Research Article

A Single-Machine Two-Agent Scheduling Problem by a Branch-and-Bound and Three Simulated Annealing Algorithms

Shangchia Liu,¹ Wen-Hsiang Wu,² Chao-Chung Kang,³
Win-Chin Lin,⁴ and Zhenmin Cheng⁵

¹ Business Administration Department, Fu Jen Catholic University, New Taipei City 24205, Taiwan

² Department of Healthcare Management, Yuanpei University, Hsinchu 30015, Taiwan

³ Department of Business Administration and Graduate Institute of Management, Providence University, Shalu, Taichung 43301, Taiwan

⁴ Department of Statistics, Feng Chia University, Taichung 40724, Taiwan

⁵ Business College, Beijing Union University, Beijing 100101, China

Correspondence should be addressed to Shangchia Liu; 056298@mail.fju.edu.tw

Received 30 June 2014; Accepted 11 August 2014

Academic Editor: Chin-Chia Wu

Copyright © 2015 Shangchia Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the field of distributed decision making, different agents share a common processing resource, and each agent wants to minimize a cost function depending on its jobs only. These issues arise in different application contexts, including real-time systems, integrated service networks, industrial districts, and telecommunication systems. Motivated by its importance on practical applications, we consider two-agent scheduling on a single machine where the objective is to minimize the total completion time of the jobs of the first agent with the restriction that an upper bound is allowed the total completion time of the jobs for the second agent. For solving the proposed problem, a branch-and-bound and three simulated annealing algorithms are developed for the optimal solution, respectively. In addition, the extensive computational experiments are also conducted to test the performance of the algorithms.

1. Introduction

Multiagent simulation is a branch of artificial intelligence that offers a promising approach to dealing with multistakeholder management systems, such as common pool resources. It provides a framework which allows analysis of stakeholders' (or agents') interactions and decision making. For example, in forest plantation comanagement, Purnomo and Guizol [1] further pointed out "Comanagement of forest resources is a process of governance that enables all relevant stakeholders to participate in the decision-making processes. Illegal logging and forest degradation are currently increasing, and logging bans are ineffective in reducing forest degradation. At the same time interest in forest plantations and concern about poverty problems of neighboring people whose livelihoods depend on forest services and products continue to increase rapidly. Governments have identified the development of

small forest plantations as an opportunity to provide wood supplies to forest industries and to reduce poverty. However, the development of small plantations is very slow due to an imbalance of power and suspicion between communities and large companies." In blood cell population dynamics, Bessonov et al. [2] gave some possible explanations of the mechanism for recovery of the system under important blood loss or blood diseases such as anemia. Agnetis et al. [3] also indicated 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, operations research, and so forth.

Scheduling with multiple agents has received growing attention in recently years. Agnetis et al. [4] and Baker and Smith [5] were among the pioneers to introduce the concept

of multiagent into scheduling problems. Yuan et al. [6] discussed two dynamic programming recursions in Baker and Smith [5] and proposed a polynomial-time algorithm for the same problem. Cheng et al. [7] addressed the feasibility model of multiagent scheduling on a single machine where each agent's objective function is to minimize the total weighted number of tardy jobs. Ng et al. [8] proposed a two-agent scheduling problem on a single machine, where the objective is to minimize the total completion time of the first agent with the restriction that the number of tardy jobs of the second agent cannot exceed a given number. Agnetis et al. [3] discussed the complexity some single-machine scheduling problems with several agents. Cheng et al. [9] studied multiagent scheduling on a single machine where the objective functions of the agents are of the max-form. Agnetis et al. [10] used a branch-and-bound method to solve several two-agent scheduling problems on a single machine. Lee et al. [11] considered a multiagent scheduling problem on a single machine in which each agent is responsible for his own set of jobs and wishes to minimize the total weighted completion time of his own set of jobs. Recently, Leung et al. [12] generalized the single machine problems proposed by Agnetis et al. [4] to the environment with multiple identical machines in parallel. Yin et al. [13] considered several two-agent scheduling problems with assignable due dates on a single machine, where the goal was to assign a due date from a given set of due dates and a position in the sequence to each job so that the weighted sum of the objectives of both agents is minimized. For different combinations of the objectives, which include the maximum lateness, total (weighted) tardiness, and total (weighted) number of tardy jobs, they provided the complexity results and solve the corresponding problems, if possible. Yin et al. [14] investigated a scheduling environment with two agents and a linear nonincreasing deterioration with the objective of scheduling the jobs such that the combined schedule performs well with respect to the measures of both agents. Three different objective functions were considered for one agent, including the maximum earliness cost, total earliness cost, and total weighted earliness cost, while keeping the maximum earliness cost of the other agent below or at a fixed level U . They proposed the optimal (nondominated) properties and present the complexity results for the problems. Yin et al. [15] addressed a two-agent scheduling problem on a single machine where the objective is to minimize the total weighted earliness cost of all jobs, while keeping the earliness cost of one agent below or at a fixed level Q . A mixed-integer programming (MIP) model was first formulated to find the optimal solution which is useful for small-size problem instances and then a branch-and-bound algorithm incorporating with several dominance properties, a lower bound and a simulated annealing heuristic algorithm were provided to derive the optimal solution for medium- to large-size problem instances. For more recent works with two-agent issues, readers can refer to Nong et al. [16], Wan et al. [17], Luo et al. [18], Yin et al. [19], and so forth. In addition, for more multiple-agent works with time-dependent, readers can refer to Liu and Tang [20], Liu et al. [21], Cheng et al. [22, 23], Wu et al. [24], Li and Hsu [25], Yin et al. [26–28], Wu [29], and Li et al. [30], and so forth.

For the importance of multiple agents competing on the usage of a common processing resource in different application environments and different methodological fields, we considered two-agent scheduling on a single machine. The objective is to minimize the total completion time of the jobs of the first agent with the restriction that an upper bound is allowed the total completion time of the jobs for the second agent which has been shown binary NP-hard by Agnetis et al. [4]. The problem is described as follows. There are n jobs which belong to one of the agents AG_0 or AG_1 . For each job j , there is a normal processing time p_j and an agent code I_j , where $I_j = 0$ if $J_j \in AG_0$ or $I_j = 1$ if $J_j \in AG_1$. All the jobs are available at time zero. Under a schedule S , let $C_j(S)$ be the completion time of job j . The objective of this paper is to find an optimal schedule to minimize $\sum_{j=1}^n C_j(S)(1 - I_j)$ subject to $\sum_{j=1}^n C_j(S)I_j < U$, where U is a control upper bound.

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

2. Dominance Properties

We will develop some adjacent dominance rules by using the pairwise interchange method. Assume that S_1 and S_2 denote two given job schedules in which the difference between S_1 and S_2 is a pairwise interchange of two adjacent jobs i and j . That is, $S_1 = (\sigma, i, j, \sigma')$ and $S_2 = (\sigma, j, i, \sigma')$, where σ and σ' each denote a partial sequence. In addition, let t be the starting time of the last job in σ .

Property 1. If jobs $i, j \in AG_0$, and $p_i < p_j$, then S_1 dominates S_2 .

Property 2. If jobs $i, j \in AG_1$, $\sum_{i \in \sigma \cap AG_1} C_i + 2t + 2p_i + p_j < U$, and $p_i < p_j$, then S_1 dominates S_2 .

Next, we give a proposition to determine the feasibility of the partial schedule. 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 with $(n - k)$ jobs. Among the unscheduled jobs, let $p_{(1)} = \min_{J_j \in \pi^c \cap AG_1} \{p_j\}$. Moreover, let $C_{[k]}$ be the completion times of the last job in π . Also, let π' and π'' denote the unscheduled jobs in AG_0 and AG_1 arranged in the smallest processing times (SPT) order, respectively.

Property 3. If there is a job $i \in AG_1$ such that $\sum_{i \in \pi \cap AG_1} C_i + C_{[k]} + p_{(1)} > U$, then (π, π^c) is not a feasible sequence.

Property 4. If all unscheduled jobs are belonging to AG_0 , then (π, π') dominates (π, π^c) .

Property 5. If all unscheduled jobs are belonging to AG_1 , then (π, π^c) can be determined by (π, π'') . That is, (π, π'') either is feasible solution or can be deleted.

2.1. Lower Bound. Assume that PS is a partial schedule in which the order of the first k jobs is determined and let US be the unscheduled part with $(n - k)$ jobs. Among the unscheduled jobs, there are n_0 jobs from agent AG_0 and n_1 jobs from agent AG_1 . Moreover, let $C_{[k]}$ denote the completion times of the k th job in PS. The completion time for the $(k + j)$ th job is

$$C_{[k+j]} \geq C_{[k]} + \sum_{i=1}^j p_{(k+i)}, \quad \text{for } 1 \leq j \leq n_0. \quad (1)$$

Then a lower bound can be obtained as follows:

$$\sum_{j=1}^n C_j(S) (1 - I_j) \geq \sum_{j=1}^k C_j(S) (1 - I_j) + \sum_{j=1}^{n_0} \widehat{C}_{(j)}(S), \quad (2)$$

where $\widehat{C}_{(j)} = C_{[k]} + \sum_{i=1}^j p_{(k+i)}$.

2.2. Simulated Annealing Algorithms. Simulated annealing has become one of the most popular metaheuristic methods to solve combinatorial optimization problems since it was proposed by Kirkpatrick et al. [31]. For example, Kim et al. [32] applied the method in scheduling of raw-material unloading from ships at a steelworks. Sun et al. [33] used the technique for allocating product lots to customer orders in semiconductor manufacturing supply chains. Moreover, the method has the advantage of avoiding getting trapped in a local optimum because of its hill climbing moves, which are governed by a control parameter. Thus, we will use SA to derive near-optimal solutions for our problem. The steps of SA algorithms are summarized as follows.

2.2.1. Initial Sequence. Three initial sequences for the three SA algorithms are adopted for our problem. For the first initial sequence in SA_1 , the sequence of jobs of AG_1 is arranged according to the smallest processing times (SPT) order, followed by arranging the sequence of jobs of AG_0 in the shortest processing time (SPT) order. In order to get a good initial one, two more initial sequences are considered. For the second initial sequence in SA_2 , the pairwise interchange is used to the initial sequence produce by SA_1 . For the third initial sequence in SA_3 , the NEH method [34] is applied to the initial sequence produce by SA_1 .

2.2.2. Neighborhood Generation. The pairwise interchange (PI) neighborhood generation method is adopted in the algorithms.

2.2.3. 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:

$$P(\text{accept}) = \exp(-\alpha \times \Delta TC), \quad (3)$$

where α is a control parameter and ΔTC is the change in the objective value. In addition, we use the method suggested by

Ben-Arieh and Maimon [35] to change α in the k th iteration as follows:

$$\alpha = \frac{k}{\delta}, \quad (4)$$

where δ is an experimental constant. After preliminary trials, $\delta = 6$ is used in our experiments.

If the total completion time of jobs from agent AG_0 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 $U(0, 1)$.

2.2.4. Stopping Rule. All three proposed SAs are stopped after $100n$ iterations, where n is the number of jobs.

3. Computational Experiments

The extensive computational experiments were conducted to test the performances of the branch-and-bound algorithm and the three simulated annealing algorithms. All the algorithms were coded in Fortran using Compaq Visual Fortran version 6.6 and performed the experiments on a personal computer powered by an Intel(R) Core(TM)2 Quad CPU 2.66 GHz with 4 GB RAM operating under Windows XP. The job processing times were generated from a uniform distribution over the integers 1–100. For the control upper bound U , we arrange the jobs of AG_1 by the smallest processing times (SPT) order and compute out the total completion times of the jobs of AG_1 , (recorded as TC_1). Then following the sequence of the jobs of AG_1 , we arrange the sequence of jobs of AG_0 in the shortest processing time (SPT) order and compute out the total completion times of jobs of AG_1 and AG_0 , (recorded as TC_{1+2}). We let $U = (1 - \alpha)TC_1 + \alpha TC_{1+2}$, where $0 < \alpha < 1$. Moreover, α was taken as the values of 0.25, 0.5, and 0.75, while the proportion of the jobs of agent AG_1 at $\text{pro} = 0.25, 0.5$, and 0.75 in the tests.

For the branch-and-bound algorithm, the average and standard deviation of the numbers of nodes and the execution times (in seconds) were recorded. For the SA heuristics, the mean and standard deviation percentage errors were recorded. The percentage error of a solution obtained from a heuristic algorithm was given by

$$\frac{V - V^*}{V^*}, \quad (5)$$

where V and V^* are denoted as the total completion time of the heuristic and the optimal solution, respectively. The computational times of the heuristic algorithms were not recorded because they all were fast in generating solutions.

The computational experiments were divided into two parts. In the first part of the experiments, three number of jobs were tested at $n = 8, 12$, and 16 . As a result, 27 experimental situations were examined. 50 instances were randomly generated for each case. The results were summarized in Table 1 that includes the CPU time (mean and standard deviation) and the number of nodes for the branch-and-bound algorithm.

TABLE 1: Performance of the branch-and-bound and heuristic algorithms ($n = 8 \sim 16$).

n	Pro	α	Branch-and-bound algorithm			SA1			SA2			SA3			SA4		
			Valid sample size		CPU time	Number of nodes		Std.	Error percentages		Mean	Error percentages		Mean	Error percentages		Mean
			Mean	Std.		Mean	Std.		Mean	Std.		Mean	Std.		Mean	Std.	
8	0.25	0.25	50	0.00	0.01	526	318	0.01	0.28	1.46	0.28	1.46	1.46	0.28	1.46	1.46	1.46
		0.50	50	0.00	0.01	748	588	0.01	1.29	4.12	0.93	3.11	0.86	0.24	3.09	0.24	1.69
		0.75	50	0.01	0.01	1383	923	0.01	3.42	7.29	2.83	5.73	2.27	0.62	4.80	0.62	2.46
		0.25	50	0.01	0.01	2229	1371	0.01	0.01	0.06	0.01	0.06	0.00	0.00	0.00	0.00	0.00
	0.50	0.50	50	0.01	0.01	2500	1412	0.01	0.50	1.78	0.14	0.56	0.20	0.04	0.86	0.04	0.20
		0.75	50	0.01	0.01	3183	1139	0.01	0.92	2.03	1.27	2.26	1.48	0.26	2.76	0.26	1.02
		0.25	50	0.01	0.01	1091	217	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.50	50	0.01	0.01	1091	217	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	50	0.01	0.01	1089	196	0.01	0.36	1.18	0.11	0.48	0.49	0.03	1.33	0.03	0.24
	Average			0.01	0.01	1538	709	0.01	0.75	1.99	0.62	1.52	0.62	0.16	1.59	0.16	0.79
12	0.25	0.25	50	0.78	0.86	97087	113419	0.86	0.29	0.93	0.61	2.30	0.30	0.05	1.38	0.05	0.20
		0.50	50	1.36	1.96	175858	261636	1.96	1.86	4.41	3.99	7.81	1.45	0.77	3.86	0.77	1.95
		0.75	50	3.92	5.24	535992	751273	5.24	4.78	7.25	4.40	6.82	5.30	2.12	8.43	2.12	4.26
		0.25	50	19.83	22.75	2278911	2794764	22.75	0.02	0.07	0.04	0.20	0.01	0.00	0.05	0.00	0.00
	0.50	0.50	50	27.91	33.86	3405293	4448908	33.86	0.64	1.24	0.53	1.16	0.32	0.17	0.73	0.17	0.45
		0.75	50	35.38	21.51	4692766	3096131	21.51	2.07	2.16	2.37	2.43	2.37	0.69	2.74	0.69	0.92
		0.25	50	6.30	2.18	794003	304461	2.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.50	50	6.30	2.18	794439	304555	2.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	50	6.60	1.82	862132	262277	1.82	0.44	0.66	0.74	1.22	0.55	0.15	1.19	0.15	0.33
	Average			12.04	10.26	1515165	1370825	10.26	1.12	1.86	1.41	2.44	1.14	0.44	2.04	0.44	0.90
16	0.25	0.25	48	846.82	1691.36	62738963	129480137	1691.36	0.36	0.92	0.39	1.06	0.35	0.18	1.16	0.18	0.77
		0.50	46	1432.63	2411.56	105554536	181805362	2411.56	1.99	3.46	2.49	4.28	1.88	0.95	4.38	0.95	2.29
		0.75	33	1446.76	2629.13	106507557	197947458	2629.13	2.72	3.14	7.79	12.00	4.30	1.83	6.64	1.83	2.28
		0.25	22	8044.31	6058.59	429572906	333869433	6058.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.50	0.50	20	7267.99	5697.96	387550386	314387045	5697.96	0.10	0.29	0.00	0.00	0.11	0.00	0.51	0.00	0.00
		0.75	3	7493.01	5776.30	411927486	331822988	5776.30	0.93	1.07	6.91	6.17	6.65	0.93	4.11	0.93	1.07
		0.25	13	9502.54	3053.15	625205281	214788228	3053.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.50	13	9478.74	3044.52	625205281	214788228	3044.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	7	9965.98	3653.10	652692683	258677186	3653.10	0.41	0.73	0.24	0.63	1.03	0.00	1.06	0.00	0.00
	Average			6164.31	3779.52	378550564	241951785	3779.52	0.72	1.07	1.98	2.68	1.59	0.43	1.98	0.43	0.71

FIGURE 1: Performance of the branch-and-bound algorithms ($n = 12$).

For the performance of the branch-and-bound algorithm, it can be observed from Table 1 that the number of nodes and the mean of the CPU time increase when n becomes bigger. The difficult situations occur at $n = 16$. Especially, the instances with a bigger value of α ($\alpha = 0.75$) are difficult to solve than those with a smaller one ($\alpha = 0.25, 0.50$). Moreover, the instances with a bigger value of pro ($\text{pro} = 0.5, 0.75$) are difficult to solve than those with a smaller one ($\text{pro} = 0.25$). The most difficult case is located at $\text{pro} = 0.5, 0.75$, and $\alpha = 0.75$ as the number of instances, which can be solved out within less 10^{-9} nodes, were declined to 10 or below. In addition, as shown in Figure 1 and Table 1, the instances with $\text{pro} = 0.5$ took more nodes or CPU time than others, and the trend became clear as the value of α got bigger.

For the performance of the proposed SAs, it can be seen in Table 1 that the means of the percentage error of SA_1 (between 0.0% and 4.78%) are lower than those of SA_2 (between 0.0% and 7.79%) and those of SA_3 (between 0.0% and 6.65%). Moreover, the trend of the standard deviations of the percentage error keeps the similar pattern. As shown in Table 1, the standard deviations of the percentage error of SA_1 , SA_2 , and SA_3 were between 0.0% and 7.29%, 0.0% and 12.0%, and 0.0% and 8.43%, respectively. It released that the performance of SA_1 was slightly better than the other two. Furthermore, Figure 2 and Table 1 indicated that the means of the percentage error of SA_1 , SA_2 , and SA_3 are slightly affected by the parameter α . For example, the means of the percentage errors of SA_1 , SA_2 , and SA_3 were larger than 2% at $\alpha = 0.25$. Specifically, the behavior became clear at $\text{pro} = 0.25$ and 0.5. It also can be observed that there is no absolutely dominance relationship among three proposed SAs. Due to get a good quality solution, we further combined three proposed SAs into one (recorded as SA_4 and $\text{SA}_4 = \min\{\text{SA}_i, i = 1, 2, 3\}$). The means of the percentage error of SA_4 were located between 0.0% and 2.12%. Meanwhile its standard deviations of the percentage error were between

FIGURE 2: Performance of the SA algorithms ($n = 12$).

0.0% and 4.26%. Moreover, the impacts of pro and α were not present for proposed three SAs.

In the second part of the experiments, the performances of the proposed SA heuristics were further tested for large numbers of jobs. Three different numbers of jobs were considered at $n = 20, 40$, and 60. The proportion of the jobs of agent AG_1 at $\text{pro} = 0.25, 0.5$, and 0.75 in the tests. Moreover, α was taken as the values of 0.25, 0.5, and 0.75. As a result, 27 experimental situations were tested. 50 instances were randomly generated for each situation. The relative deviance percentage with respect to the best known solution was reported for each instance. The mean execution time and mean relative deviance percentage were also recorded for each SA heuristic. The relative deviation percentage RDP was given by

$$\frac{\text{SA}_i - \text{SA}^*}{\text{SA}^*}, \quad (6)$$

where SA_i is the value of the objective function generated by SA_i and $\text{SA}^* = \min\{\text{SA}_i, i = 1, 2, 3\}$ is the smallest value of the objective function obtained from the three SA heuristics. The results were summarized in Table 2.

As shown in Figure 3 and Table 2, it can be seen that the RPD means of SA_1 , SA_2 , and SA_3 are getting slightly bigger as the value of α increases. In general, the RPD means of SA_3 are lower than those of SA_1 and SA_2 . Furthermore, all of the RPD means of SA_1 , SA_2 , and SA_3 were less than 2%. Figure 2 also indicated that there is no absolutely dominance relationship among three proposed SAs.

4. Conclusions

This paper explored the single-machine two-agent scheduling problem where the objective is to minimize the total completion time of the jobs belonging to the first agent with the restriction that total completion time of jobs from the second agent has an upper bound. Due to the fact that the problem under study is binary NP hard, a branch-and-bound

TABLE 2: RPD of heuristic algorithms ($n = 20 \sim 60$).

n	Pro	α	SA1				SA2				SA3			
			CPU time		RPD		CPU time		RPD		CPU time		RPD	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
20	0.25	25	0.0019	0.0051	0.45	1.44	0.0019	0.0051	0.41	1.19	0.0016	0.0047	0.51	1.75
		50	0.0019	0.0051	1.39	2.67	0.0019	0.0051	1.87	5.51	0.0022	0.0055	0.89	1.88
		75	0.0019	0.0051	2.86	4.81	0.0019	0.0051	4.96	6.80	0.0019	0.0051	2.01	3.66
	0.5	25	0.0019	0.0051	0.02	0.07	0.0019	0.0051	0.00	0.01	0.0016	0.0047	0.01	0.06
		50	0.0019	0.0051	0.51	0.81	0.0019	0.0051	0.39	0.67	0.0022	0.0055	0.51	0.78
		75	0.0019	0.0051	1.05	1.52	0.0022	0.0055	1.19	2.08	0.0022	0.0055	1.30	2.03
	0.75	25	0.0019	0.0051	0.00	0.00	0.0013	0.0043	0.00	0.00	0.0019	0.0051	0.00	0.00
		50	0.0019	0.0051	0.00	0.00	0.0016	0.0047	0.00	0.00	0.0019	0.0051	0.00	0.00
		75	0.0019	0.0051	0.50	0.58	0.0019	0.0051	0.58	0.75	0.0019	0.0051	0.34	0.55
	Average		0.0019	0.0051	0.75	1.32	0.0018	0.0050	1.04	1.89	0.0019	0.0052	0.62	1.19
40	0.25	25	0.0066	0.0078	0.10	0.30	0.0072	0.0079	0.16	0.52	0.0066	0.0078	0.00	0.00
		50	0.0072	0.0079	0.92	1.51	0.0078	0.0079	1.22	2.09	0.0072	0.0079	0.39	1.47
		75	0.0078	0.0079	3.66	4.43	0.0084	0.0079	3.91	5.84	0.0081	0.0079	2.82	4.04
	0.5	25	0.0066	0.0078	0.02	0.06	0.0066	0.0078	0.00	0.00	0.0066	0.0078	0.00	0.00
		50	0.0075	0.0079	0.39	0.43	0.0078	0.0079	0.31	0.57	0.0078	0.0079	0.46	0.79
		75	0.0084	0.0079	0.81	0.81	0.0091	0.0078	1.03	1.69	0.0091	0.0078	0.71	1.12
	0.75	25	0.0066	0.0078	0.00	0.00	0.0063	0.0077	0.00	0.00	0.0063	0.0077	0.00	0.00
		50	0.0063	0.0077	0.00	0.00	0.0063	0.0077	0.00	0.00	0.0063	0.0077	0.00	0.00
		75	0.0072	0.0079	0.32	0.45	0.0078	0.0079	0.25	0.36	0.0081	0.0079	0.25	0.31
	Average		0.0071	0.0078	0.69	0.89	0.0075	0.0078	0.76	1.23	0.0073	0.0078	0.51	0.86
60	0.25	25	0.0159	0.0022	0.07	0.20	0.0178	0.0055	0.09	0.23	0.0147	0.0037	0.02	0.11
		50	0.0175	0.0051	1.48	2.72	0.0197	0.0069	1.20	1.88	0.0163	0.0044	0.28	0.88
		75	0.0197	0.0069	2.38	2.51	0.0219	0.0077	3.74	5.10	0.0206	0.0074	2.58	4.10
	0.5	25	0.0147	0.0037	0.01	0.03	0.0147	0.0037	0.00	0.00	0.0144	0.0043	0.00	0.00
		50	0.0175	0.0060	0.39	0.57	0.0184	0.0061	0.27	0.35	0.0181	0.0066	0.23	0.42
		75	0.0209	0.0075	0.66	1.03	0.0231	0.0079	0.62	0.73	0.0222	0.0078	0.66	0.90
	0.75	25	0.0141	0.0047	0.00	0.00	0.0141	0.0047	0.00	0.00	0.0141	0.0047	0.00	0.00
		50	0.0141	0.0047	0.00	0.00	0.0141	0.0047	0.00	0.00	0.0141	0.0047	0.00	0.00
		75	0.0175	0.0051	0.21	0.27	0.0197	0.0069	0.22	0.28	0.0194	0.0067	0.21	0.25
	Average		0.0169	0.0051	0.58	0.82	0.0182	0.0060	0.68	0.95	0.0171	0.0056	0.44	0.74

FIGURE 3: Performance of the SA algorithms ($n = 60$).

algorithm incorporating several dominance properties and a lower bound was proposed for the optimal solution. Three simulated annealing algorithms were also proposed for 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 number of nodes and execution time when the number of jobs is fewer than or equal to 16. Moreover, the computational experiments also show that the proposed combined SA₄ performs well since its mean error percentage was less than 2.12% for all the tested cases. Further research lies in devising efficient and effective methods to solve the problem with significantly larger numbers of jobs.

Conflict of Interests

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

Acknowledgment

This research was supported in part by the foundation of Beijing Municipal Education on Science and Technology Plan Project (KM 20121417015).

References

- [1] H. Purnomo and P. Guizol, "Simulating forest plantation co-management with a multi-agent system," *Mathematical and Computer Modelling*, vol. 44, no. 5-6, pp. 535–552, 2006.
- [2] N. Bessonov, I. Demin, L. Pujo-Menjouet, and V. Volpert, "A multi-agent model describing self-renewal of differentiation effects on the blood cell population," *Mathematical and Computer Modelling*, vol. 49, no. 11-12, pp. 2116–2127, 2009.
- [3] A. Agnetis, D. Pacciarelli, and A. Pacifici, "Multi-agent single machine scheduling," *Annals of Operations Research*, vol. 150, no. 1, pp. 3–15, 2007.
- [4] 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.
- [5] 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.
- [6] J. J. Yuan, W. P. Shang, and Q. Feng, "A note on the scheduling with two families of jobs," *Journal of Scheduling*, vol. 8, no. 6, pp. 537–542, 2005.
- [7] T. C. 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.
- [8] C. T. Ng, T. C. 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.
- [9] T. C. 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.
- [10] A. Agnetis, G. de Pascale, and D. Pacciarelli, "A Lagrangian approach to single-machine scheduling problems with two competing agents," *Journal of Scheduling*, vol. 12, no. 4, pp. 401–415, 2009.
- [11] K. Lee, B. C. Choi, J. Y. Leung, and M. L. Pinedo, "Approximation algorithms for multi-agent scheduling to minimize total weighted completion time," *Information Processing Letters*, vol. 109, no. 16, pp. 913–917, 2009.
- [12] J. Y. Leung, M. Pinedo, and G. Wan, "Competitive two-agent scheduling and its applications," *Operations Research*, vol. 58, no. 2, pp. 458–469, 2010.
- [13] Y. Yin, S.-R. Cheng, T. C. Cheng, C.-C. Wu, and W.-H. Wu, "Two-agent single-machine scheduling with assignable due dates," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 1674–1685, 2012.
- [14] Y. Yin, S.-R. Cheng, and C.-C. Wu, "Scheduling problems with two agents and a linear non-increasing deterioration to minimize earliness penalties," *Information Sciences*, vol. 189, pp. 282–292, 2012.
- [15] Y. Yin, C.-C. Wu, W.-H. Wu, C.-J. Hsu, and W.-H. Wu, "A branch-and-bound procedure for a single-machine earliness scheduling problem with two agents," *Applied Soft Computing Journal*, vol. 13, no. 2, pp. 1042–1054, 2013.
- [16] Q. Q. Nong, T. C. Cheng, and C. T. Ng, "Two-agent scheduling to minimize the total cost," *European Journal of Operational Research*, vol. 215, no. 1, pp. 39–44, 2011.
- [17] G. Wan, S. R. Vakati, J. Y. Leung, and M. Pinedo, "Scheduling two agents with controllable processing times," *European Journal of Operational Research*, vol. 205, no. 3, pp. 528–539, 2010.
- [18] W. Luo, L. Chen, and G. Zhang, "Approximation schemes for two-machine flow shop scheduling with two agents," *Journal of Combinatorial Optimization*, vol. 24, no. 3, pp. 229–239, 2012.
- [19] 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.
- [20] P. Liu and L. Tang, "Two-agent scheduling with linear deteriorating jobs on a single machine," in *Computing and Combinatorics: Proceedings of the 14th Annual International Conference, COCOON 2008 Dalian, China, June 27–29, 2008*, vol. 5092 of *Lecture Notes in Computer Science*, pp. 642–650, Springer, Berlin, Germany, 2008.
- [21] P. Liu, N. Yi, and X. Zhou, "Two-agent single-machine scheduling problems under increasing linear deterioration," *Applied Mathematical Modelling*, vol. 35, no. 5, pp. 2290–2296, 2011.
- [22] T. C. E. Cheng, W.-H. Wu, S.-R. Cheng, and C.-C. Wu, "Two-agent scheduling with position-based deteriorating jobs and learning effects," *Applied Mathematics and Computation*, vol. 217, no. 21, pp. 8804–8824, 2011.
- [23] T. C. Cheng, Y. H. Chung, S. C. Liao, and W. C. Lee, "Two-agent single-machine scheduling with release times to minimize the total weighted completion time," *Computers & Operations Research*, vol. 40, no. 1, pp. 353–361, 2013.
- [24] C.-C. Wu, S.-K. Huang, and W.-C. Lee, "Two-agent scheduling with learning consideration," *Computers and Industrial Engineering*, vol. 61, no. 4, pp. 1324–1335, 2011.
- [25] D.-C. Li and P.-H. Hsu, "Solving a two-agent single-machine scheduling problem considering learning effect," *Computers and Operations Research*, vol. 39, no. 7, pp. 1644–1651, 2012.
- [26] Y. Yin, M. Liu, J. Hao, and M. Zhou, "Single-machine scheduling with job-position-dependent learning and time-dependent deterioration," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 1, pp. 192–200, 2012.
- [27] Y. Yin, T. C. E. Cheng, and C. C. Wu, "Scheduling with time-dependent processing times," *Mathematical Problems in Engineering*, vol. 2014, Article ID 201421, 2 pages, 2014.
- [28] Y. Yin, W.-H. Wu, T. C. E. Cheng, and C. C. Wu, "Single-machine scheduling with time-dependent and position-dependent deteriorating jobs," *International Journal of Computer Integrated Manufacturing*, 2014.
- [29] W.-H. Wu, "Solving a two-agent single-machine learning scheduling problem," *International Journal of Computer Integrated Manufacturing*, vol. 27, no. 1, pp. 20–35, 2014.
- [30] D.-C. Li, P.-H. Hsu, and C.-C. Chang, "A genetic algorithm-based approach for single-machine scheduling with learning effect and release time," *Mathematical Problems in Engineering*, vol. 2014, Article ID 249493, 12 pages, 2014.

- [31] S. Kirkpatrick, J. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [32] B.-I. Kim, S. Y. Chang, J. Chang et al., "Scheduling of raw-material unloading from ships at a steelworks," *Production Planning and Control*, vol. 22, no. 4, pp. 389–402, 2011.
- [33] Y. Sun, J. W. Fowler, and D. L. Shunk, "Policies for allocating product lots to customer orders in semiconductor manufacturing supply chains," *Production Planning and Control*, vol. 22, no. 1, pp. 69–80, 2011.
- [34] 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.
- [35] 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.