid \sum w_j^A C_j^A : f_{\text{max}}^B \leq U$, if the $A$-jobs have reversely agreeable weights, then there exists an optimal schedule where the $A$-jobs are assigned according to the nondecreasing order of $a_j^A/w_j^A$, that is, in the weighted shortest processing time (WSPT) order.
Input: \(n_A, n_B, U, p^A = (p^A_1, p^A_2, \ldots, p^A_{n_A})\) and \(p^B = (p^B_1, p^B_2, \ldots, p^B_{n_B})\).

Step 1. Set \(h = n_B, j = f^A_1 \), \(f^A_{\text{max}} = 0\) and \(t = (t_B - 1/k_B) \prod_{i=1}^{n_B} (1 - k_B i) + 1/k_B\); solve \(D^j_f\) from \(f^j_f(D^j_f) = U\) for \(j = 1, 2, \ldots, n_B\) and numerate them according to the non-decreasing order such that \(D^1_{[1]} \leq D^2_{[2]} \leq \cdots \leq D^n_{[n_B]}\).

Step 2. If \(h \geq 1\), then

- If \(t \leq D^h_{[h]}\) then
  - set \(h = h - 1, t = (t - a^B_{[h]})/(1 - k_B i_{[h]}),\) assign job \(f^B_{[h]}\) at time \(t\), and go to Step 2;
  - Else go to Step 3;
  - Else go to Step 3.

Step 3. If \(f^A_j \neq \emptyset\), then

- select the job \(f^A_j\) from \(j\) with the the smallest cost, that is, \(f^A_j(t) = \min_{f^A_j \neq \emptyset} (f^A_j(t))\),
- set \(f^A_{\text{max}} = \max(f^A_{\text{max}}, f^A_j(t)), t = (t - a^A_j)/(1 - k_A i^A_j),\) assign job \(f^A_j\) at time \(t\), delete \(f^A_j\) from \(j\), and go to Step 4;
- Elseif \(h \geq 1\) report that the instance is not feasible;
- Else go to Step 4;

Step 4. If \(j\) is not empty or \(h \geq 1\), then

- go to Step 2;
- Else stop.

Algorithm 1

**Proof.** Assume that \(S\) is an optimal schedule where \(A\)-jobs are not scheduled in the WSPT order. Let \(J^A_1\) and \(J^A_h\) be the first pair of jobs such that \(a^A_1 \geq a^A_h\) and \(w^A_1 \leq w^A_h\) due to the fact that the \(A\)-jobs have reversely agreeable weights. Assume that, in schedule \(S\), job \(J^A_1\) starts its processing at time \(l_1\), then a set of \(B\)-jobs are consecutively processed and then job \(J^A_h\) in addition, let \(\pi'\) denote the set of jobs processed after job \(J^A_h\). We construct a new scheduling \(S'\) from \(S\) by swapping jobs \(J^A_1\) and \(J^A_h\) and leaving the other jobs unchanged. We conclude the following.

1. Schedule \(S'\) is feasible. By Lemma 1, the completion time of job \(J^A_1\) in \(S\) equals that of job \(J^A_1\) in \(S'\); that is, \(C^A_1(S') = C^A_1(S)\), so the completion times of the jobs belonging to \(\pi'\) are identical in both \(S\) and \(S'\). Since \(a^A_1 \geq a^A_h\), we have \(C^A_1(S) = T + a^A_1(1-KT) \leq T + a^A_h(1-KT) = C^A_h(S)\). Hence the \(\pi\)-jobs are scheduled earlier in \(S'\), implying that their actual processing times are smaller in \(S'\), so their completion times are earlier in \(S'\). Hence \(f^B_k(C^B_k(S')) \leq f^B_k(\pi') \leq f^B_k(C^B_k(S))\) for any job \(J^B_k\) in \(\pi\), as required.

2. Schedule \(S'\) is better than \(S\). By the proof of (1), it is sufficient to show that

\[
\begin{align*}
\sum_j w^A_j C^A_j(S') + w^A_j C^A_j(S') &\leq \sum_j w^A_j C^A_j(S) + w^A_j C^A_j(S) .
\end{align*}
\]  

(4)

Since \(C^A_h(S') \leq C^A_{1}(S)\) and \(C^A_h(S') = C^A_{1}(S)\), we have

\[
\begin{align*}
&\sum_j w^A_j C^A_j(S) + w^A_j C^A_j(S) - (\sum_j w^A_j C^A_j(S') + w^A_j C^A_j(S')) \\
&\geq w^A_1 C^A_h(S) + w^A_1 C^A_h(S') - (w^A_1 C^A_h(S') + w^A_1 C^A_h(S')) \\
&= (w^A_1 - w^A_1) (C^A_h(S') - C^A_h(S')) \\
&\geq 0,
\end{align*}
\]  

as required.

Thus, repeating this swapping argument for all the \(A\)-jobs not sequenced in the WSPT order yields the theorem. \(\square\)

Based on the results of Propositions 2, 3, and 6, our algorithm to solve the problem \(| p^X = a^X(1-kT) | \sum_j w^A_j C^A_j : f^B_{\text{max}} \leq U\) for the case where the \(A\)-jobs have reversely agreeable weights can be formally described as in Algorithm 2.

**Theorem 7.** The problem \(\{ p^X = a^X(1-kT) | \sum_j w^A_j C^A_j : f^B_{\text{max}} \leq U\) can be solved in \(O(n_A \log n_A + n_B \log n_B)\) time by applying Algorithm 2 if all \(A\)-jobs have reversely agreeable weights.

**Proof.** The correctness comes from the above analysis. Now we turn to the time complexity of the algorithm. Step 1 requires two sorting operations of the \(A\)-jobs and \(B\)-jobs, respectively, which take \(O(n_A \log n_A)\) time and \(O(n_B \log n_B)\) time, respectively. Both Steps 2 and 3 take \(O(2)\) time.
Input: \( n_A, n_B, U, w^A = (w^A_1, w^A_2, \ldots, w^A_{n_A}), p^A = (p^A_1, p^A_2, \ldots, p^A_{n_A}) \) and \( p^B = (p^B_1, p^B_2, \ldots, p^B_{n_B}) \).

Step 1. Set \( i = (i_0 - 1)/k \prod_{j=0}^{i_0} (1 - ka^B_j) + 1/k, l = n_A, \) \( h = n_B, \) and \( \sum w^A_j C^A_j = 0; \)

- sort the A-jobs according to the non-decreasing order of \( a^A_j/w^A_j \), that is, \( a^A_{i_1}/w^A_{i_1} \leq a^A_{i_2}/w^A_{i_2} \leq \ldots \leq a^A_{i_{n_A}}/w^A_{i_{n_A}} \);
- calculate the deadlines of the B-jobs from \( f^B(D^B_{i_1}) = U \) and renumber them according to the non-decreasing order such that \( D^B_{i_1} \leq D^B_{i_2} \leq \ldots \leq D^B_{i_{n_B}} \);

Step 2. If \( h \geq 1 \), then

- If \( t \leq D^B_{i_0} \), then
  - set \( h = h - 1, t = (t - a^B_{i_0})/(1 - ka^B_{i_0}) \), assign job \( f^B_{i_1} \) at time \( t \), and go to Step 2;
- Else
  - go to Step 3;
- Else
  - go to Step 3;

Step 3. If \( l \geq 1 \), then

- set \( l = l - 1, \sum w^A_j C^A_j = \sum w^A_j C^A_j + w^A_j f^A_j, t = (t - a^A_{i_0})/(1 - ka^A_{i_0}) \), assign \( f^A_{i_1} \) at time \( t \), and go to Step 4;
- Else \( h \geq 1 \)
  - output that the instance is not feasible;
- Else
  - go to Step 4;

Step 4. If \( h \geq 1 \) or \( l \geq 1 \), then

- go to Step 2;
- Else
  - stop.

**Algorithm 2**

Therefore, the overall time complexity of the algorithm is indeed \( O(n_A \log n_A + n_B \log n_B) \).

5. **Problem 1** \( |p^X_j = a^X_j(1 - kt) | \sum w^A_j(1 - e^{-tC^A_j}) : f^B_{\max} \leq U \)

This section addresses the problem \( |p^X_j = a^X_j(1 - kt) | w^A_j(1 - e^{-tC^A_j}) : f^B_{\max} \leq U \). We show that it is polynomially solvable if the A-jobs have reversely agreeable weights. It is clear that Propositions 2 and 3 still hold for this problem. We give Proposition 8 as follows.

**Proposition 8.** For the problem \( |p^X_j = a^X_j(1 - kt) | w^A_j(1 - e^{-tC^A_j}) : f^B_{\max} \leq U \), if the A-jobs have reversely agreeable weights, then there exists an optimal schedule where the A-jobs are assigned according to the nondecreasing order of \( (1 - e^{-tC^A_j})/w^A_j e^{-tC^A_j} \), that is, in the weighted discount shortest processing time (WDSPT) order.

**Proof.** We adopt the same notation as that used in the proof of Proposition 6. Assume that \((1 - e^{-tC^A_j})/w^A_j e^{-tC^A_j} > (1 - e^{-tA_j})/w^A_j e^{-tA_j}\). Since A-jobs have reversely agreeable weights, we have \( w^A_j \geq a^A_j \) and \( w^A_j \leq w^A_{i_{n_A}} \). Then by the proof of Proposition 6, we know that \( C^A_j(S') \leq C_j^A(S), C_j^A(S') \leq C_j^A(S) \) for all the other jobs \( j_k \in J_A[J^A_k, I^A_k] \) and that schedule \( S' \) is feasible. To show that \( S' \) is better than \( S \), it is sufficient to show that

\[
\begin{align*}
& w^A_h \left( 1 - e^{-tC^A_h(S')} \right) + w^A_l \left( 1 - e^{-tC^A_l(S')} \right) \\
& \leq w^A_h \left( 1 - e^{-tC^A_h(S)} \right) + w^A_l \left( 1 - e^{-tC^A_l(S)} \right). 
\end{align*}
\]

(6)

In fact, since \( r \in (0, 1) \), \( C^A_h(S') \leq C^A_h(S) \), and \( C^A_l(S') = C^A_l(S) \), we have

\[
\begin{align*}
& w^A_h \left( 1 - e^{-tC^A_h(S)} \right) + w^A_l \left( 1 - e^{-tC^A_l(S)} \right) \\
& - \left( w^A_h \left( 1 - e^{-tC^A_h(S')} \right) + w^A_l \left( 1 - e^{-tC^A_l(S')} \right) \right) \\
& = w^A_h e^{-tC^A_h(S')} + w^A_l e^{-tC^A_l(S')} - w^A_h e^{-tC^A_h(S)} - w^A_l e^{-tC^A_l(S)} \\
& \geq w^A_h e^{-tC^A_h(S')} + w^A_l e^{-tC^A_l(S')} - w^A_h e^{-tC^A_h(S)} - w^A_l e^{-tC^A_l(S)} \\
& = \left( w^A_h - w^A_l \right) \left( e^{-tC^A_h(S)} - e^{-tC^A_l(S)} \right) \leq 0.
\end{align*}
\]

(7)

Hence, \( w^A_h(1 - e^{-tC^A_h(S')}) + w^A_l(1 - e^{-tC^A_l(S')}) \leq w^A_h(1 - e^{-tC^A_h(S)}) + w^A_l(1 - e^{-tC^A_l(S)}) \). Therefore, \( S' \) is not worse than \( S \). Thus, repeating this swapping argument for all the A-jobs not sequenced in the WDSPT order yields the theorem.

Based on the above analysis, our algorithm to solve the problem \( |p^X_j = a^X_j(1 - kt) | \sum w^A_j(1 - e^{-tC^A_j}) : f^B_{\max} \leq U \) for the case where the A-jobs have reversely agreeable weights can be described as in Algorithm 3.
The problem 1 | \( p_j^X \) = \( a_j^X (1 - kt) \) | \( \sum w_j^A (1 - e^{-rc_j^A}) : f_{\text{max}}^B \leq U \) can be solved in \( O(n_A \log n_A + n_B \log n_B) \) time by applying Algorithm 3 for the case that the \( A \)-jobs have reversely agreeable weights.

Proof. The proof is analogous to that of Theorem 7.

6. Conclusions

This paper introduced a new scheduling model on a single machine that involves two agents and learning effects simultaneously. We studied the problem of finding an optimal schedule for agent \( A \), subject to the constraint that the maximum cost of agent \( B \) does not exceed a given value. We derived the optimal structural properties of optimal schedules and provided polynomial time algorithms for the problem 1 | \( p_j^X \) = \( a_j^X (1 - kt) \) | \( \sum w_j^A (1 - e^{-rc_j^A}) : f_{\text{max}}^B \leq U \). We also showed that the problems 1 | \( p_j^X \) = \( a_j^X (1 - kt) \) | \( \sum w_j^A C_j^A : f_{\text{max}}^B \leq U \) and 1 | \( p_j^X \) = \( a_j^X (1 - kt) \) | \( \sum w_j^A (1 - e^{-rc_j^A}) : f_{\text{max}}^B \leq U \) can also be solved in polynomial time under certain agreeable conditions. Future research may consider the scheduling model with more than two agents.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are grateful to the Editor and two anonymous referees for their helpful comments on an earlier version of their paper. This research was supported by the National Natural Science Foundation of China (no. 71301022).

References


Submit your manuscripts at http://www.hindawi.com