Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 648082, 18 pages
http://dx.doi.org/10.1155/2013/648082

Research Article

A Two-Agent Single-Machine Scheduling Problem with Learning and Deteriorating Considerations

Wen-Hsiang Wu

Department of Healthcare Management, Yuanpei University, No. 306 Yuanpei St., Hsin Chu 30015, Taiwan

Correspondence should be addressed to Wen-Hsiang Wu; wenhsiang_wu@yahoo.com.tw

Received 26 May 2013; Accepted 1 July 2013

Academic Editor: Chin-Chia Wu

Recently, interest in scheduling with deteriorating jobs and learning effects has kept growing. However, research in this area has seldom considered the multiagent setting. Motivated by these observations, we consider two-agent scheduling on a single machine involving the learning effects and deteriorating jobs simultaneously. In the proposed model, we assume that the actual processing time of a job of the first (second) agent is a decreasing (increasing) function of the total processing time of the jobs already processed in a schedule. The objective is to minimize the total weighted completion time of the jobs of the first agent with the restriction that no tardy job is allowed for the second agent. We develop a branch-and-bound and a simulated annealing algorithms for the problem. We perform extensive computational experiments to test the performance of the algorithms.

1. Introduction

In classical scheduling, researchers routinely assume that the job processing time is known and fixed from the processing of the first job to the completion of the last job. This assumption is invalid in situations where the job processing time may be prolonged due to deterioration or shortened due to learning over time. For example, Browne and Yechiali [1] observed that the time and effort required to put out a fire increase if there is a delay in the starting of the fire-fighting effort. In such environments, a job that is processed later consumes more time than the same job when processed earlier. Scheduling in this setting is known as "scheduling deteriorating jobs." Meanwhile, Biskup [2] points out that the repeated processing of similar tasks improves the workers' skills; for example, workers are able to perform setups, deal with machine operations or software, or handle raw materials and components at a faster pace. This phenomenon is known as the "learning effect" in the literature.

The deteriorating job scheduling problem was first introduced by J. N. D. Gupta and S. K. Gupta [3] and Browne and Yechiali [1], independently. J. N. D. Gupta and S. K. Gupta [3] considered the problem using the polynomial processing time functions to minimize the makespan and propose branch-and-bound and heuristic algorithms to search for the optimal and near-optimal solutions. Browne and Yechiali [1] studied the problem using the exponential job processing times to minimize the makespan and provide insights into problem solutions. Since then, an abundance of studies of the subject have emerged. For different models of the problem dealing with different criteria, we refer readers to Alidaee and Womer [4] and Cheng et al. [5].

On the other hand, Biskup [2] and Cheng and Wang [6], independently, incorporate the concept of learning into scheduling. Many researchers have since devoted large amounts of effort to this relatively young but booming area of scheduling research. For detailed reviews of motivations, results, and applications of scheduling with learning effects, we refer the reader to a comprehensive review of scheduling research with learning considerations by Biskup [7].

Recently, there is a growing interest in scheduling research that considers deteriorating jobs and learning effects simultaneously. Wang [8] assumes that the job processing times has the following form: $p_{j[r]} = (\alpha_j + \beta t)r^a$, where $p_{j[r]}$ is the actual processing time of a job J_j scheduled in the rth position of a sequence, α_j is the basic processing time, and β

is the common deteriorating rate. Wang [9] studies a model in which the job processing time has the following form: $p_{j[r]} = p_j(\alpha(t) + \beta r^a)$, where p_j is the basic processing time and $\alpha(t)$ is an increasing deterioration function with $\alpha(0) \geq 0$. Wang and Cheng [10] consider a model in which the actual processing time is $(p_0 + \alpha_j t)r^a$, where p_0 is the common basic processing time, α_j is the growth rate, and a is the learning index. Cheng et al. [11, 12] study a new scheduling model with deteriorating jobs and learning effects in which the actual processing time of a job J_j scheduled in the rth position of a sequence is modeled as $p_{j[r]} = p_j((p_0 + \sum_{l=1}^{r-1} p_{[l]})/(p_0 + \sum_{l=1}^{n} p_l))^{a_1} r^{a_2}$, where $p_{[l]}$ denotes the normal processing time of the job scheduled in the$l$th position of the sequence, $p_0 > 0$ is a given parameter, and a_1 and a_2 denote the deteriorating and learning indices with $a_1 < 0$ and $a_2 < 0$. Toksari et al. [13] consider several scheduling problems under the assumption of the nonlinear effects of learning and deterioration, where they assume that $p_{j[r]} = (p_{[r]} + \alpha t_{[r]}^b)(1 + \sum_{k=1}^{r-1} p_{[k]})^a$, $a < 0$, $\alpha > 0$, $b > 0$, where $t_{[r]} > 0$ is the starting time of the job scheduled in position r. Huang et al. [14] consider the single-machine scheduling problem with time-dependent deterioration and an exponential learning effect, that is, the actual processing time of a job depends not only on the processing times of the jobs already processed but also on its scheduled position. Cheng [15] modeled that the actual processing time of J_j is defined as $p_j \max\{(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^a, \beta\}$ if it is scheduled in the rth position in a schedule, where $a \geq 1$ and $0 < \beta < 1$. Li and Hsu [16] modeled that the real processing time p_{ir}^X of job J_i^X varies with position r based on the learning effect, that is $p_{ir}^X = p_i^X r^a$, where a is the learning ratio with $a < 0$ and $r = 1, 2, \ldots, n$.

In classical scheduling, it is assumed that there is a single customer (i.e., agent) who seeks to minimize a scheduling criterion that is function of the order in which the customer's orders (i.e., jobs) are processed by the available processing resources (i.e., machines). In many management situations, however, multiple agents compete on the usage of common processing resources. For instance, Agnetis et al. [17] observe that multiple agents compete on the usage of a common processing resource in different application environments and different methodological fields, such as artificial intelligence, decision theory, and operations research. One major stream of research in this context is related to multiagent scheduling, in which different agents interact to perform their respective tasks, negotiating among one another for the usage of the common resources over time. For research on multiagent scheduling without learning or deteriorating jobs or both, the reader may refer to Baker and Smith [18], Agnetis et al. [19], Yuan et al. [20], Cheng et al. [21], Ng et al. [22], Agnetis et al. [17], Cheng et al. [11, 12], Yin et al. [23], Cheng et al. [24], and so forth. Scheduling in the multiagent setting provides the first motivation for this paper.

Another motivation is that research on multiagent scheduling with deteriorating jobs or learning effects is relatively limited. Liu and Tang's study [25] is probably the only scheduling study that considers deteriorating jobs and multiple agents. They assume that the actual processing time of J_j is $\delta_j t$, where $\delta_j > 0$ denotes the deterioration rate and t is the job's starting time. Under the proposed model, they consider the scheduling objectives of minimizing the makespan, maximum lateness, maximum cost, and total completion time. In this paper, we assume that the actual processing time of a job of the first agent is a decreasing function of the total processing time of the jobs already processed in a schedule, while the actual processing time of a job of the second agent is an increasing function of the total processing time of the jobs already processed in a schedule. The objective is to minimize the total weighted completion time of the jobs of the first agent with the restriction that no tardy job is allowed for the second agent.

The remainder of this paper is organized as follows. In Section 2, we present some dominant properties and develop a lower bound to speed up the search for the optimal solution, followed by discussions of branch-and-bound and simulated annealing (SA) algorithm Section 3. We present the results of extensive computational experiments to assess the performance of all of the proposed algorithms under different experimental conditions in Section 4. We conclude the paper and suggest topics for further research in the Section 5.

2. Problem Statement

We formulate the problem under study in the following. There are n jobs ready to be processed on a single machine. Each job belongs to one of the two agents, namely, AG_0 and AG_1. Associated with job J_j, there is a normal processing time p_j, a weight w_j, a due date d_j, and an agent code I_j, where $I_j = 0$ if $J_j \in AG_0$, or $I_j = 1$ if $J_j \in AG_1$. We assume that all the jobs of AG_0 have a learning rate α with $\alpha > 1$, while all the jobs of AG_1 have a deteriorating rate β with $\beta < 0$. Under the proposed model, the actual processing time of J_j is $p_j(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^a$ $(p_j(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^\beta)$ if it is a job of AG_0 (AG_1) scheduled in the rth position of a sequence, where the subscript $[l]$ denotes the job in the lth position of a sequence. For a given schedule S, let $C_j(S)$ be the completion time of J_j and let $L_j(S) = C_j(S) - d_j$ be the lateness of J_j. The objective of the scheduling problem is to find an optimal schedule to minimize $\sum_{j=1}^{n} w_j C_j(S)(1 - I_j)$ subject to $\max_{1 \leq j \leq n}\{L_j(S)I_j\} \leq 0$. Since the objective function and constraint involve regular scheduling criteria, we use the terms schedule and sequence interchangeably.

3. Branch-and-Bound and Simulated Annealing Algorithms

Ng et al. [22] show that our problem without learning or deteriorating consideration is strongly NP-hard. So, we apply branch-and-bound and SA algorithms to search for the optimal and near-optimal solutions, respectively, in this paper. In order to speed up the searching process, we first develop three adjacent pairwise interchange properties, followed by two dominant rules. We then present the procedures of the branch-and-bound and SA algorithms.

3.1. Dominant Properties. Assume that the schedule (sequence) S_1 has two adjacent jobs J_i and J_j with J_i immediately preceding J_j and that J_i and J_j are in the rth and the $(r+1)$th positions of S_1, respectively. Perform a pairwise interchange of J_i and J_j to derive a new sequence S_2.

Proposition 1. *For any two jobs J_i and $J_j \in AG_0$ to be scheduled consecutively, if $p_j/p_i \geq w_j/w_i > 1$, then S_1 dominates S_2.*

Proof. Assume that $S_1 = (\pi, J_i, J_j, \pi')$ and $S_2 = (\pi, J_j, J_i, \pi')$ denote two sequences in which π and π' denote partial sequences. To show that S_1 dominates S_2, it suffices to show that $[w_iC_i(S_1) + w_jC_j(S_1)] \leq [w_jC_j(S_2) + w_iC_i(S_2)]$ and $C_j(S_1) < C_i(S_2)$. In addition, let Abe the completion time of the last job in the subsequence π with $(r-1)$ jobs. Since J_i and $J_j \in AG_0$, the completion time of the jobs J_i and J_j in sequences S_1 and S_2 is given by the following:

$$
\begin{aligned}
C_i(S_1) &= A + p_i\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a, \\
C_j(S_1) &= A + p_i\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a + p_j\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]} + p_i}{\sum p_i}\right)^a, \\
C_j(S_2) &= A + p_j\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a.
\end{aligned} \tag{1}
$$

One also has

$$
C_i(S_2) = A + p_j\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a + p_i\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]} + p_j}{\sum p_i}\right)^a. \tag{2}
$$

Taking the difference between S_1 and S_2, we obtain the following:

$$
\begin{aligned}
&\left[w_jC_j(S_2) + w_iC_i(S_2)\right] - \left[w_iC_i(S_1) + w_jC_j(S_1)\right] \\
&= \left(w_ip_j - w_jp_i\right)\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a \\
&+ w_jp_j\left[\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a - \left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]} + p_i}{\sum p_i}\right)^a\right] \\
&- w_ip_i\left[\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a - \left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]} + p_j}{\sum p_i}\right)^a\right].
\end{aligned} \tag{3}
$$

By substituting $\theta = (p_j/w_j)/(p_i/w_i)$, $c = w_j/w_i$, and $x = p_i/(\sum p_i - \sum_{l=1}^{r-1} p_{[l]})$, we simplify (3) as follows:

$$
\begin{aligned}
&\left[w_jC_j(S_2) + w_iC_i(S_2)\right] - \left[w_iC_i(S_1) + w_jC_j(S_1)\right] \\
&= w_jp_i\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum p_i}\right)^a \\
&\quad \times\left\{(\theta - 1) + c\theta\left[1 - (1-x)^a\right] - \frac{1}{c}\left[1 - (1 - c\theta x)^a\right]\right\},
\end{aligned} \tag{4}
$$

where $c \geq 1$, $\theta \geq 1$, $a > 1$, and $0 < x < 1$. Taking the first and second derivatives of (4), we can see that (4) is nonnegative. Hence, we have

$$
\left[w_jC_j(S_2) + w_iC_i(S_2)\right] \geq \left[w_iC_i(S_1) + w_jC_j(S_1)\right]. \tag{5}
$$

On the other hand, taking the difference between (1) and (2), we have

$$
\begin{aligned}
C_i(S_2) - C_j(S_1) = {}& \left(p_j - p_i\right)\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^a \\
&+ p_i\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]} + p_j}{\sum_{l=1}^{n} p_l}\right)^a \\
&- p_j\left(1 - \frac{\sum_{l=1}^{r-1} p_{[l]} + p_i}{\sum_{l=1}^{n} p_l}\right)^a.
\end{aligned} \tag{6}
$$

By substituting $\theta = p_j/p_i$, $u = (1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))$, and $x = (p_i/\sum_{l=1}^{n} p_l)$ into (6), we have

$$
C_i(S_2) - C_j(S_1) = p_iu^a\left[(\theta - 1) + (1 - \theta x)^a - \theta(1-x)^a\right]. \tag{7}
$$

Taking the first and second derivatives of (7), and noting that $\theta > 1$, $a > 1$, and $0 < x < 1$, we have $C_i(S_2) - C_j(S_1) > 0$. Therefore, S_1 dominates S_2. □

Proposition 2. *For any two jobs J_i and $J_j \in AG_1$ to be scheduled consecutively, if $p_i < p_j$, $A + p_i(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^\beta + p_j(1 - ((\sum_{l=1}^{r-1} p_{[l]} + p_i)/(\sum_{l=1}^{n} p_l)))^\beta < d_j$, and $A + p_i(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^\beta < d_i$, then S_1 dominates S_2.*

Proposition 3. *For job $J_i \in AG_1$ and job $J_j \in AG_0$ to be scheduled consecutively, if $A + p_i(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^\beta < d_i$ and $p_i < p_j[(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^a - (1 - ((\sum_{l=1}^{r-1} p_{[l]} + p_i)/(\sum_{l=1}^{n} p_l)))^a]/(1 - ((\sum_{l=1}^{r-1} p_{[l]})/(\sum_{l=1}^{n} p_l)))^\beta$, then S_1 dominates S_2.*

We omit the proofs of Propositions 2 and 3 because they are similar to that of Proposition 1.

We next present two propositions to determine the feasibility of a partial sequence. Let (π, π^c) be a sequence of jobs, where π is the scheduled part with k jobs and π^c is the unscheduled part. Moreover, let $C_{[k]}$ be the completion time of the last job in π.

Proposition 4. *If there is a job $J_j \in AG_1 \cap \pi^c$ such that $C_{[k]} > d_j$, then sequence $(\pi,\ \pi^c)$ is not a feasible solution.*

Proof. If there is a job $J_j \in AG_1 \cap \pi^c$ such that $C_{[k]} > d_j$, this violates the constraint that no tardy job is allowed for the second agent. □

Proposition 5. *If there is a job $J_j \in AG_1 \cap \pi^c$ such that $C_{[k]} + p_j(1 - ((\sum_{l=1}^{k} p_{[l]})/(\sum_{l=1}^{n} p_l)))^{\beta} > d_j$, then sequence $(\pi,\ \pi^c)$ is not a feasible solution.*

Proof. Similar to Proposition 4. □

3.2. Lower Bound. In this subsection, we develop a lower bound for the proposed branch-and-bound algorithm. Suppose that PS is a partial schedule in which the order of the first k jobs is determined, and, US be the unscheduled part with $(n-k)$ jobs, where there are n_0 jobs of AG_0 and n_1 jobs of AG_1 with $n_0 + n_1 = n - k$. Moreover, let $p_{(k+1)} \le p_{(k+2)} \le \cdots \le p_{(n)}$ denote the normal processing time of the $(n-k)$ unscheduled jobs when arranged in nondecreasing order, and let $C_{[k]}$ be the completion time of the last job in PS. Given that the learning effect can shorten the processing time and that the deteriorating effect can lengthen the processing time, we assign the jobs with the learning effect to the first n_0 positions and the jobs with the deteriorating effect to the following n_1 positions of the remaining $(n-k)$ unscheduled positions after the kth position which contributes in the reduction the total weighted completion time of the jobs of the first agent. The completion time of the $(k+1)$th job is

$$\begin{aligned} C_{[k+1]} &= C_{[k]} + p_{[k+1]}\left[\left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{a} I_{\{[k+1]\}} + \left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{\beta}\left(1-I_{\{[k+1]\}}\right)\right] \\ &\ge C_{[k]} + p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{a}. \end{aligned} \tag{8}$$

Similarly, the completion time of the $(k+j)$th job is

$$\begin{aligned} C_{[k+j]} \ge{} & C_{[k]} + p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{a} \\ & + \sum_{i=2}^{j} p_{(k+i)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]} + \sum_{l=2}^{j} p_{(n-l+2)}}{\sum_{l=1}^{n} p_l}\right)^{a}, \\ & 2 \le j \le n_0. \end{aligned} \tag{9}$$

Next, we deriequence by arranging the jobs ofe the completion time of the jobs with the deteriorating effect assigned to the remaining n_1 positions. The completion time of the $(k+n_0+j)$th job is

$$\begin{aligned} C_{[k+n_0+j]} \ge{} & C_{[k+n_0]} \\ & + \sum_{i=1}^{j} p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]} + \sum_{l=1}^{j-1} p_{(k+l)}}{\sum_{l=1}^{n} p_l}\right)^{\beta}, \\ & \text{for } 1 \le j \le n_1. \end{aligned} \tag{10}$$

Note that $\sum_{l=1}^{0} p_{(k+l)} = 0$ and

$$\begin{aligned} C_{[k+n_0]} \ge{} & C_{[k]} + p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{a} \\ & + \sum_{i=2}^{n_0} p_{(k+i)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]} + \sum_{l=2}^{j} p_{(n-l+2)}}{\sum_{l=1}^{n} p_l}\right)^{a}. \end{aligned} \tag{11}$$

Following the same idea of Cheng et al. [26] and Wu et al. [27], we want to assign the job completion time to the jobs of agents AG_0 and AG_1. Given the constraint that the jobs of agent AG_1 cannot be tardy, we assign the completion time to the jobs of agent AG_1 as late as possible. The procedure is as follows.

3.2.1. Algorithm

Step 1. Set

$$\begin{aligned} \widehat{C}_{[k+1]} &= C_{[k]} + p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{a}, \quad \text{for } j = 1, \\ \widehat{C}_{[k+j]} &= C_{[k]} + p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]}}{\sum_{l=1}^{n} p_l}\right)^{a} \\ &\quad + \sum_{i=2}^{j} p_{(k+i)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]} + \sum_{l=2}^{j} p_{(n-l+2)}}{\sum_{l=1}^{n} p_l}\right)^{a}, \\ &\quad \text{for } 2 \le j \le n_0, \\ \widehat{C}_{[k+n_0+j]} &= \widehat{C}_{[k+n_0]} \\ &\quad + \sum_{i=1}^{j} p_{(k+1)}\left(1-\frac{\sum_{l=1}^{k} p_{[l]} + \sum_{l=1}^{j-1} p_{(k+l)}}{\sum_{l=1}^{n} p_l}\right)^{\beta}, \\ &\quad \text{for } 1 \le j \le n_1. \end{aligned} \tag{12}$$

Step 2. Sort the jobs of agent AG_0 in a nondecreasing order of their weights and the jobs of agent AG_1 in a nondecreasing order of their due dates, that is,

$$w_{(1)}^{0} \le w_{(2)}^{0} \le \cdots \le w_{(n_0)}^{0}, \qquad d_{(1)}^{1} \le d_{(2)}^{1} \le \cdots \le d_{(n_1)}^{1}. \tag{13}$$

Table 1: Performance of the branch-and-bound and SA algorithms (n = 10, pro = 0.4).

Pro	τ	R	a	β	Branch-and-bound algorithm				SA_1		SA_2		SA_3	
					CPU time		Number of nodes		Error percentages					
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	0.40	0.40	1.100	−1.100	103.98	276.35	3921045	10555474	0.543	1.381	0.530	1.452	0.247	0.883
				−1.010	63.93	217.06	2380521	8139948	0.223	0.659	0.409	0.924	0.197	0.658
				−1.001	65.47	222.48	2437016	8344800	0.216	0.507	0.165	0.483	0.170	0.381
			1.010	−1.100	95.82	253.71	3623888	9719538	0.478	1.500	0.397	1.219	0.298	1.126
				−1.010	58.55	198.37	2187372	7459183	0.272	0.725	0.239	0.630	0.182	0.586
				−1.001	59.55	201.41	2223379	7575433	0.271	0.671	0.322	0.897	0.145	0.416
			1.001	−1.100	95.15	251.83	3600246	9653456	0.420	1.255	0.656	1.637	0.175	0.620
				−1.010	58.05	196.57	2169483	7394568	0.174	0.466	0.416	1.046	0.168	0.525
				−1.001	59.06	199.77	2206236	7517115	0.200	0.625	0.208	0.618	0.153	0.497
		0.50	1.100	−1.100	85.80	248.97	3205789	9430455	0.527	1.543	0.227	0.569	0.210	0.455
				−1.010	68.71	244.97	2549809	9236732	0.581	2.138	0.318	0.891	0.171	0.419
				−1.001	68.38	248.11	2551177	9366715	0.692	1.981	0.161	0.386	0.347	0.856
			1.010	−1.100	79.98	232.32	2995531	8817308	0.575	1.440	0.439	1.103	0.219	0.667
				−1.010	63.28	226.24	2354281	8550234	0.699	1.835	0.622	1.664	0.242	0.662
				−1.001	62.63	227.25	2341949	8599304	0.316	0.789	0.297	0.777	0.143	0.348
			1.001	−1.100	79.34	230.50	2972794	8751750	0.417	1.087	0.530	1.435	0.180	0.485
				−1.010	62.76	224.44	2335360	8482895	0.285	0.771	0.567	1.530	0.321	1.380
				−1.001	62.08	225.42	2321896	8531873	0.405	0.965	0.217	0.694	0.114	0.314
		0.60	1.100	−1.100	75.42	256.62	2793897	9601920	0.546	1.308	0.447	0.970	0.359	1.051
				−1.010	64.40	245.82	2384768	9203252	0.625	1.263	0.811	2.576	0.314	0.666
				−1.001	65.53	248.86	2427465	9311518	0.369	0.874	0.352	0.859	0.285	0.676
			1.010	−1.100	71.27	242.87	2648314	9106695	0.576	1.457	0.327	0.922	0.254	0.679
				−1.010	59.88	229.35	2225401	8623062	0.381	0.693	0.429	1.125	0.405	1.064
				−1.001	60.77	231.22	2258351	8680937	0.565	1.358	0.732	2.489	0.329	1.122
			1.001	−1.100	70.71	241.02	2627453	9038121	0.360	0.943	0.357	1.033	0.135	0.320
				−1.010	60.15	230.09	2237063	8644312	0.809	1.726	0.531	1.307	0.378	1.006
				−1.001	60.20	229.00	2237648	8599211	0.567	1.440	0.462	1.334	0.266	0.614
	0.50	0.40	1.100	−1.100	19.94	59.21	780723	2349939	0.312	1.111	0.276	0.711	0.141	0.387
				−1.010	30.23	77.19	1183839	3050478	0.270	0.730	0.303	1.152	0.095	0.253
				−1.001	30.91	78.69	1209380	3107819	1.058	6.974	0.139	0.420	0.927	6.386
			1.010	−1.100	18.32	54.31	721468	2166990	0.249	0.619	0.201	0.522	0.080	0.245
				−1.010	27.57	70.55	1085563	2803070	0.202	0.511	0.355	1.077	0.480	2.424
				−1.001	28.17	71.84	1108207	2852184	0.850	3.491	0.273	0.751	0.187	0.844
			1.001	−1.100	18.16	53.83	715408	2149077	0.086	0.220	0.196	0.611	0.113	0.301
				−1.010	27.30	69.90	1075579	2778464	1.006	5.165	0.230	1.018	0.821	5.100
				−1.001	27.89	71.17	1097883	2827375	1.220	8.550	0.518	2.713	0.164	0.818
		0.50	1.100	−1.100	27.86	73.51	1088932	2909727	0.557	2.021	0.244	0.553	0.132	0.344
				−1.010	27.67	74.52	1067668	2921795	0.464	1.396	0.258	0.811	0.132	0.335
				−1.001	27.89	76.10	1078229	2982686	0.417	1.217	0.607	1.722	0.096	0.331
			1.010	−1.100	25.46	67.07	1000555	2665211	0.250	0.875	0.285	0.839	0.083	0.190
				−1.010	25.43	68.23	985378	2685319	0.512	1.747	0.380	1.148	0.158	0.697
				−1.001	25.70	69.91	997591	2751685	0.507	1.384	0.392	1.143	0.293	1.141
			1.001	−1.100	25.23	66.50	992131	2644048	0.681	2.156	0.217	0.807	0.216	0.815
				−1.010	25.20	67.61	977049	2661873	0.519	1.392	0.264	0.703	0.289	0.982
				−1.001	25.43	69.16	987380	2722447	0.311	0.823	0.311	0.963	0.296	1.136
		0.60	1.100	−1.100	32.49	83.06	1259437	3245114	0.381	1.107	0.432	1.539	0.118	0.302
				−1.010	22.45	74.47	847123	2812785	0.718	1.975	0.345	1.111	0.090	0.230
				−1.001	22.85	75.78	861566	2861344	0.295	0.705	0.598	1.635	0.151	0.324

Table 1: Continued.

					Branch-and-bound algorithm				SA_1		SA_2		SA_3	
Pro	τ	R	a	β	CPU time		Number of nodes				Error percentages			
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
			1.010	−1.100	29.96	76.43	1163916	2987954	0.535	1.425	0.378	1.250	0.098	0.269
				−1.010	21.02	69.88	797393	2652794	0.386	1.012	0.479	1.419	0.207	0.757
				−1.001	21.34	70.85	808645	2687918	0.848	2.490	0.454	1.468	0.147	0.469
			1.001	−1.100	29.76	75.79	1156347	2963377	0.319	0.960	0.360	0.984	0.296	0.930
				−1.010	20.92	69.31	793687	2631768	0.455	1.079	0.642	1.865	0.182	0.696
				−1.001	21.16	70.27	802222	2666815	0.468	1.233	0.514	1.464	0.232	0.842
	0.60	0.40	1.100	−1.100	1.95	5.15	79358	214347	0.311	1.439	0.105	0.295	0.124	0.376
				−1.010	1.73	5.65	70241	235697	0.125	0.309	0.105	0.336	0.165	0.827
				−1.001	1.73	5.70	70552	237669	0.144	0.366	0.147	0.368	0.094	0.322
			1.010	−1.100	1.82	4.83	74060	201604	0.330	1.453	0.262	1.441	0.125	0.389
				−1.010	1.57	5.10	64068	213383	0.266	1.477	0.146	0.361	0.076	0.197
				−1.001	1.56	5.14	64003	215085	0.120	0.289	0.128	0.334	0.079	0.273
			1.001	−1.100	1.81	4.82	73630	200978	0.150	0.360	0.144	0.324	0.102	0.292
				−1.010	1.55	5.05	63473	211343	0.220	0.706	0.131	0.391	0.106	0.279
				−1.001	1.55	5.10	63463	213337	0.326	1.471	0.172	0.508	0.045	0.151
		0.50	1.100	−1.100	2.32	7.36	95199	309557	0.159	0.569	0.141	0.528	0.069	0.220
				−1.010	3.06	9.20	124875	380846	0.223	0.713	0.225	0.656	0.136	0.610
				−1.001	3.11	9.33	127086	386185	0.363	1.193	0.334	1.068	0.089	0.289
			1.010	−1.100	2.11	6.73	86838	283413	0.270	0.875	0.132	0.481	0.179	0.805
				−1.010	2.78	8.44	114224	351088	0.385	1.169	0.135	0.402	0.048	0.127
				−1.001	2.83	8.56	116276	355790	0.233	0.853	0.258	0.800	0.054	0.180
			1.001	−1.100	2.09	6.66	85990	280750	0.373	1.450	0.186	0.809	0.062	0.210
				−1.010	2.75	8.36	113098	347937	0.112	0.354	0.229	0.793	0.046	0.125
				−1.001	2.80	8.48	115164	352785	0.158	0.643	0.397	1.297	0.052	0.197
		0.60	1.100	−1.100	3.44	10.66	139504	436210	0.145	0.546	0.293	0.936	0.059	0.206
				−1.010	4.91	13.22	202484	547733	0.086	0.310	0.121	0.260	0.072	0.197
				−1.001	5.02	13.44	207022	556511	0.327	1.314	0.304	1.301	0.126	0.423
			1.010	−1.100	3.12	9.69	127253	398528	0.206	0.902	0.230	0.934	0.097	0.324
				−1.010	4.47	12.10	185230	502913	0.303	1.447	0.079	0.212	0.047	0.187
				−1.001	4.57	12.29	189209	510614	0.427	2.235	0.350	2.226	0.038	0.130
			1.001	−1.100	3.09	9.61	126170	395114	0.175	0.618	0.219	0.518	0.096	0.328
				−1.010	4.43	11.99	183664	498778	0.356	1.516	0.213	0.493	0.035	0.090
				−1.001	4.53	12.19	187563	506370	0.306	1.461	0.428	2.235	0.087	0.286
Average					32.64	103.83	1234705	3960746	0.402	1.356	0.326	1.003	0.184	0.668

Step 3. Set $ic = n$, $ia = n_0$, and $ib = n_1$.

Step 4. If $ic \le n - k$, go to Step 7.

Step 5. If $\widehat{C}_{[ic]} \le d^1_{(ib)}$, set $C^1_{(ib)} = \widehat{C}_{[ic]}$ and $ib = ib - 1$. Otherwise, set $C^0_{(ia)} = \widehat{C}_{[ic]}$ and $ia = ia - 1$.

Step 6. Set $ic = ic - 1$, and go to Step 4.

Step 7. Compute $LB_{US} = \sum_{j=1}^{n_0} w^0_{(n_0-j+1)} C^0_{(j)}$. Evidently, a lower bound for the partial sequence PS is

$$LB = \sum_{j \in PS} w_{[j]} C_{[j]} \left(1 - I_j\right) + LB_{US}. \tag{14}$$

3.3. Simulated Annealing Algorithms. Simulated annealing is one of the most popular metaheuristic methods widely applied to solve combinatorial optimization problems [28–30]. SA has the advantage of avoiding getting trapped in a local optimum because of its hill climbing moves, which are governed by a control parameter. In the following, we apply SA to derive near-optimal solutions for our problem. We apply an SA algorithm with three different initials as follows.

Step 1. Initial sequence.

We use three initial sequences for the three SA algorithms. In SA_1, we use random numbers to generate an initial job sequence; if there is any tardy job in this sequence, we regenerate another job sequence until it is feasible. In SA_2, we create an initial job sequence by arranging the jobs of AG_1 in the earliest due date (EDD) order, followed by arranging the jobs of AG_0 in the shortest processing time (SPT) order. In SA_3, we create an initial job sequence by arranging the jobs of AG_1 in the EDD order, followed by arranging the jobs of AG_0

Table 2: Performance of the branch-and-bound and SA algorithms (n = 10, pro = 0.6).

Pro	τ	R	a	β	Branch-and-bound algorithm				SA_1		SA_2		SA_3	
					CPU time		Number of nodes		Error percentages					
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.6	0.40	0.40	1.100	−1.100	169.61	485.47	5142200	14422081	0.328	1.109	0.305	0.963	0.053	0.145
				−1.010	139.74	443.06	4209007	13084799	0.401	0.929	0.296	0.823	0.209	0.584
				−1.001	141.09	445.63	4247152	13156055	0.335	0.906	0.205	0.571	0.214	0.557
			1.010	−1.100	154.84	442.72	4692803	13141955	0.204	0.573	0.133	0.330	0.150	0.535
				−1.010	128.70	407.77	3878025	12037169	0.453	1.071	0.369	0.864	0.088	0.292
				−1.001	129.35	404.59	3896075	11931805	0.373	1.518	0.301	0.994	0.083	0.293
			1.001	−1.100	153.58	439.16	4655648	13037593	0.175	0.517	0.292	0.722	0.129	0.506
				−1.010	127.68	403.85	3847699	11919775	0.420	1.465	0.304	0.870	0.140	0.398
				−1.001	128.23	400.75	3863145	11818967	0.471	1.493	0.216	0.776	0.105	0.331
		0.50	1.100	−1.100	148.03	468.33	4543781	14045348	0.330	0.729	0.415	0.926	0.154	0.432
				−1.010	135.08	466.71	4045384	13826126	0.194	0.443	0.378	1.237	0.328	1.148
				−1.001	134.70	467.90	4027778	13850821	0.475	1.842	0.626	2.128	0.222	1.072
			1.010	−1.100	136.56	433.00	4186707	12960581	0.265	0.641	0.221	0.628	0.201	0.645
				−1.010	124.10	424.78	3717608	12569431	0.381	1.177	0.411	1.192	0.150	0.362
				−1.001	124.35	428.06	3721034	12662504	0.499	1.198	0.606	1.475	0.341	1.103
			1.001	−1.100	135.22	428.57	4145141	12823831	0.288	0.749	0.464	1.740	0.069	0.163
				−1.010	123.87	426.05	3710684	12606768	0.386	1.203	0.473	1.274	0.267	1.089
				−1.001	123.62	426.51	3698366	12613179	0.576	1.492	0.522	1.329	0.237	1.065
		0.60	1.100	−1.100	136.01	505.99	4109421	15175777	0.186	0.469	0.198	0.483	0.105	0.291
				−1.010	142.76	530.18	4297433	15833926	0.418	2.053	0.411	1.273	0.142	0.351
				−1.001	141.13	519.79	4243543	15506884	0.620	2.204	0.548	1.622	0.116	0.301
			1.010	−1.100	123.90	457.10	3737374	13664018	0.560	1.963	0.200	0.679	0.119	0.422
				−1.010	128.19	469.11	3852210	13970930	0.596	2.184	0.464	2.182	0.070	0.175
				−1.001	129.10	473.67	3878835	14109979	0.334	0.897	0.498	2.114	0.148	0.360
			1.001	−1.100	123.22	455.91	3715759	13624099	0.307	0.843	0.457	1.489	0.076	0.250
				−1.010	126.84	463.58	3811424	13804562	0.579	2.337	0.497	1.251	0.098	0.257
				−1.001	127.34	466.29	3824418	13884199	0.488	2.077	0.904	3.092	0.145	0.544
	0.50	0.40	1.100	−1.100	50.98	137.93	1571500	4194572	0.212	0.764	0.165	0.403	0.087	0.262
				−1.010	44.81	132.95	1358796	3955941	0.256	0.612	0.308	0.673	0.112	0.364
				−1.001	44.95	133.26	1361858	3960955	0.167	0.436	0.209	0.557	0.081	0.319
			1.010	−1.100	45.44	120.52	1403943	3668156	0.197	0.551	0.128	0.363	0.088	0.291
				−1.010	40.47	118.51	1230578	3534612	0.167	0.598	0.208	0.552	0.125	0.416
				−1.001	40.39	118.04	1226651	3514368	0.222	0.585	0.219	0.564	0.068	0.177
			1.001	−1.100	45.04	119.11	1392216	3627488	0.093	0.189	0.309	1.317	0.079	0.280
				−1.010	40.14	117.44	1221284	3504993	0.310	0.829	0.217	0.619	0.115	0.450
				−1.001	40.01	117.02	1215980	3486325	0.189	0.480	0.218	0.567	0.101	0.292
		0.50	1.100	−1.100	48.30	159.50	1462169	4805099	0.204	0.633	0.292	0.924	0.065	0.221
				−1.010	48.37	154.48	1462371	4586082	0.278	0.863	0.137	0.351	0.097	0.335
				−1.001	44.52	149.69	1363861	4460422	0.179	0.507	0.276	0.864	0.096	0.448
			1.010	−1.100	43.14	139.72	1303877	4189668	0.307	0.779	0.301	0.844	0.111	0.471
				−1.010	42.79	134.24	1292152	3971867	0.407	1.195	0.461	1.175	0.096	0.345
				−1.001	39.34	129.88	1202979	3856027	0.324	0.762	0.298	0.903	0.130	0.489
			1.001	−1.100	42.70	138.01	1290587	4136889	0.078	0.156	0.242	0.764	0.056	0.132
				−1.010	42.38	133.16	1280136	3940558	0.190	0.758	0.274	0.882	0.109	0.364
				−1.001	38.82	127.90	1186715	3795955	0.387	1.161	0.313	1.067	0.069	0.300
		0.60	1.100	−1.100	53.09	188.20	1620698	5675353	0.208	0.528	0.317	0.941	0.091	0.424
				−1.010	47.40	170.15	1443635	5040348	0.151	0.434	0.214	0.490	0.052	0.134
				−1.001	47.20	169.46	1435927	5012344	0.244	0.482	0.155	0.441	0.068	0.206

Table 2: Continued.

Pro	τ	R	a	β	Branch-and-bound algorithm				SA_1		SA_2		SA_3	
					CPU time		Number of nodes		Error percentages					
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
			1.010	−1.100	47.06	164.68	1431408	4933285	0.447	1.363	0.256	0.754	0.103	0.442
				−1.010	42.22	147.34	1282548	4342559	0.487	2.095	0.400	1.974	0.033	0.101
				−1.001	42.24	147.07	1282549	4331696	0.188	0.491	0.322	1.232	0.023	0.094
			1.001	−1.100	46.46	162.42	1412542	4862817	0.326	0.921	0.232	0.700	0.126	0.430
				−1.010	41.82	145.44	1270995	4287523	0.392	1.066	0.129	0.371	0.023	0.073
				−1.001	41.70	144.93	1265860	4267324	0.278	1.021	0.202	0.757	0.047	0.125
	0.60	0.40	1.100	−1.100	9.52	22.00	301712	689889	0.159	0.491	0.262	0.958	0.218	0.871
				−1.010	9.40	23.76	300052	754248	0.107	0.299	0.111	0.365	0.095	0.330
				−1.001	9.93	24.06	319053	765992	0.218	0.955	0.133	0.494	0.030	0.093
			1.010	−1.100	8.54	19.68	272224	621655	0.247	0.766	0.198	0.718	0.152	0.686
				−1.010	8.34	20.87	267808	667802	0.350	1.659	0.072	0.193	0.049	0.126
				−1.001	8.81	21.16	284571	678948	0.268	0.826	0.086	0.207	0.041	0.094
			1.001	−1.100	8.43	19.43	268841	613948	0.105	0.290	0.229	0.704	0.045	0.117
				−1.010	8.24	20.59	264663	659399	0.325	1.427	0.292	1.349	0.038	0.103
				−1.001	8.72	20.89	281716	670765	0.286	1.191	0.198	0.592	0.065	0.144
		0.50	1.100	−1.100	12.59	32.85	402441	1039572	0.080	0.185	0.061	0.191	0.040	0.136
				−1.010	13.08	34.50	415790	1088849	0.149	0.413	0.102	0.220	0.064	0.188
				−1.001	13.20	34.77	419385	1097323	0.115	0.350	0.211	0.597	0.064	0.213
			1.010	−1.100	11.23	29.11	360464	926431	0.185	0.681	0.037	0.095	0.032	0.071
				−1.010	11.52	30.03	367316	953452	0.081	0.260	0.256	0.747	0.057	0.240
				−1.001	11.57	30.13	368553	956769	0.178	0.488	0.098	0.226	0.106	0.346
			1.001	−1.100	11.13	28.89	357394	919613	0.115	0.411	0.272	1.049	0.042	0.177
				−1.010	11.36	29.59	362362	939892	0.207	0.593	0.129	0.379	0.062	0.256
				−1.001	11.42	29.70	364005	943583	0.108	0.337	0.186	0.628	0.060	0.249
		0.60	1.100	−1.100	17.00	47.56	540126	1496760	0.207	0.619	0.101	0.249	0.079	0.281
				−1.010	16.32	47.42	511056	1470526	0.229	0.549	0.122	0.380	0.056	0.219
				−1.001	16.26	47.28	508286	1462393	0.177	0.495	0.135	0.339	0.088	0.251
			1.010	−1.100	15.27	42.24	485259	1329067	0.197	0.635	0.176	0.465	0.083	0.387
				−1.010	14.66	41.81	460159	1302846	0.195	0.545	0.164	0.434	0.075	0.213
				−1.001	14.51	41.24	453999	1279281	0.184	0.493	0.112	0.331	0.137	0.409
			1.001	−1.100	15.14	41.77	481166	1314352	0.118	0.279	0.301	0.800	0.050	0.136
				−1.010	14.47	41.18	453836	1283057	0.160	0.487	0.204	0.512	0.049	0.155
				−1.001	14.33	40.61	448251	1259365	0.122	0.411	0.294	0.807	0.098	0.356
Average					63.62	208.26	1929543	6212496	0.277	0.870	0.272	0.841	0.104	0.352

in the weighted shortest processing time (WSPT) order. In order to get a good initial solution, we apply the NEH method [31] to the initial sequences produced by SA_2 and SA_3.

Step 2. Neighborhood generation.

Neighborhood generation plays an important role in the efficiency of SA algorithms. We use the pairwise interchange (PI) neighborhood generation method in the algorithms.

Step 3. Acceptance probability.

When a new feasible sequence is generated, it is accepted if its objective value is smaller than that of the original sequence; otherwise, it is accepted with a probability that decreases as the process evolves. The probability of acceptance is generated from an exponential distribution as follows:

$$P(\text{accept}) = \exp(-\rho \times \Delta WTC), \tag{15}$$

where ρ is a control parameter and ΔWTC is the change in the objective value. In addition, we use the method suggested by Ben-Arieh and Maimon [32] to change ρ in the kth iteration as follows:

$$\rho = \frac{k}{\delta}, \tag{16}$$

where δ is an experimental constant. After preliminary trials, we used $\delta = 0.8$ in our experiments.

If the total weighted completion time increases as a result of a random pairwise interchange, the new sequence is accepted when $P(\text{accept}) > r$, where r is randomly sampled from the uniform distribution $U(0, 1)$.

Step 4. Stopping condition.

Our preliminary trials indicate that the quality of the schedule is stable after $200n$ iterations, where n is the number of jobs.

Table 3: Performance of the branch-and-bound and SA algorithms (n = 15, pro = 0.4).

Pro	τ	R	a	β	Branch-and-bound algorithm				SA_1		SA_2		SA_3	
					CPU time		Number of nodes		Error percentages					
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	0.40	0.40	1.100	−1.100	311.88	414.22	11758285	15903115	0.837	1.546	0.887	1.652	0.293	0.483
				−1.010	191.71	349.53	7136396	13128691	0.456	0.679	0.745	1.130	0.541	1.062
				−1.001	196.36	358.32	7305845	13462940	0.519	0.683	0.491	0.745	0.397	0.537
			1.010	−1.100	287.40	379.73	10866955	14623008	0.321	0.367	0.821	1.332	0.279	0.467
				−1.010	175.58	319.29	6557095	12024063	0.543	0.917	0.347	0.504	0.443	0.911
				−1.001	178.59	324.07	6665084	12209228	0.497	0.661	0.915	1.382	0.359	0.655
			1.001	−1.100	285.39	376.85	10796044	14522042	0.855	1.659	1.354	2.175	0.220	0.281
				−1.010	174.07	316.36	6503448	11918702	0.401	0.694	1.049	1.572	0.379	0.764
				−1.001	177.12	321.43	6613663	12115285	0.320	0.576	0.313	0.531	0.401	0.804
		0.50	1.100	−1.100	257.34	384.96	9612742	14644726	1.302	2.391	0.531	0.862	0.474	0.639
				−1.010	206.07	398.32	7644147	15064816	0.744	1.464	0.588	1.103	0.267	0.370
				−1.001	205.08	404.74	7648212	15313361	1.871	3.131	0.438	0.558	0.433	0.707
			1.010	−1.100	239.89	359.34	8982129	13695175	1.282	1.814	0.821	1.005	0.600	1.058
				−1.010	189.79	368.05	7057756	13951685	1.599	2.800	1.104	2.457	0.612	1.004
				−1.001	187.82	370.71	7020711	14059100	0.580	1.139	0.729	1.177	0.305	0.446
			1.001	−1.100	237.98	356.54	8913945	13593996	0.803	1.032	1.179	1.838	0.451	0.742
				−1.010	188.22	365.16	7000983	13842236	0.633	1.160	1.384	2.292	0.945	2.301
				−1.001	186.16	367.78	6960609	13950622	0.999	1.431	0.214	0.254	0.236	0.457
		0.60	1.100	−1.100	226.20	413.41	8376937	15502069	0.799	1.222	0.921	1.162	0.507	0.894
				−1.010	193.13	404.47	7149216	15170536	1.688	1.715	1.225	1.728	0.636	0.847
				−1.001	196.52	409.14	7277271	15334414	0.781	1.150	0.665	1.036	0.736	0.988
			1.010	−1.100	213.76	391.38	7940293	14704311	1.473	2.165	0.742	1.324	0.371	0.619
				−1.010	179.59	377.57	6671217	14223196	0.831	0.898	1.027	1.720	0.526	0.744
				−1.001	182.23	380.25	6770069	14300776	1.536	2.013	0.802	1.033	0.511	0.739
			1.001	−1.100	212.06	388.42	7877739	14594634	0.999	1.440	0.665	1.477	0.328	0.477
				−1.010	180.39	378.71	6706222	14252125	1.393	1.856	0.981	1.287	0.632	0.764
				−1.001	180.54	376.59	6707971	14165604	0.962	1.708	1.115	2.115	0.580	0.853
	0.50	0.40	1.100	−1.100	59.79	92.20	2340463	3674295	0.742	1.792	0.599	1.025	0.334	0.513
				−1.010	90.67	113.76	3549507	4515071	0.599	0.900	0.461	0.727	0.264	0.388
				−1.001	92.71	115.83	3626112	4594477	0.440	0.675	0.393	0.663	0.261	0.351
			1.010	−1.100	54.94	84.53	2162741	3386252	0.581	0.882	0.424	0.726	0.187	0.339
				−1.010	82.68	104.10	3254738	4152699	0.479	0.703	0.728	1.174	0.235	0.437
				−1.001	84.48	105.83	3322651	4219459	2.067	5.768	0.738	1.153	0.142	0.179
			1.001	−1.100	54.44	83.79	2144562	3358398	0.196	0.280	0.575	0.964	0.274	0.430
				−1.010	81.88	103.15	3224787	4117010	0.760	1.991	0.586	1.682	0.138	0.224
				−1.001	83.65	104.87	3291685	4183898	0.281	0.436	0.496	0.936	0.177	0.290
		0.50	1.100	−1.100	83.54	109.89	3264865	4371726	0.624	0.706	0.660	0.789	0.350	0.522
				−1.010	82.99	112.28	3200949	4429808	1.369	2.181	0.660	1.275	0.215	0.348
				−1.001	83.65	115.22	3232620	4539750	1.125	1.877	1.135	2.213	0.153	0.205
			1.010	−1.100	76.35	100.18	2999791	3999506	0.423	0.698	0.837	1.304	0.240	0.269
				−1.010	76.27	102.66	2954127	4065085	1.511	2.811	1.110	1.802	0.416	1.179
				−1.001	77.07	105.75	2990752	4183939	0.973	1.764	1.021	1.797	0.412	1.167
			1.001	−1.100	75.65	99.35	2974524	3968512	1.006	1.300	0.245	0.309	0.346	0.479
				−1.010	75.59	101.72	2929144	4029183	1.105	1.705	0.672	1.040	0.745	1.588
				−1.001	76.26	104.60	2960124	4138909	0.786	1.267	0.804	1.519	0.442	1.168
		0.60	1.100	−1.100	97.45	122.49	3776266	4802973	1.069	1.723	0.649	0.835	0.273	0.393
				−1.010	67.32	119.31	2539341	4508240	1.642	2.544	1.020	1.764	0.243	0.349
				−1.001	68.51	121.41	2582654	4586272	0.708	0.920	1.773	2.473	0.406	0.452

Table 3: Continued.

Pro	τ	R	a	β	Branch-and-bound algorithm CPU time Mean	Std	Number of nodes Mean	Std	SA_1 Error percentages Mean	Std	SA_2 Mean	Std	SA_3 Mean	Std
			1.010	−1.100	89.87	112.61	3489778	4415326	1.060	1.675	0.695	1.240	0.286	0.411
				−1.010	63.04	112.01	2390213	4253672	1.072	1.535	1.014	1.487	0.604	1.236
				−1.001	63.99	113.55	2423954	4309165	1.790	3.392	1.329	2.338	0.405	0.753
			1.001	−1.100	89.25	111.59	3467072	4375682	0.662	1.008	0.744	0.983	0.361	0.630
				−1.010	62.73	111.02	2379099	4216867	1.321	1.547	1.916	2.870	0.545	1.139
				−1.001	63.45	112.61	2404690	4275428	1.255	1.815	1.098	1.540	0.690	1.367
	0.60	0.40	1.100	−1.100	5.85	7.71	236881	324192	0.296	0.468	0.224	0.323	0.283	0.505
				−1.010	5.16	9.04	209445	379285	0.329	0.464	0.291	0.539	0.175	0.331
				−1.001	5.16	9.12	210367	382778	0.384	0.530	0.403	0.553	0.231	0.498
			1.010	−1.100	5.44	7.26	221007	305841	0.295	0.405	0.146	0.226	0.289	0.553
				−1.010	4.68	8.14	190945	342880	0.763	2.524	0.354	0.540	0.189	0.295
				−1.001	4.66	8.23	190746	346282	0.312	0.426	0.328	0.502	0.232	0.440
			1.001	−1.100	5.40	7.24	219717	305198	0.392	0.510	0.299	0.354	0.173	0.306
				−1.010	4.64	8.06	189163	339587	0.361	0.627	0.360	0.618	0.226	0.397
				−1.001	4.62	8.16	189126	343494	0.928	2.476	0.467	0.805	0.113	0.241
		0.50	1.100	−1.100	6.94	11.68	284387	494376	0.477	0.920	0.423	0.860	0.206	0.347
				−1.010	9.15	14.39	373361	598325	0.587	1.147	0.589	1.004	0.402	1.021
				−1.001	9.32	14.58	380130	606005	1.007	1.916	0.949	1.703	0.257	0.463
			1.010	−1.100	6.31	10.69	259327	453058	0.502	0.637	0.389	0.784	0.228	0.344
				−1.010	8.33	13.23	341436	552785	1.149	1.823	0.394	0.628	0.139	0.191
				−1.001	8.49	13.40	347732	559407	0.687	1.390	0.730	1.267	0.150	0.291
			1.001	−1.100	6.25	10.59	256783	448847	0.621	1.179	0.247	0.368	0.183	0.337
				−1.010	8.24	13.11	338060	547960	0.329	0.561	0.680	1.277	0.112	0.182
				−1.001	8.40	13.28	344399	554855	0.445	1.069	1.128	2.085	0.151	0.324
		0.60	1.100	−1.100	10.29	16.81	417314	689695	0.410	0.900	0.796	1.484	0.174	0.333
				−1.010	14.72	19.92	606522	827274	0.225	0.496	0.303	0.348	0.179	0.276
				−1.001	15.05	20.21	620128	838524	0.965	2.173	0.872	2.173	0.238	0.361
			1.010	−1.100	9.34	15.30	380584	630421	0.599	1.508	0.636	1.551	0.267	0.519
				−1.010	13.41	18.26	554783	760674	0.865	2.444	0.231	0.319	0.141	0.308
				−1.001	13.70	18.52	566713	770556	1.279	3.789	1.013	3.830	0.111	0.210
			1.001	−1.100	9.26	15.17	377334	625018	0.509	1.005	0.630	0.744	0.282	0.528
				−1.010	13.29	18.10	550084	754492	1.052	2.524	0.591	0.707	0.104	0.132
				−1.001	13.58	18.37	561776	764270	0.919	2.456	1.241	3.799	0.261	0.455
Average					97.89	164.20	3701438	6277631	0.832	1.464	0.730	1.253	0.335	0.588

3.4. Details of the Branch-and-Bound Algorithm. We use the depth-first search in the branching procedure and assign jobs in a forward manner starting with the first position. In the searching tree, we adopt a branch and systematically work down the tree until we either eliminate it by virtue of the dominant properties and the lower bounds or reach its final node, in which case this sequence either replaces the initial solution or is eliminated. We summarize the main steps in the following.

3.4.1. The Branch-and-Bound Algorithm

Step 1. Apply the best solution obtained from the three proposed simulated annealing algorithms as the initial solution for the branch-and-bound algorithm.

Step 2. Use the dominant rules 1–5 to eliminate the dominated partial sequences.

Step 3. Compute the lower bound of the total weighted completion time of the first agent for the unscheduled partial sequences or the total weighted completion time of the first agent for the completed sequences. If the lower bound for an unscheduled partial sequence is greater than that of the initial solution, eliminate that node and all the nodes beyond it in the branch. If the value of the completed sequence is less than that of the initial solution, replace it as the new solution. Otherwise, eliminate it.

Step 4. Repeat Steps 2 and 3 until all the branches are explored.

4. Computational Experiments

We conducted extensive computational experiments to evaluate the efficiency of the branch-and-bound algorithm and the performance of the three simulated annealing algorithms.

Table 4: Performance of the branch-and-bound and SA algorithms (n = 15, pro = 0.6).

					Branch-and-bound algorithm				SA_1		SA_2		SA_3	
Pro	τ	R	a	β	CPU time		Number of nodes				Error percentages			
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.6	0.40	0.40	1.100	−1.100	508.76	747.32	15421452	22053238	0.962	1.783	0.916	1.513	0.151	0.221
				−1.010	419.13	702.28	12621515	20637904	0.944	1.289	0.655	1.047	0.516	0.838
				−1.001	423.19	705.68	12735894	20730667	0.864	1.318	0.479	0.725	0.453	0.721
			1.010	−1.100	464.44	681.30	14073410	20085625	0.556	0.892	0.396	0.477	0.421	0.871
				−1.010	386.03	646.22	11628829	18977908	1.008	1.105	0.885	1.240	0.158	0.233
				−1.001	387.96	639.10	11682914	18744229	1.092	2.516	0.810	1.574	0.248	0.474
			1.001	−1.100	460.68	675.83	13961961	19925977	0.459	0.816	0.786	1.068	0.329	0.833
				−1.010	382.96	639.75	11537878	18783161	1.160	2.383	0.759	1.344	0.288	0.527
				−1.001	384.61	632.89	11584157	18561838	1.290	2.378	0.538	1.226	0.232	0.481
		0.50	1.100	−1.100	444.01	741.94	13626115	22119158	0.878	1.042	0.949	1.204	0.399	0.667
				−1.010	405.16	754.23	12130822	22295569	0.458	0.568	0.486	0.672	0.241	0.277
				−1.001	404.01	756.97	12077970	22363737	0.864	2.460	1.275	3.130	0.218	0.328
			1.010	−1.100	409.60	686.34	12555127	20418894	0.713	0.937	0.648	0.968	0.600	1.020
				−1.010	372.23	685.16	11147650	20223236	0.667	0.883	0.429	0.645	0.336	0.431
				−1.001	372.98	691.25	11157930	20400814	0.846	0.881	0.988	1.254	0.420	0.469
			1.001	−1.100	405.58	679.26	12430454	20200090	0.845	1.118	0.705	0.787	0.163	0.225
				−1.010	371.52	687.89	11126894	20304074	0.743	0.970	0.796	1.162	0.222	0.314
				−1.001	370.77	689.06	11089959	20330734	1.036	1.723	0.844	0.984	0.216	0.228
		0.60	1.100	−1.100	407.95	829.01	12323340	24833584	0.477	0.681	0.558	0.713	0.173	0.274
				−1.010	428.20	868.38	12887492	25900159	0.322	0.471	0.732	1.121	0.338	0.491
				−1.001	423.31	850.13	12725974	25328476	0.944	0.908	0.493	0.828	0.324	0.451
			1.010	−1.100	371.63	747.82	11207421	22320678	1.601	3.184	0.469	1.084	0.147	0.271
				−1.010	384.51	766.39	11551885	22789194	0.784	1.413	0.410	0.668	0.166	0.236
				−1.001	387.24	774.19	11631830	23028425	0.788	1.386	0.514	0.841	0.168	0.240
			1.001	−1.100	369.58	746.26	11142589	22266752	0.446	0.570	1.112	2.195	0.229	0.396
				−1.010	380.45	757.19	11429538	22512411	0.489	0.591	1.124	1.760	0.230	0.329
				−1.001	381.94	761.87	11468577	22651834	0.633	0.831	0.521	0.692	0.152	0.236
	0.50	0.40	1.100	−1.100	152.89	208.18	4712381	6298903	0.599	1.252	0.439	0.590	0.242	0.411
				−1.010	134.40	206.96	4074096	6124008	0.559	0.843	0.669	0.919	0.311	0.582
				−1.001	134.81	207.42	4083266	6129901	0.365	0.507	0.521	0.845	0.237	0.527
			1.010	−1.100	136.30	180.49	4209770	5461139	0.520	0.850	0.375	0.559	0.257	0.466
				−1.010	121.37	183.77	3689527	5448570	0.501	0.968	0.576	0.846	0.374	0.664
				−1.001	121.14	182.92	3677737	5412926	0.470	0.787	0.590	0.844	0.163	0.257
			1.001	−1.100	135.09	178.19	4174596	5394367	0.224	0.239	0.406	0.611	0.232	0.453
				−1.010	120.38	182.07	3661654	5401469	0.899	1.257	0.604	0.969	0.315	0.747
				−1.001	120.00	181.39	3645728	5371004	0.536	0.712	0.585	0.862	0.280	0.460
		0.50	1.100	−1.100	144.86	255.28	4384365	7682740	0.587	1.003	0.817	1.479	0.184	0.358
				−1.010	145.08	245.30	4385018	7249908	0.727	1.340	0.297	0.467	0.218	0.521
				−1.001	133.52	240.54	4089473	7123797	0.477	0.789	0.784	1.369	0.274	0.754
			1.010	−1.100	129.39	222.62	3909542	6663581	0.724	1.174	0.855	1.298	0.331	0.781
				−1.010	128.34	212.22	3874419	6246018	1.114	1.876	1.283	1.748	0.198	0.435
				−1.001	118.00	207.86	3606887	6129381	0.704	0.987	0.822	1.424	0.226	0.476
			1.001	−1.100	128.07	219.79	3869681	6575772	0.215	0.208	0.614	1.188	0.160	0.190
				−1.010	127.12	210.58	3838378	6199175	0.548	1.256	0.731	1.386	0.239	0.472
				−1.001	116.43	204.61	3558106	6030835	1.064	1.816	0.919	1.717	0.111	0.338
		0.60	1.100	−1.100	159.25	305.68	4859947	9196852	0.533	0.741	0.935	1.465	0.258	0.714
				−1.010	142.16	277.02	4328607	8163288	0.417	0.676	0.571	0.712	0.138	0.199
				−1.001	141.57	275.89	4305544	8117557	0.724	0.597	0.381	0.678	0.202	0.322

Table 4: Continued.

					Branch-and-bound algorithm				SA_1		SA_2		SA_3	
Pro	τ	R	a	β	CPU time		Number of nodes		Error percentages					
					Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
			1.010	−1.100	141.16	266.80	4292168	7969042	1.210	2.149	0.640	1.138	0.288	0.739
				−1.010	126.62	238.58	3845431	6989210	1.435	3.493	1.171	3.340	0.079	0.141
				−1.001	126.70	238.03	3845492	6967975	0.523	0.745	0.959	2.018	0.044	0.112
			1.001	−1.100	139.34	263.09	4235574	7853433	0.859	1.420	0.629	1.110	0.191	0.568
				−1.010	125.43	235.35	3810783	6895185	0.870	1.285	0.385	0.569	0.045	0.084
				−1.001	125.08	234.49	3795435	6861682	0.553	1.263	0.563	1.247	0.110	0.167
	0.60	0.40	1.100	−1.100	28.56	30.78	904301	959846	0.332	0.719	0.777	1.560	0.647	1.437
				−1.010	28.18	34.87	899501	1104145	0.285	0.463	0.220	0.386	0.269	0.536
				−1.001	29.79	34.57	956497	1095900	0.650	1.593	0.394	0.805	0.090	0.145
			1.010	−1.100	25.62	27.50	815854	864274	0.741	1.199	0.594	1.163	0.450	1.150
				−1.010	25.00	30.48	802771	973960	1.041	2.791	0.214	0.289	0.141	0.189
				−1.001	26.43	30.25	853055	968158	0.424	0.648	0.200	0.265	0.108	0.139
			1.001	−1.100	25.29	27.15	805707	853584	0.304	0.445	0.502	0.886	0.122	0.178
				−1.010	24.70	30.06	793335	961321	0.881	2.390	0.813	2.275	0.105	0.160
				−1.001	26.15	29.84	844493	955428	0.845	1.978	0.576	0.928	0.186	0.203
		0.50	1.100	−1.100	37.75	48.89	1206598	1540373	0.205	0.261	0.181	0.302	0.095	0.199
				−1.010	39.24	51.55	1246699	1622670	0.426	0.637	0.289	0.306	0.167	0.299
				−1.001	39.60	51.93	1257479	1634715	0.318	0.558	0.496	0.822	0.117	0.176
			1.010	−1.100	33.68	43.19	1080679	1369768	0.547	1.110	0.109	0.141	0.090	0.101
				−1.010	34.55	44.66	1101296	1415690	0.240	0.413	0.762	1.152	0.168	0.399
				−1.001	34.69	44.80	1105004	1420683	0.532	0.737	0.280	0.323	0.239	0.479
			1.001	−1.100	33.38	42.89	1071473	1360381	0.334	0.669	0.810	1.721	0.118	0.296
				−1.010	34.08	43.99	1086439	1395117	0.585	0.916	0.383	0.587	0.183	0.424
				−1.001	34.24	44.12	1091361	1400239	0.320	0.531	0.550	1.006	0.177	0.413
		0.60	1.100	−1.100	51.00	72.66	1619576	2279361	0.427	0.609	0.301	0.360	0.234	0.456
				−1.010	48.94	73.35	1532443	2267957	0.687	0.778	0.367	0.596	0.157	0.362
				−1.001	48.77	73.14	1524128	2255341	0.530	0.750	0.351	0.477	0.238	0.387
			1.010	−1.100	45.80	64.27	1454993	2015529	0.342	0.579	0.472	0.701	0.247	0.650
				−1.010	43.98	64.28	1379773	1998789	0.467	0.735	0.473	0.653	0.203	0.326
				−1.001	43.52	63.35	1361289	1959494	0.542	0.742	0.332	0.513	0.410	0.634
			1.001	−1.100	45.40	63.51	1442718	1991271	0.344	0.400	0.650	0.924	0.146	0.206
				−1.010	43.38	63.29	1360805	1967468	0.479	0.758	0.611	0.744	0.146	0.244
				−1.001	42.97	62.32	1344046	1927063	0.361	0.658	0.730	1.147	0.294	0.578
Average					190.83	332.10	5785994	9867958	0.655	1.104	0.615	1.029	0.232	0.429

We coded all the algorithms in Fortran using the Compaq Visual Fortran version 6.6 and performed the experiments on a personal computer powered by an Intel Core2 Quad CPU 2.66 GHz with 4 GB of RAM and operating under Windows XP. The job processing times was generated from a uniform distribution over the integers 1–100. The weights of the jobs from the first agent were generated from another uniform distribution over the integers 1–100. In addition, the due date d_k of J_k of AG_1 was generated from a uniform distribution over the integers between $T(1-\tau-R/2)$ and $T(1-\tau+R/2)$, where T is the total normal processing time of the n jobs, that is, $T=\sum_{i=1}^{n} p_i$ as proposed by Fisher [33]. τ took the values 0.4, 0.5, and 0.6, while R took the values 0.4, 0.5, and 0.6. We fixed the proportions of the jobs of agent AG_1 at pro = 0.4 and 0.6 in the tests.

For the branch-and-bound algorithm, we recorded the average and standard deviation of the number of nodes as well as the average and standard deviation of the execution time (in seconds). For the SA heuristics, we recorded the mean and standard deviation percentage errors. We calculate the percentage error of a solution produced by a heuristic algorithm as follows:

$$\frac{V - V^*}{V^*}, \qquad (17)$$

where V and V^* are the total weighted completion time of the heuristic and the optimal solutions, respectively. We did not record the computational time of the heuristic algorithms because for all of the algorithms it was less than a second CPU time in generating solutions.

We conducted the computational experiments in two parts. In the first part of the experiments, we fixed the number of jobs at 10 and 15. We set the learning index at 1.001, 1.01, and 1.1, and the deteriorating index at −1.1, −1.01, and −1.001. As a result, we examined 324 experimental situations. We randomly generated a set of 20 instances for each situation.

Table 5: RPD of SA algorithms ($n = 25$).

Pro	R	a	β	$\tau = 0.4$						$\tau = 0.5$						$\tau = 0.6$					
				SA_1		SA_2		SA_3		SA_1		SA_2		SA_3		SA_1		SA_2		SA_3	
				Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	0.4	1.100	−1.100	1.276	0.984	1.451	1.549	0.116	0.105	1.252	1.239	0.848	1.056	0.069	0.059	0.981	0.860	1.018	1.303	0.073	0.084
			−1.010	2.237	3.030	1.220	1.135	0.202	0.222	1.171	1.545	1.062	1.081	0.147	0.221	1.319	1.701	1.235	1.045	0.063	0.089
			−1.001	1.327	1.370	1.320	1.232	0.137	0.177	1.442	1.389	1.295	1.792	0.156	0.236	1.381	1.328	1.529	2.077	0.050	0.065
		1.010	−1.100	0.851	1.083	1.134	1.097	0.232	0.209	1.057	1.122	1.039	1.162	0.150	0.307	1.833	3.356	1.026	1.067	0.095	0.132
			−1.010	1.442	1.442	1.356	1.360	0.163	0.162	1.280	1.100	1.707	1.473	0.060	0.090	1.266	1.500	0.905	0.905	0.096	0.166
			−1.001	1.251	1.288	1.343	1.579	0.166	0.136	1.246	1.219	1.014	1.238	0.111	0.135	1.119	0.939	1.040	1.151	0.046	0.056
		1.001	−1.100	1.164	1.648	0.694	0.785	0.306	0.291	1.608	2.032	0.845	1.239	0.151	0.310	1.610	2.162	0.988	0.958	0.077	0.121
			−1.010	1.226	1.265	1.938	1.310	0.208	0.261	1.547	1.447	1.443	1.432	0.121	0.200	0.950	1.099	0.988	1.014	0.088	0.128
			−1.001	1.127	1.132	1.309	1.533	0.245	0.316	0.978	1.244	1.078	0.837	0.120	0.161	0.921	0.828	1.282	1.554	0.103	0.154
	0.5	1.100	−1.100	1.279	1.509	1.655	1.523	0.193	0.305	1.023	0.814	1.100	0.819	0.076	0.081	1.069	0.696	0.666	0.421	0.091	0.153
			−1.010	1.406	2.069	1.301	1.285	0.206	0.317	1.476	1.692	1.976	1.672	0.096	0.159	0.785	1.046	0.910	1.003	0.086	0.130
			−1.001	1.688	2.166	1.767	1.884	0.256	0.296	1.383	1.223	1.436	1.185	0.127	0.195	1.407	1.444	1.236	1.421	0.056	0.059
		1.010	−1.100	1.540	1.576	1.256	1.211	0.127	0.167	0.969	0.844	1.713	1.460	0.265	0.455	0.995	0.707	0.721	0.836	0.065	0.093
			−1.010	2.135	2.079	2.476	2.687	0.127	0.166	1.398	0.922	1.069	1.186	0.110	0.240	1.320	1.600	0.984	1.009	0.111	0.161
			−1.001	1.572	2.298	1.697	1.159	0.184	0.223	1.476	1.193	1.381	1.002	0.100	0.204	1.227	2.268	0.696	1.131	0.137	0.224
		1.001	−1.100	1.689	1.690	1.636	1.768	0.143	0.191	1.773	2.098	1.440	1.429	0.100	0.109	0.920	1.592	0.871	1.733	0.060	0.066
			−1.010	1.811	1.276	1.275	1.484	0.167	0.195	1.515	1.000	1.207	1.194	0.140	0.303	1.204	1.728	1.271	1.913	0.070	0.111
			−1.001	1.977	2.812	1.330	1.354	0.134	0.177	1.547	1.560	1.746	1.463	0.167	0.335	1.476	1.614	1.305	1.717	0.094	0.142
	0.6	1.100	−1.100	1.364	1.807	1.177	0.992	0.231	0.379	1.603	1.655	1.387	1.473	0.111	0.225	1.433	1.494	1.084	1.241	0.142	0.196
			−1.010	2.173	3.049	2.219	2.145	0.176	0.265	1.338	1.514	2.078	2.272	0.100	0.150	1.182	1.204	1.034	1.010	0.132	0.273
			−1.001	2.136	2.370	1.742	1.657	0.127	0.170	1.853	1.322	1.712	1.909	0.078	0.098	0.839	0.793	0.962	1.013	0.081	0.100
		1.010	−1.100	1.898	1.644	1.936	2.001	0.171	0.230	1.481	1.649	1.543	1.681	0.185	0.342	0.755	0.548	0.855	1.082	0.104	0.158
			−1.010	2.004	2.399	0.903	0.734	0.121	0.125	2.320	2.198	1.635	1.670	0.179	0.342	0.901	1.147	0.952	1.035	0.181	0.429
			−1.001	1.678	1.944	1.885	2.021	0.080	0.068	1.602	1.614	2.044	2.312	0.052	0.039	0.916	0.856	0.735	0.738	0.084	0.114
		1.001	−1.100	1.144	0.949	1.375	1.479	0.182	0.468	1.678	1.463	1.971	1.689	0.062	0.057	0.720	0.605	0.868	0.850	0.137	0.217
			−1.010	3.540	3.913	2.176	1.990	0.080	0.080	2.338	2.018	1.620	1.828	0.176	0.372	1.035	1.056	0.913	0.985	0.117	0.238
			−1.001	2.201	1.911	2.537	3.343	0.109	0.130	2.001	1.998	1.986	1.543	0.070	0.081	1.116	1.083	0.897	1.308	0.084	0.123
0.6	0.4	1.100	−1.100	1.350	1.721	1.661	1.605	0.186	0.142	0.873	0.883	0.806	0.757	0.302	0.600	0.572	0.338	0.484	0.259	0.120	0.142
			−1.010	2.082	2.202	1.762	1.940	0.208	0.231	1.461	1.621	1.312	1.793	0.156	0.172	0.779	1.217	0.786	0.831	0.134	0.220
			−1.001	1.085	1.635	1.410	2.158	0.207	0.269	1.587	1.412	0.681	0.567	0.188	0.245	0.804	0.614	0.693	0.670	0.148	0.223
		1.010	−1.100	0.814	0.901	1.219	1.720	0.192	0.234	0.884	0.826	0.791	0.749	0.202	0.337	0.717	0.491	0.519	0.449	0.097	0.091
			−1.010	2.097	2.738	1.851	2.535	0.229	0.255	1.393	1.303	1.444	1.262	0.104	0.108	0.738	0.676	0.705	0.791	0.079	0.111
			−1.001	1.563	1.479	1.750	2.206	0.163	0.252	1.402	1.544	1.064	0.906	0.153	0.220	0.891	1.148	0.638	0.928	0.113	0.156
		1.001	−1.100	1.734	1.894	1.375	1.568	0.224	0.255	0.807	0.709	1.012	1.378	0.196	0.303	1.025	1.557	0.479	0.309	0.106	0.120
			−1.010	1.943	1.995	1.199	1.521	0.231	0.393	1.234	1.251	1.574	2.098	0.094	0.106	0.681	0.505	1.006	1.360	0.123	0.141
			−1.001	1.322	1.699	1.694	2.141	0.210	0.232	0.980	0.753	0.931	0.725	0.120	0.139	0.629	0.766	1.072	1.252	0.067	0.090
	0.5	1.100	−1.100	1.744	2.154	1.746	1.783	0.193	0.239	1.001	0.912	0.905	0.710	0.055	0.048	0.602	0.615	0.859	0.643	0.185	0.225
			−1.010	1.667	1.727	0.977	0.852	0.218	0.292	0.911	0.975	1.187	1.459	0.275	0.315	0.975	0.742	0.658	0.535	0.123	0.128
			−1.001	1.563	2.255	1.449	1.930	0.252	0.414	1.310	1.307	1.282	1.488	0.156	0.209	0.667	0.489	0.650	0.480	0.178	0.176
		1.010	−1.100	1.504	1.421	1.525	1.352	0.275	0.511	1.320	1.253	1.166	1.319	0.131	0.249	0.731	0.607	0.774	0.561	0.114	0.188
			−1.010	1.958	3.191	1.043	0.955	0.111	0.178	1.224	1.692	0.985	0.759	0.121	0.248	0.687	0.760	0.728	0.625	0.146	0.213
			−1.001	1.702	2.288	1.771	1.651	0.139	0.164	1.584	1.506	1.310	1.067	0.100	0.146	0.806	0.844	0.857	0.803	0.140	0.235
		1.001	−1.100	1.747	1.258	2.122	2.037	0.090	0.123	1.304	1.274	1.528	2.095	0.120	0.198	0.860	0.585	0.620	0.623	0.152	0.312
			−1.010	1.817	2.022	2.203	2.099	0.105	0.192	0.944	0.910	1.855	1.977	0.155	0.247	1.008	1.120	0.856	0.787	0.160	0.195
			−1.001	1.783	1.698	1.428	1.175	0.149	0.150	1.500	1.726	1.126	1.490	0.111	0.130	1.002	1.021	1.005	0.718	0.091	0.126

Table 5: Continued.

Pro	R	a	β	τ = 0.4						τ = 0.5						τ = 0.6					
				SA_1		SA_2		SA_3		SA_1		SA_2		SA_3		SA_1		SA_2		SA_3	
				Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.6		1.100	−1.100	2.001	2.296	2.037	1.611	0.173	0.259	1.719	2.278	1.438	1.670	0.200	0.191	0.797	0.696	0.928	0.704	0.134	0.224
			−1.010	1.933	1.773	1.871	2.590	0.179	0.277	0.608	0.490	1.129	1.337	0.238	0.269	1.196	1.227	1.045	1.245	0.144	0.180
			−1.001	2.745	2.391	1.614	2.092	0.149	0.400	1.527	1.599	1.223	1.689	0.215	0.435	1.063	1.289	1.134	1.092	0.175	0.319
		1.010	−1.100	2.176	2.464	2.824	2.469	0.131	0.191	1.229	1.199	0.999	0.781	0.151	0.237	0.739	0.692	0.780	0.578	0.136	0.187
			−1.010	1.813	2.478	2.536	2.103	0.158	0.218	1.864	1.945	0.913	0.774	0.156	0.205	1.438	2.337	0.958	1.094	0.099	0.140
			−1.001	2.404	2.671	1.896	2.346	0.129	0.194	1.136	1.136	1.946	1.998	0.187	0.377	1.045	0.954	1.299	1.311	0.116	0.172
		1.001	−1.100	1.952	2.320	1.588	1.909	0.144	0.165	1.148	0.953	1.114	1.362	0.232	0.589	1.016	0.955	0.758	0.606	0.154	0.227
			−1.010	2.130	3.139	2.191	2.195	0.120	0.165	1.518	2.083	1.488	1.477	0.184	0.245	1.402	1.157	1.490	1.744	0.205	0.374
			−1.001	1.813	2.394	2.407	3.146	0.100	0.156	1.722	2.895	1.792	1.745	0.316	0.461	1.342	2.332	1.205	0.912	0.153	0.276
Average				1.733	1.980	1.653	1.740	0.171	0.229	1.380	1.399	1.340	1.380	0.146	0.232	1.017	1.129	0.925	0.989	0.113	0.170

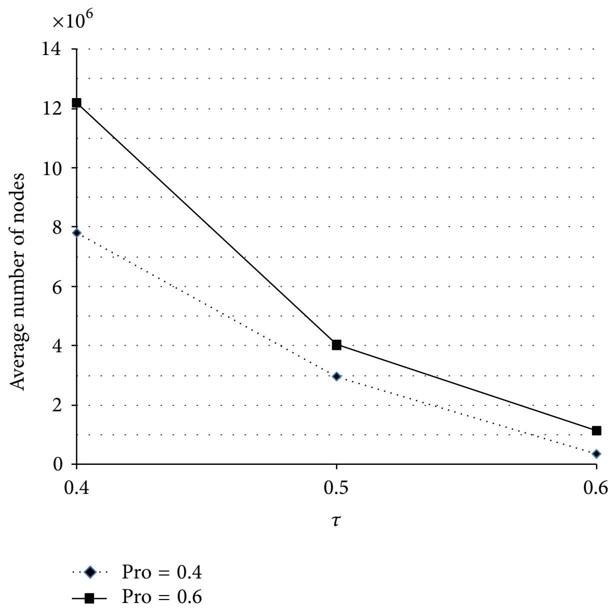

Figure 1: Performance of the branch-and-bound algorithm ($n = 15$).

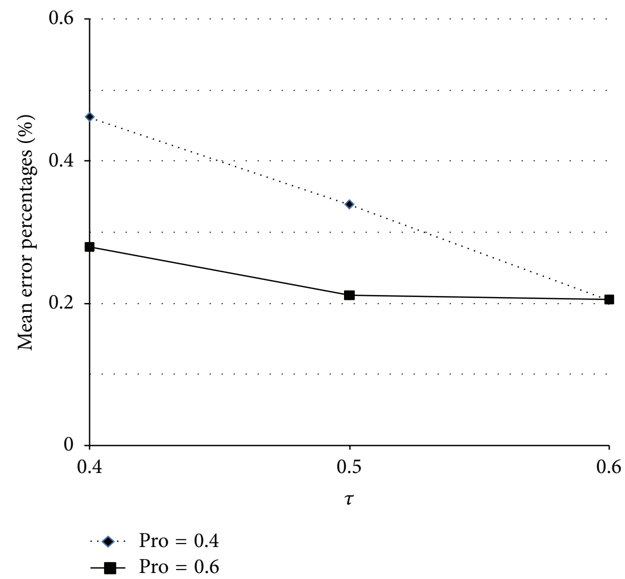

Figure 2: Performance of SA_3 algorithm ($n = 15$).

Tables 1, 2, 3, and 4 report the results, which include the CPU time (mean and standard deviation) and the number of nodes for the branch-and-bound algorithm.

As for the performance of the branch-and-bound algorithm, we see from Figure 1 and Table 1 that the number of nodes and the mean CPU time increase when n becomes bigger. When the proportion of the jobs of agent AG_1 is smaller (pro = 0.4), the number of nodes is obviously more than that with a bigger one as shown in Figure 2. In addition, the instances with a bigger value of τ (e.g., $\tau = 0.5$ and 0.6) are easier to solve than those with a smaller one. Tables 1–4 also show that the mean error percentages of SA_3 (between 0.104% and 0.335%) are lower than those of SA_1 (between 0.277% and 0.832%) and those of SA_2 (between 0.272% and 0.730%.) Moreover, the standard deviations of the percentage error follow the same pattern. In particular, the standard deviations of the percentage error of SA_1, SA_2, and SA_3 were between 0.870% and 1.464%, 0.841% and 1.253%, and 0.352% and 0.588%, respectively. This indicates that the performance of SA_3 is better than that of the other two. In addition, the means error percentages of SA_3 are affected by the parameters τ and pro. In particular, it was up to 0.462% when the value of τ became smaller (at $\tau = 0.4$) or pro was 0.6 (see Figure 2 and Tables 1–4). However, the mean error percentages of SA_3 were all below 0.5%. The results also indicate that the impact of the learning or deteriorating effect is insignificant.

In the second part of the experiments, we further assessed the performance of the proposed SA heuristics in solving instances with large numbers of jobs. Given the fact that it is not easy to generate a feasible solution when n becomes lager, we set n at 25 and 30, fixed the parameters at the values as those used for the small job-size design, and set the proportion of the jobs of agent AG_1 at pro = 0.4 and 0.6 in the tests. We set the learning index at 1.001, 1.01, and 1.1, and the deteriorating index at −1.1, −1.01, and −1.001.

Table 6: RPD of SA algorithms ($n = 30$).

Pro	R	a	β	$\tau = 0.4$						$\tau = 0.5$						$\tau = 0.6$					
				SA_1		SA_2		SA_3		SA_1		SA_2		SA_3		SA_1		SA_2		SA_3	
				Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	0.4	1.100	−1.100	1.952	1.494	1.207	1.665	0.175	0.299	1.029	1.242	1.282	1.427	0.197	0.155	1.756	1.621	1.865	1.393	0.048	0.004
			−1.010	1.587	1.564	1.636	1.632	0.155	0.195	1.820	2.590	0.962	0.721	0.098	0.071	0.889	1.282	0.864	1.280	0.096	0.096
			−1.001	1.319	0.981	2.090	1.764	0.098	0.065	1.685	1.785	1.821	2.058	0.074	0.060	1.296	1.362	2.258	2.128	0.034	0.012
		1.010	−1.100	2.153	2.579	1.025	1.349	0.120	0.073	1.373	1.427	1.176	1.723	0.200	0.302	1.592	0.730	1.442	0.570	0.047	0.003
			−1.010	1.638	1.050	1.800	1.358	0.091	0.052	1.522	1.427	1.269	1.071	0.058	0.030	0.987	1.372	1.222	1.423	0.042	0.002
			−1.001	1.584	1.474	2.365	2.117	0.182	0.194	1.908	1.890	2.039	1.723	0.067	0.049	1.328	1.644	1.945	2.166	0.073	0.050
		1.001	−1.100	1.469	1.803	1.275	1.131	0.153	0.184	1.094	1.873	0.819	0.876	0.232	0.379	1.644	0.345	2.299	1.679	0.044	0.007
			−1.010	2.084	1.844	1.850	1.604	0.134	0.118	1.485	1.341	2.046	2.087	0.091	0.065	1.074	0.745	1.102	0.673	0.042	0.001
			−1.001	1.578	1.500	1.335	1.154	0.191	0.207	1.904	1.470	2.063	1.501	0.091	0.091	1.875	2.372	1.266	1.360	0.057	0.045
	0.5	1.100	−1.100	1.926	1.485	2.517	2.784	0.255	0.326	1.779	1.904	1.338	1.006	0.086	0.116	4.414	1.022	4.207	1.019	0.013	0.000
			−1.010	2.995	2.671	1.908	1.616	0.073	0.054	3.123	3.100	1.779	1.688	0.101	0.141	2.411	1.820	1.682	1.282	0.029	0.019
			−1.001	2.289	1.520	2.681	2.919	0.091	0.077	2.099	1.855	1.584	1.273	0.074	0.103	1.525	1.139	1.846	1.358	0.028	0.021
		1.010	−1.100	1.862	0.830	1.833	1.433	0.085	0.045	2.345	1.848	1.756	1.358	0.057	0.056	4.545	1.028	4.147	1.569	0.014	0.000
			−1.010	2.452	2.384	2.123	1.575	0.080	0.059	1.627	1.389	2.396	1.642	0.084	0.116	1.756	1.379	1.399	1.252	0.035	0.024
			−1.001	2.411	3.870	2.445	1.533	0.103	0.140	2.492	2.733	2.383	2.552	0.170	0.294	1.572	1.234	1.697	1.562	0.028	0.020
		1.001	−1.100	2.389	2.254	1.257	0.844	0.127	0.157	1.865	1.405	1.897	1.379	0.101	0.153	4.135	1.387	3.754	1.535	0.014	0.002
			−1.010	2.252	2.064	2.346	1.877	0.108	0.131	2.940	3.129	2.195	1.840	0.062	0.065	1.725	1.207	1.702	1.401	0.058	0.113
			−1.001	2.178	2.644	2.907	2.485	0.133	0.206	1.721	1.428	1.880	1.554	0.093	0.114	1.691	1.277	1.660	1.316	0.027	0.020
	0.6	1.100	−1.100	2.655	1.739	2.609	1.320	0.123	0.162	1.922	1.954	1.690	1.029	0.067	0.100	1.476	1.147	1.686	1.359	0.054	0.068
			−1.010	2.629	3.121	1.962	1.714	0.227	0.359	2.513	1.823	2.135	1.607	0.059	0.063	1.664	1.617	1.468	0.993	0.175	0.340
			−1.001	2.795	2.703	2.914	2.464	0.184	0.333	2.250	1.989	1.686	1.369	0.074	0.084	1.785	1.341	1.526	1.373	0.096	0.220
		1.010	−1.100	2.666	2.523	2.282	2.228	0.299	0.672	1.998	1.461	2.376	2.048	0.041	0.029	1.783	1.217	1.481	0.897	0.028	0.013
			−1.010	2.547	2.320	2.031	1.620	0.140	0.258	1.691	1.245	2.389	1.564	0.154	0.292	1.433	1.595	1.413	1.449	0.045	0.103
			−1.001	3.222	4.776	2.353	1.606	0.109	0.157	2.284	1.822	2.229	2.088	0.153	0.319	1.579	1.479	1.605	1.344	0.056	0.095
		1.001	−1.100	2.718	2.308	1.921	2.134	0.165	0.234	2.938	3.221	1.884	1.316	0.036	0.023	1.495	1.326	1.855	1.681	0.027	0.012
			−1.010	2.765	2.250	2.040	1.549	0.167	0.266	2.531	1.881	1.988	1.331	0.105	0.171	1.480	1.855	1.439	1.261	0.101	0.221
			−1.001	2.293	1.809	2.085	2.169	0.106	0.141	2.008	1.436	2.117	1.815	0.092	0.204	1.634	1.306	2.103	2.015	0.027	0.035
0.6	0.4	1.100	−1.100	1.790	1.717	1.963	1.480	0.196	0.316	1.317	1.323	1.438	1.134	0.168	0.215	1.103	0.873	1.224	1.503	0.120	0.150
			−1.010	1.576	1.189	1.932	1.973	0.153	0.254	1.209	1.033	1.663	1.971	0.165	0.229	1.082	1.051	1.298	1.515	0.097	0.121
			−1.001	1.499	1.558	1.503	1.429	0.138	0.196	1.023	1.077	1.430	1.186	0.148	0.213	1.034	1.219	1.162	1.237	0.132	0.199
		1.010	−1.100	1.062	1.047	1.711	1.688	0.132	0.104	1.807	1.929	1.382	1.176	0.097	0.099	1.049	1.029	0.771	0.459	0.078	0.104
			−1.010	1.430	1.360	2.181	1.577	0.179	0.243	1.163	0.981	1.425	1.567	0.198	0.152	1.077	1.046	1.087	0.781	0.153	0.216
			−1.001	1.653	1.575	1.642	1.209	0.175	0.325	2.061	1.980	1.234	0.936	0.153	0.209	0.802	0.450	1.050	1.589	0.059	0.063
		1.001	−1.100	1.677	1.309	1.435	1.103	0.163	0.162	1.668	1.498	1.509	1.172	0.096	0.107	1.036	0.680	1.155	1.061	0.097	0.137
			−1.010	2.283	3.104	1.890	1.456	0.160	0.215	1.127	1.248	1.186	1.012	0.182	0.190	1.000	0.969	0.795	0.681	0.097	0.138
			−1.001	1.430	1.056	1.753	1.323	0.112	0.108	1.451	1.557	1.420	1.468	0.108	0.132	0.855	0.586	1.009	1.221	0.113	0.156
	0.5	1.100	−1.100	1.491	1.206	2.235	2.075	0.158	0.174	1.392	1.234	1.064	0.735	0.093	0.165	1.119	1.175	1.088	0.896	0.109	0.161
			−1.010	1.855	1.334	1.459	1.149	0.202	0.351	1.524	1.257	1.491	1.145	0.136	0.199	0.960	0.920	1.090	0.911	0.045	0.083
			−1.001	2.134	2.407	1.960	1.166	0.088	0.089	2.920	2.779	2.011	2.250	0.081	0.128	1.839	2.162	1.273	1.324	0.103	0.160
		1.010	−1.100	1.529	1.089	1.982	1.666	0.144	0.132	1.804	2.638	1.738	1.065	0.077	0.127	1.192	0.886	1.215	0.863	0.065	0.096
			−1.010	1.387	1.264	2.743	2.388	0.136	0.196	1.981	1.360	2.021	2.360	0.060	0.066	1.574	1.088	1.178	1.023	0.070	0.175
			−1.001	1.553	1.338	2.148	1.645	0.159	0.207	1.628	1.438	1.804	1.503	0.128	0.186	1.127	0.815	0.897	0.630	0.094	0.158
		1.001	−1.100	1.810	1.748	2.087	1.721	0.148	0.162	1.255	1.135	1.607	1.678	0.070	0.099	0.926	0.608	0.957	0.535	0.076	0.116
			−1.010	1.884	1.540	1.389	1.377	0.165	0.222	1.895	1.748	1.912	2.004	0.105	0.182	1.315	1.020	1.121	0.679	0.041	0.091
			−1.001	2.128	2.026	2.142	1.984	0.205	0.318	1.332	0.997	1.849	1.607	0.146	0.266	1.204	0.610	1.132	0.654	0.079	0.213
	0.6	1.100	−1.100	3.035	3.678	1.830	1.792	0.134	0.175	1.677	1.339	2.040	2.353	0.060	0.109	1.550	1.480	1.133	1.085	0.083	0.181
			−1.010	1.638	1.292	1.804	1.323	0.106	0.210	1.567	1.294	1.968	1.327	0.062	0.104	1.171	0.782	0.959	0.456	0.043	0.115
			−1.001	2.426	2.173	1.749	1.554	0.058	0.058	2.240	1.126	1.943	1.369	0.042	0.075	1.154	0.846	0.979	0.699	0.064	0.118

Table 6: Continued.

Pro	R	a	β	$\tau = 0.4$						$\tau = 0.5$						$\tau = 0.6$					
				SA_1		SA_2		SA_3		SA_1		SA_2		SA_3		SA_1		SA_2		SA_3	
				Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
		1.010	−1.100	2.195	2.167	1.602	1.189	0.125	0.179	1.509	0.962	1.273	1.742	0.051	0.064	1.217	0.839	1.056	0.819	0.030	0.047
			−1.010	2.499	2.162	1.141	0.841	0.095	0.148	1.860	1.068	1.845	1.465	0.031	0.046	1.072	0.641	1.563	1.337	0.030	0.095
			−1.001	1.837	1.715	2.048	1.688	0.065	0.106	1.779	1.381	1.897	1.207	0.068	0.109	1.187	1.283	1.567	1.010	0.017	0.040
		1.001	−1.100	1.705	1.238	3.076	3.955	0.161	0.278	1.958	2.291	1.453	1.162	0.050	0.085	1.154	0.564	1.337	1.554	0.032	0.076
			−1.010	1.935	2.334	1.617	1.652	0.040	0.029	2.130	1.566	2.181	1.594	0.049	0.104	1.766	1.831	1.803	1.169	0.017	0.028
			−1.001	1.300	0.989	1.756	1.308	0.213	0.384	2.072	1.963	2.212	1.811	0.037	0.062	1.325	1.322	1.260	0.863	0.032	0.057
Average				2.021	1.911	1.955	1.692	0.142	0.195	1.839	1.683	1.744	1.512	0.100	0.136	1.527	1.160	1.520	1.183	0.061	0.090

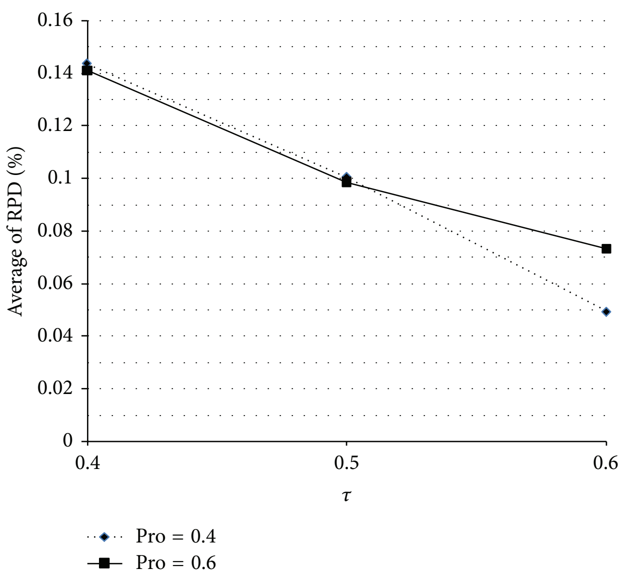

Figure 3: Performance of SA_3 algorithm ($n = 30$).

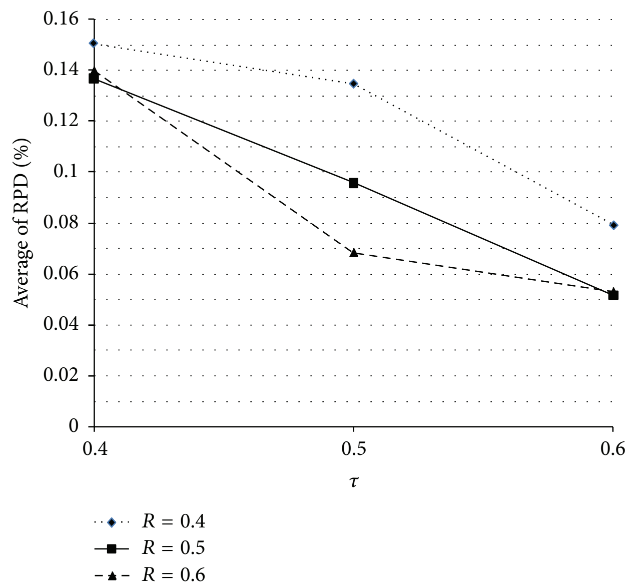

Figure 4: Performance of SA_3 algorithms ($n = 30$).

As a result, we examined 324 experimental situations. We randomly generated a set of 20 instances for each situation. We obtained the relative percentage with respect to the best known solution among SA_1, SA_2, and SA_3 for each instance. We also recorded the mean execution time and the mean relative percentage deviation for each SA heuristic. We calculate the relative percentage deviation RPD as follows:

$$\frac{SA_i - SA^*}{SA^*}, \tag{18}$$

where SA_i is the value of the objective function generated by SA_i, and $SA^* = \min\{SA_i,\ i = 1, 2, 3\}$ is the smallest value of the objective function obtained from the three SA heuristics. Tables 5-6 report the results.

As shown in Tables 5-6, we see that the mean RPDs of SA_1 and SA_2 become bigger as n increases. In general, the mean RPDs of SA_3 are lower than those of SA_1 and SA_2. Furthermore, Figure 3 and Tables 5-6 show that the RDP mean of SA_3 becomes smaller as τ becomes larger (e.g., $\tau = 0.6$). However, when $\tau = 0.6$, the mean RPD of SA_3 at pro = 0.4 was less than that at pro = 0.6. Figure 4 further shows that when $n = 30$, SA_3 has a smaller RPD value when τ or R becomes larger. In addition, all of the mean RPDs of SA_3 were less than 0.2%. In sum, SA_3 outperforms the other two SA heuristics in terms of both solution accuracy and performance stability.

5. Conclusions

In this paper, we study a two-agent single-machine scheduling problem with learning and deteriorating effects simultaneously. The objective is to minimize the total weighted completion time of the jobs of the first agent with the restriction that no tardy job is allowed for the second agent. We develop a branch-and-bound algorithm incorporating several dominant properties and a lower bound to derive the optimal solution. We also propose three simulated annealing algorithms to obtain near-optimal solutions. The computational results show that with the help of the proposed heuristic initial solutions, the branch-and-bound algorithm performs well in terms of the number of nodes and execution time, when the number of jobs is fewer than or equal to 15. Moreover, the computational experiments also show that the proposed SA_3 performs well since its mean error percentage was less than 0.4% for all the tested cases. Further research lies in the devising of efficient and effective methods to solve the problem with significantly larger numbers of jobs.

Acknowledgment

The author would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this paper under the Contract nos. NSC 101-2410-H-264-004 and NSC 102-2410-H-264-003.

References

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

[2] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[3] 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.

[4] B. Alidaee and N. K. Womer, "Scheduling with time dependent processing times: review and extensions," *Journal of the Operational Research Society*, vol. 50, no. 7, pp. 711–720, 1999.

[5] T. C. E. Cheng, Q. Ding, and B. M. T. Lin, "A concise survey of scheduling with time-dependent processing times," *European Journal of Operational Research*, vol. 152, no. 1, pp. 1–13, 2004.

[6] T. C. E. Cheng and G. Wang, "Single machine scheduling with learning effect considerations," *Annals of Operations Research*, vol. 98, no. 1-4, pp. 273–290, 2000.

[7] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[8] J.-B. Wang, "A note on scheduling problems with learning effect and deteriorating jobs," *International Journal of Systems Science*, vol. 37, no. 12, pp. 827–833, 2006.

[9] J.-B. Wang, "Single-machine scheduling problems with the effects of learning and deterioration," *Omega*, vol. 35, no. 4, pp. 397–402, 2007.

[10] X. Wang and T. C. E. Cheng, "Single-machine scheduling with deteriorating jobs and learning effects to minimize the makespan," *European Journal of Operational Research*, vol. 178, no. 1, pp. 57–70, 2007.

[11] T. C. E. Cheng, C.-C. Wu, and W.-C. Lee, "Some scheduling problems with deteriorating jobs and learning effects," *Computers and Industrial Engineering*, vol. 54, no. 4, pp. 972–982, 2008.

[12] T. C. E. Cheng, C. T. Ng, and J. J. Yuan, "Multi-agent scheduling on a single machine with max-form criteria," *European Journal of Operational Research*, vol. 188, no. 2, pp. 603–609, 2008.

[13] M. D. Toksarı, D. Oron, and E. Güner, "Single machine scheduling problems under the effects of nonlinear deterioration and time-dependent learning," *Mathematical and Computer Modelling*, vol. 50, no. 3-4, pp. 401–406, 2009.

[14] X. Huang, J.-B. Wang, L.-Y. Wang, W.-J. Gao, and X.-R. Wang, "Single machine scheduling with time-dependent deterioration and exponential learning effect," *Computers and Industrial Engineering*, vol. 58, no. 1, pp. 58–63, 2010.

[15] S.-R. Cheng, "A single-machine two-agent scheduling problem by GA approach," *Asia-Pacific Journal of Operational Research*, vol. 29, no. 2, Article ID 1250013, 2012.

[16] D.-C. Li and P.-H. Hsu, "Solving a two-agent single-machine scheduling problem considering learning effect," *Computers & Operations Research*, vol. 39, no. 7, pp. 1644–1651, 2012.

[17] A. Agnetis, D. Pacciarelli, and A. Pacifici, "Multi-agent single machine scheduling," *Annals of Operations Research*, vol. 150, pp. 3–15, 2007.

[18] K. R. Baker and J. C. Smith, "A multiple-criterion model for machine scheduling," *Journal of Scheduling*, vol. 6, no. 1, pp. 7–16, 2003.

[19] A. Agnetis, P. B. Mirchandani, D. Pacciarelli, and A. Pacifici, "Scheduling problems with two competing agents," *Operations Research*, vol. 52, no. 2, pp. 229–242, 2004.

[20] J. J. Yuan, W. P. Shang, and Q. Feng, "A note on the scheduling with two families of jobs," *Journal of Scheduling*, vol. 8, pp. 537–542, 2005.

[21] T. C. E. Cheng, C. T. Ng, and J. J. Yuan, "Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs," *Theoretical Computer Science*, vol. 362, no. 1–3, pp. 273–281, 2006.

[22] C. T. Ng, T. C. E. Cheng, and J. J. Yuan, "A note on the complexity of the problem of two-agent scheduling on a single machine," *Journal of Combinatorial Optimization*, vol. 12, no. 4, pp. 387–394, 2006.

[23] Y. Yin, S. R. Cheng, T. C. E. Cheng, W. H. Wu, and C. C. Wu, "Two-agent single-machine scheduling with release times and deadlines," *International Journal of Shipping and Transport Logistics*, 2012.

[24] T. C. E. Cheng, Y.-H. Chung, S.-C. Liao, and W.-C. Lee, "Two-agent singe-machine scheduling with release times to minimize the total weighted completion time," *Computers & Operations Research*, vol. 40, no. 1, pp. 353–361, 2013.

[25] P. Liu and L. Tang, "Two-agent scheduling with linear deteriorating jobs on a single machine," in *Computing and Combinatorics*, vol. 5092 of *Lecture Notes in Computer Science*, pp. 642–650, Springer, Berlin, Germany, 2008.

[26] T. C. E. Cheng, S.-R. Cheng, W.-H. Wu, P.-H. Hsu, and C.-C. Wu, "A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations," *Computers and Industrial Engineering*, vol. 60, no. 4, pp. 534–541, 2011.

[27] C. C. Wu, S. K. Huang, and W. C. Lee, "Two-agent scheduling with learning consideration," *Computers & Industrial Engineering*, vol. 61, no. 4, pp. 1324–1335, 2011.

[28] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *American Association for the Advancement of Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[29] R. Stahlbock and S. Voß, "Efficiency considerations for sequencing and scheduling of double-rail-mounted gantry cranes at maritime container terminals," *International Journal of Shipping and Transport Logistics*, vol. 2, no. 1, pp. 95–123, 2010.

[30] C.-L. Li and K.-W. Pang, "An integrated model for ship routing and berth allocation," *International Journal of Shipping and Transport Logistics*, vol. 3, no. 3, pp. 245–260, 2011.

[31] M. Nawaz, E. E. Enscore Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[32] D. Ben-Arieh and O. Maimon, "Annealing method for PCB assembly scheduling on two sequential machines," *International*

Journal of Computer Integrated Manufacturing, vol. 5, pp. 361–367, 1992.

[33] M. L. Fisher, "A dual algorithm for the one-machine scheduling problem," *Mathematical Programming*, vol. 11, no. 3, pp. 229–251, 1976/77.