Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 754826, 9 pages
http://dx.doi.org/10.1155/2013/754826

Research Article

Competitive Two-Agent Scheduling with Learning Effect and Release Times on a Single Machine

Der-Chiang Li[1] and Peng-Hsiang Hsu[1,2]

[1] *Department of Industrial and Information Management, National Cheng Kung University, No. 1, University Road, Tainan 70101, Taiwan*
[2] *Department of Business Administration, Kang-Ning Junior College, Taipei, Taiwan*

Correspondence should be addressed to Peng-Hsiang Hsu; r38991036@mail.ncku.edu.tw

Received 16 May 2013; Accepted 2 July 2013

Academic Editor: Yunqiang Yin

The learning effect has gained much attention in the scheduling research recently, where many researchers have focused their problems on only one optimization. This study further addresses the scheduling problem in which two agents compete to perform their own jobs with release times on a common single machine with learning effect. The aim is to minimize the total weighted completion time of the first agent, subject to an upper bound on the maximum lateness of the second agent. We propose a branch-and-bound approach with several useful dominance properties and an effective lower bound for searching the optimal solution and three simulated-annealing algorithms for the near-optimal solutions. The computational results show that the proposed algorithms perform effectively and efficiently.

1. Introduction

In traditional scheduling problems, most studies assumed that each of the operations of all job processing times is known and fixed. But in some circumstances, the jobs processing times are affected by learning effect. The "learning effect" is the phenomenon that unit costs reduce as firms produce more of a product and gain knowledge or experience. Biskup [1] and Cheng and Wang [2] first brought the phenomenon of learning effect into scheduling field. Afterwards the learning effect is getting much to pay attention in the scheduling research in the last decade such as Mosheiov [3], Mosheiov and Sidney [4], Bachman and Janiak [5], Lee and Wu [6], Kuo and Yang [7], and Koulamas and Kyparisis [8]. Biskup [9] provided the most recent learning effects survey paper in scheduling research. More recent studies involving learning effects were by Wang et al. [10], Janiak and Rudek [11], Yin et al. [12], Toksari and Güner [13], Wang et al. [14], Lee et al. [15], J.-B. Wang and C. Wang [16], Wu et al. [17], Zhang et al. [18], Yin et al. [19], and Yang et al. [20], Yang et al. [21], Wang et al. [22], J.-B. Wang and J.-J. Wang [23], Cheng et al. [24], and so forth.

In addition, most research assumed that all jobs met a single criterion. But jobs might come from different agents. There might be multiple agents who compete on the same resources, and each agent has its own objective. This concept was first initiated and considered into scheduling field by Baker and Smith [25] and Agnetis et al. [26]. After that time, many researchers focused on multiagent in scheduling field. However, little research has been done on scheduling problem with learning effect and multiagent. Liu et al. [27] studied the optimal polynomial time algorithms to solve a single-machine scheduling problem with two-agent and position-dependent processing time aging and learning effect. The objective is to find a schedule that minimizes the total completion time of the first agent with a maximum cost limit of the second agent. Cheng et al. [28] investigated a two-agent single-machine scheduling problem with a truncated sum-of-processing times-based learning effect and developed algorithms to minimize the total weighted completion time of the jobs of the first agent, subject to the restriction that no tardy job is allowed for the second agent. Li and Hsu [29] investigated the job scheduling problem of two agents competing for the usage of a common single machine with learning effect. The objective is to minimize the total weighted completion time of both agents with the restriction that the makespan of either agent cannot exceed an upper bound. Wu et al. [30] address a two-agent single-machine scheduling problem with the co-existing sum-of-processing times-based deteriorating and learning effects. The goal is to minimize the total weighted completion time of the jobs of the first agent given that no tardy job is allowed for the second agent.

The previous both scheduling issues were yet relatively unexplored. In this paper, we therefore study a two-agent scheduling problem with position-based learning competing on a common single machine and further consider each job a different release time. The objective is to minimize the total weighted completion time of first agents, subject to the constraint that the maximum lateness of second agent the jobs cannot exceed an upper bound. In the classical scheduling notation, the problem can be notated by a triplet as $1 \mid r_j^X, p_{jk}^X = p_j^X k^a \mid \sum w_j^A C_j^A : L_{\max}^B \leq Q$. As shown by Agnetis et al. [26], it is a Binary NP-hard problem, even without release time and learning effect $(1 \parallel \sum w_j^A C_j^A : f_{\max}^B)$.

The rest of this paper is organized as follows: the problem formulation is introduced in the next section. The branch-and-bound and simulated-annealing algorithms are employed to find the optimal solution and the near-optimal solutions, respectively. Dominance properties and a lower bound are developed to be used in the branch-and-bound algorithm in Section 3. The details of simulated-annealing algorithm are described in Section 4, and the computational experiment results are given in Section 5. In the last section, some conclusions and extensions are presented.

2. Problem Formulation

In this section, we describe a formal definition of the model as there are n jobs from two competing agents (X = agent A or agent B) to be scheduled. The number of jobs in the two sets is recorded as n^A and n^B, such that $n = n^A + n^B$. The processing time and weight of job j are known and denoted as p_j^X and w_j^X, respectively. Based on the learning effect, the actual processing time p_{jk}^X of job j changes with position k and learning ratio a, that is $p_{jk}^X = p_j^X k^a$, where $a < 0$ and $k = 1, 2, \ldots, n$.

Furthermore, we consider the problem of scheduling a set S of n independent jobs with integer release times (r_j^X) on single machine. The goal is to minimize the weighted sum of completion times of the jobs from agent A, subject to the constraint that the maximum lateness of the jobs from agent B is not more than a given upper bound Q.

3. Branch-and-Bound Algorithm

The computational complexity of this problem does not consider ready time and learning effect $(1 \parallel \sum w_j C_j^A : f_{\max}^B)$, which is showed a Binary NP-hard problem by Agnetis et al. [26]. Therefore, the addressed problem here $(1 \mid r_j^X, p_{jk}^X = p_j^X k^a \mid \sum w_j C_j^A : L_{\max}^B)$ is more complex than the problem $(1 \parallel \sum w_j C_j^A : f_{\max}^B)$. We basically try to employ the branch-and-bound algorithm to gain the optimal solution, and for speeding up the searching process, several dominance properties and a lower bound were presented in the following.

3.1. Dominance Property. Suppose that there are two contiguous jobs (J_i^X and J_j^X) in sequence $S_1 = (\pi, J_i^X, J_j^X, \pi^c)$, where π and π^c denote the scheduled and unscheduled partial sequence, respectively. One can perform the contiguous jobs J_i^X and J_j^X interchanges to obtain another sequence $S_2 = (\pi, J_j^X, J_i^X, \pi^c)$. To show that S_1 dominates S_2, it is sufficient to ensure $C_j^X(S_1) \leq C_i^X(S_2)$ or $[w_i^X C_i^X(S_1) + w_j^X C_j^X(S_1)] \leq [w_i^X C_i^X(S_2) + w_j^X C_j^X(S_2)]$. Before proving the proposed properties, we let t denote the completion time of the last job in the scheduled partial sequence π to determine the following properties.

Property 1. If $J_i^X, J_j^X \in J^A$, $\max\{r_j^A, t\} \geq \max\{r_i^A, t\}$, $p_i^A \leq p_j^A$, and $w_i^A > w_j^A$, then S_1 dominates S_2.

Proof. By the definition, the completion times of jobs J_i^A and J_j^A in S_1 and S_2 are, respectively,

$$\begin{aligned} C_i^A(S_1) &= \max\{r_i^A, t\} + p_i^A k^a, \\ C_j^A(S_1) &= \max\{\max\{r_i^A, t\} + p_i^A k^a, r_j^A\} + p_j^A (k+1)^a, \\ C_j^A(S_2) &= \max\{r_j^A, t\} + p_j^A k^a, \\ C_i^A(S_2) &= \max\{\max\{r_j^A, t\} + p_j^A k^a, r_i^A\} + p_i^A (k+1)^a \\ &= \max\{r_j^A, t\} + p_j^A k^a + p_i^A (k+1)^a. \end{aligned} \tag{1}$$

Since $\max\{r_j^A, t\} \geq \max\{r_i^A, t\}$ and $p_i^A \leq p_j^A$, $C_j^A(S_1) \leq C_i^A(S_2)$. Moreover,

$$\begin{aligned} & w_j^A C_j^A(S_2) + w_i^A C_i^A(S_2) - \left(w_i^A C_i^A(S_1) + w_j^A C_j^A(S_1)\right) \\ &= w_j^A \left(\max\{r_j^A, t\} + p_j^A k^a\right) \\ &\quad + w_i^A \left(\max\{r_j^A, t\} + p_j^A k^a + p_i^A (k+1)^a\right) \\ &\quad - w_i^A \left(\max\{r_i^A, t\} + p_i^A k^a\right) \\ &\quad - w_j^A \left(\max\{\max\{r_i^A, t\} + p_i^A k^a, r_j^A\} + p_j^A (k+1)^a\right) \\ &\geq w_j^A \left(\max\{r_j^A, t\} + p_j^A k^a\right) \\ &\quad + w_i^A \left(\max\{r_j^A, t\} + p_j^A k^a + p_i^A (k+1)^a\right) \\ &\quad - w_i^A \left(\max\{r_i^A, t\} + p_i^A k^a\right) \\ &\quad - w_j^A \left(\max\{r_i^A, t\} + p_i^A k^a + p_j^A (k+1)^a\right) \\ &= \left(w_i^A + w_j^A\right)\left(\max\{r_j^A, t\} - \max\{r_i^A, t\}\right) \end{aligned}$$

$$+ w_j^A p_j^A k^a + w_i^A \left(p_j^A k^a + p_i^A (k+1)^a\right) - w_i^A p_i^A k^a - w_j^A \left(p_i^A k^a + p_j^A (k+1)^a\right). \quad (2)$$

By $p_i^A \le p_j^A$ and $w_i^A > w_j^A$, we have

$$w_j^A p_j^A k^a + w_i^A \left(p_j^A k^a + p_i^A (k+1)^a\right) - w_i^A p_i^A k^a - w_j^A \left(p_i^A k^a + p_j^A (k+1)^a\right) > 0, \quad (3)$$

hence $[w_i^A C_i^A(S_1) + w_j^A C_j^A(S_1)] \le [w_i^A C_i^A(S_2) + w_j^A C_j^A(S_2)]$, as required. □

The proofs of Properties 2–4 are omitted since they are similar to that of Property 1.

Property 2. If $J_i^X, J_j^X \in J^A$, $r_j^A \ge \max\{r_i^A, t\} + p_i^A k^a$, and $p_i^A \le p_j^A$, then S_1 dominates S_2.

Property 3. If J_i^X, $J_j^X \in J^B$, $\max\{\max\{r_i^B, t\} + p_i^B k^a, r_j^B\} + p_j^B (k+1)^a - d_j^B \le 0$, and $\max\{r_i^B, t\} + p_i^B k^a - d_i^B \le 0 < \max\{\max\{r_j^B, t\} + p_j^B k^a, r_i^B\} + p_i^B (k+1)^a - d_i^B$, then S_1 dominates S_2.

Property 4. If $J_i^X \in J^A$, $J_j^X \in J^B$, $\max\{r_j^B, t\} \ge \max\{r_i^A, t\}$, $p_i^A \le p_j^B$, and $\max\{\max\{r_i^A, t\} + p_i^A k^a, r_j^B\} + p_j^B (k+1)^a - d_j^B \le Q$, then S_1 dominates S_2.

In addition, let (π, π^c) be a sequence of the jobs where π is the scheduled part with k jobs, and π^c is the unscheduled part with $(n-k)$ jobs. The following property is found for determining the sequence feasibility by the unscheduled J_j^B. Moreover, $C_{[k]}$ is the completion time of the last job in π.

Property 5. If there is a J_j^B in π^c such that $\max\{C_{[k]}, r_j^B\} + p_j^B n^a - d_j^B > Q$, then sequence (π, π^c) is not a feasible solution.

Proof. Since $\max\{C_{[k]}, r_j^B\} + p_j^B n^a - d_j^B > Q$, lateness of the unscheduled jobs J_j^B must exceed the given bound Q. So (π, π^c) is not a feasible sequence. □

The next property is for assigning the unscheduled J_j^A in $(k+1)$th position.

Property 6. If all the unscheduled jobs belong to J^A and there exists an J_j^A such that $\max\{C_{[k]}, r_j^A\} + p_j^A (k+1)^a \le r^*$, where $r^* = \min\{r_i^A\}$ for all jobs $J_i^A \in \pi^c$ and $J_i^A \ne J_j^A$, then job J_j^A may be assigned to the $(k+1)$th position.

3.2. Lower Bound. In addition to the previous properties, we hatch up a lower bound to speed up the building of searching trees in the branch-and-bound algorithm. Assume that PS and US are the two partial scheduling sequences. PS is the scheduled k jobs, and US is the remaining $(n-k)$ unscheduled jobs in which n_1 is agent A jobs and n_2 is agent B jobs, where $n_1 + n_2 = n - k$. The lower bound is obtained by scheduling agent A jobs first and then scheduling agent B jobs in any order. To elaborate this, let $C_{[k]}^X$ be the completion time of the last job in PS; then the completion time for the $(k+1)$th job is

$$\begin{aligned} C_{[k+1]}^A(S) &= \max\left\{C_{[k]}^X(S), r_{[k+1]}^A\right\} + p_{[k+1]}^A (k+1)^a \\ &\ge r_{[k+1]}^A + p_{[k+1]}^A (k+1)^a \\ &\ge r_{[k+1]}^A + p_{[k+1]}^A n^a. \end{aligned} \quad (4)$$

Similarly, the completion time for the $(k+l)$ job is

$$\begin{aligned} C_{[k+l]}^A(S) &= \max\left\{C_{[k+l-1]}^A(S), r_{[k+l]}^A\right\} + p_{[k+l]}^A (k+l)^a \\ &\ge r_{[k+l]}^A + p_{[k+l]}^A n^a, \quad 2 \le l \le n_1. \end{aligned} \quad (5)$$

Hence the lower bound of the partial sequence PS can thus be found as follows:

$$\mathrm{LB} = \sum_{J_j^A \in \mathrm{AS}} w_j^A C_j^A + n^a \sum_{J_j^A \in \mathrm{US}} w_j^A \left(r_j^A + p_j^A\right). \quad (6)$$

4. Simulated-Annealing Algorithm

The simulated-annealing algorithm is one of the meta-heuristic methods to solve large-scaled combinatorial minimization problems [28, 31–34]. It was first described by Kirkpatrick et al. [35] based upon the research of Metropolis et al. [36]. The major advantage of this approach is that it avoids getting trapped in local minima for global optimization by controlling the parameter which influences the probability of accepting a worse solution in the iterative process. Here, we employ SA algorithm to obtain near-optimal solutions as described in the following.

Step 1 (initial feasible solution). An initial feasible sequence was generated by putting agent B in front of agent A for considering the conditionality objective. Thus, we employ the earliest due date (EDD) rule for agent B first, and for agent A, four different rules are employed as EDD rule, shortest processing time (SPT) rule, shortest release time (SRT) rule, and weighted shortest processing time (WSPT) rule; they were denoted by SA_1, SA_2, SA_3, and SA_4, respectively.

Step 2 (adjusting the solution). To improve on the initial schedule, we shift the neighborhood job schedules. The exchange sequence strategy procedures were to choose two different locations randomly and irregularly select one of the three resources (pairwise interchanges, extraction, and forward/backward shifted reinsertions) to ameliorate the quality of the SA.

Step 3 (acceptance probability). If there exists a new schedule that improves the value of the objective, it replaces the previous schedule. Besides, the SA algorithms prevent it to get stuck to local minima with an acceptance probability. The acceptance probability, given in the next equation, is based on the exponential distribution:

$$P(\text{accept}) = \exp(-\lambda \times \Delta\mathrm{WC}), \quad (7)$$

where λ is the control parameter and $\Delta\mathrm{WC}$ is the variation of the objective value. If $P(\text{accept}) > \text{rand}(0,1)$, the new sequence is accepted, otherwise new sequence will be rejected. Ben-Arieh and Maimon [37] suggested that λ in the kth iteration is

$$\lambda = \frac{k}{\beta}, \tag{8}$$

where β is an experimental constant. After preliminary trials, $\beta = 2$ was used in our experiment.

4.1. Stopping Condition Iterations. The SA algorithms were terminated after $300n$ iterations in our preliminary experiments, where n is the number of jobs.

5. Computational Experiments

A computational experiment was conducted to assess the performance of the proposed branch-and-bound algorithm and the accuracy of the SA algorithms. All the algorithms were coded in Compaq Visual Fortran version 6.6 and run on an Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz with 4 GB RAM operating system under Windows 7 environment. The experimental design followed Chu's [38] and Fisher [39] framework, where the normal job processing times were generated randomly from a uniform distribution over the integers between 1 and 100; the release times were from a uniform distribution over the integers $(0, 50.5n\lambda)$, where n was the job size and λ was a control variable; the due dates were from a uniform distribution over the range of integers $T(1-\tau-\gamma/2)$ to $T(1-\tau+\gamma/2)$, where T, τ, and γ are the sum of the processing times of all the jobs $T = \sum_{i=1}^{n} p_i$, the tardiness factor, and the due date range, respectively. Furthermore, the bound Q was fixed at 0 and each agent has half of the total number of jobs.

5.1. The Accuracy of SAs for Small Job Size. First, to assess the accuracy of SA algorithms, the error percentage was calculated as

$$\frac{(\mathrm{SA}_i - \mathrm{OP})}{\mathrm{OP}} \times 100\%, \tag{9}$$

where SA_i $(i = 1,2,3,4,5)$ was the solution obtained from the SA algorithm and OP was the optimal solution of the objective function obtained from the branch-and-bound algorithm. In the first simulation experiment, the job size (n), tardiness factor (τ), the due date range (γ), and learning effect (le) were fixed at $(n,\tau,\gamma,\mathrm{le}) = (12, 0.25, 0.25, 80)$. The values of λ were taken as 0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0, and 3.0. For each situation, 100 replications were randomly generated. The mean and maximum of the error percentages were recorded in Table 1. The results showed that SA_1, SA_2, SA_3, and SA_4 are not affected by the variation of λ. It was observed that the mean error percentage of the SA algorithm was less than 0.1352%. In order to diminish some peculiar worst case of SAs, we combined the four SAs to obtain SA_5 which was the smallest value of the objective function obtained from the SAs. The mean error percentage of the SA_5 was less than 0.0429%.

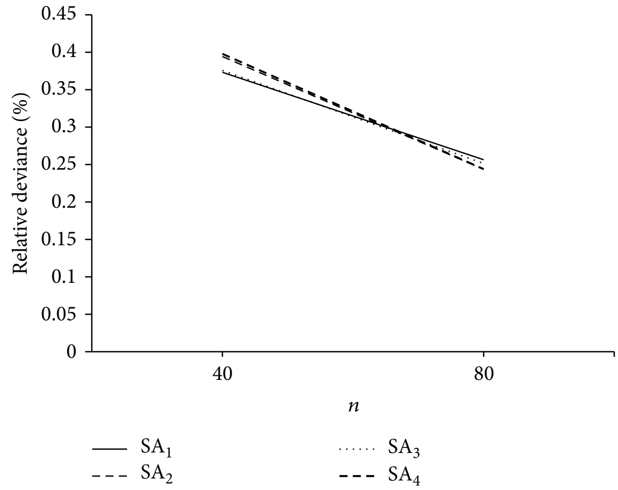

Figure 1: The performance of SA algorithm with respect to n.

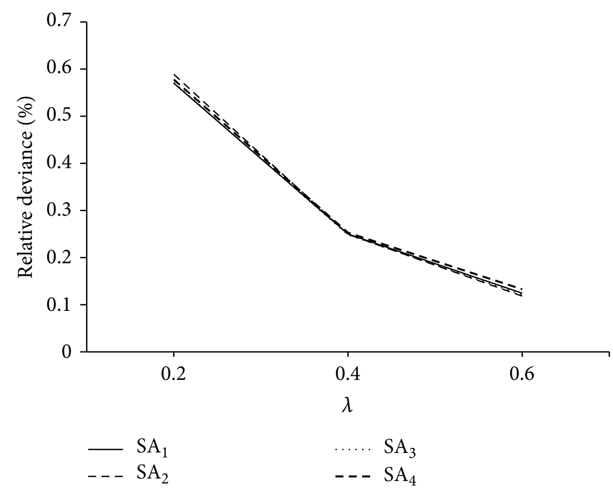

Figure 2: The performance of SA algorithm with respect to λ.

In another simulation experiment, variables were fixed at $(n,\lambda,\mathrm{le}) = (12, 0.4, 80)$. The values of (τ,γ) were taken as $(0.25, 0.25)$, $(0.25, 0.5)$, $(0.25, 0.75)$, $(0.5, 0.25)$, $(0.5, 0.5)$, and $(0.5, 0.75)$. For each situation, 100 replications were randomly generated. The mean and maximum of the percentage errors were also recorded in Table 2. The results were similar to the former experiment. The mean error percentage of the SA_5 was less than 0.0151%. By the two experiment results, it was recommended to combine SAs into SA_5.

5.2. Performance of the Branch-and-Bound Algorithm for Small Job Size. To evaluate the performance of the branch-and-bound algorithm, the mean and maximum numbers of nodes were recorded as well as the computation times (in seconds) to calculate the mean and maximum computation times. In this simulation experiment, the parameters were set as the job size (n = 12 and 14), release times control value

Table 1: The performance of the simulated-annealing algorithms at $(n, \tau, \gamma, \mathrm{le}) = (12, 0.25, 0.25, 80)$.

λ	SA_1		SA_2		SA_3		SA_4		SA_5	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
0.20	0.0252	0.6711	0.0326	1.2731	0.0294	1.2731	0.0239	1.2731	0.0020	0.2014
0.40	0.0539	1.9013	0.0724	1.9013	0.0911	1.9013	0.1352	5.8067	0.0151	1.4364
0.60	0.0433	0.9016	0.0954	5.5725	0.0803	4.4405	0.0919	4.6113	0.0103	0.3001
0.80	0.0595	1.8690	0.0412	0.4131	0.0465	1.6664	0.0543	1.7206	0.0062	0.1241
1.00	0.0710	3.7936	0.1013	3.7936	0.0994	3.7936	0.0907	3.7936	0.0429	3.7936
1.25	0.0423	0.2853	0.1068	5.7572	0.0433	0.4495	0.0371	0.6049	0.0045	0.0665
1.50	0.0359	0.4688	0.0414	0.2745	0.0331	0.3646	0.0331	0.2364	0.0048	0.0651
1.75	0.0671	3.3128	0.0414	0.3602	0.0671	3.3128	0.0413	0.3486	0.0057	0.1112
2.00	0.0358	0.3843	0.0329	0.3610	0.0504	0.3250	0.0314	0.3368	0.0091	0.2887
3.00	0.0355	0.2558	0.0420	0.2762	0.0321	0.2738	0.0316	0.3117	0.0066	0.1416

Table 2: The performance of the simulated-annealing algorithms at $(n, \lambda, \mathrm{le}) = (12, 0.4, 80)$.

τ	γ	SA_1		SA_2		SA_3		SA_4		SA_5	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
0.25	0.25	0.0252	0.6711	0.0326	1.2731	0.0294	1.2731	0.0239	1.2731	0.0020	0.2014
	0.50	0.0539	1.9013	0.0724	1.9013	0.0911	1.9013	0.1352	5.8067	0.0151	1.4364
	0.75	0.0433	0.9016	0.0954	5.5725	0.0803	4.4405	0.0919	4.6113	0.0103	0.3001
0.50	0.25	0.0671	3.3128	0.0414	0.3602	0.0671	3.3128	0.0413	0.3486	0.0057	0.1112
	0.50	0.0358	0.3843	0.0329	0.3610	0.0504	0.3250	0.0314	0.3368	0.0091	0.2887
	0.75	0.0355	0.2558	0.0420	0.2762	0.0321	0.2738	0.0316	0.3117	0.0066	0.1416

(λ = 0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0, and 3.0), tardiness factor (τ = 0.25 and 0.5), the due date range (γ = 0.25 and 0.75), and the learning effect (le = 70% and 90%). For each situation, 100 replications were randomly generated to yield a total of 16000 instances. Table 3 summarized the performance of the branch-and-bound and SA algorithms. It indicated as follows.

First, the mean and maximum numbers of nodes increased as the job size became larger, as λ became smaller, as τ became smaller, as γ became smaller, or as learning effect became strong. The CPU times were observed to increase exponentially as the job size increased. Some of instances could not be solved in the reasonable time. We recorded the number of solvable instances (NSI) if the numbers of nodes were less than 10^8. There were 15,848 solvable instances for all instances. When $(n, \lambda, \tau, \gamma, \mathrm{le}) = (14, 0.2, 0.25, 0.25, 70)$, the least NSI was 73 which implied that the aforesaid parameters affected the nodes. Second, the accuracy of SA algorithms was assessed, and the results presented in Table 3 indicated that the mean error percentage of SA_5 was less than 0.7376%. The performance of error percentage of SA_5 trend was not subject to the parameter influenced.

5.3. The Performance of the SA Algorithms for Large Job Size. On evaluating the accuracy of SA algorithms for large job size, we carried out the jobs size (n = 40 and 80) simulation experiments. The other parameters were taken as (λ = 0.2, 0.4, and 0.6), (τ = 0.25 and 0.5), (γ = 0.25, 0.5, and 0.75), and (le = 70%, 80%, and 90%). For each situation, 100 replications were randomly generated. For evaluating the accuracy of SA algorithms, the mean relative deviance percentage was calculated as

$$\frac{(SA_i - SA_5)}{SA_5} \times 100\%, \qquad (10)$$

where SA_i ($i = 1, 2, 3, 4$) was the solution obtained from SA algorithms and SA_5 was the smallest value of the objective function obtained from the ith SA algorithms. We get the total average for each SA_i algorithm experimental results as depicted in Figures 1–5. In Figure 4, parameters could slightly

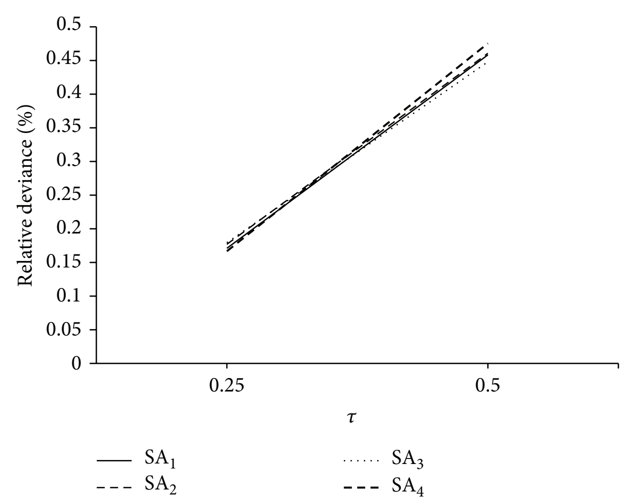

Figure 3: The performance of SA algorithm with respect to τ.

Table 3: The performance of the branch-and-bound and simulated-annealing algorithms in $n = 12$ and $n = 14$.

				$n = 12$ Branch-and-bound algorithm							$n = 14$ Branch-and-bound algorithm						
λ	τ	γ	le	Node Mean	Node Max	Node Mean	Node Max	NSI	SA_5 Mean	SA_5 Max	Node Mean	Node Max	Mean Mean	Mean Max	NSI	SA_5 Mean	SA_5 Max
0.2	0.25	0.25	70	7953180	57938701	60.47	492.31	100	0.0012	0.0524	234798074	490877044	2081.69	4386.73	73	0.0201	0.8312
			90	4740856	27074133	44.13	259.24	100	0.1540	8.7447	130221904	456315410	1437.82	5207.04	93	0.0044	0.2050
		0.75	70	7266869	68305607	56.12	577.05	100	0.0008	0.0672	181473222	499210262	1646.57	4692.44	81	0.0107	0.8222
			90	1669247	17432807	15.03	161.55	100	0.0157	0.6238	74978873	422464059	830.62	4819.28	97	0.2485	20.4500
	0.5	0.25	70	4885377	30357870	37.28	260.68	100	0.0117	1.1747	185730807	467124530	1815.88	4535.62	83	0.0136	0.8498
			90	420242	4048018	4.20	40.48	100	0.6183	14.4734	13870486	163643109	173.24	2134.31	100	0.5811	15.4566
		0.75	70	2168845	35858649	16.14	291.38	100	0.0736	2.7652	95878107	488272101	956.07	5041.92	96	0.1735	6.8935
			90	553857	8010627	5.23	77.77	100	0.2046	11.3446	14254450	139078335	162.34	1740.41	100	0.4214	23.7347
0.4	0.25	0.25	70	7095766	25477833	52.13	195.94	100	0.0032	0.0972	224196847	494946864	2117.99	5289.59	69	0.0057	0.1331
			90	1437139	8454985	13.72	88.31	100	0.0661	5.5203	53829459	368620154	640.41	4722.94	99	0.1643	6.3549
		0.75	70	3266325	34176162	23.38	240.77	100	0.0077	0.4400	139740047	458655160	1351.59	5074.59	92	0.0066	0.2338
			90	899555	7207133	8.27	66.58	100	0.3497	10.3461	25911608	163977599	293.97	2093.06	99	0.4423	18.5009
	0.5	0.25	70	1547759	9729009	11.64	73.84	100	0.0688	2.7329	70707385	375376204	715.40	3852.09	99	0.0359	1.6350
			90	90231	626524	0.89	6.21	100	0.4160	13.1596	4073837	54407366	51.53	721.13	100	0.7376	21.6059
		0.75	70	1351836	10319114	9.67	75.99	100	0.0049	0.1158	53013074	403629962	540.22	4223.36	98	0.0106	0.3034
			90	197252	2130612	1.83	20.01	100	0.3840	14.2713	5725593	47250298	64.77	564.13	100	0.2255	4.8593
0.6	0.25	0.25	70	3862510	16167618	27.75	122.24	100	0.0071	0.1072	171949416	449957311	1674.82	4512.44	85	0.0149	0.1546
			90	639113	2425615	6.18	23.70	100	0.1761	12.3540	29277986	233944079	356.17	2956.13	100	0.1433	3.3191
		0.75	70	2091778	15560184	14.85	119.36	100	0.0055	0.0945	111408553	497252899	1019.55	4674.72	99	0.0284	1.4289
			90	435481	4364092	3.89	42.06	100	0.1133	8.4358	13830333	162070288	152.22	1924.88	100	0.1332	3.4404
	0.5	0.25	70	706061	5309118	5.16	40.39	100	0.0324	2.8857	35336302	316279317	357.65	3325.72	100	0.0107	0.4723
			90	44338	273034	0.41	2.31	100	0.4392	13.6008	1421321	15206208	16.72	195.00	100	0.3227	11.0501
		0.75	70	1087937	14668547	7.95	108.73	100	0.0208	1.0697	27409801	442248409	265.04	4704.04	99	0.0083	0.1406
			90	115643	2408792	1.08	23.57	100	0.1092	8.6434	2959763	25796932	32.88	302.44	100	0.1744	5.1725
0.8	0.25	0.25	70	2228168	11825715	16.03	95.74	100	0.0094	0.1390	115343373	461244567	1167.23	4819.64	93	0.0171	0.1883
			90	330955	1919421	3.23	21.20	100	0.2375	16.7504	11697110	120462234	139.35	1375.25	100	0.2476	4.8784
		0.75	70	1860095	41493678	13.49	321.69	100	0.0076	0.0942	67104475	432826308	646.54	4682.28	98	0.0127	0.1003
			90	362757	3964085	3.36	39.09	100	0.0609	5.0439	14443132	373521531	169.66	4464.73	100	0.0286	0.9466
	0.5	0.25	70	524558	3799338	3.96	29.16	100	0.0061	0.2385	9415219	80038607	95.53	833.19	100	0.0324	2.5209
			90	36666	577706	0.34	5.74	100	0.2641	6.3281	823660	5949505	10.63	83.68	100	0.2739	9.5707
		0.75	70	403985	5962730	2.83	43.99	100	0.0062	0.0928	22181919	380549351	222.19	4369.47	100	0.0098	0.1969
			90	72633	1101780	0.67	10.53	100	0.1546	4.0201	1523540	30875624	18.57	395.32	100	0.1881	8.0855
1.00	0.25	0.25	70	1830472	16744441	13.43	121.90	100	0.0095	0.1007	59443784	330818627	564.87	3015.11	98	0.0133	0.1125
			90	215190	2035038	2.08	20.64	100	0.1032	4.0359	4466746	28799550	51.63	350.30	100	0.0403	2.8493
		0.75	70	1129202	18263919	7.96	128.09	100	0.0065	0.1021	37106372	459794226	335.53	4123.83	99	0.0139	0.1279
			90	228020	2337809	2.06	21.34	100	0.0061	0.1508	8313461	106599701	93.90	1371.25	100	0.0125	0.4674
	0.5	0.25	70	214900	3407180	1.58	25.43	100	0.0068	0.4455	6115145	60562961	57.14	570.64	100	0.0087	0.0862
			90	29404	276253	0.27	2.81	100	0.0589	4.6160	629722	5126101	7.19	58.67	100	0.1737	5.3500
		0.75	70	301944	4639599	2.11	34.49	100	0.0064	0.2709	11218748	269394563	104.08	2817.75	100	0.0090	0.0958
			90	69288	1953549	0.65	19.42	100	0.0102	0.9705	1357448	13996545	15.29	171.25	100	0.1471	4.7189

Table 3: Continued.

λ	τ	γ	le	$n = 12$							$n = 14$						
				Branch-and-bound algorithm				NSI	SA_5		Branch-and-bound algorithm				NSI	SA_5	
				Node		Node					Node		Mean				
				Mean	Max	Mean	Max		Mean	Max	Mean	Max	Mean	Max		Mean	Max
1.25	0.25	0.25	70	1089215	11257092	8.21	95.29	100	0.0090	0.0845	32149788	369070833	326.03	3946.48	100	0.0146	0.1287
			90	121436	1254725	1.16	13.21	100	0.0505	2.4323	5560244	114005664	70.15	1528.11	100	0.0049	0.0518
		0.75	70	442342	2908517	3.06	20.06	100	0.0079	0.0759	21335243	266345547	194.90	2685.73	99	0.0127	0.1111
			90	135323	1799997	1.23	17.18	100	0.0272	1.8397	2880368	22008680	30.82	231.41	100	0.0050	0.0754
	0.5	0.25	70	100924	1251332	0.75	9.97	100	0.0065	0.1067	8748002	445990918	84.10	4346.73	100	0.0054	0.0692
			90	19309	331734	0.18	3.17	100	0.1289	3.6122	415265	4518256	4.64	49.28	100	0.0714	6.6160
		0.75	70	133234	4014823	0.93	27.98	100	0.0073	0.2137	5501857	53869685	49.83	514.11	100	0.0113	0.0758
			90	28751	250665	0.27	2.36	100	0.0037	0.2513	730907	10862579	7.95	117.42	100	0.0049	0.0849
1.50	0.25	0.25	70	664098	9051000	4.84	69.75	100	0.0105	0.1576	29590335	325690474	299.12	3285.10	99	0.0171	0.1274
			90	101999	951368	0.96	8.89	100	0.0099	0.4903	2879670	35619654	35.16	391.36	100	0.0198	1.2207
		0.75	70	460782	6318122	3.27	44.57	100	0.0118	0.0894	14381778	329708866	135.18	3225.48	100	0.0127	0.1105
			90	88765	959182	0.80	8.91	100	0.0040	0.0662	2681801	98041164	29.74	1127.96	100	0.0064	0.0727
	0.5	0.25	70	90684	718580	0.66	5.07	100	0.0055	0.1499	1217974	26068657	11.74	278.99	100	0.0073	0.0970
			90	17842	97492	0.16	0.98	100	0.0534	3.6591	396960	6700590	4.53	77.83	100	0.0851	4.8506
		0.75	70	69694	549161	0.50	3.90	100	0.0041	0.1045	2519744	27555493	22.84	251.41	100	0.0066	0.1271
			90	25354	255908	0.23	2.43	100	0.0138	1.1731	675377	11025856	7.61	135.48	100	0.0389	2.0427
1.75	0.25	0.25	70	346213	12937042	2.56	102.62	100	0.0091	0.0727	10231727	183248629	98.28	1629.76	100	0.0139	0.1443
			90	72082	962963	0.67	8.94	100	0.0192	1.5155	1524548	12214703	18.99	144.91	100	0.0159	0.7236
		0.75	70	155745	1597658	1.12	11.67	100	0.0048	0.0759	8349472	233610726	82.41	2322.54	100	0.0112	0.0788
			90	54346	556872	0.49	4.99	100	0.0045	0.0677	1193311	14839106	13.61	164.88	100	0.0060	0.0762
	0.5	0.25	70	33618	401940	0.26	3.17	100	0.0050	0.1512	1037401	29751952	10.05	286.85	100	0.0061	0.0863
			90	22010	588097	0.21	5.80	100	0.0165	1.3437	277896	2630825	3.28	30.73	100	0.0337	2.5780
		0.75	70	60324	938322	0.43	6.44	100	0.0057	0.0852	1345847	17163412	12.75	169.89	100	0.0076	0.0834
			90	19976	223194	0.18	1.97	100	0.0138	1.1915	450183	4753419	5.18	56.54	100	0.0053	0.1587
2.00	0.25	0.25	70	263896	5361429	1.92	42.43	100	0.0076	0.1149	7966404	294171852	80.90	3238.28	100	0.0122	0.1022
			90	41536	303945	0.37	2.50	100	0.0069	0.1556	1257091	30578774	14.30	353.53	100	0.0257	1.3044
		0.75	70	174184	1725739	1.25	13.27	100	0.0057	0.0808	3265403	32146416	29.53	315.98	100	0.0128	0.1015
			90	48914	669835	0.42	5.74	100	0.0039	0.0705	1195686	15899030	13.15	187.01	100	0.0127	0.5283
	0.5	0.25	70	34243	391787	0.25	3.09	100	0.0043	0.0473	673277	19665442	6.18	181.05	100	0.0090	0.1181
			90	11611	154462	0.11	1.69	100	0.1822	16.9507	187134	2683667	2.05	30.06	100	0.0214	1.5299
		0.75	70	34586	471487	0.24	3.35	100	0.0056	0.0758	1271906	25609588	11.54	234.23	100	0.0098	0.0715
			90	6815	55855	0.06	0.48	100	0.0028	0.0490	225415	3858777	2.49	45.84	100	0.0081	0.3787
3.00	0.25	0.25	70	84306	3685077	0.64	29.47	100	0.0096	0.0978	1998031	52564160	19.32	556.56	100	0.0080	0.0852
			90	26775	344199	0.24	3.65	100	0.0074	0.1978	781780	14099992	9.08	148.56	100	0.0063	0.0869
		0.75	70	78675	2304815	0.57	18.53	100	0.0071	0.1111	1449705	38705731	13.90	405.38	100	0.0104	0.0937
			90	23334	592065	0.20	5.27	100	0.0074	0.0876	318573	3935413	3.55	45.32	100	0.0085	0.0990
	0.5	0.25	70	15785	616369	0.11	4.66	100	0.0028	0.0627	420907	2543861	3.95	26.30	100	0.0046	0.0764
			90	3863	49539	0.03	0.50	100	0.0025	0.0815	108292	614015	1.14	6.43	100	0.1739	2.4366
		0.75	70	11775	162353	0.08	1.18	100	0.0045	0.0894	276874	4007313	2.53	37.96	100	0.0074	0.0642
			90	8133	270480	0.07	2.57	100	0.0033	0.0798	108638	1842254	1.21	21.81	100	0.0018	0.0236

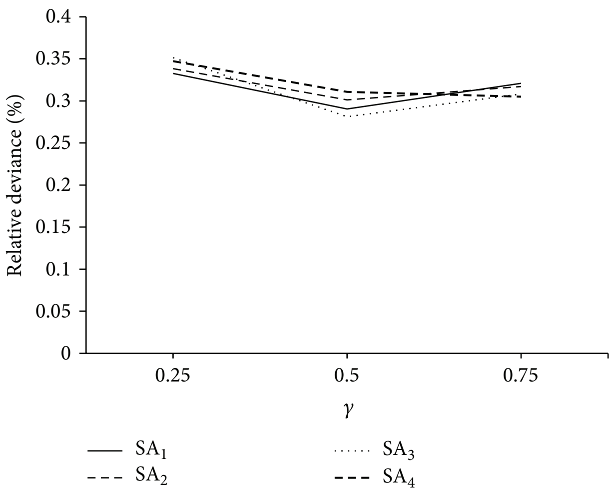

Figure 4: The performance of SA algorithm with respect to γ.

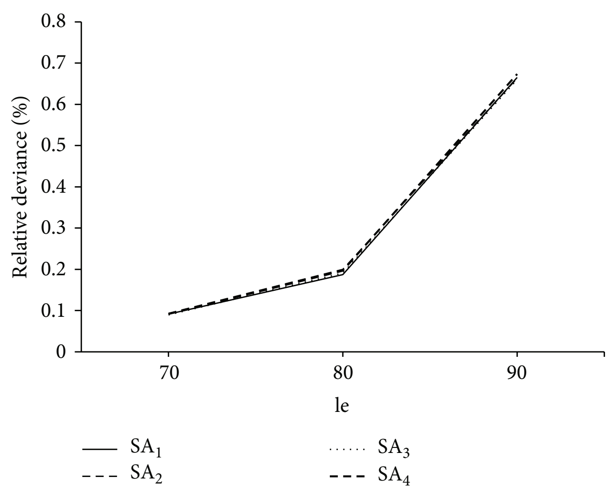

Figure 5: The performance of SA algorithm with respect to le.

affect the performance of SAs algorithm. It was observed that there was no significant difference among the performance of SA algorithms. Besides, the CPU times of the SA algorithms were not recorded since they were completed within one second. Thus, the SA_5 is recommended for solving the large job size method.

6. Conclusions

In this paper, we study the two-agent single-machine scheduling problems in which jobs with arbitrary release times in learning effect condition. The objective is minimizing the total weighted completion time of one agent, subject to an upper bound on the maximum lateness of the second agent. To solve the problem, a branch-and-bound algorithm incorporated with several dominance properties and a lower bound is developed. In addition, simulated-annealing algorithms are also developed to test the accuracy. Computational results show that the branch-and-bound algorithm can find the optimum up to 14 jobs in a reasonable time, and the simulated-annealing algorithm is effective and efficient in obtaining near-optimal solutions. Suggested further research includes considering other optimization criteria or multiobjective optimization to further verify the proposed approach.

References

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

[2] 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.

[3] G. Mosheiov, "Scheduling problems with a learning effect," *European Journal of Operational Research*, vol. 132, no. 3, pp. 687–693, 2001.

[4] G. Mosheiov and J. B. Sidney, "Scheduling with general job-dependent learning curves," *European Journal of Operational Research*, vol. 147, no. 3, pp. 665–670, 2003.

[5] A. Bachman and A. Janiak, "Scheduling jobs with position-dependent processing times," *Journal of the Operational Research Society*, vol. 55, pp. 257–264, 2004.

[6] W. C. Lee and C. C. Wu, "Minimizing total completion time in a two-machine flowshop with a learning effect," *International Journal of Production Economics*, vol. 88, no. 1, pp. 85–93, 2004.

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

[8] C. Koulamas and G. J. Kyparisis, "Single-machine and two-machine flowshop scheduling with general learning functions," *European Journal of Operational Research*, vol. 178, no. 2, pp. 402–407, 2007.

[9] 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.

[10] J. B. Wang, C. Ng, T. C. E. Cheng, and L. Liu, "Single-machine scheduling with a time-dependent learning effect," *International Journal of Production Economics*, vol. 111, no. 2, pp. 802–811, 2008.

[11] A. Janiak and R. Rudek, "Experience-based approach to scheduling problems with the learning effect," *IEEE Transactions on Systems, Man and Cybernetics A*, vol. 39, no. 2, pp. 344–357, 2009.

[12] Y. Yin, D. Xu, K. Sun, and H. Li, "Some scheduling problems with general position-dependent and time-dependent learning effects," *Information Sciences*, vol. 179, no. 14, pp. 2416–2425, 2009.

[13] M. D. Toksari and E. Güner, "Parallel machine scheduling problem to minimize the earliness/tardiness costs with learning effect and deteriorating jobs," *Journal of Intelligent Manufacturing*, vol. 21, no. 6, pp. 843–851, 2010.

[14] X. R. Wang, J. B. Wang, W. J. Gao, and X. Huang, "Scheduling with past-sequence-dependent setup times and learning effects on a single machine," *The International Journal of Advanced Manufacturing Technology*, vol. 48, no. 5–8, pp. 739–746, 2010.

[15] W. C. Lee, C. C. Wu, and P. H. Hsu, "A single-machine learning effect scheduling problem with release times," *Omega*, vol. 38, no. 1-2, pp. 3–11, 2010.

[16] J.-B. Wang and C. Wang, "Single-machine due-window assignment problem with learning effect and deteriorating jobs,"

Applied Mathematical Modelling, vol. 35, no. 8, pp. 4017–4022, 2011.

[17] C. C. Wu, Y. Yin, and S. R. Cheng, "Some single-machine scheduling problems with a truncation learning effect," *Computers & Industrial Engineering*, vol. 60, no. 4, pp. 790–795, 2011.

[18] X. Zhang, G. Yan, W. Huang, and G. Tang, "A note on machine scheduling with sum-of-logarithm-processing-time-based and position-based learning effects," *Information Sciences*, vol. 187, pp. 298–304, 2012.

[19] Y. Yin, D. Xu, S. R. Cheng, and C. C. Wu, "A generalisation model of learning and deteriorating effects on a single-machine scheduling with past-sequence-dependent setup times," *International Journal of Computer Integrated Manufacturing*, vol. 25, no. 9, pp. 804–813, 2012.

[20] D.-L. Yang, T. C. E. Cheng, S.-J. Yang, and C.-J. Hsu, "Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities," *Computers & Operations Research*, vol. 39, no. 7, pp. 1458–1464, 2012.

[21] D. L. Yang, T. C. E. Cheng, and W. H. Kuo, "Scheduling with a general learning effect," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1–4, pp. 217–229, 2013.

[22] J.-B. Wang, C.-J. Hsu, and D.-L. Yang, "Single-machine scheduling with effects of exponential learning and general deterioration," *Applied Mathematical Modelling*, vol. 37, no. 4, pp. 2293–2299, 2013.

[23] J.-B. Wang and J.-J. Wang, "Scheduling jobs with a general learning effect model," *Applied Mathematical Modelling*, vol. 37, no. 4, pp. 2364–2373, 2013.

[24] T. C. E. Cheng, C. C. Wu, J. C. Chen, W. H. Wu, and S. R. Cheng, "Two-machine flowshop scheduling with a truncated learning function to minimize the makespan," *International Journal of Production Economics*, vol. 141, no. 1, pp. 79–86, 2013.

[25] 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.

[26] 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.

[27] P. Liu, X. Zhou, and L. Tang, "Two-agent single-machine scheduling with position-dependent processing times," *The International Journal of Advanced Manufacturing Technology*, vol. 48, no. 1–4,, pp. 325–331, 2010.

[28] 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 & Industrial Engineering*, vol. 60, no. 4, pp. 534–541, 2011.

[29] 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.

[30] W. H. Wu, S. R. Cheng, C. C. Wu, and Y. Yin, "Ant colony algorithms for a two-agent scheduling with sum-of processing times-based learning and deteriorating considerations," *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1985–1993, 2012.

[31] A. Baykasoğlu, "Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems," *Journal of Intelligent Manufacturing*, vol. 17, no. 2, pp. 217–232, 2006.

[32] S. H. Mirsanei, M. Zandieh, M. J. Moayed, and M. R. Khabbazi, "A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times," *Journal of Intelligent Manufacturing*, vol. 22, no. 6, pp. 965–978, 2011.

[33] T. S. F. Chan, A. Prakash, H. L. Ma, and C. S. Wong, "A hybrid Tabu sample-sort simulated annealing approach for solving distributed scheduling problem," *International Journal of Production Research*, vol. 51, no. 9, pp. 2602–2619, 2013.

[34] 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*, vol. 5, no. 1, pp. 75–94, 2013.

[35] 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.

[36] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[37] D. Ben-Arieh and O. Maimon, "Annealing method for PCB assembly scheduling on two sequential machines," *International Journal of Computer Integrated Manufacturing*, vol. 5, no. 6, pp. 361–367, 1992.

[38] C. Chu, "A branch-and-bound algorithm to minimize total flow time with unequal release dates," *Naval Research Logistics*, vol. 39, no. 6, pp. 859–875, 1992.

[39] M. L. Fisher, "A dual algorithm for the one-machine scheduling problem," *Mathematical Programming*, vol. 11, no. 1, pp. 229–251, 1976.