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 \( j \in AG_0 \) or \( I_j = 1 \) if \( j \in AG_1 \). All the jobs are available at time zero. Under a schedule \( S \),
let \( C_j(S) \) be the completion time of job \( j \), \( T_j(S) = \max\{0, C_j(S) - d_j\} \) 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 \( \sigma \) and \( \sigma' \) each denote a partial sequence. In addition, let \( t \) be the completion time of the last job in \( \sigma \).

**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

\[
T_j(S_1) = t + p_j - d_j ,
\]

\[
T_j(S_2) = t + p_i + p_j - d_j ,
\]

\[
T_j(S_2) = t + p_j - d_j ,
\]

and

\[
T_j(S_2) = t + p_i + p_j - d_i ,
\]

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

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

and

\[
C_j(S_1) = C_j(S_2) .
\]

Therefore, \( S_1 \) dominates \( S_2 \). The proof is completed.

**Property 2.** If job \( i \in AG_0 \), job \( j \in AG_1 \), \( t + p_i < d_i < t + p_i + p_j \), 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 implies that \( T_i(S_1) = 0 \) and \( T_j(S_2) = t + p_i + p_j - d_i \). Meanwhile, because \( t + p_i + p_j < d_i \), we have \( T_j(S_1) = 0 \). Therefore, we have \( [T_j(S_1) + T_j(S_2)] > [T_i(S_1) + T_j(S_1)] \).

**Property 3.** If job \( i \in AG_0 \), job \( j \in AG_1 \), \( t + p_i < d_i \) and \( t + p_i + p_j < d_i \), 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 implies 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 \( t + p_i + p_j < d_i \), we have \( T_i(S_1) = 0 \) and \( T_i(S_2) = 0 \). Therefore, we have \( [T_j(S_1) + T_j(S_2)] > [T_i(S_1) + T_j(S_1)] \).

Next, we give a proposition to determine the feasibility of the partial schedule. Let \( (\pi, \pi') \) be a sequence of jobs where \( \pi \) is the scheduled part with \( k \) jobs and \( \pi' \) is the unscheduled part with \( (n-k) \) jobs. Among the unscheduled jobs, let \( p_{i(1)} = \min\{p_j\} \) and \( d_{j(1)} = \min\{d_j\} \). Moreover, let \( C_{[k]} \) be the completion times of the last job in \( \pi \). Also, let \( \pi' \) and \( \pi'' \) 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 \( C_{[k]} \geq \max_{j \in \pi''} \{d_j\} \), then schedule \( (\pi, \pi') \) is dominated by schedule \( (\pi, \pi'') \).

**Proof:** Since \( C_{[k]} \geq \max_{j \in \pi''} \{d_j\} \), all the unscheduled jobs are from \( AG_0 \) and tardy. So the SPT rule yields an optimal sub-schedule.

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

**Proof:** Similar to Property 4.

**Property 6.** If \( C_{[k]} + p_{i(1)} > d_{j(1)} \), then \( (\pi, \pi') \) is not a feasible sequence.

**Proof:** Since \( C_{[k]} + p_{i(1)} > d_{j(1)} \), one job of \( AG_1 \) among the unscheduled jobs must be tardy. So \( (\pi, \pi'') \) 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 \( \pi \) is a partial schedule
in which the order of the first \( k \) jobs is determined and let \( \pi' \) 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 \( \pi \). The completion time for the \((k+j)\)th job is

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

Then a lower bound can be obtained as follows

\[
\sum_{j=1}^{n} T_j(S)(1-I_j) \geq \sum_{j=1}^{n} L_j(S)(1-I_j)
= \sum_{j=1}^{n} (C_j(S) - d_j(S))(1-I_j)
= \sum_{j=1}^{n} C_{(j)}(S) - \sum_{j=1}^{n} d_{(j)} = LB
\]

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 recombinates them using a crossover operator to generate offsprings that are submitted to a mutation operator 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 GA1, initial sequences are improved by pairwise interchange. While in GA2, initial sequences are improved by forward-shifted reinsertion. In GA3, 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:

\[
f(S_i(v)) = \max_{1 \leq j \leq n} \left( \sum_{j=1}^{n} T_j(S_i(v)) \right) - \sum_{j=1}^{n} T_j(S_i(v))
\]

where \( S_i(v) \) is the \( i \)th 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:

\[
P(S_i(v)) = f(S_i(v)) / \sum_{j=1}^{N} f(S_j(v))
\]

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 GAs 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.
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 from 1 to 100. The instances with number of nodes exceeded 10^9. The instances with a bigger value of number of nodes in the instances is getting larger as the number of jobs increases. The instances with a bigger value of 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 GA1, GA2, and GA3 were less than 2% or below, except one case at (τ, R) = (0.25, 0.75) in GA1 has a bigger mean error percentage. However, the situation was disappeared when we further combined three

<p>| Table 1. Performance of the branch-and-bound and GA algorithms (n = 10, 12, 14). |
|---|---|---|---|---|---|---|---|---|---|---|---|</p>
<table>
<thead>
<tr>
<th>n</th>
<th>τ</th>
<th>R</th>
<th>valid sample size</th>
<th>CPU time</th>
<th>number of nodes</th>
<th>GA1</th>
<th>GA2</th>
<th>GA3</th>
<th>GA</th>
<th>GA∗</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.25</td>
<td>0.25</td>
<td>50</td>
<td>4.6</td>
<td>0.9</td>
<td>785914</td>
<td>171846</td>
<td>0.02</td>
<td>0.12</td>
<td>0.00</td>
</tr>
<tr>
<td>0.50</td>
<td>50</td>
<td>2.9</td>
<td>1.4</td>
<td>483862</td>
<td>253246</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.09</td>
<td>0.66</td>
</tr>
<tr>
<td>0.75</td>
<td>50</td>
<td>0.4</td>
<td>0.7</td>
<td>69607</td>
<td>114075</td>
<td>0.43</td>
<td>1.74</td>
<td>0.17</td>
<td>1.19</td>
<td>0.31</td>
</tr>
<tr>
<td>0.50</td>
<td>0.25</td>
<td>50</td>
<td>0.8</td>
<td>0.2</td>
<td>136659</td>
<td>34564</td>
<td>0.03</td>
<td>0.24</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.50</td>
<td>50</td>
<td>0.7</td>
<td>0.3</td>
<td>120369</td>
<td>58854</td>
<td>0.68</td>
<td>4.81</td>
<td>0.78</td>
<td>4.84</td>
<td>0.09</td>
</tr>
<tr>
<td>0.75</td>
<td>50</td>
<td>0.5</td>
<td>0.5</td>
<td>89639</td>
<td>80064</td>
<td>0.29</td>
<td>2.02</td>
<td>0.03</td>
<td>0.23</td>
<td>0.04</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>1.7</td>
<td>0.7</td>
<td>281008</td>
<td>118775</td>
<td>0.26</td>
<td>1.60</td>
<td>0.16</td>
<td>1.04</td>
</tr>
<tr>
<td>12</td>
<td>0.25</td>
<td>0.25</td>
<td>50</td>
<td>519.8</td>
<td>911.7</td>
<td>64757081</td>
<td>11917319</td>
<td>0.00</td>
<td>0.00</td>
<td>1.33</td>
</tr>
<tr>
<td>0.50</td>
<td>50</td>
<td>270.2</td>
<td>136.4</td>
<td>33403302</td>
<td>17547217</td>
<td>0.01</td>
<td>0.07</td>
<td>1.00</td>
<td>6.99</td>
<td>0.03</td>
</tr>
<tr>
<td>0.75</td>
<td>50</td>
<td>44.2</td>
<td>68.9</td>
<td>5424919</td>
<td>8514648</td>
<td>5.99</td>
<td>35.73</td>
<td>0.09</td>
<td>0.65</td>
<td>0.23</td>
</tr>
<tr>
<td>0.50</td>
<td>0.25</td>
<td>50</td>
<td>60.6</td>
<td>19.6</td>
<td>7258565</td>
<td>2358229</td>
<td>0.29</td>
<td>1.86</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>0.50</td>
<td>50</td>
<td>49.1</td>
<td>24.6</td>
<td>5925444</td>
<td>3015943</td>
<td>0.24</td>
<td>1.26</td>
<td>0.15</td>
<td>0.52</td>
<td>0.56</td>
</tr>
<tr>
<td>0.75</td>
<td>50</td>
<td>29.5</td>
<td>24.6</td>
<td>3556758</td>
<td>3000702</td>
<td>0.35</td>
<td>1.40</td>
<td>0.35</td>
<td>1.90</td>
<td>0.30</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>162.3</td>
<td>60.9</td>
<td>20054345</td>
<td>7725676</td>
<td>1.15</td>
<td>6.72</td>
<td>0.49</td>
<td>2.97</td>
</tr>
<tr>
<td>14</td>
<td>0.25</td>
<td>0.25</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.50</td>
<td>5</td>
<td>5541.0</td>
<td>5297.7</td>
<td>447631420</td>
<td>427969908</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.75</td>
<td>44</td>
<td>1034.3</td>
<td>2533.8</td>
<td>85038885</td>
<td>209388961</td>
<td>489.26</td>
<td>3241.15</td>
<td>0.62</td>
<td>4.11</td>
<td>0.00</td>
</tr>
<tr>
<td>0.50</td>
<td>0.25</td>
<td>50</td>
<td>5052.6</td>
<td>2019.8</td>
<td>440447020</td>
<td>179742131</td>
<td>0.05</td>
<td>0.25</td>
<td>0.10</td>
<td>0.45</td>
</tr>
<tr>
<td>0.50</td>
<td>50</td>
<td>4292.1</td>
<td>2391.4</td>
<td>380001168</td>
<td>216586322</td>
<td>1.07</td>
<td>3.82</td>
<td>0.91</td>
<td>2.46</td>
<td>1.29</td>
</tr>
<tr>
<td>0.75</td>
<td>49</td>
<td>2037.4</td>
<td>1639.4</td>
<td>178910871</td>
<td>146037371</td>
<td>1.11</td>
<td>7.62</td>
<td>0.60</td>
<td>1.64</td>
<td>1.71</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>3591.5</td>
<td>2776.4</td>
<td>306405873</td>
<td>235944211</td>
<td>98.30</td>
<td>650.57</td>
<td>0.44</td>
<td>1.73</td>
</tr>
</tbody>
</table>
The overall mean RDP of GA1 was less than 2%. However, the performance of GA3 was significantly better with a mean RDP of 0.01%. The results indicated that GA3 was the most effective heuristic algorithm for the scheduling problem. The proposed GAi algorithms were also compared with the branch-and-bound algorithm (BnB) for different problem sizes. The BnB algorithm was found to be the most accurate but also the most computationally expensive. The GAi algorithms were found to be a viable alternative to the BnB algorithm with a significant reduction in computation time. Overall, the GAi algorithms demonstrated the potential for solving complex scheduling problems with a high degree of accuracy.
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