Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times

Pei-Chann Chang a,*, Shih-Hsin Chen b

a Department of Information Management, Yuan-Ze University, 135 Yuan Tung Road, Chung-Li 32026, Taiwan, ROC
b Department of Electronic Commerce Management, Nanhua University, 32, Chungkeng, Dalin Chiayi 62248, Taiwan, ROC

Article history:
Received 8 September 2007
Received in revised form 18 February 2010
Accepted 14 March 2010
Available online 19 March 2010

Keywords:
Scheduling
Setup times
Dominance properties
Genetic algorithm
Unrelated parallel machine

Abstract

This paper deals with an unrelated parallel machine scheduling problem with the objective of minimizing the makespan. There are machine-dependent and job sequence-dependent setup times and all jobs are available at time zero. This is a NP-hard problem and a set of dominance properties are developed including machine (adjacent and non-adjacent interchange) and intra-machine switching properties as necessary conditions of job sequencing orders in an optimal schedule. As a result, by applying these dominance properties for a given sequence, a near-optimal solution can be derived. In addition, a new meta-heuristic is introduced by integrating the dominance properties with genetic algorithm to further improve the solution quality for larger problems. The performance of this meta-heuristic is evaluated by using benchmark problems from the literature. The intensive experimental results show that GADP can find all optimal solutions for the small problems and outperformed the solutions obtained by the existing heuristics for larger problems.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

In most scheduling problems, the setup times are either neglected or assumed to be part of the processing times. This assumption is acceptable when the ratio of the setup time to the processing time is small. However, in the make-to-order production environment the role of the setup time cannot be neglected because of the frequent changeovers and large amount of setups. Negligence of setup time leads to unrealistic results, that is, the resulting schedules are not informative any more. Applications for parallel machine scheduling with setup are common in many industries including painting, plastic, textile, glass, semiconductor, chemical, and paper manufacturing (e.g., Guinet and Dussauchy [20]; Chang et al. [5, 11]—Franca et al. [15]; Radhakrishnan and Ventura [36]; Kurz and Askin [25]; Pinedo [34]; Randhawa and Kuo [29]; Zhang and Wu [40]).

The unrelated parallel machines scheduling problem (PMSP) is the scheduling of n jobs available at time zero on m unrelated machines (Rm) to minimize the makespan, C max. If the jobs’ processing times are dependent on the machine assigned and there is no relationship among these machines, then the machines are considered unrelated. In addition, machine-dependent and sequence-dependent setup times, i.e., Sijk, for job i sequenced before j on machine k are considered. The setup times are sequence-dependent as their amounts depend on job sequence. They are also machine-dependent because each machine has its own matrix of setup times and each machine is not exactly identical to each other.

The problem will be referred to as Rm||Sijk/C max. The basic identical PMSP Rm||C max is NP-hard even when m = 2 according to Garey and Johnson [16]. Since Rm||Sijk/C max is a generalization of the former problem, and then it is also NP-hard.

Recently, meta-heuristics are proposed to solve different kind of machine scheduling problems as in Chang et al. [6–10], Chou et al. [12], Hsieh et al. [21] and Connolly [13]. However, it is observed that the convergence of genetic algorithm is slow. One way of improving the convergence in genetic algorithms is to include the knowledge from problem domain. In this research, dominance properties are included in Meta-heuristics to further improve the convergence. Dominance properties of the optimal schedule are developed based on the switching of two adjacent jobs i and j. These dominance properties are in the mathematic form which provides necessary conditions for any given schedule to be optimal. For a given schedule, by applying these DP in advance, we can derive the best sequence between any two jobs i and j. In addition, the operation of DP is very efficient.

The dominance properties obtain efficient solutions before the meta-heuristics are carried out. Once the DP generates good initial solutions efficiently, the meta-heuristics are able to converge faster. It is true that DP needs additional computational efforts, but helps
GA to converge faster so that meta-heuristics requires less number of generations.

2. Literature review

State-of-art reviews on parallel machines’ research can be found in Graves [19], and more recently in Mokotoff [32]. In addition, Allahverdi et al. [1] presented a survey on scheduling problems involving setup time’s constraints. While the focus of our review will be limited to unrelated PMSP, it is important to note that many papers addressed identical parallel machine scheduling with and without setup consideration referring to recent researches by Dunstall and Wirth [14], Kurz and Askim [25], and Lin and Li [28].

Allahverdi et al. [1] provided a good review about scheduling with setup time. Picard and Queyranne [33], Asano and Ohta [3], Brucker et al. [4], have developed branch and bound algorithms for this type of problems. Liu and Chang [29] proposed a Lagrangian relaxation-based approach. Martello et al. [39] proposed a mixed integer programming model and heuristic algorithms. Kim and Bobrowski [24] combined neural networks with some dispatching rules. Tan et al. [39] applied four different methods in solving the total tardiness minimization problem of the single machine with sequence-dependent setup times. Missbauer [31] investigated the topic about order release and sequence-dependent setup times. Due to the complexity of the problem, finding optimal solutions for large problems is very time consuming and sometimes computationally infeasible. Developing heuristic algorithms to derive near-optimal solutions becomes much more practical and useful.

Some authors have considered the use of meta-heuristic approaches for unrelated PMSP. Kim et al. [22,23] developed heuristics for the problem including a Simulated Annealing (SA) to minimize the total tardiness with machine-independent sequence-dependent setup times. Glass et al. [18] compared genetic algorithms (GA), SA, and Tabu Search (TS) for $R_m|C_{\text{max}}$ without setup times. The authors concluded that the quality of solutions generated by GA was poor. However, a hybrid method that incorporated GA was comparable to performances by SA and TS. Srivastava [38] presented effective TS for the same problem without setup times and reported that TS can provide good quality solutions for practical size problems within a reasonable amount of time.

Ghirardi and Potts [17] also addressed $R_m|C_{\text{max}}$ where the heuristic they used was an application of the Recovering Beam Search. The authors reported that their algorithm produced good results on large instances (up to 50 machines and 1000 jobs). Some researchers developed exact algorithms for unrelated PMSP including Liaw et al. [27] and Lancia [26] who developed branch and bound algorithms to find optimal solutions for the problem without setup times. The objective functions are the total weighted tardiness and $C_{\text{max}}$, respectively. Martello et al. [30] developed lower bounds for $R_m|C_{\text{max}}$ based on Lagrangian relaxation, which showed to be better than previous bounds. They utilized their results to develop effective approximate algorithms.

The motivation for this paper comes from the scheduling problem in a real-world factory. There are many types of products and the production of various types of products on the shop floor requires different setup times in the changeovers. In the past, researchers have spent much effort in solving the setup time scheduling problems. This research will develop dominance properties including inter-machine (i.e., adjacent and non-adjacent interchange) and intra-machine switching properties as necessary conditions of job sequencing orders in an optimal schedule. Therefore, by applying these dominance properties for a given sequence, a near-optimal solution can be derived. In addition, a new meta-heuristic is introduced by integrating the dominance properties with genetic algorithm to further improve the solution quality for larger problems.

3. Problem definition

A mixed integer program (MIP) is formulated to find optimal solutions for the unrelated parallel machine scheduling problems with sequence-dependent times. Similar formulation was used by Guinet and Dussauchy [20].

Minimize $C_{\text{max}}$

subject to

$$\sum_{i=0}^{n} \sum_{k=1}^{m} x_{i,j,k} = 1 \quad \forall j = 1, ..., n$$ (2)

$$\sum_{j=0}^{n} \sum_{i=1}^{m} x_{i,j,k} = 0 \quad \forall h = 1, ..., n$$ (3)

$$C_j \geq C_i + \sum_{k=1}^{m} x_{i,j,k} (S_{i,j,k} + p_{j,k}) + M \sum_{k=1}^{m} x_{i,j,k} - 1$$ (4)

$$\forall i = 0, ..., n \quad \forall j = 1, ..., n$$

$$\sum_{j=0}^{n} x_{0,j,k} = 1 \quad \forall k = 1, ..., m$$ (5)

$$x_{i,j,k} \in \{0, 1\} \quad \forall i = 0, ..., n \quad \forall j = 0, ..., n \quad \forall k = 1, ..., m$$ (6)

$$C_0 = 0$$ (7)

$$C_j \geq 0 \quad \forall j = 1, ..., n$$ (8)

where, $C_j$: Completion time of job $j$, $p_{j,k}$: processing time of job $j$ on machine $k$, $S_{i,j,k}$: sequence-dependent setup time to process job $j$ after job $i$ on machine $k$, $S_{0,k}$: setup time to process job $j$ first on machine $k$, $x_{i,j,k}$: 1 if job $j$ is processed directly after job $i$ on machine $k$ and 0 otherwise, $x_{0,k}$: 1 if job $j$ is the first job to be processed on machine $k$ and 0 otherwise, $S_{0,k}$: 1 if job $j$ is the last job to be processed on machine $k$ and 0 otherwise, $M$: a large positive number.

The objective (1) is to minimize the makespan. Constraints (2) ensure that each job is scheduled only once and processed by one machine. Constraints (3) make sure that there is no need for another set of constraints to guarantee that there is no need for another set of constraints to guarantee that the completion time for the dummy job is zero and constraints (8) ensure that completion times are non-negative. Optimal solutions for the problem then can be obtained by solving the MIP software solver.

4. Derivations of dominance properties

We consider the problem of scheduling $n$ jobs into unrelated parallel machines and to derive the dominance properties (necessary conditions) of the optimal schedule. In this section, we use the objective function $(Z(\Pi))$ for the makespan of schedule $\Pi$. 
In order to derive the dominance properties for schedule $\Pi_x$, we consider interchanging two jobs on the same machine or on different machines to prove some intermediate results. Fig. 1 shows a schematic diagram of a parallel machine schedule. $[j]$: The job is at position $[j]$, $P[i][j]$; the processing time of the job at position $[j]$ on machine $[k]$, $S[i][j][k]$; the setup time of the job at position $[j]$ is after the job $[i]$ on machine $[k]$, $AP[i][j][k]$; the adjusted processing time of the job at position $[j]$ is after the job $[i]$ on machine $[k]$. Thus, $AP[i][j][k]$ is actually equal to $P[i][j] + S[i][j][k]$. $G_k$: the completion time of job $i$ on machine $k$, $G'[i]$: the job set before job $i$ on machine $k$, $G''[i]$: the job set between job $i$ and job $[i + 1]$ on machine $k$, $G'[i]$: the job set after job $i$ on machine $k$.

There are two conditions in exchanging jobs and they include the in-machine (two jobs are on the same machine) and inter-machine (The exchanging jobs come from different machines). The dominance properties of inter-machine and intra-machine are presented at Sections 4.1 and 4.2, respectively.

### 4.1. Inter-machine exchange

There are two cases to be considered within the inter-machine exchange, and they are the adjacent exchange (see Fig. 2) and non-adjacent exchange (see Fig. 3). The following lemma is for the adjacent exchange.

**Lemma 1a.** When the following condition exists, the exchanged schedule is better than the original one:

\[
\begin{align*}
(\text{AP}[i-1][j][k] - \text{AP}[i-1][j][k]) & + (\text{AP}[j][j][k] - \text{AP}[j][j][k]) \\
+ (\text{AP}[j][j][k] - \text{AP}[j][j+1][k]) < 0
\end{align*}
\]

**Proof.** Suppose we exchange job $i$ and job $j$ in schedule $\Pi_x$, which are adjacent to each other on the same machine. After exchanging these two neighborhood jobs, schedule $\Pi_x$ is changed to $\Pi_y$. From Fig. 2, the jobs in the set $G_x[i]$ and $G'[i][k]$ remain the same, whose objective does not change. Thus, the completion time of $G_x[i]$ and $G'[i][k]$ is shown as follows:

\[
G_x[i] = \sum_{a=1}^{i-1} \text{AP}[a-1][a][k] = C'_1[i][k] \quad \square
\]

Except for the $G_x[i]$ and $G'[i][k]$ are not changed, the completion time of the job sets $G'_2[k]$, $G'_3[k]$, $G'_2[k]$, and $G'_3[k]$ are listed in the following.

\[
\begin{align*}
G'_2[k] &= G_x[i] + \text{AP}[i-1][j][k] + \text{AP}[j][j][k] + \text{AP}[j][j+1][k] \\
G'_3[k] &= G_x[i] + \sum_{a=j+2}^{n} \text{AP}[a-1][a][k]
\end{align*}
\]

When it comes to calculate the difference before and after we exchange the job $i$ and job $j$, we subtract $\Pi_x$ with $\Pi_y$, whose difference $\Delta$ is actually equal to $G_2[k] - G'_3[k]$.

\[
\Delta = \prod_{x} - \prod_{y} = G_2[k] - G'_3[k] = (\text{AP}[i-1][j][k] + \text{AP}[j][j][k] + \text{AP}[j][j+1][k]) + (\text{AP}[i-1][j][k] + \text{AP}[j][j][k] + \text{AP}[j][j+1][k]) < 0
\]

As a result, if the difference $\Delta$ is less than zero, the job $i$ and job $j$ should be exchanged.

In term of non-adjacent interchange, the procedure is similar. The dominance property of the non-adjacent interchange and the derivation are shown below.

**Lemma 1b.** When the following condition exists, the job $i$ and job $j$ are exchanged.

\[
(\text{AP}[i-1][j][k] - \text{AP}[i-1][j][k]) + (\text{AP}[j][j][k] - \text{AP}[j][j][k]) + (\text{AP}[j][j][k] - \text{AP}[j][j+1][k]) < 0
\]

**Proof.** The objective of $G_x[i]$ and $G'_1[j][k]$ is the same, so it will be eliminated after we subtract $\Pi_x$ with $\Pi_y$. The equation of $G_x[i]$ and $G'_1[j][k]$ is shown as follows:

\[
G_x[i] = \sum_{a=1}^{i-1} \text{AP}[a-1][a][k] = C'_1[i][k] \quad \square
\]

Similarly with the **Lemma 1a**, the completion time of the job sets $G_2[k]$, $G'_3[k]$, $G'_2[k]$, and $G'_3[k]$ are changed. Thus, they are demonstrated as follows.

\[
\begin{align*}
G'_2[k] &= G_x[i] + \text{AP}[i-1][j][k] + \text{AP}[j][j][k] + \text{AP}[j][j+1][k] + \text{AP}[j][j][k] + \text{AP}[j][j+1][k] \\
G'_3[k] &= G_x[i] + \sum_{a=j+2}^{n} \text{AP}[a-1][a][k]
\end{align*}
\]
4.2.1. The objective difference of machine k1

\[ \Delta_{k1} = \prod_x - \prod_y = (\prod_x A_{p[i-1][j][k1]} - A_{p[i-1][j][k1]}) + (A_{p[i][j+1][k1]} - A_{p[i][j+1][k1]}) \]

Consequently, the job i and job j is exchanged only if the difference \( \Delta \) is less than zero.

4.2. Intra-machine exchanging

This section discusses the interchange of job i and job j on any two different machines (see Fig. 4). Because the number of parallel machine is equal to or more than 2, the notation of k1 and k2 indicates the machine number for job i and job j. Job i and job j exchanged from \( \Pi_x \) and \( \Pi_y \) (i.e., a new schedule) is obtained thereafter. Lemma 2 presents the dominance property which satisfies this exchanging condition.

**Lemma 2.** When the following condition exists, the job i and job j are exchanged.

\[ \text{Max} \{ (G_{k1}[i] + (A_{p[i-1][j][k1]} - A_{p[i][j+1][k1]})) + (A_{p[i][j+1][k1]} - A_{p[i-1][j][k1]}) \} < C_{max} \]

**Proof.** We compare the makespan \( C_{max} \), which is \( \text{Max}(C_{k1}[i], C_{k2}[i]) \), of original schedule with \( C_{max} \) of the exchanged schedule. If \( C_{max} \) is less than \( C_{max} \) job i and job j are exchanged. In order to verify the relationship of the objective belonged to the two machines. The objective difference of machine k1 is calculated before machine k2. □

4.2.1. The objective difference of machine k1

The detailed completion time of each job sets on machine k1 is shown as follows:

\[
\begin{align*}
G_1[k1] &= \sum_{a=1}^{i-1} A_{p[a-1][a][k1]} = G_1[i] \\
G_2[k1] &= G_1[k1] + A_{p[i][j+1][k1]} + A_{p[i][j+1][k1]} \\
G_3[k1] &= G_2[k1] + \sum_{a=j+2}^{n} A_{p[a-1][a][k1]} \\
G_4[k1] &= G_3[k1] + A_{p[i][j+1][k1]} + A_{p[i][j-1][k1]} \\
G_5[k1] &= G_4[k1] + \sum_{a=j+2}^{n} A_{p[a-1][a][k1]} 
\end{align*}
\]

4.2.2. The objective difference of machine k2

The detailed completion time of each job sets on machine k2 is shown as follows:

\[
\begin{align*}
G_1[k2] &= \sum_{a=1}^{j-1} A_{p[a-1][a][k2]} = G_1[k2] \\
G_2[k2] &= G_1[k2] + A_{p[j-1][j][k2]} + A_{p[j][j+1][k2]} \\
G_3[k2] &= G_2[k2] + \sum_{a=j+2}^{n} A_{p[a-1][a][k2]} \\
G_4[k2] &= G_3[k2] + A_{p[j][j+1][k2]} + A_{p[j-1][j][k2]} \\
G_5[k2] &= G_4[k2] + \sum_{a=j+2}^{n} A_{p[a-1][a][k2]} 
\end{align*}
\]

After the above information is obtained, we are able to calculate the objective difference of machine k1 by subtracting \( \Pi_x \) to \( \Pi_y \).

\[
\Delta_{k1} = \prod_x - \prod_y = (\prod_x A_{p[i-1][j][k1]} - A_{p[i-1][j][k1]}) + (A_{p[i][j+1][k1]} - A_{p[i][j+1][k1]})
\]

4.3. A case study for dominance properties application

Take a 6 job 2 machine scheduling problem for example. Tables 1 and 2 are the adjusted processing times for machine 1 and machine 2, respectively. \( A_{p25} \) in Table 1 represents the adjusted processing times for job 5 sequenced after job2 in machine 1 with a value of 170. \( A_{p25} \) in Table 2 represents the adjusted processing times for job 5 sequenced after job2 in machine 2 with a value of 165. Different jobs sequenced differently may cause different processing times after taking the setup times into consideration.

An initial sequence of the case example:

Step one: Min{116, 142, 130, 109, 157, 152, 163, 135, 149, 136, 127, 131,} = 109
The first job to be assigned in machine 1 is job 4.

Step two: Min{163, 135, 149, 136, 127, 131} = 127
The first job to be assigned in machine 2 is job 5.

Step three: Again, second job on machine 1 will be job 1.
Second job on machine 2 will be job 6.

Step four: The third job on machine one is job 3 and the third job on machine 2 is job 2.
The adjusted processing times in machine 1.

<table>
<thead>
<tr>
<th>$A_P^1$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>116</td>
<td>142</td>
<td>130</td>
<td>109</td>
<td>157</td>
<td>152</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>167</td>
<td>166</td>
<td>122</td>
<td>179</td>
<td>141</td>
</tr>
<tr>
<td>2</td>
<td>127</td>
<td>-</td>
<td>120</td>
<td>145</td>
<td>170</td>
<td>180</td>
</tr>
<tr>
<td>3</td>
<td>143</td>
<td>165</td>
<td>-</td>
<td>150</td>
<td>143</td>
<td>145</td>
</tr>
<tr>
<td>4</td>
<td>124</td>
<td>165</td>
<td>157</td>
<td>-</td>
<td>173</td>
<td>137</td>
</tr>
<tr>
<td>5</td>
<td>118</td>
<td>162</td>
<td>158</td>
<td>108</td>
<td>-</td>
<td>137</td>
</tr>
<tr>
<td>6</td>
<td>132</td>
<td>170</td>
<td>136</td>
<td>151</td>
<td>181</td>
<td>-</td>
</tr>
</tbody>
</table>

The adjusted processing times in machine 2.

<table>
<thead>
<tr>
<th>$A_P^2$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>163</td>
<td>135</td>
<td>149</td>
<td>136</td>
<td>127</td>
<td>131</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>135</td>
<td>171</td>
<td>126</td>
<td>139</td>
<td>156</td>
</tr>
<tr>
<td>2</td>
<td>144</td>
<td>-</td>
<td>128</td>
<td>158</td>
<td>165</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>154</td>
<td>-</td>
<td>145</td>
<td>127</td>
<td>136</td>
</tr>
<tr>
<td>4</td>
<td>147</td>
<td>152</td>
<td>159</td>
<td>-</td>
<td>170</td>
<td>140</td>
</tr>
<tr>
<td>5</td>
<td>144</td>
<td>142</td>
<td>168</td>
<td>171</td>
<td>-</td>
<td>127</td>
</tr>
<tr>
<td>6</td>
<td>166</td>
<td>157</td>
<td>172</td>
<td>172</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

At this moment, job sequence on machine 1 is [4, 1, 3] and the finish time for machine 1 is $C_{m1} = 109 + 124 + 166 = 399$. Job sequence on machine 2 is [5, 6, 2] and the finish time for machine 2 is $C_{m2} = 127 + 127 + 157 = 411$. Then, the total completion time will be $C_{max} = 411$ as shown in Fig. 5:

Inter-machine exchange: The exchanging on the same machine:

(1) The job i and job j are adjacent jobs

(i) Machine 1:

Step 1:

$\Pi_X: [4, 1, 3]$ and $\Pi_Y: [4, 1, 3]$

$\Delta = (59 - 52) + (122 - 124) + (157 - 166) = -4 < 0$

Because $\Pi_Y$ is better than $\Pi_X$, the job 1 is changed with job 4. The new sequence is $[1, 4, 3]$.

The completion time on machine one is $C_{m1} = 116 + 122 + 157 = 395$.

$C_{max} = 411$

Step 2:

$\Pi_X: [1, 4, 3]$ and $\Pi_Y: [1, 3, 4]$

$\Delta = (166 - 122) + (150 - 157) = 37 > 0$

Because $\Pi_Y$ is not better than $\Pi_X$, the job 4 and job 3 are not exchanged.

(ii) Machine 2:

Step 1:

$\Pi_X: [5, 6, 2]$ and $\Pi_Y: [6, 5, 2]$

$\Delta = (131 - 127) + (173 - 127) + (142 - 157) = 35 > 0$

Because $\Pi_Y$ is not better than $\Pi_X$, the job 5 and job 6 are not exchanged.

(2) The job i and job j are not adjacent jobs

Machine 1:

$\Pi_X: [1, 4, 3]$ and $\Pi_Y: [3, 4, 1]$

$\Delta = (130 - 116) + (150 - 122) + (124 - 157) = 9 > 0$

Because $\Pi_Y$ is not better than $\Pi_X$, the schedule is not exchanged.

Machine 2:

$\Pi_X: [5, 6, 2]$ and $\Pi_Y: [6, 2, 5]$

$\Delta = (131 - 127) + (157 - 142) + (165 - 110) = 74 > 0$

Because $\Pi_Y$ is not better than $\Pi_X$, the schedule is not exchanged.

Intra-machine exchange: The job i and job j are not adjacent jobs

Machine 1: [1, 4, 3], $C_{m1} = 395$

Machine 2: [5, 2, 6], $C_{m2} = 379$

$C_{max} = 395$

Step 1:

$\Pi_X: Machine 1: [1, 4, 3]$ and Machine 2: [5, 2, 6]

$\Pi_Y: Machine 1: [5, 4, 3]$ and Machine 2: [1, 2, 6]

$\Delta_m1 = (157 - 116) + (108 - 122) = 27$

$\Delta_m2 = (163 - 127) + (135 - 142) = 19$

$C_{max} = \max(395 + 27, 379 + 19) = 422$

$422 > 395$

Because $\Pi_Y$ is not better than $\Pi_X$, we do not exchange the schedule.

Step 2:

$\Pi_X: Machine 1: [1, 4, 3]$ and Machine 2: [5, 2, 6]

$\Pi_Y: Machine 1: [2, 4, 3]$ and Machine 2: [5, 1, 6]

$\Delta_m1 = (142 - 116) + (145 - 122) = 49$

$\Delta_m2 = (144 - 142) + (156 - 110) = 48$

$C_{max} = \max(395 + 49, 379 + 48) = 444$

$444 > 395$

Because $\Pi_Y$ is not better than $\Pi_X$, we do not exchange the schedule.

Step 3:

$\Pi_X: Machine 1: [1, 4, 3]$ and Machine 2: [5, 2, 6]
5. Methodology

These dominance properties can function as a standalone heuristic or to be integrated with meta-heuristic. The procedure of DP is described in Section 4.1. Moreover, this paper also proposes a heuristic or to be integrated with meta-heuristic. The procedure of algorithms with dominance properties (GADP) and simulated annealing. They are genetic algorithms with dominance properties (SADP). Both of them are explained in Sections 5.2 and 5.3, respectively.

5.1. Implementation of dominance properties

The dominance properties consider the exchanging jobs of inter-machine interchange and intra-machine interchange. As a result, this research utilizes the general pair-wise interchange (GPI) to do the exchanges. In the beginning of DP, a random sequence is generated and all jobs are assigned into the machine. The dominance properties test whether the sequence on the same machine satisfies the optimal condition when the algorithm does inter-machine interchange. For the inter-machine exchange, all the combinations of the two jobs come from two machines that are tested. The whole procedures are iterated thirty times or the solution remains the same from previous iteration. The advantage of the standalone DP heuristics has a time-complexity of only \( O(k(n/m)^2) \) to obtain a solution where \( k \) is the number of iterations and \( m \) is the number of machines.

Even though the proposed DP heuristics is able to find an effective solution, it might fail to find out global optimal solution. Consequently, the research seeks to integrate DP with GA and SA, which are very effective at searching for global optimal. The hybrid algorithms are shown in Sections 5.2 and 5.3.

5.2. Genetic algorithm with DP

There are two versions of genetic algorithm with dominance properties, which are named GADP and GADP2. First, GADP applies the initial solutions generated by DP heuristics as the initial population. Based on these initial solutions, GA explores the solution spaces with regular genetic operators, including the selection, crossover, and mutation operator. The research employs the binary tournament selection, two-point crossover, and swap mutation in the GADP.

In GADP2, DP heuristics plays an important role in generating the initial solutions and to enhance the solution quality produced by Genetic algorithm. The hybrid algorithm takes the advantage of DP which provides good initial solutions to avoid the blind search of GA at the beginning while exploring the solution space. Fig. 7 demonstrates the detailed procedures of GADP2.

It is important to note that there is a parameter \( k \) to control the introduction time of the DP algorithm. When the generation \( t \) is divided by \( k \) completely, the current chromosomes will be inputted to DP heuristics which will readjust the sequence of each job and generate a new sequence. The only difference is that the input solutions here for DP heuristics are the solutions which have been evolved for \( k \) generations. After DP heuristics have been applied to further improve the solutions, the hybrid algorithm will continue to evolve the chromosomes. In other words, GADP2 will apply DP many times to fine-tune the intermediate chromosomes generated by the GA procedure.

Finally, the parameters of GA should be optimized because they influence the performance of GA significantly. This study has utilized the design of the experiment to select appropriate parameters. Table 3 shows the levels of the population size, crossover rate, and mutation rate. The final setup of each parameter after DOE is shown in Table 4 presenting the best parameter configuration of GA.

5.3. Simulated annealing with DP

Because SADP is compared with the proposed algorithm GADP and GADP2, this section demonstrates the detail procedures of the

### Table 3

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (PopSize)</td>
<td>100, 200</td>
</tr>
<tr>
<td>Crossover rate (Pc)</td>
<td>0.6, 0.9</td>
</tr>
<tr>
<td>Mutation rate (Pm)</td>
<td>0.1, 0.5</td>
</tr>
</tbody>
</table>

### Table 4

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.6</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.5</td>
</tr>
<tr>
<td>Max number of iterations</td>
<td>500 \times \text{number of jobs}</td>
</tr>
</tbody>
</table>
SADP. The idea of SADP is similar to GADP. In the very beginning, DP generates a population of solutions and then we select a best solution among them. SA utilizes this solution to improve the solution quality. SADP is defined in the following pseudo code.

\begin{verbatim}
initialParameters()
initialStage();
while counter < numberOfSolutionsExamined do
    for i=0 to numberOfMoves do
        tempChromosomes = getMoves();
        calcObjectiveValue();
        tempChromosomes[0] = selectBetterMoves();
        acceptanceRule();
    End for
    currentTemperature *= \alpha;
End while
\end{verbatim}

Line 1 initiates the parameters used by the SA and the required scheduling information could be processed by DP and SA. In the initial stage (Line 2), a set of solutions is generated and then a best solution is selected. After this, SA continues the exploration from this solution from Line 3 to Line 10. Line 3 shows the stopping criterion which sets the number of examined solutions. We fix the number of examined solution as 100,000. Line 5 is an important step to vibrate the current solution. The purpose of move is to do a variation on current solution by local search. The move strategies include swap move, 2-opt, 3-opt, k-opt, shift move, and inverse move. The swap move is to swap two points of original path and the 2-opt is to replace original two arcs, which are not nearby and then connect two new arcs into the path. The work applies the swap move, shift move, move, and inverse move together. They are described below.

No matter for the shift move or inverse move, it needs to randomly generate two cut points. We may call it “cut point 1” and “cut point 2”. For shift move as shown in Fig. 8, we move the cut point 2 ahead the position of the range so that it replaces the original cut point 1. Then, shifting all point forward for one space until at the end of element on cut point 2. (Because it has been moved to the place of cut point 1) Fig. 7 shows how the shift move works which supposes there are 10 jobs.

The inverse move is to inverse the current position. Take Fig. 8 for instance, the original sequence between the two cut points is 9-5-3-4-8-0. After the inverse, the new sequence becomes 0-8-4-3-9-5 (Fig. 9).

Finally, the swap move is very easy to implement because it just has to set two positions and exchange the two values of its position. The result is shown in Fig. 10.
Line 6 is to evaluate the candidates generated by the three move strategies. The best neighborhood solution among them is selected in Line 7. This best move is then tested by the acceptance rule defined in Line 8. If the neighborhood solution is better than current solution, it goes without saying that it replaces the current solution. On the other hand, if the neighborhood solution is worsened, simulated annealing creates a chance to accept this solution depending on the following condition. The characteristic of SA is different from traditional heuristics that may discard the neighborhood solution. The following condition will accept the worse solution when the random probability is less than or equal to the value of energy function.

\[
U(0, 1) < e^{\frac{f(x') - f(x)}{T}}
\]

where: \(U(0,1)\): a random generated value is between 0 and 1; \(f(x)\): the current objective value of \(x\); \(f(x')\): The objective value of the neighborhood solution of \(x'\); \(T\): the current temperature.

Then, the neighborhood solution is also compared with current best solution. If it is better than current optimal solution, SA replaces the current best solution by the neighborhood solution.

After certain iterations, we decrease the current temperature in Line 10. Finally, the termination condition is that when the current Temperature < finalTemperature, the SA stops.

Because there are some parameters of SA to be configured, the study follows the suggestions of Connolly [4]. The parameter configuration of SA is shown in Table 5.

The bound information used here takes the average processing time of each job; it selects the minimum and maximum processing for each job on all the machines, respectively. The lower bound of energy function.
Table 8
The experimental result of the dominant processing time instances.

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>DP</th>
<th>SGA</th>
<th>GADP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>40</td>
<td>3952</td>
<td>3980</td>
<td>4012</td>
<td>15.01</td>
</tr>
<tr>
<td>60</td>
<td>5926</td>
<td>5959</td>
<td>6005</td>
<td>18.59</td>
</tr>
<tr>
<td>80</td>
<td>7879</td>
<td>7930</td>
<td>7988</td>
<td>31.20</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>749</td>
<td>757</td>
<td>764</td>
</tr>
<tr>
<td>40</td>
<td>1336</td>
<td>1356</td>
<td>1378</td>
<td>10.31</td>
</tr>
<tr>
<td>60</td>
<td>2020</td>
<td>2041</td>
<td>2061</td>
<td>9.47</td>
</tr>
<tr>
<td>80</td>
<td>2642</td>
<td>2676</td>
<td>2709</td>
<td>16.73</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>386</td>
<td>392</td>
<td>401</td>
</tr>
<tr>
<td>40</td>
<td>750</td>
<td>767</td>
<td>784</td>
<td>8.64</td>
</tr>
<tr>
<td>60</td>
<td>1034</td>
<td>1069</td>
<td>1093</td>
<td>13.87</td>
</tr>
<tr>
<td>80</td>
<td>1362</td>
<td>1407</td>
<td>1460</td>
<td>24.47</td>
</tr>
</tbody>
</table>

and the upper bound of the algorithm are set by the following equations.

\[
\text{lower bound} = \frac{\sum_{j=1}^{n} \min_{i \in \text{jobs}, k \in \text{machines}}(AP_{ijk})}{m},
\]

\[
\text{upper bound} = \frac{\sum_{j=1}^{n} \max_{i \in \text{jobs}, k \in \text{machines}}(AP_{ijk})}{m},
\]

6. Experimental results

The test instances of this unrelated parallel machine problem are provided by Rabadi et al. [35]. The number of jobs includes 20, 40, 60, and 80. The data distribution for processing time \(p_{ij}\) and setup time \(s_{ij}\) are balanced. Because there are 15 instance replications of each combination, the total number of instances is 540.

The parameter configuration of GA is done by the Design-of-Experiment before. The levels of the three factors are listed in the following Table 5. Each parameter combination is replicated 30 times.

![Fig. 11. Interaction plot of Pc and Pm.](image-url)
Table 9
The experimental result of the dominant setup time instances.

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>DP</th>
<th>SGA</th>
<th>GADP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>3942</td>
<td>3970</td>
<td>4011</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>5908</td>
<td>5942</td>
<td>5976</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>7850</td>
<td>7891</td>
<td>7965</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>748</td>
<td>758</td>
<td>766</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>1337</td>
<td>1356</td>
<td>1375</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>2019</td>
<td>2040</td>
<td>2060</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>2643</td>
<td>2676</td>
<td>2706</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>383</td>
<td>390</td>
<td>398</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>750</td>
<td>765</td>
<td>783</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>1031</td>
<td>1069</td>
<td>1093</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>1361</td>
<td>1404</td>
<td>1454</td>
</tr>
</tbody>
</table>

Table 10
ANOVA test of the experiment.

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>Mean square</th>
<th>F-Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>1</td>
<td>2.105E+10</td>
<td>2.105E+10</td>
<td>4.25E+07</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Instances</td>
<td>155</td>
<td>2.94E+11</td>
<td>1.902E+09</td>
<td>3.840,723</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Type × instances</td>
<td>155</td>
<td>1.526E+10</td>
<td>98,428,645</td>
<td>198,755</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Method</td>
<td>5</td>
<td>757,891,007</td>
<td>151,578,201</td>
<td>306,078</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Type × method</td>
<td>5</td>
<td>470,046.79</td>
<td>94,009,357</td>
<td>189,83</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Instances × method</td>
<td>775</td>
<td>651,513,926</td>
<td>840,663.13</td>
<td>1697.53</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Type × instance × method</td>
<td>775</td>
<td>303,1145.1</td>
<td>391.155</td>
<td>7.9</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Instances × method</td>
<td>775</td>
<td>651,513,926</td>
<td>840,663.13</td>
<td>1697.53</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Type × instance × method</td>
<td>775</td>
<td>303,1145.1</td>
<td>391.155</td>
<td>7.9</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>95,328</td>
<td>47,209,017</td>
<td>495,22718</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected total</td>
<td>97,199</td>
<td>4.203E+11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
When we examine the significance of these factors, the ANOVA table as shown in Table 6 indicates that there is significant between the crossover rate and mutation rate. Consequently, we further use the interaction plot in Fig. 11 to select the setting of Pc and Pm. We found when we set the crossover rate and mutation rate at the level 0.6 and 0.5, respectively; the algorithm will yield a better solution quality. Because there is no significant difference for population size when they are set as 100 and 200, the population size is set as the 100 based on the main effect plot in Fig. 12.

This research compares the performance of DP heuristics, GA, SA, GADP, GADP2, SADP, and Partition Heuristic Algorithm. PH’s Algorithm is proposed by Al-Salem [35] in solving the parallel machine scheduling problem (PMSP) with machine-dependent and sequence-dependent setup times to minimize the makespan. This research derives the dominance properties for the scheduling problems. GADP is able to find optimal solutions for all small problems. For large problems, GADP when compared with PH’s algorithm again. As shown in Table 12, we found GADP2 outperforms PH’s algorithm consistently except the job sets are 60 and 80 on machine 6 and 12.

### Table 11
Duncan pair-wise comparison of these six methods through all instances.

<table>
<thead>
<tr>
<th>Duncan grouping</th>
<th>Mean</th>
<th>N</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2435.43</td>
<td>16,200</td>
<td>SA</td>
</tr>
<tr>
<td>B</td>
<td>2311.14</td>
<td>16,200</td>
<td>SGA</td>
</tr>
<tr>
<td>C</td>
<td>2206.63</td>
<td>16,200</td>
<td>DP</td>
</tr>
<tr>
<td>D</td>
<td>2200.37</td>
<td>16,200</td>
<td>SADP</td>
</tr>
<tr>
<td>E</td>
<td>2195.94</td>
<td>16,200</td>
<td>GADP</td>
</tr>
<tr>
<td>F</td>
<td>2189.06</td>
<td>16,200</td>
<td>GADP2</td>
</tr>
</tbody>
</table>

### Table 12
The comparison between GADP2 and PH’s Algorithm.

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>Balanced processing time</th>
<th>Dominant processing time</th>
<th>Dominant setup time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LB</td>
<td>GADP2</td>
<td>PH’s Algorithm</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>1186</td>
<td>1254</td>
<td>1296</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2345</td>
<td>2495</td>
<td>2521</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>3510</td>
<td>3675</td>
<td>3733</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>4665</td>
<td>4872</td>
<td>4927</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>362</td>
<td>454</td>
<td>480</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>717</td>
<td>831</td>
<td>848</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>1071</td>
<td>1246</td>
<td>1232</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>1429</td>
<td>1648</td>
<td>1626</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>175</td>
<td>239</td>
<td>250</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>347</td>
<td>455</td>
<td>473</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>519</td>
<td>649</td>
<td>627</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>690</td>
<td>849</td>
<td>830</td>
</tr>
</tbody>
</table>

### 7. Conclusions and future direction of research

The problem addressed in this paper is referred to the unrelated parallel machine scheduling problem (PMSP) with machine-dependent and sequence-dependent setup times to minimize the makespan. This research derives the dominance properties for unrelated parallel machine scheduling problem. According to the empirical results, the proposed DP heuristic outperforms GA and SA in effectiveness (the solution quality) and efficiency (less computational time). Moreover, DP heuristics can be integrated with meta-heuristic and the hybrid algorithm is a novel approach in solving the scheduling problems. GADP is able to find optimal solutions for all small problems. For large problems, GADP when compared to GA, SA and GADP still outperforms the other two approaches in nearly all instances.

It will be interesting to extend this work by including local heuristic such as PH algorithm in GADP and investigate whether more improvement can be further accomplished. Or, solutions from these combinations of different heuristics can be injected into the population during the evolution of the GA. Benchmark data sets and solutions for both heuristics used in this paper are made available in our website for other researchers to compare their solution methodologies.

### References
