Applied Mathematics and Computation 219 (2013) 10831–10841

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A single-machine bi-criterion scheduling problem with two agents

Wen-Chiung Lee [a,*], Yu-Hsiang Chung [b], Zong-Ren Huang [a]

[a] *Department of Statistics, Feng Chia University, Taichung, Taiwan*
[b] *Department of Industrial & Engineering Management, National Chiao Tung University, Hsinchu, Taiwan*

ARTICLE INFO

Keywords:
Scheduling
Total completion time
Maximum tardiness
Two-agent
Single-machine

ABSTRACT

The multiple-agent scheduling problems have received increasing attention recently. However, most of the research focuses on studying the computational complexity of the intractable cases or examining problems with a single criterion. Often a decision maker has to decide the schedule based on multiple criteria. In this paper, we consider a single machine problem where the objective is to minimize a linear combination of the total completion time and the maximum tardiness of jobs from the first agent given that no tardy jobs are allowed for the second agent. We develop a branch-and-bound algorithm and several simulated annealing algorithms to search for the optimal solution and near-optimal solutions for the problem, respectively. Computational experiments show that the proposed branch-and-bound algorithm could solve problems of up to 24 jobs in a reasonable amount of time and the performance of the combined simulated annealing algorithm is very good with an average error percentage of less than 0.5% for all the tested cases.

1. Introduction

In most scheduling models, there is a common goal to minimize for all the jobs [1–6]. However, jobs might be from different customers which have their own goals to pursue. For instance, Peha [7] gave a telecommunication service example where various types of packets and service compete for the use of a commercial satellite, and the problem is to satisfy the service requirements of individual agents to transfer voice, image and text files for their clients. Kim et al. [8] pointed out that in project scheduling the problem is concerned with negotiation to resolve conflicts whenever the agents find their own schedules unacceptable. Kubzin and Strusevich [9] presented another example that maintenance operations complete with the real jobs for machine occupancy on maintenance planning. Balasubramanian et al. [10] provided the manufacturing examples where the agents belong to different subjects competing for the usage of machines.

Agnetis et al. [11] and Baker and Smith [12] were the pioneers that brought the multi-agent problems into scheduling field. For more articles with multiple-agent scheduling, readers can refer to [13–23].Since then, many researchers have devoted efforts to this area. Recently, Lee et al. [24] studied a two-agent problem with deteriorating jobs on a single machine to minimize the total weighted completion time of jobs from agent 1 given that no tardy jobs are allowed for agent 2. In their model, the actual job processing time is $\alpha_j + \beta t$ if its starting time is t where α_j is the normal processing time of job j, and β is the common deterioration rate. They provided the branch-and-bound algorithm to search for the optimal solution for problems with up to 20 jobs. Moreover, they provided three heuristic algorithms based on linear combinations of processing times and weights for jobs from agent 1 and linear combinations of processing times and due dates for jobs from agent 2.

* Corresponding author.
E-mail address: wclee@fcu.edu.tw (W.-C. Lee).

0096-3003/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2013.05.025

Wu et al. [25] considered a single-machine scheduling problem with learning effects where the objective is to minimize the total tardiness of jobs from the first agent with the constraint that no tardy job is allowed for the second agent. The actual processing time of job j under the model is $p_j r^a$ if it is scheduled in the rth position where p_j is the processing time of job j, a is learning effect and $a \leq 0$. The proposed branch-and-bound algorithm could solve problems with up to 24 jobs optimally, and the proposed heuristic algorithms sort jobs according to the linear combination of the job processing times and the due dates. They presented the computational experiment for their heuristics with 50 jobs. Lee et al. [26] studied a two-agent scheduling problem on two-machine permutation flowshop where the objective is to minimize the total tardiness of jobs from the first agent given that no tardy job of the second agent is allowed. They provided a branch-and-bound algorithm to solve problems with 12 jobs optimally. In addition, they presented the computational results of two simulated annealing algorithms for problems with 25, 50 and 75 jobs. Cheng et al. [27] considered a two-agent single-machine scheduling problem with deteriorating jobs and learning effects simultaneously. The objective is to minimize the total weighted completion time of jobs from the first agent with the restriction that no tardy job is allowed for the second agent. They assumed that jobs from the first agent have a position-based learning effect, and jobs from the second agent have a position-based deteriorating effect. Their branch-and-bound algorithm could present the optimal solutions for problems with up to 16 jobs. Moreover, they conducted a computational experiment for the proposed simulated algorithm for problems with 40 and 50 jobs.

Most of the research on multiple-agent focuses on studying the computational complexity of the intractable cases or examining problems of minimizing a single criterion of one agent given a bound of a single criterion of the other agent. However, a decision maker has to decide the schedule based on many aspects. For instance, the manufacturer might be interested in reducing the inventory cost as well as reducing the impact of late delivery. To the best of our knowledge, multi-agent scheduling problem with the consideration of multiple objectives for the same agent is never been discussed in literature. Since the total completion time is commonly used to represent the internal efficiency while the maximum tardiness is commonly used to represent the external efficiency. We consider these two criteria in this paper. That is, we study a two-agent scheduling problem on a single machine where the objective is to minimize the weighted combination of the total completion time and the maximum tardiness of jobs from one agent given that the number of tardy jobs from the other agent is zero. The rest of the paper is organized as follows. In the next section we describe the formulation of our problem. In Section 3, we construct a branch-and-bound algorithm incorporating several elimination rules and a lower bound to speed up the search for the optimal solution. In Section 4, several simulated annealing algorithms are proposed to solve this problem. In Section 5, a computational experiment is conducted to evaluate the efficiency of the branch-and-bound algorithm and the performance of the proposed heuristic algorithms. A conclusion is given in the last section.

2. Problem description

The problem description is given as follows. There are n jobs ready to be processed on a single machine. Each job belongs to either one of the two agents AG_0 or AG_1. For each job j, there is a processing time p_j, a due date d_j, and an agent code I_j, where $I_j = 0$ if $j \in AG_0$ or $I_j = 1$ if $j \in AG_1$. Under a schedule S, let $C_j(S)$ be the completion time of job j, $T_j(S) = \max\{0, C_j(S) - d_j\}$ be the tardiness of job j, $T_{\max}(S) = \max_{1 \leq j \leq n}\{T_j(S)(1 - I_j)\}$ be the maximum tardiness of jobs from agent AG_0 under schedule S, and $U_j(S) = 1$ if $T_j(S) > 0$ and zero otherwise. The objective of this paper is to find a schedule that minimizes the weight combination of the total completion time and the maximum tardiness of jobs from AG_0 with the restriction that no tardy jobs from AG_1 are allowed. Using the conventional three fields notation, this problem is denoted as $1|\sum U_j I_j = 0|\alpha \sum C_j(1 - I_j) + (1 - \alpha)T_{\max}$ where $0 < \alpha < 1$.

3. A branch-and-bound algorithm

Agnetis [28] pointed out that the two-agent problem is NP-hard if both the agents have the sum-type objectives. Thus, the branch-and-bound algorithm is developed to obtain the optimal solution.

3.1. Dominance properties

In this subsection, we first provide several adjacent dominance properties. Suppose that S and S' are two job schedules and the difference between S and $S\prime$ is a pairwise interchange of two adjacent jobs i and j. That is, $S = (\pi, i, j, \pi')$ and $S' = (\pi, j, i, \pi')$, where π and π' each denote a partial sequence. In addition, let t denote the completion time of the last job in π and B be the maximum tardiness of jobs from agent AG_0 under partial sequence π, that is, $B = \max_{k \in \pi \cap AG_0}\{T_k(S)\}$. The completion times of jobs i and j in S are

$$C_i(S) = t + p_i \tag{1}$$

and

$$C_j(S) = t + p_i + p_j, \tag{2}$$

whereas the completion times of jobs j and i in S' are

$$C_j(S') = t + p_j, \tag{3}$$

and

$$C_i(S') = t + p_j + p_i. \tag{4}$$

Property 1. *If jobs* $i, j \in AG_0$, $p_i < p_j$, $d_i < d_j$ *and* $t - B > \max\{d_i - p_i, d_j - p_j\}$, *then S dominates S′.*

Proof. Since jobs in π are processed in the same order in both S and S', we have from Eqs. (2) and (4) that

$$C_k(S) = C_k(S') \text{ if job } k \in \pi \text{ or } \pi'.$$

Thus, to show S dominates S', it suffices to show

$$\alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_{\max}(S) < \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_{\max}(S'). \tag{5}$$

Since $t - B > \max\{d_i - p_i, d_j - p_j\}$, we have

$$T_i(S) = t + p_i - d_i, \tag{6}$$

$$T_j(S) = t + p_i + p_j - d_j, \tag{7}$$

$$T_j(S') = t + p_j - d_j, \tag{8}$$

and

$$T_i(S') = t + p_j + p_i - d_i. \tag{9}$$

To show Eq. (5) hold, we divide it into the following three cases:
Case I: $T_{\max}(S) = T_i(S)$. From $p_i < p_j$ and Eqs. (6) and (9), we have

$$\begin{aligned}\alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_{\max}(S) &= \alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_i(S) < \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_i(S') \\ &\le \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_{\max}(S').\end{aligned}$$

Case II: $T_{\max}(S) = T_j(S)$. From $p_i < p_j$, $d_i < d_j$ and Eqs. (7) and (9), we have

$$\begin{aligned}\alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_{\max}(S) &= \alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_j(S) < \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_i(S') \\ &\le \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_{\max}(S').\end{aligned}$$

Case III: $T_{\max}(S) = T_k(S)$ for some $k \in \pi$ or π'. From $p_i < p_j$, we have

$$\begin{aligned}\alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_{\max}(S) &= \alpha C_i(S) + \alpha C_j(S) + (1-\alpha)T_k(S) < \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_k(S') \\ &\le \alpha C_j(S') + \alpha C_i(S') + (1-\alpha)T_{\max}(S').\end{aligned}$$

Thus, S dominates S' since Eq. (5) holds for all the three cases.

Property 2. *If jobs* $i, j \in AG_0$, $p_i < p_j$, *and* $d_i < t + p_i - B \le d_j - p_j$, *then S dominates S′.*

Property 3. *If jobs* $i, j \in AG_0$, $p_i < p_j$, $t + p_i - B \le d_i$, *and* $t + p_i + p_j - B \le d_j$, *then S dominates S′.*

Property 4. *If jobs* $i, j \in AG_0$, $p_i < p_j$, $\alpha(p_i - p_j) + (1-\alpha)(t + p_i + p_j - d_j - B) < 0$, *and* $d_j - p_i < t + p_j - B \le \min\{d_i - p_i, d_j\}$, *then S dominates S′.*

Property 5. *If jobs* $i, j \in AG_0$, $p_i < \alpha p_j$ *and* $d_j < t + p_j - B \le d_i - p_i$, *then S dominates S′.*

Property 6. *If jobs* $i, j \in AG_0$, $p_i < p_j$, $d_i < d_j$, $0 \le B - (t + p_j - d_j) < p_i$, *and* $t + p_i - B > d_i$, *then S dominates S′.*

Property 7. *If jobs* $i, j \in AG_0$, $p_i < \alpha p_j$, *and* $t - B > \max\{d_i - p_i, d_j - p_j\}$, *then S dominates S′.*

Property 8. *If jobs* $i, j \in AG_0$, $p_i < p_j$, $\alpha(p_i - p_j) + (1-\alpha)(d_i - d_j) < 0$, $t + p_j - B > d_j$, *and* $0 \le B - (t + p_i - d_i) < p_j$, *then S dominates S′.*

Property 9. *If jobs* $i, j \in AG_0$, $p_i < \alpha p_j$, $0 \le B - (t + p_i - d_i) < p_j$, $t + p_j - B > d_j$ *and* $d_i - p_i > d_j$, *then S dominates S′.*

Property 10. *If jobs* $i,j \in AG_0$, $p_i < p_j$, $t + p_i + p_j - B > d_j$, $t - B \le \min\{d_i - p_i, d_j - p_j\}$, *and* $d_i < d_j$, *then S dominates S'.*

Property 11. *If job* $i \in AG_0$, *job* $j \in AG_1$, *and* $t + p_i + p_j - d_j \le 0$, *then S dominates S'.*

To further facilitate the search process, we provide a proposition to determine the feasibility of a partial schedule and a proposition to determine the ordering of the remaining unscheduled jobs. Assume that (PS, US) is a sequence of jobs where PS is the scheduled part with k jobs and US is the unscheduled part with $(n–k)$ jobs. It is assumed among the unscheduled jobs, there are n_0 jobs from agent AG_0 and n_1 jobs from agent AG_1 where $n_0 + n_1 = n - k$. Let $S^* = (PS^*, US^*)$ be a sequence in which n_0 jobs from agent AG_0 are scheduled first in the shortest processing time (SPT) rule, followed by n_1 jobs from agent AG_1 in the earliest due date (EDD) rule. In addition, let $d^1_{(1)} \le d^1_{(2)} \le \ldots \le d^1_{(n_1)}$ denote the due dates of the remaining n_1 jobs from agent AG_1 when they are arranged in the EDD rule. Moreover, let $p^1_{(1)}, p^1_{(2)}, \ldots, p^1_{(n_1)}$ denote their corresponding processing times and $C_{[k]}$ be the completion times of the last job in PS.

Proposition 1. *If there is a unscheduled job j from agent* AG_1 *such that* $C_{[k]} + \sum_{i=1}^{j} p^1_{(i)} - d^1_{(j)} > 0$, *then sequence* (PS, US) *is infeasible.*

Proposition 2. *If* $\max_{k\in PS^*\cap AG_0}\{T_k(S^*)\} \le \max_{k\in PS\cap AG_0}\{T_k(S^*)\}$ *and* $\sum_{k\in PS^*\cap AG_1} U_k(S^*) = 0$, *then* $S^* = (PS^*, US^*)$ *dominates sequences of the type* (PS, US).

3.2. A lower bound

The performance of the branch-and-bound algorithm also depends on the efficiency of the lower bound. Assume that $S = (PS, US)$ be a schedule in which PS is the scheduled part with k jobs and US is the unscheduled part with $(n–k)$ jobs. Let t be the completion time of the last job in partial schedule PS, and $T^{PS}_{\max}$ be the maximum tardiness of jobs from agent AG_0 in PS. That is, $T^{PS}_{\max} = \max_{j\in AG_0\cap PS}\{0, C_j(S) - d_j\}$. In addition, it is assumed that among the unscheduled jobs, there are n_0 jobs from agent AG_0 and n_1 jobs from agent AG_1. Moreover, let $d^1_{(1)} \le d^1_{(2)} \le \ldots \le d^1_{(n_1)}$ denote the due dates of the remaining n_1 jobs from agent AG_1 when they are arranged in the EDD rule and $p^1_{(1)}, p^1_{(2)}, \ldots, p^1_{(n_1)}$ denote their associated processing times. In addition, let $p^0_{(1)} \le p^0_{(2)} \le \ldots \le p^0_{(n_0)}$ denote the processing times of the remaining n_0 jobs from agent AG_0 when they are arranged in the SPT rule and $d^0_{(1)} \le d^0_{(2)} \le \ldots \le d^0_{(n_0)}$ denote the due dates of the remaining n_0 jobs from agent AG_0 when they are arranged in the EDD rule. Note that $p^1_{(j)}$ and $d^1_{(j)}$ are from the same jobs, but $p^0_{(j)}$ and $d^0_{(j)}$ are not necessarily from the same job. We then construct n_0 pseudo jobs from agent AG_0 such that job j from agent AG_0 has processing time $p^0_{(j)}$ and due date $d^0_{(j)}$ for $j = 1, \ldots, n^0$. The basic idea to develop the lower bound is (1) to complete jobs from agent AG_1 as late as possible without violating the assumption of no tardy jobs from agent AG_1 are allowed, and (2) to insert n_0 pseudo jobs from agent AG_0 into the machine available periods where job preemption is allowed for jobs from agent AG_0. The procedures are divided into two phases, and given as follows.

```
Phase I:
Step 1: Set $i = n_1$, $st = \infty$, and $t_{n_1+1} = \infty$.
Step 2: If $d^1_{(i)} < st$, set $t_i = d^1_{(i)} - p^1_{(i)}$. Otherwise, set $t_i = st - p^1_{(i)}$.
Step 3: Set $st = t_i$ and $i = i - 1$. If $i \ge 1$, go to Step 2.
Step 4: Output $(t_1, t_2, \ldots, t_{n_1}, t_{n_1+1})$.
Phase II:
Step 1: Set $i = 1$, and $k = 0$.
Step 2: Set $k = k + 1$.
Step 3: If $t + p^0_{(i)} > t_k$, set $p^0_{(i)} = p^0_{(i)} - (t_k - t)$, $t = t_k + p^1_{(k)}$ and go to Step 2. Otherwise, set $t = t + p^0_{(i)}$ and $C^0_{(i)} = t$.
Step 4: If $i < n_0$, set $i = i + 1$ and go to Step 3.
Step 5: Output $\left(C^0_{(1)}, C^0_{(2)}, \ldots, C^0_{(n_0)}\right)$.
```

The purpose of Phase I is to calculate t_j, the latest time to start processing job j from agent AG_1 without violating the no-tardy jobs assumptions, whereas the main goal of Phase II is to schedule jobs from agent AG_0 into the machine available periods, and the output $C^0_{(j)}$ is the completion time for pseudo job j. Thus, the lower bound of the weighted combination of the total completion time and the maximum tardiness for sequence S is $LB(S) = \alpha\left[\sum_{i=1}^{k} C_{[i]}(PS)(1 - I_{[i]}) + \sum_{i=1}^{n_0} C^0_{(i)}\right] + (1-\alpha)\max\{T^{PS}_{\max}, \max_{1\le i\le n_0}\{C^0_{(i)} - d^0_{(i)}\}\}$.

3.3. Description of the branch-and-bound algorithm

A depth-first search is adopted in the branching procedure. In this paper, the algorithm assigns jobs in a forward manner starting from the first position. We first implement the proposed *SA* algorithms (discussed in the next section) to obtain a sequence as the initial incumbent solution. In the branching tree, we systematically work down the tree. We apply Proposition 1, Properties 1 to 11 to eliminate the dominated partial sequence. For the non-dominated nodes, we apply Proposition 2 to determine the sequence of the unscheduled jobs or compute the lower bound on the weighted combination of the total completion time and the maximum tardiness of jobs from agent AG_0. If the lower bound on the objective function for the partial sequence is greater than the incumbent solution, eliminate that node. If the objective function of the completed sequence is less than the incumbent solution, replace it as the new solution. The procedure is repeated until we explore the whole tree.

4. The simulated annealing algorithm

Evolutionary heuristic algorithms have been successfully applied to solve many combinatory optimization problems [29–33]. In this paper, the simulated annealing (*SA*) algorithm [34] is utilized to derive a near-optimal solution for the proposed problem. The essential elements of the *SA* algorithm included:

(1) Initial sequence: In this study, two initial sequences are used. In the first initial sequence, jobs from agent AG_1 are first placed according to the EDD rule, followed by jobs from agent AG_0 according to the SPT first rule, denoted as $SA_{\text{EDD+SPT}}$ in later analysis. In the second initial sequence, jobs from agent AG_1 are first placed according to the EDD rule, followed by jobs from agent AG_0 in the EDD rule, denoted as $SA_{\text{EDD+EDD}}$.

(2) Neighborhood generation: Three neighborhood generation methods are analyzed in this study. They are the pairwise interchange (*PI*), the extraction and forward-shifted reinsertion (*EFSR*), and the extraction and backward-shifted reinsertion (*EBSR*) movements. Depending on the initial sequences and the neighborhood generation movements, there are six *SA* algorithms considered in this study, and denoted as $SA^{PI}_{\text{EDD+SPT}}$, $SA^{EFSR}_{\text{EDD+SPT}}$, $SA^{EBSR}_{\text{EDD+SPT}}$, $SA^{PI}_{\text{EDD+EDD}}$, $SA^{EFSR}_{\text{EDD+EDD}}$, and $SA^{EBSR}_{\text{EDD+EDD}}$.

(3) Acceptance probability: The probability of acceptance is generated from an exponential distribution,

$$P(accept) = \exp(-k \times \Delta TC/\theta),$$

where ΔTC is the change in the objective function [35]. After some pretests, we choose $\theta = 5000$. If the weight combination of the total completion time and the maximum tardiness of jobs from agent AG_0 increases, the new sequence is accepted with probability r, where r is a uniform random number between 0 and 1.

(4) **Stopping condition:** According to our pretests, the procedure is stopped after $300n$ iterations, where n is the number of jobs.

5. Computational experiments

In this section, the computational experiments were conducted to evaluate the performance of the branch-and-bound and the *SA* algorithms. They were coded in Fortran 90 and run on a personal computer with 3.20 GHz Intel(R) Core(TM) i5 CPU 650 3.21 GHz and 3.45 GB RAM under Windows XP. The job processing times were generated from a uniform distribution over the integers 1–100 and the due dates of jobs were generated from another uniform distribution over the integers between $T(1-\tau-R/2)$ and $T(1-\tau+R/2)$, where R is the due date range factor, τ is the tardiness factor, and T is the total processing times of all the jobs.

The computational experiment consisted of five parts. In the first part of the experiment, the job size was fixed at 10, the weighted combination of the total completion time and the maximum tardiness α was 0.5, and the proportion of jobs from agent AG_1 was fixed at 50% to test the efficiency of the dominance properties and the lower bound separately, and the results are compared with the enumeration method. Eight combinations of (τ, R) values were tested, i.e. (0.25, 0.25), (0.25, 0.50),

Table 1
Results of the branch-and-bound algorithm and enumeration method with $n = 10$, $P = 0.5$, α=0.5.

τ	R	*BB_L* Number of nodes		*BB_P* Number of nodes		*BB_P + L* Number of nodes		Enumeration Number of nodes
		Mean	SD	Mean	SD	Mean	SD	
0.25	0.25	111755.8	85379.2	299.5	134.5	295.1	129.5	3628800
	0.50	79758.3	61617.9	299.6	132.8	273.9	104.5	3628800
	0.75	51875.5	62046.2	310.1	389.7	247.4	229.5	3628800
0.50	0.25	10078.9	14995.6	107.7	74.7	104.1	72.1	3628800
	0.50	9408.5	16062.1	142.6	156.9	136.2	151.5	3628800
	0.75	12953.7	29669.8	164.0	167.9	156.5	152.9	3628800
0.75	0.25	812.5	1635.7	41.6	24.0	41.5	24.0	3628800
	0.50	772.8	1155.4	45.5	33.7	44.8	32.8	3628800

Table 2
The performance of the branch-and-bound algorithm with $n = 12$, $P = 0.5$, α=0.5.

τ	R	Number of nodes		CPU time	
		Mean	SD	Mean	SD
0.25	0.25	952.26	443.83	0.006	0.008
	0.5	961.62	442.47	0.005	0.007
	0.75	1239.66	1704.13	0.008	0.011
0.5	0.25	300.98	193.29	0.001	0.004
	0.5	465.16	447.05	0.003	0.006
	0.75	479.26	440.50	0.003	0.006
0.75	0.25	110.99	81.45	0.001	0.003
	0.5	117.65	159.42	0.001	0.004

Table 3
One-way ANOVA table for the number of nodes with $n = 12$, $P = 0.5$, α=0.5.

Source	SS	DF	MS	F	p-value
Due date factors	125427020	7	17918146	38.122	0.000
Error	372253902	792	470018		
Total	497680922	799			

Table 4
Two-way ANOVA table for the number of nodes with $n = 12$, $P = 0.5$, α=0.5.

Source	SS	DF	MS	F	p-value
Factor (τ)	60683304	1	60683304	97.664	0.000
Factor (R)	5538663	2	2769331	4.457	0.012
Interaction	1759012	2	879506	1.415	0.244
Error	369081068	594	621349		
Total	437062047	599			

(0.25, 0.75), (0.5, 0.25), (0.5, 0.50), (0.5, 0.75), (0.75, 0.25), and (0.75, 0.50). The branch-and-bound algorithm with the dominance properties is denoted by BB_P, and the branch-and-bound algorithm with only the lower bound is denoted by BB_L, while the branch-and-bound algorithm with the properties and the lower bound is denoted by BB_P + L. The average and standard deviation of the number of nodes were reported for the branch-and-bound algorithms, while the number of nodes, 10!, was given for the enumeration method. 100 replications were randomly generated for each condition and the results were presented in Table 1. It is seen that the dominance properties and the lower bound benefit the searching process in terms of the number of nodes explored. Thus, the branch-and-bound algorithm with the properties and the lower bound was used in later analysis.

In the second part of the experiment, the effects of the due date factors τ and R to the performance of the branch-and-bound algorithm were studied. The values of parameters were the same as those in Table 1, except the job size was fixed at 12. The mean and the standard deviation of the number of nodes and the mean and the standard deviation of the CPU time (in seconds) were reported for the branch-and-bound algorithm. 100 replications were randomly generated for each case and the results were presented in Table 2. A one-way analysis of variance (ANOVA) on the number of nodes of the branch-and-bound algorithm was constructed and given in Table 3. The resulting F-value was 38.122 with a p-value of close to 0, which indicated that the due date factors τ or R have statistically significant effects on the difficulty of the problem. To further analyze the impact of τ and R, a two-way ANOVA on the number of nodes for the first six cases was conducted and the results were presented in Table 4. It indicated that τ is a significant factor since its resulting F-value is 97.664 and a p-value of close to 0. In addition, the impact of R is significant with a resulting F-value of 4.457 and a p-value of 0.012. However, there is no significant indication of the interaction effects between τ and R since its resulting F-value is 1.415 and its associated p-value is 0.244. A closer look at Table 2 revealed that the problems are harder to solve as the value of τ is smaller. The main reason is that Proposition 1 and the lower bound are less powerful in that case. In addition, it was noted that the case $(\tau, R) = (0.25, 0.75)$ has the most number of nodes with an average of 1239.66 among the eight cases. Thus, it was used in the third part of the experiment.

Similar to the second part of the experiment, the third part was to test the effects of the coefficient α and the proportion of jobs from the second agent P. The job size was fixed at 12, and (τ, R) was (0.25, 0.75). Three different values of α (0.25, 0.5, 0.75), and of P (0.25, 0.50, 0.75) were used. 100 replications were randomly generated for each case and the results were presented in Table 5. A two-way ANOVA on the number of nodes was utilized to test the effects of the parameters to the performance of the branch-and-bound algorithm, and the result was reported in Table 6. The resulting F-value of α was

Table 5
The performance of the branch-and-bound algorithm with $n = 12$ and $\tau = 0.25$, $R = 0.75$.

P	α	Number of nodes		CPU time	
		Mean	SD	Mean	SD
0.25	0.25	566.6	667.1	0.004	0.007
	0.5	474.3	392.7	0.002	0.006
	0.75	453.9	374.0	0.003	0.006
0.5	0.25	1136.4	1113.0	0.006	0.009
	0.5	1239.7	1704.1	0.008	0.011
	0.75	1234.9	1576.9	0.007	0.011
0.75	0.25	958.7	1324.8	0.005	0.009
	0.5	737.6	945.6	0.005	0.008
	0.75	1109.8	1765.0	0.006	0.012

Table 6
ANOVA table for the number of nodes with $n = 12$ and $\tau = 0.25$, $R = 0.75$.

Source	SS	DF	MS	F	p-value
Factor (P)	76059350	2	38029675	26.061	0.000
Factor (α)	2037967	2	1018983	0.698	0.498
Interaction	6371767	4	1592942	1.092	0.359
Error	1300175652	891	1459232		
Total	1384644736	899			

Table 7
The performance of the branch-and-bound algorithm (α=0.5).

n	τ	R	P	Number of nodes		CPU time	
				Mean	SD	Mean	SD
16	0.25	0.50	0.25	3654.0	7230.1	0.03	0.06
			0.50	14300.7	27645.3	0.12	0.22
			0.75	28873.3	43994.4	0.27	0.37
		0.75	0.25	5045.4	7508.4	0.04	0.06
			0.50	15669.7	18501.6	0.13	0.15
			0.75	12879.9	31172.2	0.12	0.27
	0.50	0.50	0.25	15844.5	44713.8	0.12	0.35
			0.50	5679.1	6499.3	0.05	0.05
			0.75	2137.1	2936.6	0.02	0.03
		0.75	0.25	18269.0	56922.6	0.13	0.42
			0.50	15293.9	23433.1	0.12	0.18
			0.75	7006.8	16149.6	0.06	0.14
20	0.25	0.50	0.25	16499.3	21739.4	0.19	0.24
			0.50	187894.2	436495.2	2.17	4.93
			0.75	653635.3	1258141.5	8.85	15.88
		0.75	0.25	111090.4	511446.1	1.13	5.11
			0.50	272520.9	560226.2	3.09	6.23
			0.75	317470.9	805000.9	4.35	10.93
	0.50	0.50	0.25	479755.2	3290233.5	5.30	37.44
			0.50	200706.0	539656.7	2.21	5.82
			0.75	36039.8	93098.6	0.50	1.30
		0.75	0.25	383251.0	1124434.6	4.08	12.01
			0.50	843572.2	3247301.9	9.12	36.53
			0.75	112184.1	395111.2	1.41	4.95
24	0.25	0.50	0.25	252867.7	730185.0	3.58	10.20
			0.50	2182444.9	5099456.3	32.73	73.43
			0.75	10881767.2	20926257.4	198.34	362.17
		0.75	0.25	739842.0	1664482.6	10.44	23.52
			0.50	12731415.3	28723857.3	186.32	412.87
			0.75	28187420.7	110415273.5	486.52	1862.78
	0.50	0.50	0.25	21904170.1	94524057.3	299.96	1299.99
			0.50	7246913.0	19021221.9	100.86	266.49
			0.75	519383.6	1598286.6	8.94	25.94
		0.75	0.25	7114554.9	17944690.0	95.45	245.22
			0.50	22155286.8	75160846.0	321.92	1102.85
			0.75	5764492.3	22341614.3	100.18	375.64

Table 8
The error percentages of the proposed simulated annealing algorithms.

n	τ	R	P	$SA^{PI}_{\mathrm{EDD}+SPT}$		$SA^{EFSR}_{\mathrm{EDD}+SPT}$		$SA^{EBSR}_{\mathrm{EDD}+SPT}$		$SA^{PI}_{\mathrm{EDD+EDD}}$		$SA^{EFSR}_{\mathrm{EDD+EDD}}$		$SA^{EBSR}_{\mathrm{EDD+EDD}}$		CombinedSA	
				Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
16	0.25	0.50	0.25	0.25	0.87	1.60	2.76	8.09	7.07	0.13	0.32	1.28	2.09	8.22	7.89	0.04	0.07
			0.50	0.25	1.06	5.38	8.57	14.57	14.99	0.32	1.46	4.09	7.01	11.21	13.38	0.02	0.06
			0.75	0.98	5.00	5.78	9.04	14.62	29.03	1.50	6.48	6.18	12.68	12.54	27.20	0.02	0.20
		0.75	0.25	0.14	0.26	0.69	1.45	7.76	9.19	0.12	0.26	0.74	1.29	5.64	5.08	0.04	0.12
			0.50	0.20	0.67	2.06	4.10	12.15	20.18	0.17	0.66	2.42	5.13	8.15	9.89	0.03	0.09
			0.75	0.12	0.54	1.63	6.26	4.62	12.04	0.58	4.70	0.62	3.32	5.65	18.90	0.00	0.00
	0.50	0.50	0.25	0.08	0.13	0.61	1.73	9.68	10.16	0.10	0.15	0.80	1.90	4.91	5.57	0.03	0.06
			0.50	0.16	1.11	1.20	2.94	9.10	12.99	0.21	1.39	1.04	2.71	5.22	8.00	0.02	0.06
			0.75	0.17	1.66	0.18	1.65	0.81	3.36	0.17	1.65	0.33	2.16	0.81	3.63	0.16	1.65
		0.75	0.25	0.14	0.28	0.79	1.28	6.71	9.95	0.10	0.21	0.62	1.02	4.59	4.51	0.03	0.11
			0.50	0.29	1.14	1.62	3.26	7.26	11.31	0.20	0.82	1.85	3.07	6.04	9.06	0.01	0.04
			0.75	0.49	2.65	1.42	6.23	3.99	7.08	0.49	2.64	1.22	4.18	2.93	5.98	0.10	0.95
20	0.25	0.50	0.25	0.13	0.27	1.30	2.13	10.34	8.06	0.16	0.40	1.49	2.27	10.00	10.01	0.04	0.08
			0.50	0.74	2.51	3.71	5.36	9.00	8.35	0.65	2.56	4.84	5.73	10.28	10.64	0.13	1.13
			0.75	1.63	6.52	7.69	11.29	10.34	14.33	1.82	7.34	4.14	7.60	11.97	19.44	0.02	0.12
		0.75	0.25	0.10	0.17	1.07	1.64	10.19	8.91	0.11	0.20	0.94	1.62	8.85	8.45	0.04	0.07
			0.50	0.58	2.43	2.66	5.74	8.38	8.82	0.34	1.26	2.66	4.66	9.97	13.18	0.12	0.79
			0.75	0.10	0.38	1.01	4.05	9.05	35.58	0.13	0.39	0.94	3.29	7.28	19.98	0.00	0.02
	0.50	0.50	0.25	0.09	0.20	1.14	1.75	12.70	12.11	0.12	0.24	0.87	1.48	8.38	8.42	0.03	0.05
			0.50	0.10	0.33	1.47	2.79	13.48	11.05	0.12	0.75	2.14	3.87	9.01	11.05	0.02	0.03
			0.75	0.12	1.08	0.25	1.23	1.26	4.31	0.12	1.08	0.44	2.21	1.16	4.24	0.11	1.08
		0.75	0.25	0.11	0.20	0.66	0.91	10.60	8.18	0.17	0.34	0.90	1.30	7.43	6.16	0.06	0.16
			0.50	0.33	1.32	2.77	4.08	9.75	9.53	0.36	1.36	2.76	3.86	9.45	10.19	0.10	0.65
			0.75	0.24	1.72	1.01	3.39	6.37	11.37	0.30	1.76	2.72	9.07	5.62	13.62	0.03	0.24
24	0.25	0.50	0.25	0.26	0.84	1.58	2.11	13.84	10.22	0.16	0.68	1.63	2.30	10.38	8.43	0.04	0.11
			0.50	0.68	2.17	4.38	6.01	8.45	8.41	0.70	2.30	4.87	7.75	13.12	14.02	0.14	0.64
			0.75	1.57	5.79	7.15	10.05	9.97	11.91	0.88	4.59	5.66	13.40	14.58	22.88	0.45	3.51
		0.75	0.25	0.17	0.33	1.20	1.50	12.53	8.45	0.12	0.33	1.07	1.39	9.90	8.53	0.06	0.22
			0.50	0.56	1.87	4.05	6.44	8.08	7.77	0.54	1.95	3.66	5.11	9.10	8.39	0.08	0.37
			0.75	0.55	2.71	1.98	6.33	9.50	21.49	0.52	2.75	1.84	6.15	9.73	25.50	0.31	2.21
	0.50	0.50	0.25	0.13	0.40	1.20	1.50	15.18	8.21	0.11	0.31	1.34	1.73	11.18	8.06	0.04	0.14
			0.50	0.34	1.57	2.36	3.75	18.33	12.91	0.10	0.74	2.00	3.56	10.19	9.37	0.08	0.73
			0.75	0.02	0.05	0.76	3.42	2.43	5.37	0.15	1.27	0.19	1.47	2.01	5.50	0.00	0.04
		0.75	0.25	0.14	0.30	0.93	1.01	11.57	7.35	0.12	0.26	1.10	1.62	9.55	7.19	0.06	0.22
			0.50	0.28	1.06	2.69	3.93	12.04	9.52	0.28	1.25	2.34	3.01	10.34	9.73	0.04	0.22
			0.75	0.47	2.29	2.74	5.80	7.93	10.49	0.95	3.66	2.56	5.73	8.07	10.62	0.17	1.48

0.698 with a p-value of 0.498, which implied that α does not affect the performance of the branch-and-bound algorithm. On the other hand, the statistical test showed that the proportion of jobs from the second agent P is a significant factor with an F-value of 26.061 and a p-value of close to 0. There was also an indication of no interaction effects between these two factors since its corresponding F-value and p-value were 1.092 and 0.359. It was seen from Table 5 that the problems are easier to solve when the value of P is smaller. The main reason was that the properties are more powerful in that case.

The main purpose of the fourth part of the experiment was to study the impact of the number of jobs to the performance of the branch-and-bound algorithms, and the accuracy of the proposed simulated annealing algorithms. Three different job sizes (n = 16, 20 and 24) were tested. Since problems were harder to solve if the due date factor τ was smaller or R was larger, and the proportion of jobs from the second agent P were also found to be the significant factors in the previous experiments, they were also considered in the experiment. Two different values of τ (0.25, 0.50), two values of R (0.5, 0.75) and three values of P (0.25, 0.50, 0.75) were used. We recorded the mean and standard deviation of the number of nodes and of the CPU time (in seconds) for the branch-and-bound algorithm, while only recording the mean and standard deviation of the error percentages of the SA algorithms, where the last one, denoted as the combined one, is the minimum value of the first six proposed SA algorithms. The execution time of SA algorithm was not recorded since they were finished within a second. For each condition, 100 replications were generated and the results were given in Tables 7 and 8. It was seen that the number of nodes and execution time grows exponentially as the number of jobs increases, and the standard deviations of the numbers of nodes are many times of their mean numbers of nodes since the problem is NP-hard. It was also observed that the branch-and-bound algorithm could solve problems of up to 24 jobs in a reasonable amount of time. The most time-consuming case took an average of 486 s when $(\tau, R, P) = (0.25, 0.75, 0.75)$. As to the performance of the SA algorithms, it was seen that $SA^{PI}_{\mathrm{EDD+SPT}}$ and $SA^{PI}_{\mathrm{EDD+EDD}}$ perform better, followed by $SA^{EFSR}_{\mathrm{EDD+SPT}}$ and $SA^{EFSR}_{\mathrm{EDD+EDD}}$, and $SA^{EBSR}_{\mathrm{EDD+SPT}}$ and $SA^{EBSR}_{\mathrm{EDD+EDD}}$ have the worst performance. It implied that the PI movement yields a better result, and the $EBSR$ yields the worst result overall. However, there was no clear dominance relation between these six algorithms, since the error percentages of the combined SA were much smaller than those of the six SA algorithms. Moreover, it was noted that average error percentage the combined SA

Table 9
The RDP and n_T of the proposed simulated annealing algorithms for large job-sized problems.

n	τ	R	P	$SA^{PI}_{\text{EDD}+SPT}$ RDP			$SA^{EFSR}_{\text{EDD}+SPT}$ RDP			$SA^{EBSR}_{\text{EDD}+SPT}$ RDP			$SA^{PI}_{\text{EDD+EDD}}$ RDP			$SA^{EFSR}_{\text{EDD+EDD}}$ RDP			$SA^{EBSR}_{\text{EDD+EDD}}$ RDP		
				Mean	SD	n_T	Mean	SD	n_T	Mean	SD	n_T	Mean	SD	n_T	Mean	SD	n_T	Mean	SD	n_T
100	0.25	0.50	0.25	0.15	0.21	39	1.21	0.77	2	3.67	1.78	0	0.08	0.17	59	2.85	1.74	0	5.28	4.12	0
			0.50	1.73	1.79	23	0.83	0.98	35	12.07	5.80	0	1.53	1.60	24	2.83	3.35	20	15.32	8.81	0
			0.75	5.22	9.33	37	3.83	4.01	25	33.40	27.77	3	4.42	8.12	40	2.35	5.42	49	33.05	33.92	2
		0.75	0.25	0.17	0.21	28	1.55	0.71	0	4.61	2.06	0	0.04	0.09	73	2.39	1.32	0	6.74	3.88	0
			0.50	0.91	1.04	17	0.52	0.73	41	10.84	5.13	0	0.53	0.85	40	5.76	4.24	2	14.52	7.75	0
			0.75	0.48	2.67	62	0.01	0.07	94	11.44	21.89	39	0.48	2.70	61	0.01	0.08	94	10.00	17.43	35
	0.50	0.50	0.25	0.04	0.06	42	1.67	0.81	0	6.99	2.53	0	0.02	0.05	60	1.93	1.07	0	14.69	4.61	0
			0.50	0.22	0.56	40	0.81	1.08	15	24.34	6.04	0	0.30	0.67	45	3.21	2.88	5	27.90	8.56	0
			0.75	0.11	0.63	48	0.30	1.05	16	8.23	5.79	3	0.09	0.52	57	0.56	1.54	24	7.35	6.70	2
		0.75	0.25	0.03	0.06	56	2.10	0.89	0	10.11	2.54	0	0.03	0.06	54	2.19	1.00	0	16.64	5.13	0
			0.50	0.13	0.21	44	0.99	0.81	2	20.87	5.39	0	0.11	0.30	56	7.01	3.33	0	24.65	8.41	0
			0.75	0.60	1.65	46	1.10	1.78	15	23.14	8.67	0	0.94	2.27	42	4.54	3.92	6	25.71	11.96	0
200	0.25	0.50	0.25	0.53	0.32	1	0.11	0.15	48	2.68	1.13	0	0.08	0.12	51	4.12	1.67	0	3.01	1.45	0
			0.50	3.96	1.93	1	0.16	0.42	83	13.99	7.47	1	2.75	1.66	3	3.44	3.44	12	12.77	6.93	0
			0.75	7.01	9.93	16	3.45	4.03	18	25.56	23.28	2	6.02	8.71	31	2.58	4.59	33	27.58	22.33	1
		0.75	0.25	0.45	0.36	6	0.65	0.27	1	3.53	1.34	0	0.01	0.04	93	3.46	1.25	0	3.75	1.99	0
			0.50	3.15	1.57	1	0.04	0.13	89	16.54	6.78	0	1.38	0.99	10	10.69	4.51	0	15.55	8.11	0
			0.75	0.27	1.50	4	0.00	0.00	47	1.60	9.52	69	0.18	1.17	9	0.00	0.01	34	1.05	4.85	60
	0.50	0.50	0.25	0.09	0.12	26	0.69	0.20	0	5.81	1.70	0	0.02	0.05	74	3.27	0.93	0	11.94	4.24	0
			0.50	0.59	0.56	13	0.21	0.49	61	27.69	5.18	0	0.60	0.60	25	2.43	1.64	1	35.95	7.40	0
			0.75	0.12	0.48	24	0.24	0.64	38	10.65	6.66	0	0.05	0.27	40	0.87	1.52	12	9.97	6.58	1
		0.75	0.25	0.08	0.08	16	0.75	0.13	0	9.26	2.02	0	0.00	0.01	84	3.24	0.94	0	15.19	4.23	0
			0.50	0.34	0.34	19	0.16	0.19	37	25.69	4.22	0	0.16	0.23	44	10.78	3.10	0	33.12	5.90	0
			0.75	1.16	1.41	18	0.21	0.46	53	31.28	8.76	0	1.22	1.42	22	6.15	4.57	7	39.13	11.25	0

Table 10
The CPU times (s) of the proposed simulated annealing algorithms for large job-sized problems.

n	τ	R	P	$SA^{PI}_{\text{EDD}+SPT}$		$SA^{EFSR}_{\text{EDD}+SPT}$		$SA^{EBSR}_{\text{EDD}+SPT}$		$SA^{PI}_{\text{EDD+EDD}}$		$SA^{EFSR}_{\text{EDD+EDD}}$		$SA^{EBSR}_{\text{EDD+EDD}}$	
				Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
100	0.25	0.50	0.25	0.12	0.01	0.09	0.01	0.16	0.01	0.12	0.01	0.09	0.01	0.16	0.01
			0.50	0.11	0.01	0.09	0.01	0.16	0.01	0.11	0.01	0.09	0.01	0.17	0.02
			0.75	0.12	0.01	0.11	0.01	0.15	0.01	0.12	0.01	0.11	0.01	0.15	0.01
		0.75	0.25	0.13	0.01	0.09	0.01	0.19	0.02	0.13	0.01	0.09	0.01	0.20	0.02
			0.50	0.13	0.01	0.09	0.01	0.23	0.03	0.13	0.01	0.10	0.01	0.25	0.03
			0.75	0.13	0.01	0.11	0.01	0.20	0.02	0.13	0.01	0.11	0.01	0.20	0.02
	0.50	0.50	0.25	0.16	0.01	0.09	0.01	0.27	0.03	0.17	0.01	0.10	0.01	0.30	0.04
			0.50	0.18	0.02	0.12	0.01	0.29	0.03	0.18	0.02	0.12	0.01	0.30	0.03
			0.75	0.21	0.03	0.21	0.03	0.26	0.03	0.22	0.03	0.22	0.03	0.26	0.02
		0.75	0.25	0.20	0.02	0.10	0.01	0.47	0.08	0.20	0.02	0.10	0.01	0.51	0.09
			0.50	0.24	0.03	0.13	0.01	0.63	0.12	0.25	0.03	0.13	0.01	0.64	0.11
			0.75	0.31	0.06	0.20	0.03	0.52	0.10	0.31	0.06	0.21	0.04	0.54	0.09
200	0.25	0.50	0.25	0.45	0.01	0.32	0.01	0.60	0.03	0.45	0.02	0.33	0.01	0.62	0.03
			0.50	0.43	0.02	0.35	0.01	0.68	0.04	0.44	0.02	0.36	0.01	0.70	0.05
			0.75	0.45	0.02	0.41	0.02	0.60	0.03	0.45	0.02	0.41	0.02	0.59	0.03
		0.75	0.25	0.51	0.02	0.33	0.01	0.77	0.05	0.51	0.02	0.34	0.01	0.78	0.05
			0.50	0.52	0.02	0.37	0.01	1.05	0.08	0.52	0.02	0.38	0.02	1.07	0.10
			0.75	0.49	0.03	0.42	0.02	0.82	0.05	0.49	0.03	0.42	0.02	0.84	0.06
	0.50	0.50	0.25	0.66	0.03	0.36	0.01	1.06	0.09	0.68	0.03	0.37	0.01	1.20	0.11
			0.50	0.69	0.04	0.44	0.02	1.21	0.10	0.70	0.04	0.44	0.02	1.23	0.11
			0.75	0.82	0.07	0.81	0.09	1.03	0.08	0.84	0.07	0.81	0.08	1.03	0.07
		0.75	0.25	0.82	0.05	0.37	0.01	2.17	0.34	0.83	0.05	0.37	0.01	2.47	0.43
			0.50	0.95	0.06	0.46	0.02	3.12	0.43	0.95	0.06	0.50	0.03	3.39	0.46
			0.75	1.22	0.22	0.75	0.10	2.42	0.33	1.23	0.22	0.80	0.12	2.42	0.33

algorithm was less than 0.5% for all the tested cases, thus, it was recommended as the ultimate algorithm since all the *SA* can be easily implemented.

The last part of the computational experiments was used to test the performance of the proposed *SA* algorithm when the number of jobs is large. We tested two job sizes, i.e., $n = 100$ and 200. We randomly generated 100 instances for each situation and we reported the results in Table 9. We recorded the mean and standard deviation of the relative deviation percentage (RDP). For instance, the RDP of the solution produced by $SA^{PI}_{\text{EDD+SPT}}$ is calculated as

$$\left(V^{PI}_{\mathrm{EDD+SPT}} - V^*\right)/V^* \times 100\%$$

where $V^{PI}_{\mathrm{EDD+SPT}}$ is the value of the weighted combination of the completion time and the maximum tardiness of jobs from the first agent generated by $SA^{PI}_{\mathrm{EDD+SPT}}$ and V^* is the minimum value obtained from the six algorithms. In addition, we recorded the number of times (n_T) it yields the minimum value. The result was presented in Table 9. We also recorded the mean and standard deviation of the execution time (in seconds) in Table 10. It was seen from Table 9 that $SA^{PI}_{\mathrm{EDD+EDD}}$ has the best performance in terms of mean RDP and the number of times it yields the minimum value. However, it did not perform well for all the cases. For instance, the mean RDP and n_T of $SA^{PI}_{\mathrm{EDD+EDD}}$ were 2.75% and 3 when $(n, \tau, R, P) = (200, 0.25, 0.5, 0.5)$, in this case $SA^{EFSR}_{\mathrm{EDD+SPT}}$ is the best with a mean RDP of 0.16% and $n_T = 83$. Moreover, it was seen from Table 10 that the execution time of the algorithms was only a few seconds. Thus, the combined simulated annealing algorithm is recommended when the number of jobs is large.

6. Conclusions

In this paper we studied a two-agent single-machine scheduling problem where the objective is to minimize the weighted combination of the completion time and maximum tardiness of the jobs of the first agent, given that no tardy jobs are allowed for the second agent. We proposed a branch-and-bound algorithm to solve the problem, and a combined simulated annealing algorithm to find near-optimal solutions. We conducted computational experiments to evaluate the performance of the proposed algorithms. The computational results showed that the branch-and-bound algorithm can solve problems with up to 24 jobs in a reasonable amount of time. It also showed that the performance of the combined *SA* algorithm is very good, yielding an average error percentage error of less than 0.5% for all the tested cases.

Acknowledgements

The authors are grateful to the editor and the referees, whose constructive comments have led to a substantial improvement in the presentation of the paper. This work was supported by the NSC of Taiwan, under NSC 101-2221-E-035-021.

References

[1] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, second ed., Prentice Hall, New Jersey, 2002.
[2] B. Naderi, S.M.T. Fatemi Ghomi, M. Aminnayeri, M. Zandieh, Scheduling open shops with parallel machines to minimize total completion time, J. Comput. Appl. Math. 235 (2011) 1275–1287.
[3] J.B. Wang, C.M. Wei, Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties, Appl. Math. Comput. 217 (2011) 8093–8099.
[4] R. Rudek, The strong NP-hardness of the maximum lateness minimization scheduling problem with the processing-time based aging effect, Appl. Math. Comput. 218 (2012) 6498–6510.
[5] Y.Y. Xiao, R.Q. Zhang, Q.H. Zhao, I. Kaku, Permutation flow shop scheduling with order acceptance and weighted tardiness, Appl. Math. Comput. 218 (2012) 7911–7916.
[6] W.C. Lee, Z.S. Lu, Group scheduling with deteriorating jobs to minimize the total weighted number of late jobs, Appl. Math. Comput. 218 (2012) 8750–8757.
[7] J.M. Peha, Heterogeneous-criteria scheduling: minimizing weighted number of tardy jobs and weighted completion time, Comput. Oper. Res. 22 (1995) 1089–1100.
[8] K. Kim, B.C. Paulson, C.J. Petrie, V.R. Lesser, Compensatory negotiation for agent-based schedule coordination, CIFE working paper #55, Stanford University, Stanford, CA, 1999.
[9] M.A. Kubzin, V.A. Strusevich, Planning machine maintenance in two-machine shop scheduling, Oper. Res. 54 (2006) 789–800.
[10] H. Balasubramanian, J.W. Fowler, A.B. Keha, M.E. Pfund, Scheduling interfering job sets on parallel machines, Eur. J. Oper. Res. 199 (2009) 55–67.
[11] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, A. Pacifici, Scheduling problems with two competing agents, Oper. Res. 52 (2004) 229–242.
[12] K.R. Baker, J.C. Smith, A multiple-criterion model for machine scheduling, J. Sched. 6 (2003) 7–16.
[13] J.J. Yuan, W.P. Shang, Q. Feng, A note on the scheduling with two families of jobs, J. Sched. 8 (2005) 537–542.
[14] T.C.E. Cheng, C.T. Ng, J.J. Yuan, Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs, Theor. Comput. Sci. 362 (2006) 273–281.
[15] C.T. Ng, T.C.E. Cheng, J.J. Yuan, A note on the complexity of the problem of two-agent scheduling on a single machine, J. Comb. Optim. 12 (2006) 387–394.
[16] A. Agnetis, D. Pacciarelli, A. Pacifici, Multi-agent single machine scheduling, Ann. Oper. Res. 150 (2007) 3–15.
[17] T.C.E. Cheng, C.T. Ng, J.J. Yuan, Multi-agent scheduling on a single machine with max-form criteria, European Journal of Operational Research 188 (2008) 603–609.
[18] K.B. Lee, B.C. Choi, J.Y.T. Leung, M.L. Pinedo, Approximation algorithms for multi-agent scheduling to minimize total weighted completion time, Inf. Process. Lett. 109 (2009) 913–917.
[19] A. Agnetis, G. Pascale, D. Pacciarelli, A Lagrangian approach to single-machine scheduling problems with two competing agents, J. Sched. 12 (2009) 401–415.
[20] J.Y.T. Leung, M. Pinedo, G.H. Wan, Competitive two agents scheduling and its applications, Oper. Res. 58 (2010) 458–469.
[21] G.H. Wan, S.R. Vakati, J.Y.T. Leung, M. Pinedo, Scheduling two agents with controllable processing times, Eur. J. Oper. Res. 205 (2010) 528–539.
[22] P. Liu, L. Tang, X. Zhou, Two-agent group scheduling with deteriorating jobs on a single machine, Int. J. Adv. Manuf. Technol. 47 (2010) 657–664.
[23] P. Liu, X. Zhou, L. Tang, Two-agent single-machine scheduling with position- dependent processing times, Int. J. Adv. Manuf. Technol. 48 (2010) 325–331.
[24] W.C. Lee, W.J. Wang, Y.R. Shiau, C.C. Wu, A single-machine scheduling problem with two-agent and deteriorating jobs, Appl. Math. Modell. 34 (2010) 3098–3107.
[25] C.C. Wu, S.K. Huang, W.C. Lee, Two-agent scheduling with learning consideration, Comput. Ind. Eng. 61 (2011) 1324–1335.
[26] W.C. Lee, S.K. Chen, C.C. Wu, Branch-and-bound and simulated annealing algorithms for a two-agent scheduling problem, Expert Syst. Appl. 37 (2010) 6594–6601.

[27] T.C.E. Cheng, W.H. Wu, S.R. Cheng, C.C. Wu, Two-agent scheduling with position-based deteriorating jobs and learning effects, Appl. Math. Comput. 217 (2011) 8804–8824.
[28] A. Agnetis, Combinatorial models for multi-agent scheduling problems, in: Proceedings of the 12th International Conference Devoted to Project Management and Scheduling, Tours, France, 2010, pp. 37–40.
[29] C.H. Liu, Using genetic algorithms for the coordinated scheduling problem of a batching machine and two-stage transportation, Appl. Math. Comput. 217 (2011) 10095–10104.
[30] T.M. Gwizdałła, The role of crossover operator in the genetic optimization of magnetic models, Appl. Math. Comput. 217 (2011) 9368–9379.
[31] R. Thangaraj, M. Pant, A. Abraham, P. Bouvry, Particle swarm optimization: hybridization perspectives and experimental illustrations, Appl. Math. Comput. 217 (2011) 5208–5226.
[32] C.Y. Low, C.J. Hsu, C.T. Su, A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance, Expert Syst. Appl. 37 (2010) 6429–6643.
[33] A. Azadeh, M.S. Sangari, A.S. Amiri, A particle swarm algorithm for inspection optimization in serial multi-stage processes, Appl. Math. Modell. 36 (2012) 1455–1464.
[34] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[35] D. Ben-Arieh, O. Maimon, Annealing method for PCB assembly scheduling on two sequential machines, Int. J. Comput. Integr. Manuf. 5 (1992) 361–367.