Journal of Marine Science and Technology

Volume 21 | Issue 2 Article 14

AN EXACT AND META-HEURISTIC APPROACH FOR TWO-AGENT SINGLE-MACHINE SCHEDULING PROBLEM

Wen-Hung Wu
Department of Business Administration, Kang-Ning Junior College of Medical Care and Management, Taipei, Taiwan, R.O.C., wu410226@knjc.edu.tw

Recommended Citation

Wu, Wen-Hung (2013) "AN EXACT AND META-HEURISTIC APPROACH FOR TWO-AGENT SINGLE-MACHINE SCHEDULING PROBLEM," *Journal of Marine Science and Technology*: Vol. 21 : Iss. 2 , Article 14.
DOI: 10.6119/JMST-013-0128-1
Available at: https://jmstt.ntou.edu.tw/journal/vol21/iss2/14

AN EXACT AND META-HEURISTIC APPROACH FOR TWO-AGENT SINGLE-MACHINE SCHEDULING PROBLEM

Acknowledgements

We are grateful to the Editor and two anonymous referees for their constructive comments on the previous version of our paper.

Journal of Marine Science and Technology, Vol. 21, No. 2, pp. 215-221 (2013)
DOI: 10.6119/JMST-013-0128-1

AN EXACT AND META-HEURISTIC APPROACH FOR TWO-AGENT SINGLE-MACHINE SCHEDULING PROBLEM

Wen-Hung Wu

Key words: scheduling, single-machine, two-agent, genetic algorithm.

ABSTRACT

In many real-life applications, it can be often found 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, etc. Moreover, scheduling with multiple agents is relatively unexplored. Based on this observation, this paper attempts to study a single-machine scheduling problem where the objective is to minimize the total tardiness of the first agent with the constraint that no tardy job is allowed for the second agent. In this study, we provide a branch-and-bound algorithm and a genetic algorithm for the optimal and near-optimal solutions. We also report a computational experiment to evaluate the impact of the parameters involving with proposed problem simulation settings.

I. INTRODUCTION

Scheduling with multiple agents has received growing attention in recently years. Agnetis *et al*. [1] and Baker and Smith [3] were independently the first authors to introduce the concept of multi-agent into scheduling problems. Yuan *et al*. [30] addressed two dynamic programming recursions in Baker and Smith [3] and developed a polynomial-time algorithm for the same problem. Cheng *et al*. [9] considered the feasibility model of multi-agent scheduling on a single machine where each agent's objective function is to minimize the total weighted number of tardy jobs. Ng *et al*. [23] studied 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*. [2] considered the scheduling problems when several agents, each owning a set of non-preemptive jobs, compete to perform their respective jobs on one shared processing resource. Each agent wants to minimize a certain cost function, which depends on the completion times of its jobs only. Cheng *et al*. [9] studied multi-agent scheduling on a single machine where the objective functions of the agents are of the max-form. Lee *et al*. [18] considered a multi-agent 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. Besides, for more multiple-agent works with time-dependent, we refer readers to Liu and Tang, Cheng *et al*., Wan *et al*., Liu *et al*., Wu *et al*., Mor and Mosheiov, Nong *et al*., and Yin *et al*., etc. [7, 10, 19-22, 24, 26-29]. For more recent scheduling problems faced by the manufacturing industry, but are from the same agent, the reader can refer to Hsu *et al*. [16], Shyr and Lee [25].

Due to the importance of multiple agents competing on the usage of a common processing resource in different application environments and different methodological fields, we studied two-agent scheduling on a single machine. The objective is to minimize the total tardiness of the jobs of the first agent with the restriction that no tardy job is allowed for the second agent.

The remainder of this paper is organized as follows: In Section II, the problem statement is given. In Section III, some dominance properties and a lower bound are presented. In Section IV, the details of three genetic algorithms are described. In Section V, the extensive computational experiments to assess the performance of all of the proposed algorithms are reported. The conclusion is given in the last section.

II. PROBLEM FORMULATION

The problem is described as follows. There are n jobs which belongs to one of the agents AG_0 or AG_1. For each job j, there is a normal processing time p_j, a due date d_j, and an agent code I_j, where $I_j = 0$ if $J_j \in AG_0$ or $I_j = 1$ if $J_j \in AG_1$. All the jobs are available at time zero. Under a schedule S,

Paper submitted 06/08/12; revised 12/11/12; accepted 01/28/13. *Author for correspondence: Wen-Hung Wu (e-mail: wu410226@knjc.edu.tw).*
Department of Business Administration, Kang-Ning Junior College of Medical Care and Management, Taipei, Taiwan, R.O.C.

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 J_j and $U_j(S) = 1$ if $T_j(S) > 0$, and zero otherwise. The objective of this paper is to find an optimal schedule to minimize $\sum_{j=1}^{n} T_j(S)(1-I_j) = 0$ subject to $\sum_{j=1}^{n} U_j(S) I_j = 0$.

III. BRANCH-AND-BOUND ALGORITHM

The classical single-machine total tardiness problem without agents was proved to be NP-hard. Thus, our problem is also NP-hard. Moreover, no relative computational results from the algorithm viewpoints for the problem have been reported. Thus, we will attempt to use the branch-and-bound technique and a genetic algorithm to search for the optimal solution and near optimal solution, respectively.

Below we will develop the branch-and-bound technique incorporating with some dominance rules to help searching for the optimal solution. Below are some adjacent properties.

1. Dominance Properties

In this subsection, some adjacent dominance rules are first derived by using the pairwise interchange method. Let 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 completion time of the last job in σ.

Property 1. If jobs $i, j \in AG_0$, $p_i < p_j$, and $t > \max\{d_i - p_i, d_j - p_j\}$, then S_1 dominates S_2.

Proof: From $t > \max\{d_i - p_i, d_j - p_j\}$, we have

$$T_i(S_1) = t + p_i - d_i, \quad (1)$$

$$T_j(S_1) = t + p_i + p_j - d_j. \quad (2)$$

$$T_j(S_2) = t + p_j - d_j, \quad (3)$$

and

$$T_i(S_2) = t + p_j + p_i - d_i, \quad (4)$$

From Eqs. (1)-(4), and $p_i < p_j$, we have

$$[T_j(S_2) + T_i(S_2)] - [T_i(S_1) + T_j(S_1)] = [2p_j + p_i] - [2p_i + p_j] > 0$$

and

$$C_j(S_1) = C_i(S_2).$$

Therefore, S_1 dominates S_2. The proof is completed.

Property 2. If job $i \in AG_0$, job $j \in AG_1$, $t + p_i < d_i < t + p_i + p_j$, and $t + p_i + p_j < d_j$, then S_1 dominates S_2.

Proof: From job $i \in AG_0$, job $j \in AG_1$, and $t + p_i < d_i < t + p_i + p_j$, it imply that $T_i(S_1) = 0$ and $T_i(S_2) = t + p_i + p_j - d_i$. Meanwhile, because $t + p_i + p_j < d_j$, we have $T_j(S_1) = 0$. Therefore, we have $[T_j(S_2) + T_i(S_2)] > [T_i(S_1) + T_j(S_1)]$.

Property 3. If job $i \in AG_0$, job $j \in AG_1$, $t + p_i < d_i$ and $t + p_i + p_j < d_j$, then S_1 dominates S_2.

Proof: From job $i \in AG_0$, job $j \in AG_1$, and $t + p_i > d_i$, it imply that job i is tardy in S_1 and S_2. $T_i(S_1) = t + p_i - d_i$ and $T_i(S_2) = t + p_i + p_j - d_i$. Meanwhile, because $t + p_i + p_j < d_j$, we have $T_j(S_1) = 0$ and $T_j(S_2) = 0$. Therefore, we have $[T_j(S_2) + T_i(S_2)] > [T_i(S_1) + T_j(S_1)]$.

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 $(n\text{-}k)$ jobs. Among the unscheduled jobs, let $p_{(1)} = \min_{J_j \in \pi^c}\{p_j\}$ and $d^1_{(1)} = \min_{J_j \in \pi^c \cap AG_1}\{d_j\}$. 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 arranged in the weighted smallest processing times (SPT) order and the unscheduled jobs in AG_1 arranged in the earliest due date rule (EDD) order, respectively.

Property 4. If all the unscheduled jobs belong to AG_0 and $C_{[k]} \geq \max_{j \in \pi^c}\{d_j\}$, then schedule (π, π^c) is dominated by schedule (π, π').

Proof: Since $C_{[k]} \geq \max_{j \in \pi^c}\{d_j\}$, all the unscheduled jobs are from AG_0 and tardy. So the SPT rule yields an optimal subschedule.

Property 5. If all the unscheduled jobs belong to AG_1 and no tardy job can be found in schedule (π, π''), then schedule (π, π^c) is dominated by schedule (π, π'').

Proof: Similar to Property 4.

Property 6. If $C_{[k]} + p_{(1)} > d^1_{(1)}$, then (π, π^c) is not a feasible sequence.

Proof: Since $C_{[k]} + p_{(1)} - d^1_{(1)} > 0$, one job of AG_1 among the unscheduled jobs must be tardy. So (π, π^c) is not a feasible solution.

2. A Lower Bound

A simple lower bound of the partial sequence will be developed in the following. Assume that π is a partial schedule

in which the order of the first k jobs is determined and let π^c 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 kth job in π. The completion time for the (k+j)th job is

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

Then a lower bound can be obtained as follows

$$\begin{aligned} \sum_{j=1}^{n} T_j(S)(1-I_j) &\geq \sum_{j=1}^{n} L_j(S)(1-I_j) \\ &= \sum_{j=1}^{n} (C_j(S) - d_j(S))(1-I_j) \\ &= \sum_{j=1}^{n_0} C_{(j)}(S) - \sum_{j=1}^{n_0} d_{(j)} = LB \end{aligned}$$

IV. GENETIC ALGORITHM

The branch-and-bound becomes very time consuming when the job size is getting larger. Meanwhile, a heuristic algorithm can supply time-saving approximate solution with small margin of error. Thus, we adopted three genetic algorithms (GAs) for near-optimal solution.

Genetic algorithms (GAs) are intelligent random search strategies which have been successfully applied to find near-optimal solutions of many complex problems [5-6, 16]. A genetic algorithm starts with a set of feasible solutions (population) and iteratively replaces the current population by a new population. It requires a suitable encoding for the problem and a fitness function that represents a measure of the quality of each encoded solution (chromosome or individual). The reproduction mechanism selects the parents and recombines them using a crossover operator to generate offsprings that are submitted to a mutation operator in order to alter them locally [12]. The procedures of the GA applied to solve the proposed problem were summarized in the following.

Representation of structure- In this study we adopt the method proposed by Etiler *et al*. [13] that a structure can be described as a sequence of the jobs in the problem.

Initial population- We randomly generate the initial population based on Bean [4]. In order to arrive at the final solution more quickly, three improvement techniques are applied in initial sequences. There are including pairwise interchange, backward-shifted reinsertion, and forward-shifted reinsertion [11]. In GA_1, initial sequences are improved by pairwise interchange. While in GA_2, initial sequences are improved by forward-shifted reinsertion. In GA_3, initial sequences are adopted by backward-shifted reinsertion.

Population size- The population size plays an important role in the computational process of GA. In a preliminary trial, the population size N is set at 40 in our computational experiment.

Fitness function- Following Iyer and Saxena [17], the fitness function assigns to each member of the population a value reflecting their relative superiority or inferiority. Our objective is to minimize the total tardiness. The fitness function of the strings can be calculated as follows:

$$f(S_i(v)) = \max_{1 \leq l \leq N} \left\{ \sum_{j=1}^{n} T_j(S_l(v)) \right\} - \sum_{j=1}^{n} T_j(S_i(v)),$$

where $S_i(v)$ is the ith string chromosome in the v-th generation, $\sum_{j=1}^{n} T_j(S_i(v))$ is the total tardiness of $S_i(v)$, and $f(S_i(v))$ is the fitness function of $S_i(v)$. Therefore, the probability, $P(S_i(v))$, of selection for a schedule is to ensure that the probability of selection for a sequence with lower value of the objective function is higher. Here $P(S_i(v))$ can be calculated as follows:

$$P(S_i(v)) = f(S_i(v)) / \sum_{l=1}^{N} f(S_l(v)).$$

This is also the criterion used for the selection of parents for the reproduction of children.

Crossover- This study adopts linear order crossover (LOX) method which is developed by Falkenauer and Bouffouix [14]. In a pilot study, in order to protect the best schedule which has the minimum total tardiness at each generation, we transfer this schedule to the next population with no change. This operation enables us to choose the higher crossover with the crossover rate P_c = 100%.

Mutation- In this study, the mutation rates (P_m) are set at 0.3 based on our preliminary experiment.

Selection- It is a procedure to select offspring from parents to the next generation. In our study, the population sizes are fixed at 40 from generation to generation. Excluding the best 10% schedule which has the minimum total tardiness, the rest 90% of the offsprings are generated from the parent chromosomes by the roulette wheel method.

Termination- The proposed GA's are terminated after 500 generations or the objective with zero in our preliminary experiment.

V. COMPUTATIONAL EXPERIMENT

A computational experiment was conducted to test the branch-and-bound algorithm and proposed genetic algorithms.

Table 1. Performance of the branch-and-bound and GA algorithms (n = 10, 12, 14).

				branch-and-bound algorithm				GA_1		GA_2		GA_3		GA^*	
n	τ	R	valid	CPU time		number of nodes		error percentages							
			sample size	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
10	0.25	0.25	50	4.6	0.9	785914	171846	0.02	0.12	0.00	0.00	0.49	3.11	0.00	0.00
		0.50	50	2.9	1.4	483862	253246	0.09	0.66	0.00	0.00	0.09	0.66	0.00	0.00
		0.75	50	0.4	0.7	69607	114075	0.43	1.74	0.17	1.19	0.31	1.52	0.17	1.19
	0.50	0.25	50	0.8	0.2	136659	34564	0.03	0.24	0.00	0.00	0.00	0.00	0.00	0.00
		0.50	50	0.7	0.3	120369	58854	0.68	4.81	0.78	4.84	0.09	0.62	0.00	0.00
		0.75	50	0.5	0.5	89639	80064	0.29	2.02	0.03	0.23	0.04	0.19	0.00	0.00
		Average		1.7	0.7	281008	118775	0.26	1.60	0.16	1.04	0.17	1.02	0.03	0.20
12	0.25	0.25	50	519.8	91.1	64757081	11917319	0.00	0.00	1.33	7.76	0.37	2.00	0.00	0.00
		0.50	50	270.2	136.4	33403302	17547217	0.01	0.07	1.00	6.99	0.03	0.14	0.00	0.00
		0.75	50	44.2	68.9	5424919	8514648	5.99	35.73	0.09	0.65	0.23	1.13	0.09	0.65
	0.50	0.25	50	60.6	19.6	7258565	2358229	0.29	1.86	0.00	0.00	0.05	0.33	0.00	0.00
		0.50	50	49.1	24.6	5925444	3015943	0.24	1.26	0.15	0.52	0.56	2.86	0.00	0.00
		0.75	50	29.5	24.6	3556758	3000702	0.35	1.40	0.35	1.90	0.30	1.37	0.00	0.00
		Average		162.3	60.9	20054345	7725676	1.15	6.72	0.49	2.97	0.25	1.30	0.02	0.11
14	0.25	0.25	0	-	-	-	-	-	-	-	-	-	-		
		0.50	5	5541.0	5297.7	447631420	427969908	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	44	1034.3	2533.8	85038885	209388961	489.26	3241.15	0.62	4.11	0.00	0.00	0.00	0.00
	0.50	0.25	50	5052.6	2019.8	440447020	179742131	0.05	0.25	0.10	0.45	0.01	0.04	0.01	0.04
		0.50	49	4292.1	2391.4	380001168	216586322	1.07	3.82	0.91	2.46	1.29	5.80	0.05	0.36
		0.75	49	2037.4	1639.4	178910871	146033731	1.11	7.62	0.60	1.64	1.71	7.72	0.02	0.10
		Average		3591.5	2776.4	306405873	235944211	98.30	650.57	0.44	1.73	0.60	2.71	0.01	0.10

The algorithms were coded in Fortran and run on Compaq Visual Fortran version 6.6 on a Intel(R) Core(TM)2 Quad CPU 2.66 GHz with 4 GB RAM on Windows XP. The experimental design follows Fisher's [16] framework. The job processing times were generated from a uniform distribution over the integers between 1 and 100. The due dates were generated from a uniform distribution over the range of integers $T(1 - \tau - R/2)$ to $T(1 - \tau + R/2)$, where τ is the tardiness factor, R is the due date range, and T is the sum of the processing times of all the jobs, i.e., $T = \sum_{i=1}^{n} p_i$. The combination of (τ, R) took the values (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 the branch-and-bound algorithm, the average and standard deviation numbers of nodes as well as the average and standard deviation execution times (in seconds) were recorded. For the three genetic algorithms, the mean and standard deviation error percentages were recorded, where the error percentage was calculated as

$$(GA_i - OP)/OP * 100\%,$$

where GA_i is the total tardiness obtained from the genetic algorithm and OP is the total tardiness of the optimal schedule. The computational times of the heuristic algorithms were not recorded since they were finished within a second.

The computational experiment consisted of small job numbers and big job numbers. In the first part of the experiment with small job numbers, three job sizes (n = 10, 12 and 14) were examined in the branch-and-bound algorithm. The same sets of instances were used to test the performance of the branch-and-bound and the genetic heuristic algorithms. As a consequence, 18 experimental situations were tested. A set of 50 instances were randomly tested for each case. Moreover, the algorithms were set to skip to the next set of data if the number of nodes exceeded 10^9. The instances with number of nodes less than 10^9 were denoted as solvable instances (valid sample size). The results are presented in Table 1.

As shown in Fig. 1 and Table 1, it indicated that the number of nodes in the instances is getting larger as the number of jobs increases. The instances with a bigger value of τ (τ = 0.5) is easily to solve than those with a smaller value of τ (τ = 0.25). The performance of R also has the same situation. For example, the instances with a bigger value of R (R = 0.75) is easily to solve than those with a smaller value of R (R = 0.25, 0.5). It can be observed in Table 1 that fixed n = 14, the most difficult case occurs at (τ, R) = (0.25, 0.25) where no instance can be solved out.

As to the performance of the proposed GA algorithms, out of the 18 cases, the performances of proposed genetic heuristics were not affected as the values of τ or R varied. Most of the mean error percentages of GA_1, GA_2, and GA_3 were less than 2% or below, except one case at (τ, R) = (0.25, 0.75) in GA_1 has a bigger mean error percentage. However, the situation was disappeared when we further combined three

Table 2. RDP of heuristic algorithms (n = 60, 80, 100).

			GA_1				GA_2				GA_3			
n	τ	R	CPU time		RDP		CPU time		RDP		CPU time		RDP	
			mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
60	0.25	0.25	0.09	0.01	0.86	2.76	0.08	0.01	10.77	9.18	0.08	0.01	7.27	9.22
		0.50	0.09	0.01	3.49	14.49	0.08	0.01	115.71	259.07	0.09	0.01	125.68	296.91
		0.75	0.08	0.01	0.00	0.00	0.09	0.01	4978.00	22320.87	0.08	0.01	3696.00	14394.23
	0.50	0.25	0.09	0.01	0.04	0.24	0.09	0.01	5.90	4.28	0.09	0.01	4.79	3.14
		0.50	0.09	0.01	0.01	0.06	0.09	0.01	12.35	8.24	0.09	0.01	12.40	7.70
		0.75	0.09	0.01	0.24	1.64	0.09	0.01	23.32	18.47	0.09	0.01	25.46	20.50
Average			0.09	0.01	0.77	3.20	0.09	0.01	857.67	3770.02	0.09	0.01	645.27	2455.28
80	0.25	0.25	0.12	0.01	0.80	2.91	0.12	0.01	7.91	6.01	0.12	0.01	7.82	5.99
		0.50	0.11	0.01	7.24	27.64	0.11	0.01	654.20	2161.53	0.11	0.01	220.52	443.90
		0.75	0.11	0.01	0.00	0.00	0.11	0.00	946.00	4787.28	0.11	0.01	3808.00	24982.86
	0.50	0.25	0.12	0.01	0.03	0.23	0.13	0.01	6.52	3.26	0.13	0.01	7.24	3.78
		0.50	0.12	0.01	0.04	0.28	0.13	0.01	18.21	9.42	0.13	0.01	17.87	9.99
		0.75	0.13	0.01	0.00	0.00	0.13	0.01	36.00	21.58	0.13	0.01	42.74	20.76
Average			0.12	0.01	1.35	5.18	0.12	0.01	278.14	1164.85	0.12	0.01	684.03	4244.55
100	0.25	0.25	0.15	0.01	0.07	0.51	0.17	0.02	13.36	6.06	0.16	0.01	12.41	7.31
		0.50	0.14	0.01	1.29	5.54	0.14	0.02	575.99	2032.50	0.15	0.01	1556.42	6256.08
		0.75	0.14	0.01	0.00	0.00	0.14	0.01	882.00	5714.77	0.14	0.01	7096.00	27727.35
	0.50	0.25	0.16	0.01	0.03	0.22	0.20	0.02	8.47	4.91	0.19	0.01	8.19	4.00
		0.50	0.17	0.02	0.00	0.00	0.19	0.02	20.40	9.35	0.19	0.01	20.61	8.72
		0.75	0.18	0.02	0.00	0.00	0.19	0.02	46.23	25.67	0.18	0.01	46.40	19.09
Average			0.15	0.01	0.23	1.05	0.17	0.01	257.74	1298.87	0.17	0.01	1456.67	5670.43

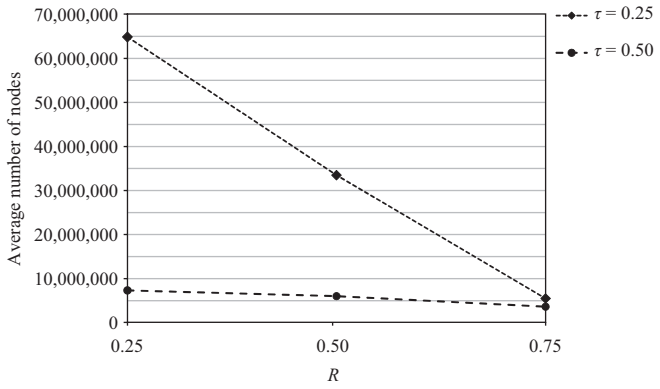

Fig. 1. Performance of the branch-and-bound algorithms (n = 12).

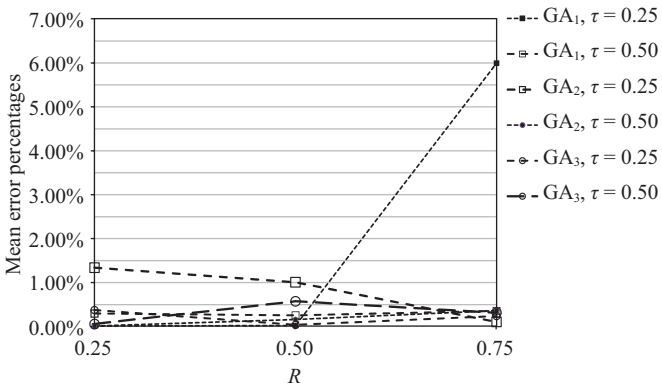

Fig. 2. Performance of the $GA_{1\sim3}$ algorithms (n = 12).

proposed three GAs into GA^* in which ($GA^* = \min\{GA_i, i = 1, 2, 3\}$. Table 1 further indicated that the mean error percentages of GA^* were reduced to 0.2% or below no matter that the values of τ or R varied.

In the second part of the experiment for large job-sized problems, the proposed heuristic algorithms were tested with three different numbers of jobs at n = 60, 80, and 100. The mean execution time and the mean relative deviance percentage were recorded for each heuristic. The relative deviance percentage (RDP) was given by

$$(GA_i - GA^*)/GA^* * 100\%,$$

where GA_i is the value of the objective function generated by the ith heuristic, and $GA^* = \min\{GA_i, i = 2, 3\}$ is the smallest value of the objective function obtained from the heuristics. The results are summarized in Table 2.

As shown in Fig. 3 and Table 2, it was observed that the mean RDP of GA_1 is lower than those of GA_2 and GA_3. The overall mean RDP of GA_1 was less than 2%. However, there is no absolutely dominance between the performances

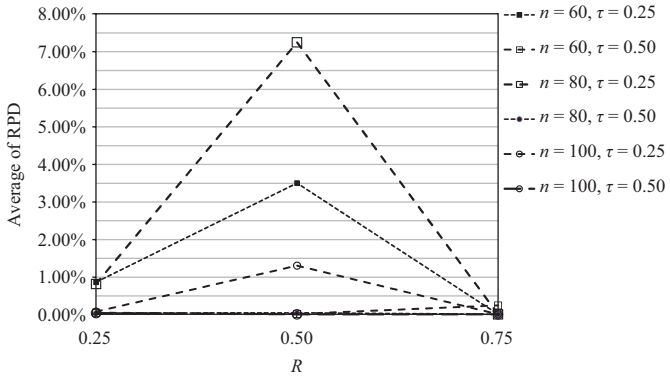

Fig. 3. Performance of the $GA_{1\sim3}$ algorithms (n = 60, 80, 100).

of the first three genetic algorithms. Thus, it is recommended to use the GA^* algorithm since it has both accuracy and the smallest RDP.

VI. CONCLUSIONS

This paper studied a single-machine two-agent scheduling problem here the objective is to minimize the total tardiness of the first agent with the constraint that no tardy job is allowed for the second agent. The contributions of this paper were; Firstly, a branch-and-bound algorithm incorporating with several dominances and a lower bound was proposed to derive an optimal solution, and then three genetic algorithms were provided for near-optimal solution. Finally, the impacts of the relative parameters about proposed problem were tested and reported.

The computational results also showed that with the help of the proposed heuristic initial solution, the branch-and-bound algorithm can solve the instances up to n = 14. Moreover, the computational experiments also showed that the proposed GA^* algorithm performed quite well in terms of accuracy and the smallest RDP.

ACKNOWLEDGMENTS

We are grateful to the Editor and two anonymous referees for their constructive comments on the previous version of our paper.

REFERENCES

1. Agnetis, A., Mirchandani, P. B., Pacciarelli, D., and Pacifici, A., "Scheduling problems with two competing agents," *Operations Research*, Vol. 52, pp. 229-242 (2004).
2. Agnetis, A., Pacciarelli, D., and Pacifici, A., "Multi-agent single machine scheduling," *Annals of Operations Research*, Vol. 150, pp. 3-15 (2007).
3. Baker, K. R. and Smith, J. C., "A multiple-criterion model for machine scheduling," *Journal of Scheduling*, Vol. 6, pp. 7-16 (2003).
4. Bean, J. C., "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal of Computing*, Vol. 6, pp. 154-160 (1994).
5. Beasley, D., Bull, D., and Martin, R. R., "An overview of genetic algorithms, part 1: fundamentals," *Journal of University Computing*, Vol. 15, pp. 58-69 (1993).
6. Chen, J. S., Pan, J. C. H., and Lin, C. M., "A hybrid genetic algorithm for the reentrant flowshop scheduling problem," *Expert Systems with Applications*, Vol. 34, pp. 570-577 (2008).
7. Cheng, T. C. E., Cheng, S.-R., Wu, W.-H., Hsu, P.-H., and Wu, C.-C., "A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations," *Computers and Industrial Engineering*, Vol. 60, pp. 534-541 (2011).
8. Cheng, T. C. E., Ng, C. T., and Yuan, J. J., "Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs," *Theoretical Computer Science*, Vol. 362, pp. 273-281 (2006).
9. Cheng, T. C. E., Ng, C. T., and Yuan, J. J., "Multi-agent scheduling on a single machine with max-form criteria," *European Journal of Operational Research*, Vol. 188, pp. 603-609 (2008).
10. Cheng, T. C. E., Wu, W. H., Cheng, S. R., and Wu, C. C., "Two-agent scheduling with position-based deteriorating jobs and learning effects," *Applied Mathematics and Computation*, Vol. 217, pp. 8804-8824 (2011).
11. Della Croce, F., Narayan, V., and Tadei, R., "The two-machine total completion time flow shop problem," *European Journal of Operational Research*, Vol. 90, pp. 227-237 (1996).
12. Essafi, I., Matib, Y., and Dauzere-Peres, S., "A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem," *Computers and Operations Research*, Vol. 35, pp. 2599-2616 (2008).
13. Etiler, O., Toklu, B., Atak, M., and Wilson, J., "A generic algorithm for flow shop scheduling problems," *Journal of Operations Research Society*, Vol. 55, No. 8, pp. 830-835 (2004).
14. Falkenauer, E. and Bouffoix, S., "A genetic algorithm for job shop," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 824-829 (1991).
15. Fisher, M. L., "A dual algorithm for the one-machine scheduling problem," *Mathematical Program*, Vol. 11, pp. 229-251 (1971).
16. Hsu, C.-J., Yang, Y.-J., and Yang, D.-L., "Due-date assignment and optimal maintenance activity scheduling problem with linear deteriorating jobs," J*ournal of Marine Science and Technology*, Vol. 19, No. 1, pp. 97-100 (2011).
17. Iyer, S. K. and Saxena, B. S., "Improved genetic algorithm for the permutation flowshop scheduling problem," *Computers and Operations Research*, Vol. 31, pp. 593-606 (2004).
18. Lee, K. B., Choi, B. C., Leung, J. Y. T., and Pinedo, M. L., "Approximation algorithms for multi-agent scheduling to minimize total weighted completion time," *Information Processing Letters*, Vol. 109, pp. 913-917 (2009).
19. Liu, P. and Tang, L., "Two-agent scheduling with linear deteriorating jobs on a single machine," *Lecture Notes in Computer Science*, Vol. 5092, pp. 642-650 (2008).
20. Liu, P., Yi, N., and Zhou, X., "Two-agent single-machine scheduling problems under increasing linear deterioration," *Applied Mathematical Modelling*, Vol. 35, No. 5, pp. 2290-2296 (2011).
21. Mor, B. and Mosheiov, G., "Scheduling problems with two competing agents to minimize minmax and minsum earliness measures," *European Journal of Operational Research*, Vol. 206, pp. 540-546 (2010).
22. Mor, B. and Mosheiov, G., "Single machine batch scheduling with two competing agents to minimize total flowtime," *European Journal of Operational Research*, Vol. 215, pp. 524-531 (2011).
23. Ng, C. T., Cheng, T. C. E., and Yuan, J. J., "A note on the complexity of the problem of two-agent scheduling on a single machine," *Journal of Combinatorial Optimization*, Vol. 12, pp. 387-394 (2006).
24. Nong, Q. Q., Cheng, T. C. E., and Ng, C. T., "Two-agent scheduling to minimize the total cost," *European Journal of Operational Research*, Vol. 215, pp. 39-44 (2011).
25. Shyr, O. F. F. and Lee, Y.-L., "Modeling pricing and scheduling strategies for air cargo carriers as non-cooperative games," *Journal of Marine Science and Technology*, Vol. 20, No. 2, pp. 216-222 (2012).
26. Wan, G., Vakati, S. R., Leung, J. Y. T., and Pinedo, M., "Scheduling two agents with controllable processing times," *European Journal of Opera-*

tional Research, Vol. 205, pp. 528-539 (2010).
27. Wu, C. C., Huang, S. K., and Lee, W. C., "Two-agent scheduling with learning consideration," *Computers & Industrial Engineering*, Vol. 61, No. 4, pp. 1324-1335 (2011).
28. Yin, Y., Cheng, S. R., Cheng, T. C. E., Wu, W. H., and Wu, C. C., "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).
29. Yin, Y., Wu, W. H., Cheng, S. R., and Wu, C. C., "An investigation on a two-agent single-machine scheduling problem with unequal release dates," *Computers & Operations Research*, Vol. 39, pp. 3062-3073 (2012).
30. Yuan, J. J., Shang, W. P., and Feng, Q., "A note on the scheduling with two families of jobs," *Journal of Scheduling*, Vol. 8, pp. 537-542 (2005).