Applied Soft Computing 11 (2011) 1263–1274

Contents lists available at ScienceDirect

Applied Soft Computing

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

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 INFO

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 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. 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.

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., S_{ijk}, 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 $R_m/S_{ijk}/C_{\max}$. The basic identical PMSP $P_m||C_{\max}$ is NP-hard even when $m = 2$ according to Garey and Johnson [16]. Since $R_m/S_{ijk}/C_{\max}$ 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

* Corresponding author. Tel.: +886 936101320; fax: +886 34638884.
E-mail address: iepchang@saturn.yzu.edu.tw (P.-C. Chang).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.03.003

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 Askin [25], and Lin and Li [28].

Allahverdi et al. [1] has 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 $R_m||C_{\max}$ 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 $R_m||C_{\max}$ 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 $C_{\max}$, respectively. Martello et al. [30] developed lower bounds for $R_m||C_{\max}$ 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].

$$\textit{Minimize} \quad C_{\max} \tag{1}$$

subject to

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{k=1}^{m} x_{i,j,k} = 1 \quad \forall j = 1, \ldots, n \tag{2}$$

$$\sum_{\substack{i=0 \\ j \neq h}}^{n} x_{i,h,k} - \sum_{\substack{i=0 \\ j \neq h}}^{n} x_{h,j,k} = 0 \quad \forall h = 1, \ldots, n \tag{3}$$

$\forall k = 1, \ldots, m$

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

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

$$\sum_{j=0}^{n} x_{0,j,k} = 1 \quad \forall k = 1, \ldots, m \tag{5}$$

$$x_{i,j,k} \in \{0,\ 1\} \ \forall i = 0, \ldots, n, \quad \forall j = 0, \ldots, n, \quad \forall k = 1, \ldots, m \tag{6}$$

$$C_0 = 0 \tag{7}$$

$$C_j \geq 0 \quad \forall j = 1, \ldots, n \tag{8}$$

where, C_j: Completion time of job j, p_{jk}: processing time of job j on machine k, S_{ijk}: sequence-dependent setup time to process job j after job i on machine k, S_{0jk}: setup time to process job j first on machine k, x_{ijk}: 1 if job j is processed directly after job I on machine k and 0 otherwise, x_{0jk}: 1 if job j is the first job to be processed on machine k and 0 otherwise, S_{j0k}: 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 each job must neither be preceded nor succeeded by more than one job. Constraints (4) are used to calculate completion times and to ensure that no job can precede and succeed the same job. Constraints (5) ensure that no more than one job can be scheduled first at each machine. Note that there is no need for another set of constraints to guarantee that only one is scheduled last on each machine because this is guaranteed by constrains (5) in conjunction with (3). Constraints (6) specify that the decision variable x is binary over all domains. Constraints (7) state that the completion time for the dummy job 0 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 Π.

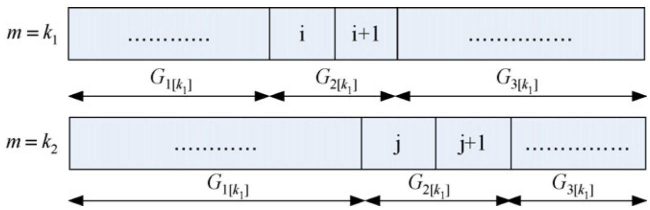

Fig. 1. A parallel machine schedule.

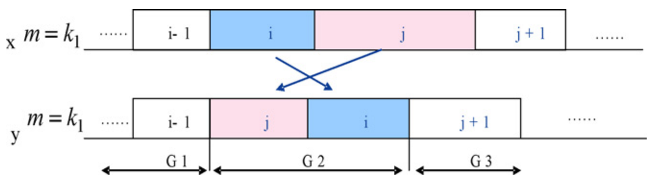

Fig. 2. The interchange on the same machine (k_1).

In order to derive the dominance properties for schedule Π, 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 $[j]$, $P_{[j][k]}$: 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 $P_{[j][k]}$ plus$S_{[i][j][k]}$. C_{k_1}: the completion time on k_1, $G_{1[k]}$: the job set before job $[i]$ on machine k, $G_{2[k]}$: the job set between job $[i]$ and job $[j+1]$ on machine k, $G_{2[k]}$: the job set after job $[i]$ on machine k.

There are two conditions in exchanging jobs and they include the intra-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 interchange

There are two cases to be considered within the inter-machine interchange, 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*:

$$(AP_{[i-1][j][k]} - AP_{[i-1][i][k]}) + (AP_{[j][i][k]} - AP_{[i][j][k]}) + (AP_{[i][j+1][k]} - AP_{[j][j+1][k]}) < 0$$

Proof. Suppose we exchange job i and job j in schedule Π_x which are adjacent to each other on the same machine. After exchanging these two neighborhood jobs, schedule Π_x is changed to Π_y. From Fig. 2, the jobs in the set $G_{1[k]}$ and $G'_{1[k]}$ remain the same, whose objective does not change. Thus, the completion time of $G_{1[k]}$ and $G'_{1[k]}$ is shown as follows:

$$G_{1[k]} = \sum_{a=1}^{i-1} AP_{[a-1][a][k]} = G'_{1[k]} \quad \square$$

Except for the $G_{1[k]}$ and $G'_{1[k]}$ 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.

$$G_{2[k]} = G_{1[k]} + AP_{[i-1][i][k]} + AP_{[i][j][k]} + AP_{[j][j+1][k]}$$

$$G_{3[k]} = G_{2[k]} + \sum_{a=j+2}^{n} AP_{[a-1][a][k]}$$

$$G'_{2[k]} = G_{1[k]} + AP_{[i-1][j][k]} + AP_{[j][i][k]} + AP_{[i][j+1][k]}$$

$$G'_{3[k]} = G'_{2[k]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k]}$$

When it comes to calculate the difference before and after we exchange the job i and job j, we subtract Π_y with Π_x, whose difference Δ is actually equal to $G'_{2[k]} - G'_{3[k]}$.

$$\begin{aligned} \therefore \Delta &= \prod_y - \prod_x = G'_{2[k]} - G_{2[k]} \\ &= (AP_{[i-1][j][k]} - AP_{[i-1][i][k]}) + (AP_{[j][i][k]} - AP_{[i][j][k]}) + (AP_{[i][j+1][k]} - AP_{[j][j+1][k]}) \end{aligned}$$

As a result, if the difference Δ 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.*

$$(AP_{[i-1][j][k]} - AP_{[i-1][i][k]}) + (AP_{[j][i+1][k]} - AP_{[i][i+1][k]}) + (AP_{[j-1][i][k]} - AP_{[j-1][j][k]}) + (AP_{[i][j+1][k]} - AP_{[j][j+1][k]}) < 0$$

Proof. The objective of $G_{1[k]}$ and $G'_{1[k]}$, is the same, so it will be eliminated after we subtract Π_y with Π_x. The equation of $G_{1[k]}$ and $G'_{1[k]}$ is shown as follows:

$$G_{1[k]} = \sum_{a=1}^{i-1} AP_{[a-1][a][k]} = G'_{1[k]} \quad \square$$

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.

$$G_{2[k]} = G_{1[k]} + AP_{[i-1][i][k]} + AP_{[i][i+1][k]} + AP_{[j-1][j][k]} + AP_{[j][j+1][k]}$$

$$G_{3[k]} = G_{2[k]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k]}$$

$$G'_{2[k]} = G_{1[k]} + AP_{[i-1][j][k]} + AP_{[j][i+1][k]} + AP_{[j-1][i][k]} + AP_{[i][j+1][k]}$$

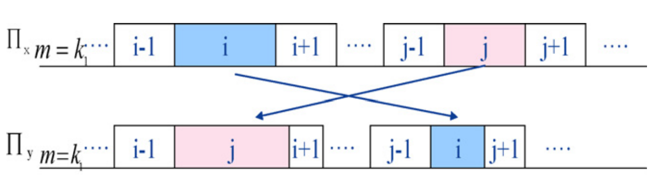

Fig. 3. Exchanging job i and job j on the same machine.

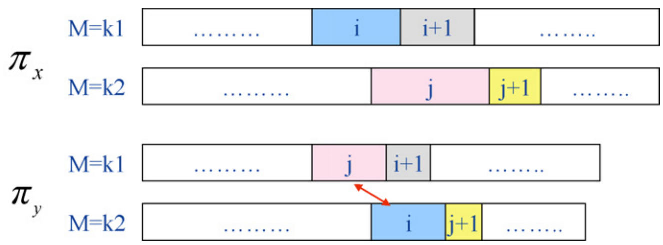

Fig. 4. Exchanging job *i* and job *j* on the different machines.

$$G'_{3[k]} = G'_{2[k]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k]}$$

When we subtract $\prod_y$ with $\prod_x$, the difference Δ is actually the $G'_{2[k]} - G_{2[k]}$. The equation of Δ is:

$$\begin{aligned}\Delta &= \prod\nolimits_y - \prod\nolimits_x = G'_{2[k]} - G_{2[k]}\\ &= (AP_{[i-1][j][k]} - AP_{[i-1][i][k]}) + (AP_{[j][i+1][k]} - AP_{[i][i+1][k]}) + (AP_{[j-1][i][k]} - AP_{[j-1][j][k]}) + (AP_{[i][j+1][k]} - AP_{[j][j+1][k]})\end{aligned}$$

Consequently, the job *i* and job *j* is exchanged only if the difference Δ 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 k_1 and k_2 indicates the machine number for job *i* and job *j*. Job *i* and job *j* exchanged from $\prod_x$ and $\prod_y$ (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.*

$$Max\left\{(G_{3[k_1]}+(AP_{[i-1][j][k_1]}-AP_{[i-1][i][k_1]})+(AP_{[j][i+1][k_1]}-AP_{[i][i+1][k_1]}),\ G_{3[k_2]}+(AP_{[j-1][i][k_2]}-AP_{[j-1][j][k_2]})+(AP_{[i][j+1][k_2]}-AP_{[j][j+1][k_2]})\right\} < C_{\max}$$

Proof. We compare the makespan $C_{\max}$, which is $Max(G_{3[k_1]}, G_{3[k_2]})$, 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 k_1 is calculated before machine k_2. □

4.2.1. The objective difference of machine k_1

The detailed completion time of each job sets on machine k_1 is shown as follows:

$$G_{1[k_1]} = \sum_{a=1}^{i-1} AP_{[a-1][a][k_1]} = G'_{1[k_1]}$$

$$G_{2[k_1]} = G_{1[k_1]} + AP_{[i-1][i][k_1]} + AP_{[i][i+1][k_1]}$$

$$G_{3[k_1]} = G_{2[k_1]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k_1]}$$

$$G'_{2[k_1]} = G'_{1[k_1]} + AP_{[i-1][j][k_1]} + AP_{[j][i+1][k_1]}$$

$$G'_{3[k_1]} = G'_{2[k_1]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k_1]}$$

After the above information is obtained, we are able to calculate the objective difference of machine k_1 by subtracting Π'_x to Π_x.

$$\begin{aligned}\Delta_{k_1} &= \prod\nolimits'_x - \prod\nolimits_x = (\prod\nolimits_x AP_{[i-1][j][k_1]} - AP_{[i-1][i][k_1]})\\ &\quad + (AP_{[j][i+1][k_1]} - AP_{[i][i+1][k_1]})\end{aligned}$$

4.2.2. The objective difference of machine k_2

The detailed completion time of each job sets on machine k_2 is shown as follows:

$$G_{1[k_2]} = \sum_{a=1}^{j-1} AP_{[a-1][a][k_2]} = G'_{1[k_2]}$$

$$G_{2[k_2]} = G_{1[k_2]} + AP_{[j-1][j][k_2]} + AP_{[j][j+1][k_2]}$$

$$G_{3[k_2]} = G_{2[k_2]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k_2]}$$

$$G'_{2[k_2]} = G'_{1[k_2]} + AP_{[j-1][i][k_2]} + AP_{[i][j+1][k_2]}$$

$$G'_{3[k_2]} = G'_{2[k_2]} + \sum_{a=j+2}^{n} AP_{[a\text{-}1][a][k_2]}$$

After the above information is obtained, we are able to calculate the objective difference of machine k_2 by subtracting Π'_y to Π_y.

$$\begin{aligned}\Delta_{k_2} &= \prod\nolimits'_y - \prod\nolimits_y = (AP_{[j-1][i][k_2]} - AP_{[j-1][j][k_2]})\\ &\quad + (AP_{[i][j+1][k_2]} - AP_{[j][j+1][k_2]})\end{aligned}$$

Finally, $C'_{\max}$ of the exchanged schedule is $Max(G_{3[k_1]} + \Delta_{k_1}, G_{3[k_2]} + \Delta_{k_2})$. Therefore, if $C'_{\max}$ is less than $C_{\max}$, the job *i* and job *j* are exchanged.

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. AP_{25} in Table 1 represents the adjusted processing times for job 5 sequenced after job2 in machine 1 with a value of 170. AP_{25} 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.

Table 1
The adjusted processing times in machine 1.

AP_1	1	2	3	4	5	6
0	116	142	130	109	157	152
1	–	167	166	122	179	141
2	127	–	120	145	170	180
3	143	165	–	150	143	145
4	124	165	157	–	173	137
5	118	162	158	108	–	137
6	132	170	136	151	181	–

Table 2
The adjusted processing times in machine 2.

AP_2	1	2	3	4	5	6
0	163	135	149	136	127	131
1	–	135	171	126	139	156
2	144	–	128	158	165	110
3	15	154	–	145	127	136
4	147	152	159	–	170	140
5	144	142	168	171	–	127
6	166	157	172	172	173	–

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:

Π_X: [**4**, **1**, 3] and Π_Y: [**1**, **4**, 3]

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

Because Π_Y is better than Π_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:

Π_X: [1, **4**, **3**] and Π_Y: [1,**3**, **4**]

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

Because Π_Y is not better than Π_X, the job 4 and job 3 are not exchanged.

(ii) Machine 2:

Step 1 :

Π_X: [**5**, **6**, 2] and Π_Y: [**6**, **5**, 2]

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

Because Π_Y is not better than Π_X, the job 5 and job 6 are not exchanged.

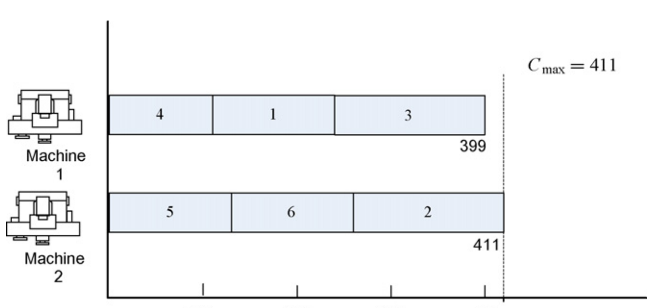

Fig. 5. The job sequence in each machine as an initial sequence.

Step 2

Π_X: [5, **6**, **2**] and Π_Y: [5,**2**, **6**]

$\Delta = (142 - 127) + (110 - 157) = -32 < 0$

Because Π_Y is better than Π_X, the job 6 is changed with job 2. The new sequence is [5, 2, 6]

The completion time of Machine 2 is $127 + 142 + 110 = 379$

$C_{\max} = 395$

(2) The job *i* and job *j* are adjacent jobs

Machine 1:

Π_X: [**1**, 4, **3**] and Π_Y: [**3**, 4, **1**]

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

Because Π_Y is not better than Π_X, the schedule is not exchanged.

Machine 2:

Π_X: [**5**, 2, **6**] and Π_Y: [**6**, 2, **5**]

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

Because Π_Y is not better than Π_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:

Π_X: Machine 1: [**1**, 4, 3] and Machine 2: [5, 2, 6]
Π_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 Π_Y is not better than Π_Y, we do not exchange the schedule.

Step 2:

Π_X : Machine 1: [**1**, 4, 3] and Machine 2: [5, **2**, 6]
Π_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 Π_Y is not better than Π_X, we do not exchange the schedule.

Step 3:

Π_X: Machine 1: [**1**, 4, 3] and Machine 2: [5, 2, **6**]

Π_Y : Machine 1: [**6**, 4, 3] and Machine 2: [5,2, **1**]

$$\Delta_{m1} = (152 - 116) + (151 - 122) = 65$$

$$\Delta_{m2} = (144 - 110) = 34$$

$$C'_{\max} = \max(395 + 65, 379 + 34) = 460$$

$$460 > 395$$

Because Π_Y is not better than Π_X, we do not exchange the schedule.

As a result, the final sequences for the two machines are:

Machine 1: [1,4, 3], $C_{m1} = 395$
Machine 2: [5,2, 6], $C_{m2} = 379$
The makespan is $C_{\max} = 395$ which is shown in Fig. 6.

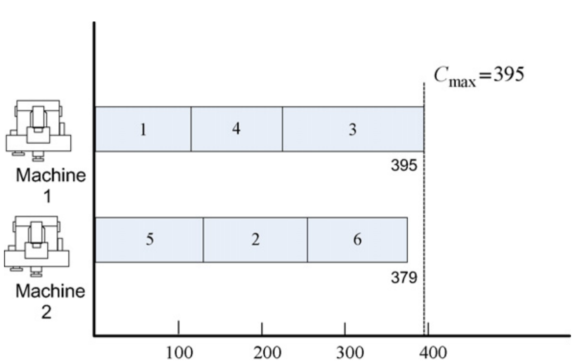

Fig. 6. The representation of the final sequence of the two machines.

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 hybrid algorithm that combines these dominance properties with genetic algorithm and simulated annealing. They are genetic algorithms with dominance properties (GADP) and simulated annealing 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.

Table 3
The levels of the population size, crossover rate, and mutation rate.

Factors	Levels
Population size (PopSize)	100, 200
Crossover rate (Pc)	0.6, 0.9
Mutation rate (Pm)	0.1, 0.5

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 4
The parameter settings of GA.

Parameter	Value
Population size	100
Crossover rate	0.6
Mutation rate	0.5
Max number of iterations	500 × number of jobs

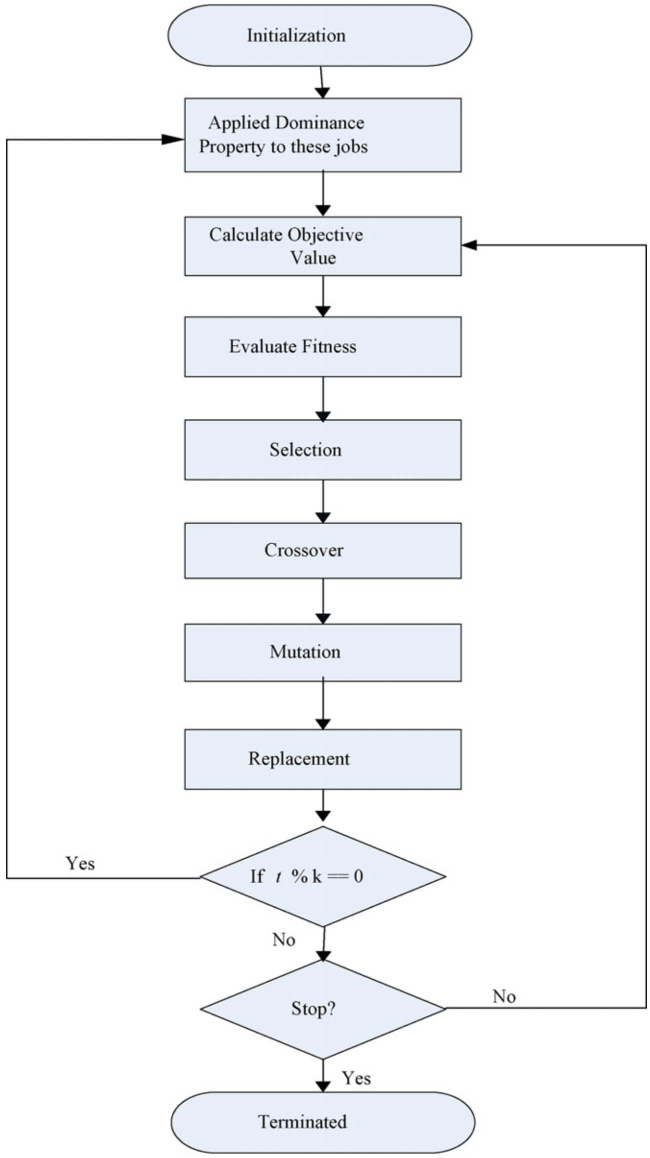

Fig. 7. Procedure of GADP2.

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.

initialTemperature: The initial temperature
α: The cooling parameter
currentTemperature: The current temperature
finalTemperature: The final temperature and it is used as the termination criterion
numberOfMoves: The number of local search within the same temperature
chromosome1: The solution object stores current solution and corresponding objective values
tempChromosomes: It is also a solution object which stores temporarily solution by different kinds of moves
best: The best solution found so far

Main()://the main procedures of SADP

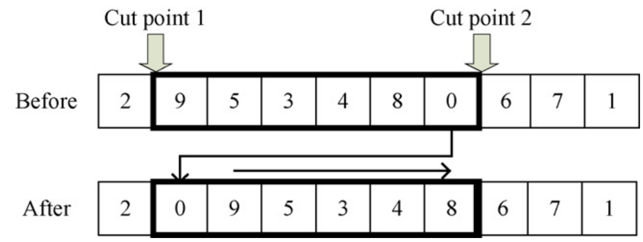

Fig. 8. Shift move.

```
1.  initialParameters()
2.  initialStage();
3.  while counter < numberOfSolutionsExamined do
4.    for i = 0 to numberOfMoves do
5.      tempChromosomes = getMoves();
6.      calcObjectiveValue();
7.      tempChromosomes[0] = selectBetterMoves ();
8.      acceptanceRule();
9.    End for
10.   currentTemperature * = α;
11. End while
```

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, 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.

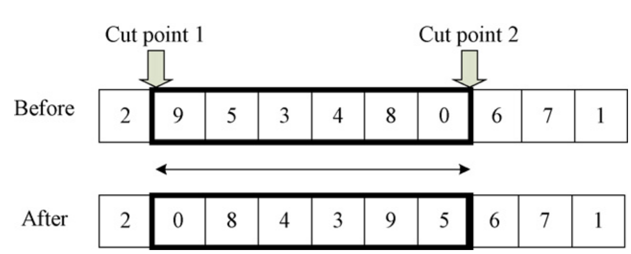

Fig. 9. Inverse move.

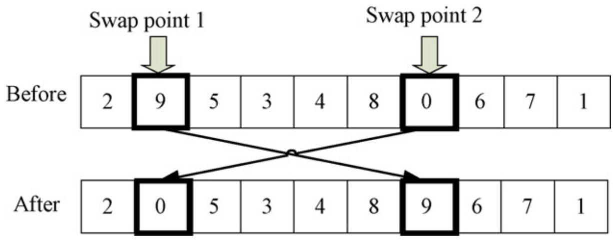

Fig. 10. The swap move.

Table 5
The parameter settings.

Parameter	Value
Initial temperature	T_0 = (lower bound) + (upper bound − lower bound)/10
Final temperature	T_f = lower bound
Cooling coefficient α	0.99
Max number of iterations	500 × number of jobs

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.

Table 6
ANOVA results of these three factors (together with block factors: instance and machine type).

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Instances	89	1.18E+11	1.18E+11	1.33E+09	6491.68	0
Type	2	8.19E+09	8.19E+09	4.1E+09	20015.06	0
Popsize	1	567,604	567,604	567,604	2.77	0.096
Pc	1	64,951	64,951	64,951	0.32	0.573
Pm	1	92,720	92,720	92,720	0.45	0.501
Popsize × Pc	1	35,661	35,661	35,661	0.17	0.676
Popsize × Pm	1	347,090	347,090	347,090	1.7	0.193
Pc × Pm	1	909,950	909,950	909,950	4.45	0.035
Popsize × Pc × Pm	1	57,683	57,683	57,683	0.28	0.596
Error	21,501	4.4E+09	4.4E+09	204,697		
Total	21,599	1.31E+11				

$$U(0,1) \le e^{-\frac{f(X')-f(x)}{T'}}$$

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 *currentTemperature* < *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

Table 7
The experimental result of the balanced instances.

m	n	DP Min	DP Avg	DP Max	DP StDev	SGA Min	SGA Avg	SGA Max	SGA StDev	GADP Min	GADP Avg	GADP Max	GADP StDev
2	20	1242	1255	1268	6.30	1269	1301	1330	15.23	1242	1255	1267	6.23
	40	2446	2469	2487	10.17	2551	2608	2676	31.59	2446	2469	2487	10.06
	60	3665	3698	3740	18.62	3864	3952	4052	46.79	3664	3695	3734	16.67
	80	4863	4893	4925	14.73	5188	5291	5424	58.71	4859	4892	4923	15.13
6	20	450	458	466	4.22	455	472	488	8.36	448	456	463	4.04
	40	821	848	873	12.86	884	914	942	13.91	815	838	861	11.73
	60	1238	1267	1294	13.65	1343	1378	1414	17.27	1225	1252	1277	12.79
	80	1639	1675	1709	16.76	1805	1849	1887	20.59	1623	1655	1684	14.52
12	20	234	241	248	3.29	248	260	270	5.54	234	240	247	3.04
	40	451	468	484	8.66	474	491	509	8.58	444	456	471	6.66
	60	647	674	698	13.45	687	708	728	10.28	629	652	673	11.02
	80	845	891	937	23.16	928	954	979	12.08	821	852	889	16.88

m	n	GADP2 Min	GADP2 Avg	GADP2 Max	GADP2 StDev	SA Min	SA Avg	SA Max	SA StDev	SADP Min	SADP Avg	SADP Max	SADP StDev
2	20	1242	1254	1266	6.03	1297	1395	1467	31.20	1196	1255	1338	33.80
	40	2441	2459	2474	8.26	2718	2814	2903	35.50	2371	2462	2550	35.90
	60	3652	3675	3695	10.59	4145	4265	4369	42.20	3588	3680	3764	41.20
	80	4846	4872	4896	11.89	5569	5700	5843	49.70	4753	4879	5045	59.10
6	20	448	454	459	3.08	470	505	531	10.24	441	455	481	8.80
	40	809	831	853	11.00	933	970	997	10.48	796	841	892	18.53
	60	1219	1246	1270	12.05	1395	1444	1481	11.90	1210	1259	1295	15.50
	80	1622	1648	1672	12.47	1883	1928	1964	13.40	1606	1662	1705	18.80
12	20	235	239	244	2.29	259	280	294	5.37	229	241	265	5.15
	40	444	455	468	5.91	501	522	536	6.45	440	466	493	10.84
	60	809	649	667	10.08	714	743	759	6.46	628	669	715	15.25
	80	819	849	884	16.51	967	995	1009	7.05	819	891	959	24.34

Table 8
The experimental result of the dominant processing time instances.

m	n	DP Min	DP Avg	DP Max	DP StDev	SGA Min	SGA Avg	SGA Max	SGA StDev	GADP Min	GADP Avg	GADP Max	GADP StDev
2	20	1989	2002	2014	6.18	2021	2048	2081	14.77	1989	2002	2014	6.17
	40	3952	3980	4012	15.01	4037	4100	4174	32.80	3952	3979	4012	14.66
	60	5926	5959	6005	18.59	6089	6182	6290	49.80	5925	5957	5994	16.66
	80	7879	7930	7988	31.20	8116	8256	8420	73.42	7878	7924	7974	27.41
6	20	749	757	764	4.07	749	766	786	9.11	748	754	761	3.81
	40	1336	1356	1378	10.31	1403	1434	1463	14.71	1332	1349	1369	9.12
	60	2020	2041	2061	9.47	2092	2128	2164	17.53	1996	2032	2053	12.96
	80	2642	2676	2709	16.73	2802	2852	2901	23.57	2630	2661	2690	14.75
12	20	386	392	401	3.64	399	410	419	5.16	385	392	399	3.43
	40	750	767	784	8.64	770	791	809	9.39	744	756	769	6.21
	60	1034	1069	1093	13.87	1063	1083	1103	10.03	1005	1031	1058	13.38
	80	1362	1407	1460	24.47	1450	1477	1501	12.93	1339	1368	1408	17.23

m	n	GADP2 Min	GADP2 Avg	GADP2 Max	GADP2 StDev	SA Min	SA Avg	SA Max	SA StDev	SADP Min	SADP Avg	SADP Max	SADP StDev
2	20	1987	1999	2011	5.83	2050	2145	2218	30.60	1920	2001	2079	37.30
	40	3944	3971	4010	15.18	4229	4325	4413	32.60	3878	3977	4087	35.60
	60	5916	5943	5967	12.54	6365	6524	6644	48.10	5851	5949	6096	48.10
	80	7857	7890	7918	14.71	8555	8715	8854	50.80	7792	7900	8058	61.60
6	20	747	752	757	3.00	765	805	829	9.95	731	753	781	7.32
	40	1326	1341	1359	7.84	1453	1495	1522	11.40	1307	1348	1398	16.10
	60	2004	2028	2043	9.33	2133	2201	2239	14.10	2001	2033	2073	12.50
	80	2621	2650	2678	14.35	2887	2955	2992	15.70	2603	2664	2713	20.00
12	20	385	391	396	2.68	409	432	445	5.77	378	393	406	5.15
	40	743	754	767	5.80	796	826	842	7.61	736	763	797	11.33
	60	1003	1028	1055	13.03	1097	1122	1139	6.21	1007	1065	1103	15.30
	80	1337	1365	1402	15.72	1498	1523	1541	7.73	1329	1407	1480	26.50

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

$$\text{lower bound} = \frac{\sum_{j=1}^{n} arg\ \min(AP_{ijk})}{m}, \quad i \in jobs, k \in machines \tag{9}$$

$$\text{upper bound} = \frac{\sum_{j=1}^{n} arg\ \max(AP_{ijk})}{m}, \quad i \in jobs, k \in machines \tag{10}$$

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.

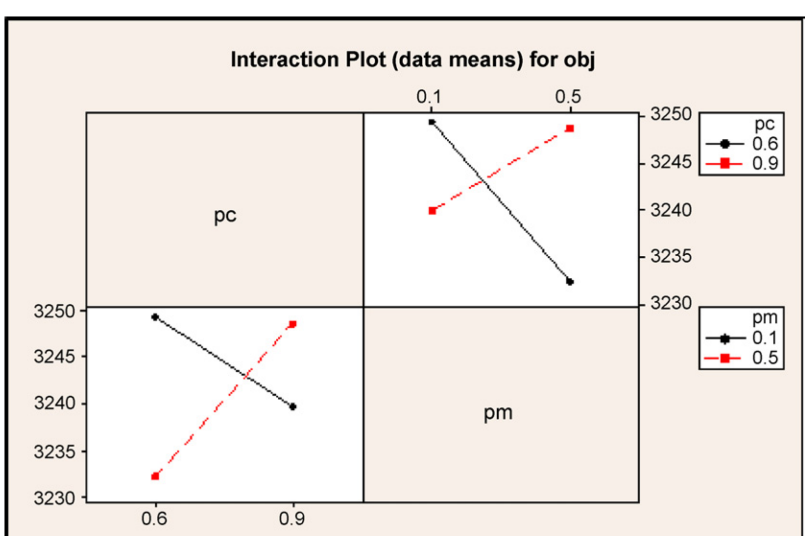

Fig. 11. Interaction plot of Pc and Pm.

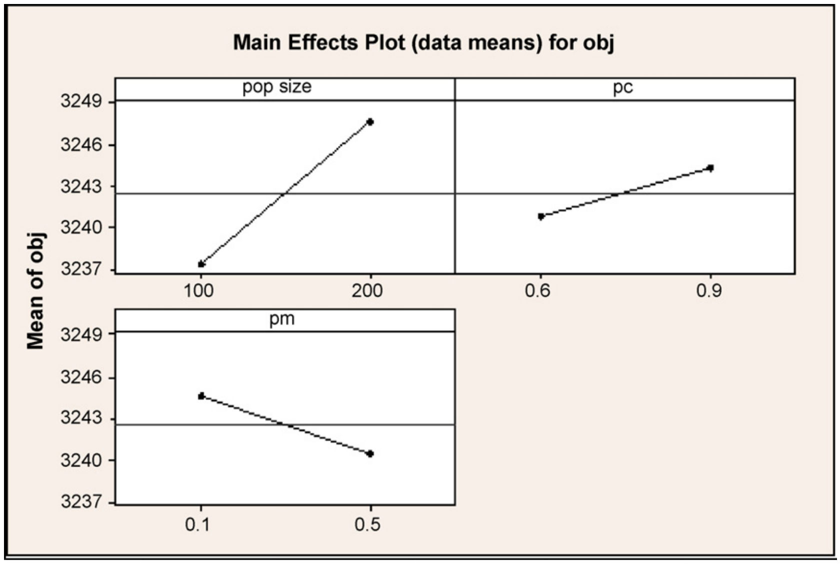

Fig. 12. Main effect plot of the three factors of the GA.

Table 9
The experimental result of the dominant setup time instances.

m	*n*	DP				SGA				GADP			
		Min	Avg	Max	StDev	Min	Avg	Max	StDev	Min	Avg	Max	StDev
2	20	1994	2007	2020	6.38	2023	2051	2085	14.45	1993	2006	2019	6.21
	40	3942	3970	4011	17.01	4031	4097	4162	32.59	3941	3969	4009	16.66
	60	5908	5942	5976	16.39	6073	6166	6272	51.55	5907	5940	5973	15.97
	80	7850	7891	7965	25.27	8102	8236	8380	69.01	7850	7888	7934	19.99
6	20	748	758	766	4.87	751	768	786	8.76	746	754	764	4.84
	40	1337	1356	1375	8.60	1404	1436	1466	14.99	1334	1350	1368	8.40
	60	2019	2040	2060	9.79	2090	2130	2163	17.99	2006	2032	2052	11.09
	80	2643	2676	2706	15.25	2805	2850	2898	22.87	2631	2660	2689	14.07
12	20	383	390	398	3.58	398	410	421	5.67	383	389	396	3.26
	40	750	765	783	8.95	769	790	808	9.66	743	755	768	6.63
	60	1031	1069	1093	14.75	1061	1082	1102	10.73	1005	1030	1056	12.87
	80	1361	1404	1454	21.59	1453	1478	1502	12.25	1340	1366	1401	15.55
m	*n*	GADP2				SA				SADP			
		Min	Avg	Max	StDev	Min	Avg	Max	StDev	Min	Avg	Max	StDev
2	20	1990	2002	2014	6.02	2030	2146	2212	28.70	1927	2005	2086	27.80
	40	3935	3964	4001	15.67	4203	4323	4430	35.10	3889	3970	4066	33.10
	60	5898	5926	5950	12.25	6376	6514	6618	42.10	5830	5932	6023	38.90
	80	7829	7864	7894	15.62	8519	8698	8827	48.20	7734	7875	8002	54.50
6	20	747	753	759	3.25	766	806	830	10.35	736	755	783	9.30
	40	1329	1344	1360	7.49	1458	1498	1522	11.90	1315	1349	1392	13.70
	60	2004	2027	2042	9.04	2155	2205	2236	12.30	1991	2032	2072	15.70
	80	2627	2653	2676	12.51	2907	2953	2996	15.30	2606	2666	2717	18.10
12	20	383	388	394	2.66	414	432	444	5.17	379	390	414	4.81
	40	741	752	765	6.14	797	826	846	7.86	734	760	790	10.61
	60	1000	1028	1057	13.35	1091	1122	1137	6.54	1008	1066	1103	16.30
	80	1334	1362	1398	15.00	1496	1523	1540	7.96	1326	1400	1471	24.50

Table 10
ANOVA test of the experiment.

Source	DF	SS	Mean square	*F*-Value	Pr > *F*
Type	1	2.105E+10	2.105E+10	4.25E+07	<.0001
Instances	155	2.948E+11	1.902E+09	3,840,723	<.0001
Type × instances	155	1.526E+10	98,428,645	198,755	<.0001
Method	5	757,891,007	151,578,201	306,078	<.0001
Type × method	5	470046.79	94009.357	189.83	<.0001
Instances × method	775	651,513,926	840663.13	1697.53	<.0001
Type × instance × method	775	3031145.1	3911.155	7.9	<.0001
Instances × method	775	651,513,926	840663.13	1697.53	<.0001
Type × instance × method	775	3031145.1	3911.155	7.9	<.0001
Error	95,328	47,209,017	495.22718		
Corrected total	97,199	4.203E+11			

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

Duncan grouping	Mean	*N*	Method
A	2435.43	16,200	SA
B	2311.14	16,200	SGA
C	2206.63	16,200	DP
D	2200.37	16,200	SADP
E	2195.94	16,200	GADP
F	2189.06	16,200	GADP2

Table 12
The comparison between GADP2 and PH's Algorithm.

m	*n*	Balanced processing time			Dominant processing time			Dominant setup time		
		LB	GADP2	PH's Algorithm	LB	GADP2	PH's Algorithm	LB	GADP2	PH's Algorithm
2	20	1186	1254	1296	1933	1999	2046	1936	2002	2050
	40	2345	2459	2521	3844	3971	4019	3834	3964	4031
	60	3510	3675	3733	5767	5943	5997	5755	5926	5980
	80	4665	4872	4927	7672	7890	7933	7649	7864	7914
6	20	362	454	480	613	752	779	611	753	776
	40	717	831	848	1214	1341	1361	1218	1344	1371
	60	1071	1246	1232	1823	2028	1965	1822	2027	1969
	80	1429	1648	1626	2426	2650	2673	2428	2653	2670
12	20	175	239	250	300	391	402	299	388	398
	40	347	455	473	597	754	769	597	752	769
	60	519	649	627	894	1028	986	894	1028	990
	80	690	849	830	1191	1365	1348	1192	1362	1357

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 machines scheduling problems. There are three heuristics sequentially applied in the PH algorithm to minimize the makespan, including the constructive, improvement, and a traveling salesman problem (TSP)-like heuristic. Constructive is the first heuristic which assigns the jobs to machines. Secondly, an improvement procedure is used to improve the previous constructed solution. Finally, similar to the heuristic of traveling salesman problem, a TSP-like heuristic is applied, which sets the sequence of jobs on each machine.

The following Tables 7–9 are the empirical result of the different types of instances. DP heuristic performs the best among these algorithms such as GA, SA and DP. Moreover, when GA and SA are combined with DP, the overall performances are further enhanced.

In order to validate the significance among these algorithms, ANOVA is employed to test it, which shows significant differences of these factors in Table 10. The Duncan pair-wise comparison tests the significance among these methods in Table 11. The Duncan grouping presents each algorithm located in different group because they share different alphabet. As a result, there is statistical significance between any two algorithms. The best algorithm is GADP2, the second best is GADP, and the worst is SA.

Finally, because GADP2 is the best among all algorithms, it is compared with PH' 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.

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

[1] Allahverdi, J.N.D. Gupta, T. Aldowaisan, A review of scheduling research involving setup considerations, Omega 27 (2) (1999) 219–239.
[3] M. Asano, H. Ohta, A heuristic for job shop scheduling to minimize total weighted tardiness, Computers and Industrial Engineering 42 (2–4) (2002) 137–147.
[4] P. Brucker, S. Knust, A. Schoo, O. Thiele, A branch and bound algorithm for the resource-constrained project scheduling problem, European Journal of Operational Research 107 (2) (1998) 272–288.
[5] P.C. Chang, J.C. Hsieh, Y.W. Wang, Genetic algorithms applied in BOPP film scheduling problems: minimizing total absolute deviation and setup times, Applied Soft Computing 3 (2) (2003) 139–148.
[6] P.C. Chang, S.H. Chen, K.L. Lin, Two phase sub-population genetic algorithm for parallel machine scheduling problem, Expert Systems with Applications 29 (3) (2005) 705–712.

[7] P.C. Chang, J.C. Hsieh, C.H. Liu, A case-injected genetic algorithm for single machine scheduling problems with release time, International Journal of Production Economics 103 (2) (2006) 551–564.
[8] P.C. Chang, J.C. Hsieh, C.Y. Wang, Adaptive multi-objective genetic algorithms for scheduling to drilling operation of printed circuit board industry, Applied Soft Computing Journal 7 (3) (2007) 800–806.
[9] P.C. Chang, S.-H. Chen, C.-Y. Fan, Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems, Applied Soft Computing Journal 8 (1) (2008) 767–777.
[10] P.C. Chang, S.H. Chen, The development of a sub-population genetic algorithm II (SPGAII) for the Multi-objective combinatorial problems, Applied Soft Computing Journal 9 (1) (2009) 173–181.
[11] P.C. Chang, T.W. Liao, Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory, Applied Soft Computing 6 (2) (2006) 198–206.
[12] F.D. Chou, P.C. Chang, H.M. Wang, A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem, International Journal of Advanced Manufacturing Technology 31 (3/4) (2006) 350–359.
[13] D.T. Connolly, An improved annealing scheme for the quadratic assignment problems, European Journal of Operational Research 46 (1) (1990) 93–100.
[14] S. Dunstall, A. Wirth, Heuristic methods for the identical parallel machine flow time problem with set-up times, Computers and Operations Research 32 (9) (2005) 2479–2491.
[15] P.M. França, M. Gendreau, G. Laporte, F.M. Müller, A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times, International Journal of Production Economics 43 (2/3) (1996) 79–89.
[16] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1979.
[17] M. Ghirardi, C.N. Potts, Makespan minimization for scheduling unrelated parallel machines: a recovering beam search approach, European Journal of Operational Research 165 (2) (2005) 457–467.
[18] C.A. Glass, C.N. Potts, P. Shade, Unrelated parallel machine scheduling using local search, Mathematical and Computer Modelling 20 (2) (1994) 41–52.
[19] S.C. Graves, A review of production scheduling, Operations Research 29 (4) (1981) 646–675.
[20] A. Guinet, A. Dussauchy, Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria, International Journal of Production Research 31 (1993) 1579–1594.
[21] J.C. Hsieh, P.C. Chang, L.C. Hsu, Scheduling of drilling operations in printed-circuit-board factory, Computers and Industrial Engineering 44 (3) (2003) 461–473.
[22] D.W. Kim, K.H. Kim, W. Jane, F.F. Chen, Unrelated parallel machine scheduling with setup times using simulated annealing, Robotics and Computer Integrated Manufacturing 18 (3/4) (2002) 223–231.
[23] D.W. Kim, D.G. Na, F.F. Chen, Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective, Robotics and Computer Integrated Manufacturing 19 (1/2) (2003) 173–181.
[24] S.C. Kim, P.M. Bobrowski, Scheduling jobs with uncertain setup times and sequence dependency, Omega 25 (4) (1997) 437–447.
[25] M.E. Kurz, R.G. Askin, Heuristic scheduling of parallel machines with sequence-dependent set-up times, International Journal of Production Research 39 (16) (2001) 3747–3769.
[26] G. Lancia, Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the Makespan, European Journal of Operational Research 120 (2) (2000) 277–288.
[27] C.Y. Liaw, Y.K. Lin, C.Y. Chen, M. Chen, Scheduling unrelated parallel machines to minimize total weighted tardiness, Computers & Operations Research 30 (12) (2003) 1777–1789.
[28] Y. Lin, W. Li, Parallel machine scheduling of machine-dependent jobs with unit-length, European Journal of Operational Research 156 (1) (2004) 261–266.
[29] C.Y. Liu, S.C. Chang, Scheduling flexible flow shops with sequence-dependent setup effects, IEEE Transactions on Robotic Automation 16 (4) (2000) 408–429.
[30] S. Martello, F. Soumis, P. Toth, Exact and approximation algorithms for Makespan minimization on unrelated parallel machines, Discrete Applied Mathematics 75 (2) (1997) 169–188.
[31] H. Missbauer, Order release and sequence-dependent setup times, International Journal of Production Economics 49 (2) (1997) 131–143.
[32] E. Mokotoff, Parallel machine scheduling problems: a survey, Asia-Pacific, Journal of Operational Research 18 (2) (2001) 193–242.
[33] J.-C. Picard, M. Queyranne, On the integer-valued variables in the linear vertex packing problem, Mathematical Programming 12 (1) (1977) 97–101.
[34] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice Hall, NJ, 1995.
[35] G. Rabadi, R.J. Moraga, A. Al-Salem, Heuristics for the unrelated parallel machine scheduling problem with setup times, Journal of Intelligent Manufacturing 17 (1) (2006) 85–97.
[36] S. Radhakrishnan, J.A. Ventura, Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times, International Journal of Production Research 38 (10) (2000) 2233–2252.
[38] B. Srivastava, An effective heuristic for minimizing makespan on unrelated parallel machines, Journal of the Operational Research Society 49 (8) (1998) 886–894.
[39] K.-C. Tan, R. Narasimhan, P.A. Rubin, G.L. Ragatz, A Comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times, Omega 28 (3) (2000) 313–326.
[40] R. Zhang, C. Wu, A hybrid immune simulated annealing algorithm for the job shop scheduling problem, Applied Soft Computing 10 (1) (2010) 79–89.